Annual Reviews in Control 41 (2016) 147–158

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Full Length Article

Real-time management of complex resource allocation systems: Necessity, achievements and further challenges☆☆☆

Spyros Reveliotis*

School of Industrial & Systems Engineering, Georgia Institute of Technology, United States

ARTICLE INFO

Article history:
Received 18 November 2015
Revised 4 January 2016
Accepted 10 January 2016
Available online 26 April 2016

Keywords:
Resource Allocation Systems
Discrete Event Systems
Supervisory control
Deadlock avoidance
(Stochastic) scheduling

ABSTRACT

Many contemporary applications, ranging from flexibly automated production systems, to automated material handling and intelligent transportation systems, to internet-based workflow management systems, and more recently, to the massively parallelized software systems that emerge in the context of the novel multi-core computing architectures, can be perceived as a set of finite resources that support a number of concurrently running processes. These processes execute in a staged manner and, at each stage, they vie for the allocation of various subsets of the system resources. To effectively support and manage the extensive levels of concurrency and operational flexibility that are contemplated for these environments, and the ensuing complexity, there is a substantial need for formal models and tools that will enable the modeling, analysis and eventually the control of the aforementioned resource allocation function so that the resulting dynamics are, both, behaviorally correct and operationally efficient. This article overviews a research program that seeks to address the aforementioned need by using the unifying abstraction of the resource allocation system (RAS) and supporting modeling frameworks, like automata, Petri nets, and Markov reward and decision processes, borrowed from the area of Discrete Event Systems (DES) theory. The presented results take advantage of the special structure that exists in the considered RAS classes, and they are characterized by their analytical rigor and computational tractability. The article also highlights the further challenges that must be addressed for the successful completion and promotion of the pursued framework.

1. Introduction

1.1. The ever-increasing need for concurrency and flexibility in many contemporary operations and the corresponding resource allocation function

The efficient and expedient allocation of a finite set of resources to a number of contesting processes is a ubiquitous problem, arising in various operational settings of our contemporary technological civilization. Indeed, cost-effectiveness and responsiveness are predominant concepts in modern corporate strategy and typical requirements for many everyday functions (Curry & Feldman, 2011; Heizer & Render, 2010). In the context of the resource allocation functions considered in this paper, cost-effectiveness is based on the ability to "make the most" – i.e., maintain a high utilization – of the engaged resources. On the other hand, the posed requirement for high responsiveness traditionally has implied the ability to fill an arising demand or to support an emerging service need in a timely manner. But in the context of many contemporary operations, responsiveness also implies the ability to diversify these operations on the basis of various attributes of their target groups, and evolve them according to externally determined conditions and trends. As a more concrete example of this quest for extensive diversification, one can consider the mass-customization practices that have been adopted by various industries, like the automotive and computer manufacturers. The effective support of all the aforestated requirements necessitates the further deployment, at the operational level, of high levels of concurrency and flexibility; i.e., a need to support, through the same set of resources, the simultaneous execution of a broad set of diversified workflows, while each of these workflows takes place at a low or moderate rate, and evolves continually into new operational patterns.

In more concrete terms, the general trends described in the above paragraph have been pretty conspicuous in the manufacturing domain, where they have given rise to the proverbial concept

☆ This article is based on a keynote presentation that was given by the author at the 5th International Workshop on Dependable Control of Discrete Systems (DCDS 2015).

☆☆ This work has been partially supported by NSF grant ECCS-1405156.

* Corresponding author.
E-mail address: spyros.reveliotis@isye.gatech.edu

http://dx.doi.org/10.1016/j.arcontrol.2016.04.009
1367-5788/

of the "flexible manufacturing system" (FMS) (Groover, 1996; Tolio, 2009). The prototypical FMS consists of a set of numerically controlled workstations interconnected by an automated material handling system, and with the workflow taking place in the entire facility being integrated and coordinated by a computerized controller (Groover, 1996).

More recently, the aforementioned trend for a flexible automation/mechanization of the workflows taking place in the contemporary production systems has been extended to the service sector through the concept of the "workflow management system" (Lawrence, 1997). This is a computerized tool that supports the definition, the enactment and the coordination of the execution of business workflows, by monitoring their progress and assigning the necessary resources to them; the resources involved can range from data files, to supporting processing software, to printing and communication means, and even the humans that might be necessary for the support and the authorization of certain steps. It is currently believed that various routine transactions in the banking sector, in the supply chain management and logistics services, in insurance claim processing, and even in the broader health-care sector, can be rendered more responsive and efficient through their mechanization by the successful deployment of a workflow management system.

In the transportation sector, the aforementioned resource allocation paradigm manifests itself in any set-up where a set of concurrently traveling vehicles have to negotiate the necessary traveling space in the context of some "zoning scheme" that ensures collision-free and safe operation. Trains being successively allocated a sequence of segments of the underlying railway network is a typical example of such a "zone"-based operation (Giua, Fanti, & Seatzu, 2006). Also, in the industrial sector, similar zoning schemes have been implemented in the operation of unit-load automated guided vehicle (AGV) and monorail-based material-handling systems (Reveliotis, 2000; Roszkowska & Reveliotis, 2008), and in the operation of the hoist and the crane systems that support the material-handling operations at many ports and freight terminals (Tompkins, White, Bozer, & Tanchoco, 2010).

Finally, another domain where the aforementioned resource allocation functions (and problems) are very prominent, is in the software and the computational platforms that control the aforementioned operations as well as any other operation that is supported by our modern technologies. Indeed, since its early days, our modern computing technology has employed extensive levels of (actual or virtual) concurrency, where a number of software threads run in parallel, each of them tasked with a particular role and function. These threads need to share the limited resources of the underlying computer platform (CPUs, registers, I/O devices, files, etc.) in a way that provides exclusive access to the requested resources; the corresponding coordination is attained through the use of a set of tokens, that are known as "mutually exclusive (mutex) locks" or "semaphores", and constitute a "pass" for accessing the corresponding resource (Holt, 1972; Wang et al., 2010).

1.2. The additional quest for extensive automation and autonomy and its implications

The above discussion regarding the increased levels of efficiency/cost-effectiveness, responsiveness, concurrency and flexibility that are requested for many contemporary operations, and the accompanying examples, also render pretty clear that these requirements lead to operations that are characterized by a high level of operational complexity. And this complexity translates to some challenging scheduling problems for the underlying resource allocation functions.

In many cases, including all of the aforementioned examples, things are further complicated by the extensive levels of automation and autonomy that is requested by the considered operations. The need for automation can arise from technological and/or feasibility considerations (as in the case of the multithreaded software mentioned above, and in the operations taking place in the modern semiconductor fabs that must be isolated from the polluting effect of the human element), or from considerations concerning the operational and financial efficiency of the underlying operation (as in the aforementioned workflow management systems and the driverless transportation systems). In either case, the removal of the human element from the underlying processes implies that the deployed controllers, and especially the resource allocation functions involved, must not only be efficient, but also correct and robust to logical problems and errors that, in more traditional settings, have been addressed by human intervention and improvisation.

A typical such logical problem in the context of the considered resource allocation functions is the formation of (partial) deadlock, i.e., situations where a set of the concurrently executing processes are entangled in a circular waiting pattern, each of them waiting for some of the other processes to release resources that are necessary for its advancement. Clearly, the formation of any of the aforementioned deadlocks is a pernicious situation that stalls the further advancement of the processes involved and drives to zero the utilization of all the resources that have been allocated to these processes. At the same time, it should be obvious that deadlock is a natural consequence of the concurrency and the flexibility, and, finally, the arbitrary structure of the corresponding resource allocation function that is implied by the first two requirements; therefore, it constitutes a ubiquitous problem for the operational environments discussed in the previous paragraphs.

The bottom line of all the above discussion is that, in the context of the considered automated operations, the underlying resource allocation functions must be controlled for operational efficiency, cost effectiveness and responsiveness, and also for correctness and robustness to certain problems of a more qualitative or "logical" nature, like the formation of partial deadlock. In the rest of this article we present the results of a research program that has sought to provide a systematic and rigorous solution to this challenging resource allocation problem by employing and extending results from modern control theory.[1]

2. Resource Allocation Systems and the proposed control framework

The presented research program has sought to address the control problem that was outlined in the introductory section, in a systematic and rigorous manner, by (i) abstracting the considered operations through the notion of a (sequential) Resource Allocation System (RAS) (Reveliotis, 2005), and (ii) employing and extending results coming from the controls area of Discrete Event Systems (DES) (Cassandras & Lafortune, 2008). In this section, first we introduce the formal notion of the sequential RAS, and subsequently we outline the DES-based control framework that has been proposed for these RAS. We also present a RAS taxonomy that has

[1] We also want to render clear at this point that we do not claim an exhaustive cataloging of every single work pertaining to the aforementioned problem of controlling the considered resource allocation functions for logical correctness and operational efficiency. Apart from the intractability of this task given the volume of the existing literature, especially w.r.t. the logical control problem of deadlock avoidance, we also believe that such an approach would not be very constructive for the reader. Hence, our choice has been to focus on a body of results that provide a coherent and principled approach to the considered problem, and define a rigorous analytical base for the necessary modeling, analysis and control synthesis functions w.r.t. this problem. Furthermore, our selection has tried to cover all the key insights and the primary methodologies that are currently available for the considered problems, but at the same time it is confined by the typical space and other limitations that characterize such a survey paper.

been instrumental in the investigation of the relevant control problems. These problems and the currently available results for them, as well as the remaining open challenges, will be addressed in subsequent sections.

2.1. Sequential Resource Allocation Systems

A sequential RAS is formally defined in Reveliotis (2005) by a quintuple $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A}, \mathcal{D} \rangle$, where: (i) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*. (ii) $C : \mathcal{R} \to \mathbb{Z}^+$ – the set of strictly positive integers – is the system *capacity* function, specifying the number of identical units from each resource type available in the system. Resources are assumed to be *reusable*, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore, $C(R_i) \equiv C_i$ constitutes a system *invariant* for each i. (iii) $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ denotes the set of the system *process types* supported by the considered system configuration. Each process type Π_j is a composite element itself, in particular, $\Pi_j = \langle \Delta_j, \mathcal{G}_j \rangle$, where: (a) $\Delta_j = \{\Xi_{j1}, \ldots, \Xi_{j,l_j}\}$ denotes the set of *processing stages* involved in the definition of process type Π_j, and (b) $\mathcal{G}_j$ is an additional data structure that encodes the sequential logic that integrates the set of the processing stages Δ_j into a set of potential process flows. (iv) $\mathcal{A} : \Delta \to \prod_{i=1}^{m} \{0, \ldots, C_i\}$ is the *resource allocation function* associating every processing stage Ξ_{jk} with the *resource allocation vector* $\mathcal{A}(\Xi_{jk}) \neq \mathbf{0}$ required for its execution. (v) $\mathcal{D}$ is a function mapping each processing stage Ξ_{jk} in $\Delta \equiv \bigcup_{j=1}^{n} \Delta_j$ to a distribution with positive support that characterizes the *"processing time"*of the corresponding processing stage. Finally, we also set $\xi \equiv |\Delta|$, and for purposes of complexity considerations, we define the *size* $|\Phi|$ of RAS Φ by $|\Phi| \equiv |\mathcal{R}| + \xi + \sum_{i=1}^{m} C_i$.

At any point in time, the system contains a certain number of (possibly zero) instances of each process type that execute one of the corresponding processing stages; this distribution of the active process instances across the various processing stages defines a notion of *"state"* for the considered RAS. Obviously, this RAS state must respect the resource capacities; i.e., no resource type $R_i \in \mathcal{R}$ can be over-allocated w.r.t. its capacity C_i at any point in time. Furthermore, in order to model the "hold-while-waiting" effect that is exhibited in the resource allocation dynamics of the considered processes, the adopted resource allocation protocol stipulates that a process instance J_j executing a non-terminal stage Ξ_{jk} and seeking to advance to a next stage $\Xi_{jk'}$, must first be allocated the resource differential $(\mathcal{A}(\Xi_{ik'}) - \mathcal{A}(\Xi_{ik}))^+$ and only then will it release the resource units $|(\mathcal{A}(\Xi_{jk'}) - \mathcal{A}(\Xi_{jk}))^-|$, that are not needed anymore.[2] Then, in the resulting operational context, the *RAS deadlock* can be formally defined as a RAS state containing a set of active process instances, DJ, such that every instance $J_j \in DJ$, in order to advance to any of its next processing stages, requests some resources currently held by some other process instance $J_k \in DJ$.

2.2. A real-time control framework for the considered RAS

As remarked in the introductory section, an effective real-time controller for the considered RAS must ensure the attainment of some set of performance objectives typically defined w.r.t. the timed RAS behavior, while keeping the RAS away from problematic behavioral patterns like the aforementioned deadlock states. This last control requirement is frequently known as the RAS *behavioral* or *logical* control problem, because the corresponding problematic behavior can be effectively avoided by controlling only the sequencing of the relevant resource allocation events and not their exact timing. Furthermore, it is generally accepted by the relevant research community that, due to the stochasticity that is generally present in the timed dynamics of the considered RAS, any robust solution to the RAS behavioral and performance control problems should rely on some feedback control scheme and not on the open-loop execution of some precomputed plan.

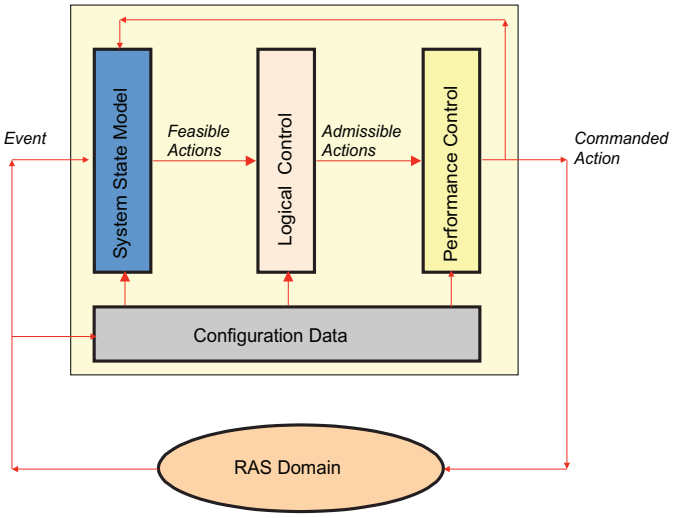

Fig. 1. An event-driven control scheme for the real-time management of the considered RAS (Reveliotis, 2005).

Such a feedback-based controller is presented in Fig. 1. The depicted control paradigm is an *event-driven* approach, where the applied control function monitors the *events* taking place in the underlying RAS and commands a certain *action sequence* in response to these events. More specifically, the proposed controller maintains a representation of the RAS *state*, which enables it to monitor the system status and to identify the entire set of *feasible actions* that can be executed by the system at any given time. Hence, this information is instrumental for enabling the controller to determine the scope of its possible responses to a certain event. However, the controller must eliminate (*"filter out"*) all those actions that can result in problematic behavior. Such problematic behavior includes the formation of deadlock, but in the more general case, this part of the depicted control scheme will address additional specifications that might be defined, for instance, by quality concerns or some policy considerations, like those arising from a notion of "fairness" to the contesting processes. All the aforementioned concerns boil down to the systematic exclusion of certain resource allocation patterns from the RAS behavior, and, as already mentioned, the resulting problem is generally known as the *logical* or *behavioral* control problem to be addressed by the controller. The set of actions eventually accepted by the logical controller defines the space of the *"admissible"* behavior for the considered RAS. Then, the second stage of the proposed control logic must shape/bias this admissible behavior in a way that aligns best with the system *performance objectives*; this biasing is effectively achieved through the selection of the particular admissible action to be commanded upon the system, at each decision stage. The corresponding problem is known as the RAS *performance-oriented* control or *scheduling*.

The effective deployment of the RAS control scheme that is described in the previous paragraph necessitates a pertinent formal characterization of the RAS state, and the reference of the RAS logical and performance-oriented control problems to appropriate formal modeling frameworks. These frameworks will enable a rigorous analysis of the corresponding RAS dynamics and the

[2] This assumption is not restrictive since resource releases that do not adhere to this protocol can be modeled by the insertion of additional processing stages in the underlying process plan.

Table 1
A RAS taxonomy (Reveliotis, 2005)

Based on the process sequential logic	Based on the requirement vectors
Linear: Each process is defined by a linear sequence of stages **Disjunctive:** A number of alternative process plans encoded by an acyclic digraph **Merge-split:** Each process is a fork-join network **Complex:** A combination of the above behaviors	**Single-unit:** Each stage requires a single unit from a single resource **Single-type:** Each stage requires an arbitrary number of units, but all from a single resource **Conjunctive:** Stages require different resources at arbitrary levels

effective synthesis of the necessary policies. The aforementioned capabilities have been conveniently provided to the developing RAS theory by the areas of *qualitative* and *quantitative analysis of Discrete Event Systems (DES)* (Cassandras & Lafortune, 2008).

Generally speaking, DES theory is a field of modern control theory investigating the behavior of dynamical systems that evolve their state discontinuously over time, in response to the occurrence of certain critical, instantaneous events. In this general setting, *qualitative DES theory* uses formal linguistic frameworks (Hopcroft & Ullman, 1979) borrowed from theoretical computer science, augmented with control-theoretic concepts and techniques, in order to analyze and control the event sequences that are generated and observed by the underlying DES dynamics (Cassandras & Lafortune, 2008; Ramadge & Wonham, 1989). On the other hand, *quantitative DES theory* analyzes and controls the timed DES dynamics, using models and tools that are borrowed from (stochastic) OR (Puterman, 1994; Ross, 1983) and simulation theory (Asmussen & Glynn, 2007; Cassandras & Lafortune, 2008).

However, while the aforementioned DES-theoretic frameworks are very powerful in their ability to provide a rigorous characterization of the RAS behavioral and scheduling problems and the necessary theoretical base for the analysis of these problems, they are not equally powerful when it comes to the synthesis of the necessary controllers for any given RAS instance. This limitation is due to the very high complexity of the general representations and the corresponding algorithms that are employed by these frameworks in the context of the considered RAS. But in the subsequent parts of this article we shall demonstrate how the relevant research community has leveraged the representational and analytical capabilities that are provided by DES theory in order to develop customized *practical* solutions to the two aforementioned control problems that arise in the RAS operational context. These solutions result from the pertinent exploitation of the special structure that exists in the considered problems, and the corresponding developments have substantially enriched and extended the capabilities of the supporting DES theory itself.

2.3. A RAS taxonomy

A tool that has been very helpful in the management of the complexity that is inherent in the considered RAS problems, and especially the RAS behavioral control problem of deadlock avoidance, is the RAS taxonomy that is depicted in Table 1. This taxonomy was initially proposed in Reveliotis (2005) and it defines a number of RAS classes on the basis of (i) the structure that is supported for the process sequential logic, and (ii) the structure of the resource allocation requests that are posed by the various processing stages.

Furthermore, more recent developments in the area of RAS deadlock avoidance have revealed the significance of some additional RAS attributes when it comes to the analytical characterization of the qualitative RAS dynamics and their control for deadlock avoidance. These new attributes include (iii) the absence of resources with *non-unit* capacities, (iv) the presence of *cycling* in the sequential logic of the RAS process types, and (v) the presence of RAS dynamics of an *uncontrollable* nature; this last feature can be further differentiated into (a) uncontrollability w.r.t. the exact timing of a certain resource allocation and (b) uncontrollability of the branching decisions of some underlying process that possesses alternate routings.

We shall use this taxonomy in the next sections, especially when we discuss the existing theory on the RAS deadlock avoidance problem.

2.4. Some further remarks

We close the discussion of this section by providing a few remarks that will enable the further positioning of the presented research program in the context of some broader control-theoretic developments and perspectives, and of some other endeavors to address the resource allocation problems that are considered in this paper.

First, it should be evident from the previous discussion and the block diagram that is depicted in Fig. 1, that the proposed control scheme relies heavily on (i) a pertinent characterization/encoding of the RAS state, and (ii) the effective decomposition of the RAS real-time control problem to logical and performance-oriented control. Both of these elements are in line with the basic DES theory (Cassandras & Lafortune, 2008). Indeed, as already remarked, the classical DES theory decomposes the overall DES control problem to logical/behavioral and performance-oriented control, and each of these two sub-problems is addressed respectively by the qualitative and the quantitative DES theory. In particular, the qualitative DES theory sets as its primary objective the enforcement of the target behavior while maximizing the behavioral latitude of the underlying DES; this is a notion of optimal control that is known as "maximal permissiveness" in the context of the relevant theory. From the standpoint of the quantitative DES theory, the aforementioned maximal permissiveness enriches the policy space that is available to the performance-oriented controller, and therefore, it maintains a high performance potential for the underlying DES. The latter must be materialized by a pertinent scheduling methodology. When viewed from this standpoint, the overall RAS control problem that is addressed in this paper, specializes the generic DES control problem to the operational context of the sequential RAS that was introduced at the beginning of this section; and each of the next two sections addresses respectively the corresponding sub-problems of logical and performance-oriented control.

Our second remark on the material that was provided in the earlier parts of this section, pertains to the particular problem of the effective resolution of the RAS deadlock. This is a problem of considerable historical depth in the corresponding literature, as it was initially investigated in the context of the resource allocation taking place in the multi-tasking operating systems of the mainframe computers that were used in the 1960's and 70's (Coffman, Elphick, & Shoshani, 1971; Habermann, 1969; Havender, 1968; Holt, 1972). In an effort to provide additional structure in its endeavors, the literature of that time had classified the approaches to that problem in three categories, characterized as (i) *"deadlock prevention"*, (ii) *"deadlock avoidance"*, and (iii) *"deadlock detection and recovery"*. Generally speaking, prevention methods comprise those methods that seek to ensure deadlock-free operation by defining the underlying RAS structure in a way that would guarantee the deadlock-freedom of the resultant operation. (ii) On the other hand, the term "deadlock avoidance" refers to those methods that try to predict any possible deadlock formation during

the real-time operation of the system, and control the resource allocation function in a way that steers the system away from these potential deadlock states. (iii) Finally, as the name suggests, deadlock detection and recovery methods, allow the system to enter any deadlock state, but they provide (a) a detection mechanism that recognizes such a development, and (b) a recovery routine that brings the system back to a deadlock-free state by forcefully redefining the processing status of some of the processes that are involved in the deadlock. However, in the context of the DES-theoretic approaches that are currently used to address the RAS deadlock, the developed controller frequently takes the form of an additional structure that is super-imposed on the original formal representation of the underlying RAS.[3] Hence, the distinction between the deadlock prevention and the deadlock avoidance methods is not very pertinent (and useful) anymore, and this is the position that is adopted in the rest of this paper. More specifically, in the sequel we shall adopt the term "deadlock avoidance" in order to refer to any method that seeks to prevent the deadlock formation during the real-time operation of the underlying RAS.[4]

On the other hand, detection and recovery is a method for the RAS deadlock problem that is fundamentally different from the methods that are pursued in this document. Clearly, such an approach is much more reactive to the deadlock problem than the previous two, and leaves the underlying RAS much more exposed to the disruptive effects of a deadlock. Hence, the deadlock detection and recovery method is typically applicable in cases where either (a) there is not adequate information about the future evolution of the resource requirements of the various RAS processes to exercise effective deadlock prevention and/or avoidance, or (b) deadlock is a rather rare event and the corresponding recovery process is not very costly in terms of resources and time. Such conditions are encountered in some computational platforms involving multithreaded programming, and it is these environments that have been the primary application context for deadlock detection and recovery methods (Kelly, Wang, Lafortune, & Mahlke, 2009). Also, the work of Reveliotis (2000) provides a prototypical analytical investigation of the selection between the deadlock avoidance and the deadlock detection and recovery methods based on the impact of these two methods on the time-based performance of the underlying RAS; the modeling framework used in that study is similar to some formal modeling frameworks that are discussed in Section 4. But overall, it is true that, for the reasons explained above, the deadlock detection and recovery method currently is outside the scope of the RAS control framework that was defined in the earlier parts of this section.

3. The RAS logical control problem

3.1. Formal modeling of the RAS behavior

A straightforward way to represent formally the behavior of a given RAS Φ, and the corresponding logical control problem of deadlock avoidance and liveness-enforcing supervision, is by means of a finite state automaton (FSA), $G(\Phi)$ (Cassandras & Lafortune, 2008). The state s of this FSA is a vector of dimensionality ξ – i.e., the number of the distinct processing stages of the underlying RAS Φ – where each component indicates the number of active process instances that execute the corresponding processing stage. The state space S consists of all those vectors that define a feasible resource allocation w.r.t. the RAS resource allocation function $\mathcal{A}$ and the resource capacities C_i. The events that evolve the RAS state comprise (i) the "loading" events that initiate new process instances, setting them to their first processing stage, (ii) the "advancing" events, that advance an active process instance from its current processing stage to a successor processing stage, and (iii) the "unloading" events that terminate those process instances that have completed their last processing stage and remove them from the system. The initial state of this automaton corresponds to the state **0** where the system is idle and empty of any jobs. State **0** is also the only marked state of G, a fact that expresses the desire for complete process runs.

In the context of the FSA $G(\Phi)$, the feasible behavior of the underlying RAS Φ is defined by the entire set of states, S_r, that are reachable from state **0**. On the other hand, the admissible behavior is characterized by the set of states S_s that are co-reachable to state **0**; in the relevant terminology, these states are also characterized as "safe", while their complement w.r.t. the state set S is the set of the "unsafe" states S_u. The state set S_u contains the set of the "deadlock" states, S_d, i.e., states that exhibit a (partial) deadlock, according to the definition of this concept in the previous section. But S_u can also contain states that do not exhibit such a partial deadlock but lead unavoidably to a deadlock state; this last set of states are characterized as "deadlock-free unsafe" states.

3.2. Optimal RAS deadlock avoidance and its complexity

Given the above classification of the RAS state space S, the sought supervisor should restrict the RAS behavior in the subspace $S_r \cap S_s \equiv S_{rs}$, i.e., the set of reachable and safe states. This restriction ensures the ability of every activated process instance to terminate, and at the same time, it is the *minimal* required restriction of the uncontrolled RAS behavior for establishing deadlock-free operation. Hence, in view of the first of the two remarks that are provided in Section 2.4, the corresponding supervisor is characterized as the *"maximally permissive deadlock avoidance policy (DAP)"*. Also, the computation and deployment of the maximally permissive DAP for any given RAS defines an "optimal control" problem in the context of the logical control of the considered RAS.

A natural implementation of the maximally permissive DAP would be through an one-step-lookahead scheme that would permit a transition in automaton G on the basis of the "safety" (i.e., the co-reachability) of the resulting state. But it has been shown that assessing the safety of any given RAS state is an NP-complete problem even for the simplest class of the Linear-Single-Unit-RAS (Reveliotis & Roszkowska, 2010). Hence, the deployment of the maximally permissive DAP is an NP-Hard proposition. In the rest of this section we discuss how the relevant research community has coped with this negative result, developing a rich methodology that has enabled the implementation of the optimal or near-optimal DAPs for RAS instances of very large size and behavioral complexity.

3.3. Suboptimal Polynomial-Kernel DAPs

As has been the case with many other optimization problems that exhibit super-polynomial complexity, the first reaction of the research community on the deadlock avoidance problem was to identify surrogate conditions to safety that are testable with a polynomial complexity w.r.t. the underlying RAS size, $|\Phi|$, and lead to a significant coverage of the target state space S_{rs}. Such policies are known as "Polynomial Kernel" (PK-) DAPs in the relevant literature (Reveliotis, 2005).

[3] c.f. Chapter 3 in Cassandras and Lafortune (2008) on the compositional methods that are employed by the DES supervisory control theory, and also the discussion in the next section, especially the part on addressing the problem of the RAS deadlock in the Petri net modeling framework.

[4] Besides the term "deadlock avoidance", another term that is also used extensively for characterizing the prevention of deadlock formation during the real-time operation of any given RAS, is that of "liveness-enforcing supervision"; this term is motivated by the Petri net-based modeling of the corresponding problem (c.f. the corresponding part in Section 3). Both terms will be used interchangeably in the rest of this document.

A characteristic example of such a policy is Dijkstra's Banker's algorithm (Dijkstra, 1965), that restricts the RAS behavior to a subspace of S_{rs} for which the existence of paths to the target state **0** is polynomially verifiable. A RAS state s that avails of a polynomially detectable path to the empty state **0** is characterized as an "ordered" state in the relevant literature (Lawley, Reveliotis, & Ferreira, 1998), and the polynomial detectability of the corresponding paths is due to their decomposition to event subsequences that advance and terminate each of the active process instances in s one at a time. More specifically, by removing one of the active process instances from the running state, the execution of each of these subsequences leads to an increase of the pool of free resources. Hence, in the search procedure that looks for such feasible sequences starting from the considered state s, there is no need for backtracking, and therefore, the corresponding computational cost is polynomial w.r.t. the underlying RAS size.

Some other classes of PK-DAPs for the considered RAS take the form of a polynomial set of linear inequalities on the RAS state s. These PK-DAPs are characterized as "algebraic", and some specific examples of such policies are the Resource Upstream Neighborhood (RUN) Policy (Lawley, Reveliotis, & Ferreira, 1997a, 1997b) and the Resource Ordering (RO) policy (Lawley, Reveliotis, & Ferreira, 1998); these policies were originally developed for Linear-Single-Unit RAS, but the RUN policy has also been extended to the case of Disjunctive-Conjunctive RAS, as well (Park & Reveliotis, 2001). An important challenge in the design of a PK-DAP is to ensure that it is induced-deadlock-free; i.e., for every RAS state s that is admitted by the considered policy there is a complete path of policy-admissible states that leads back to the empty state **0**. PK-DAPs that are induced-deadlock-free are characterized as "correct" in the relevant literature. As we will discuss in more detail in the sequel, one of the major achievements of the current RAS supervisory control theory is the development of a methodology that can assess automatically the ability of any given set of inequalities to define a correct DAP for a given RAS Φ (Reveliotis, 2005).

Finally, it should also be pointed out that the disjunction of a set of PK-DAPs for a given RAS Φ – i.e., the policy that admits a state s of Φ if it is admitted by any of the constituent policies – is another correct DAP with admissible state space equal to the union of the state spaces that are admitted by the constituent policies. Hence, such a DAP disjunction can be perceived as an attempt to reconstruct (an approximation of) the notion of RAS state safety by "patching" together a set of surrogate concepts, each contributing a particular facet of the target concept.

3.4. Special RAS structure admitting optimal deadlock avoidance of polynomial complexity

A second typical reaction to an NP-Hardness result, is the effort to identify "special structures" of practical interest – i.e., certain subclasses of the considered problem – that admit solutions of polynomial complexity. In the case of the deployment of the maximally permissive DAP for the considered RAS classes, such special structure has been identified effectively by two lines of analysis.

The first line has sought to identify RAS structure that renders the search for a path to the marked state **0** a task of polynomial complexity w.r.t. the underlying RAS size $|\Phi|$ (Araki, Sugiyama, & Kasami, 1977; Gold, 1978). Generally speaking, this RAS structure guarantees that each safe state s will possess a process-completion sequence consisting of easily identifiable process advancement subsequences that lead to a monotonic increase of the pool of free resources; therefore, the search for these process-completion sequences can be effected by a "greedy" algorithm (Glasserman & Yao, 1994) (i.e., a search algorithm without the need for backtracking, as in the case of the ordered RAS states discussed in the previous paragraphs).

The second line of analysis that has enabled the identification of RAS special structure admitting polynomially deployable maximally permissive DAP, is based on the realization that, while the assessment of RAS state safety is NP-complete, the detection of a deadlock in a RAS state is a task of polynomial complexity in many RAS classes of the taxonomy of Table 1, including the broad class of Disjunctive-Conjunctive RAS (Reveliotis, 2005). This realization subsequently implies that in any RAS classes with no deadlock-free unsafe states (i.e., RAS where $S_u = S_d$), unsafety is polynomially recognizable by substituting, in the aforementioned one-step-lookahead scheme that implements the maximally permissive DAP, the original tests for (un-)safety with the polynomial test for deadlock. The current literature characterizes a number of RAS subclasses with no deadlock-free unsafe states (Fanti, Maione, Mascolo, & Turchiano, 1997; Fanti, Maione, & Turchiano, 1998; Lawley & Reveliotis, 2001; Reveliotis, Lawley, & Ferreira, 1997). Furthermore, these subclasses are of substantial practical interest, as they have enabled, for instance, maximally permissive buffer space allocation in flexibly automated cells and semiconductor manufacturing re-entrant lines (Lawley & Reveliotis, 2001), and the synthesis of a distributed control scheme for the safe and live traffic management of free-ranging mobile agents (Roszkowska & Reveliotis, 2013). On the other hand, it is also true that currently all of the aforementioned RAS classes are sub-classes of the Disjunctive-Single-Unit-RAS (i.e., they do not allow for conjunctive resource allocation at their processing stages) (Reveliotis, 2005).

3.5. The optimal DAP as a classifier of the RAS state space: the parametric approach

Substantial advances in the deployment of the maximally permissive DAP for the considered RAS have been obtained more recently through the realization that the NP-Hardness of this policy is essentially a "worst-case" result, while practical, efficient implementations of the policy can still be obtained by (i) isolating the hardest part of its computation to an off-line phase, and (ii) employing a pertinent, parsimonious representation of the final result of that first computational step, that will enable the on-line / real-time instantiation of the policy with a manageable computational cost.

In a basic implementation of the above idea, the off-line phase can be supported by the enumerative, set-theoretic techniques that are employed by the DES Supervisory Control (SC) theory (Cassandras & Lafortune, 2008; Ramadge & Wonham, 1989). On the other hand, the development of a parsimonious representation of the obtained maximally permissive supervisor is based on the realization that such a supervisor essentially dissects the vector set S (or S_r) into two subsets containing, respectively, the admissible and the inadmissible states. Hence, at the end, the maximally permissive DAP acts as a "classifier" of the elements of this set. But then, the sought parsimonious representations of the maximally permissive DAP can be provided by more general results of classification theory (Nilsson, 1990) adapted to the considered problem context (Reveliotis & Nazeem, 2014).

More specifically, it can be shown that the finite vector set containing the RAS states can be dichotomized to its safe and unsafe subsets by a two-layered classifier where the first layer consists of a number of systems of linear inequalities, and the second layer consists of a Boolean function that tests the satisfaction of at least one of the linear-inequality systems at the lower level; these classifiers have been characterized as "disjunctive" in the relevant literature (Nazeem & Reveliotis, 2012). In fact, in the considered problem context of the maximally permissive DAP, the aforementioned inequalities present additional structure that results from the relative topology of the underlying safe and unsafe subspaces and enables the effective design of the sought classifier

while considering only the maximal safe and the minimal unsafe states under the partial ordering of these vectors that is based on a componentwise comparison. Furthermore, there are additional interesting and practical cases where the sought classifier can be expressed as a single set of linear inequalities; such a classifier has been characterized as "linear" in the relevant literature (Chen & Li, 2011; Nazeem, Reveliotis, Wang, & Lafortune, 2011).

Finally, this line of research has also revealed a strong affinity between the considered classification problem and the optimal set covering problem. In particular, the inequalities of the disjunctive and the linear classifiers that separate some subset of unsafe states from the set of safe states can be considered as a "cover" for the corresponding set of unsafe states. This realization subsequently has enabled the development of customized algorithms for the construction of a structurally minimal classifier from the considered classes of disjunctive and linear classifiers (Cordone, Nazeem, Piroddi, & Reveliotis, 2013; Reveliotis & Nazeem, 2013), and the adaptation of efficient heuristics for the set covering problem that can be applied in the case that the aforementioned algorithms turn out to be computationally too costly (Nazeem et al., 2011).

3.6. The optimal DAP as a classifier of the RAS state space: the non-parametric approach

A complementary line of research to the research line that was discussed in the previous paragraph has sought to express the target DAP in a "non-parametric" manner, by simply identifying and storing the set of unsafe states that are on the "boundary" between the safe and the unsafe subspaces; i.e., those unsafe states that can be reached from the safe subspace in one transition. In this case, the aforementioned monotonicity of the RAS state safety, and the relative topology of the safe and unsafe subspaces that it implies, render the target set of the boundary unsafe states a "right-closed" set. Therefore, this set can be represented compactly by storing only its minimal elements (Nazeem & Reveliotis, 2011).

The relevant literature also avails of very efficient algorithms that can detect the minimal boundary unsafe states through only a partial exploration of the underlying state space. This exploration starts from a pertinent reconstruction of the minimal deadlocks and backtraces "intelligently" from them in search for the other minimal unsafe but deadlock-free RAS states (Nazeem & Reveliotis, 2014). Some of these exploration algorithms are also employing the power of the symbolic computation that has been developed in the recent years for the efficient representation and processing of Boolean functions with a very large number of variables, as well as the representation and processing of more general very large finite sets stored under a binary representation (Fei, Reveliotis, Miremadi, & Akesson, 2015). As a result, currently we are in a position to develop very compact representations of the maximally permissive DAP for RAS with very large sizes $|\Phi|$, and with billions of states in the corresponding state spaces.

Finally, by focusing on the efficient identification and storage of only the minimal boundary unsafe states, the above results have also enabled the effective implementation of the maximally permissive DAP even for RAS with infinite state spaces. In particular, the recent work of Nazeem and Reveliotis (2015) has extended the techniques that were described in the previous paragraphs in order to compute the maximally permissive DAP for the RAS class that corresponds to multi-threaded software with reader/writer (R/W-) locks; since these resources can be accessed in their reading mode by an arbitrary number of processes, the underlying state space can grow infinitely large. The effective deployment of the maximally permissive supervisor for a DES structure exhibiting a behavior that evolves over an infinite state space is an important result that transcends the synthesis capabilities of the standard supervisory control theory (Cassandras & Lafortune, 2008; Ramadge & Wonham, 1989).

3.7. Petri net-based approaches to the study of RAS deadlock and deadlock avoidance

We close our discussion of the RAS logical control problem by overviewing another line of research that has been very active and very prominent in the relevant literature, and with substantial theoretical and practical contributions in our efforts to cope with this problem. This research line is employing the formal modeling framework of Petri nets (PNs) (Cassandras & Lafortune, 2008; Murata, 1989) for representing the qualitative RAS dynamics. Compared to the FSA modeling framework, that enumerates explicitly the underlying state space S, PNs enjoy a much more compact representation of the RAS dynamics, and even more importantly, they express explicitly the linkage of those dynamics to the underlying system structure. Hence, this modeling framework is particularly amenable for pursuing a "structural analysis" of the considered problem, i.e., a line of analysis that links particular behavioral properties of the system to structural formations and properties.

In the context of the RAS deadlock avoidance, a highly celebrated structural result concerns the association of the RAS partial deadlock, and the lack of liveness and reversibility for the corresponding PN model, to the structural objects of "empty", or, more generally, "deadly marked siphons" (Ezpeleta, Colom, & Martinez, 1995; Jeng, Xie, & Peng, 2002; Park & Reveliotis, 2001; Reveliotis, 2003; 2005; Tricas, Garcia-Valles, Colom, & Ezpeleta, 2005). More specifically, these results attribute the non-liveness of the RAS-modeling PN to the formation of such badly marked siphons in the net markings or, in certain cases, in the projection of these markings to appropriately selected subspaces. A powerful feature of this theory is that it has managed to express the aforementioned condition for (non-)liveness in the form of some mixed Integer programming (MIP) formulations that are obtained from, and are polynomially related in terms of their variables and constraints to, the structure of the underlying RAS (Chu & Xie, 1997; Park & Reveliotis, 2000; 2001; Reveliotis, 2005). The derived tests will either conclude the absence of deadlock in the dynamics of the assessed RAS, or they will return a partial deadlock in the form of the corresponding deadly marked siphon.

Furthermore, in the case of DAPs that can be expressed as additional places superimposed on the structure of the RAS-modeling PN, which are known as "monitor" or "control" places (Giua, DiCesare, & Silva, 1992; Moody & Antsaklis, 1998), the aforementioned tests can also assess the ability of these policies to establish correct deadlock avoidance for the considered RAS, and therefore, they can support a "DAP synthesis" process. Of particular interest are such synthesis processes where the maximally permissive DAP is obtained iteratively through the detection and control of partial deadlocks existing in the original RAS behavior, or maybe brought about by the policy logic that is incrementally defined through these iterations; this second type of partial deadlocks are the (policy-)induced deadlocks that were discussed in earlier parts of this article. The literature avails of specific results where such an incremental synthesis will converge to the maximally permissive DAP (Liao, Lafortune, Reveliotis, Y., & Mahlke, 2013; Liao et al., 2013). Obviously, a necessary condition for such a convergence is the ability to represent the maximally permissive DAP by a set of "monitor" places, a requirement that further translates to the possibility of representing the dichotomy of the state set S that is effected by the maximally permissive DAP as a linear classifier (Giua et al., 1992; Moody & Antsaklis, 1998). For the corresponding RAS classes, the target policy can be obtained while avoiding

completely an explicit (even partial) enumeration of the underlying RAS state space.[5]

Finally, we also mention, for completeness, that there have been additional attempts that seek to design the maximally permissive DAP for a given RAS through the standard DES SC theory, and subsequently, rehash the obtained policy in a PN representation, in order to exploit the compactness of this modeling framework (Ghaffari, Rezg, & Xie, 2003); these approaches constitute an adaptation of the, so called, "theory of regions" (Badouel & Darondeau, 1998) to the considered problem, but they tend to suffer from a high representational complexity for the final result, and also they lack completeness (since, as discussed above, the maximally permissive DAP must admit a linear representation in order to be effectively represented as a set of "monitor" places).

4. The RAS performance control problem

4.1. A basic characterization of the RAS scheduling problem

As already mentioned in the previous sections, the RAS performance control problem is essentially a scheduling problem that must be formulated and solved on the admissible subspace that is defined by the RAS logical control policy. From a theoretical standpoint, these scheduling problems belong to the broader class of problems that seek to schedule stochastic networks with blocking (Perros, 1994), a particular area in the scheduling theory that has not been extensively researched up to this point.

To a large extent, the current lack of results for this class of problems is due to the fact that blocking introduces strong "coupling" effects among the underlying workstations, or more generally the involved resources, that render the characterization of the optimal scheduling policy a very challenging undertaking. Furthermore, the permanent blocking – i.e., the deadlocking – effects that can arise in these environments, in addition to the transient blocking, have been another important source of complexity and a very practical roadblock to the endeavors of the more traditional stochastics community w.r.t. this set of scheduling problems.

On the other hand, the current availability of the results on the RAS logical control problem that were described in the previous section provides a systematic methodology to deal with these blocking and deadlocking effects, and opens the corresponding set of scheduling problems to more active and systematic research. In the rest of this section, we shall briefly outline some ongoing endeavors of ours w.r.t. this set of problems. But as already mentioned above, it can be safely argued that the research on the RAS performance control has not reached the maturity and the richness of the results that characterize its logical control counterpart.

4.2. Formulating the RAS scheduling problem through MDP theory

A typical framework for tackling the scheduling problems that are discussed in the previous paragraph is that of the Markov Decision Processes (MDPs) (Puterman, 1994). Under an approximation of the "processing time" distributions of the various processing stages by pertinently selected phase-type distributions (Cassandras & Lafortune, 2008), and assuming some additional stationarity in the behavior of the involved process types, the MDP modeling framework enables the structured modeling and analysis of all the stochasticity that might characterize the operations of the considered RAS.

Furthermore, the co-reachability of the admissible states w.r.t. the initial state **0** that is established by the applied logical control policy, implies that the obtained MDP formulations belong to the class of MDPs that, in principle, are solvable by effective and fairly efficient algorithms (Bertsekas, 2005; Puterman, 1994). The corresponding characterizations, and a further adaptation of these general MDP results to the considered RAS scheduling problem, can be found in Choi and Reveliotis (2003); Reveliotis (2005) and Choi and Reveliotis (2005).

Things are complicated, however, by the explosive size of the underlying state space. In fact, this state space explosion implies that the sought scheduling policies are not only hard to compute, but they also have a very high (frequently prohibitive) representational complexity.[6]

4.3. Controlling the complexity of the RAS scheduling problem

Motivated by the above remarks, in a recently initiated research program we have sought to confine the aforementioned scheduling problems in policy spaces that admit a more parsimonious representation. The current results of this research program are presented in Li and Reveliotis (2015) and Li and Reveliotis (0000), and an outline of the pursued approach is provided in Fig. 2.

Instrumental to this approach has been the expression of the underlying dynamics in the modeling framework of the Generalized Stochastic Petri Nets (GSPNs) (Ajmone Marsan, Balbo, & Conte, 1986). The basic PN structure of this modeling framework enables the effective representation of the qualitative RAS dynamics and the corresponding logical control (e.g., deadlock avoidance) policy. At the same time, the partitioning of the net transitions to "timed" and "untimed", according to the standard GSPN semantics, enables the explicit expression of the corresponding scheduling problem by means of the externally provided distributions that will regulate the conflicts of the enabled untimed transitions at the various net markings. These distributions are known as "random switches" in the GSPN terminology and, in the considered problem context, essentially define randomized stationary policies for the corresponding MDP formulation.

An optimal set of random switches can be computed, at least in principle, through a non-linear programming (NLP) formulation that has as its primary variables the elements (i.e., the probabilities) of these random switches, and as secondary variables the

[5] Essentially, the considered methods substitute the explicit search of the RAS state space for partial deadlocks with an *implicit* search that is effected by the aforementioned MIP formulations.

[6] More generally, the MDP modeling framework, and its companion computational framework of dynamic programming (DP), are the primary formal frameworks for addressing sequential decision-making problems like the RAS scheduling problems that are considered in this paper. And the state space explosion that is observed in the considered application context is a typical problem that is frequently encountered in the application of these frameworks, and has come to be known as the "curse of dimensionality" (Bellman, 1957). Also, in principle, these frameworks can further address non-stationary versions of the considered scheduling problems, especially when embedded in 'receding" or "rolling-horizon" frameworks that decompose the overall scheduling problem into a sequence of sub-problems. These subproblems are defined on overlapping finite time-horizons with a length that is determined by (i) the time scale of the various operations in the underlying RAS, (ii) the availability of pertinent information about the more distant periods in the considered planning horizon(s), and (iii) the computational scalability of the resulting formulations w.r.t. this length. Every time that a sub-problem is solved, typically only a certain part of the derived schedule is executed, and then the problem is reformulated and resolved with a revised set of data that reflect the emergent situation since the solution of the last sub-problem. This mechanism provides the necessary feedback for closing the control-loop. Also, frequently the principle of certainty equivalence (Bertsekas, 1995) is adopted in the formulation of the aforementioned subproblems, which are eventually solved as deterministic combinatorial optimization problems (Cook, Cunningham, Pulleyblank, & Schrijver, 1998), through generic or more customized approaches. Overall, the rolling-horizon methods that are described above tend to be more *ad hoc* and less structured than the stationary formulations that are discussed in the main part of this section, and their technical complexity and interest seems to depend on the particular non-stationarities that are present in the corresponding RAS. Also, the current application of these methods in the sequential RAS classes that are considered in this paper, is very limited. Hence, we confine the discussion of these methods at the very end of this section, where we provide a few pointers to some relevant results.

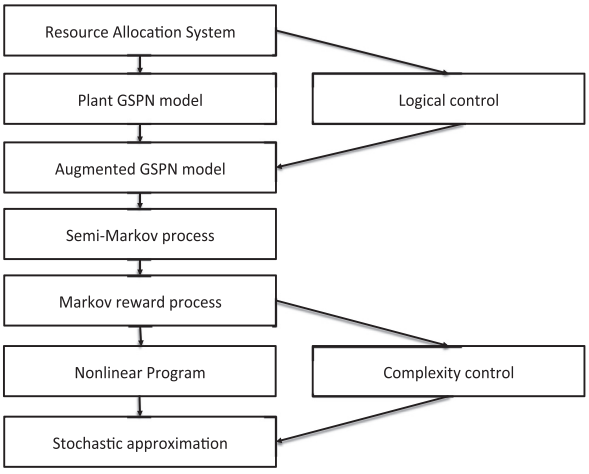

Fig. 2. A schematic outline of the RAS scheduling methodology that is presented in this article, based on the developments of Li and Reveliotis (2015) and Li and Reveliotis (0000).

stationary distributions of the corresponding policies. Furthermore, this NLP formulation can be solved by stochastic approximation algorithms (Kushner & Yin, 2003) and results drawn from the theory of regenerative Markov reward processes (Cao, 2007; Marbach & Tsitsiklis, 2001); these results enable the estimation of the performance objective of the formulation and its gradients through simulation, and avoid the enumeration of the underlying state spaces.

On the other hand, the association of a distinct random switch to every GSPN marking with more than one enabled untimed transition leads to a very high representational cost for the defined policies that is similar to that experienced by the aforementioned MDP formulations. But the explicit representation of the system structure and dynamics by the underlying GSPN also enables the redefinition of the employed random switches in a way that reduces the representational complexity of the resulting scheduling policies, and still renders these policies quite pertinent and practical in the context of the considered operations.

A particular such redefinition that is currently explored by the considered research program seeks the replacement of the original set of random switches by a "static" version of this set which assigns the same distribution to every GSPN marking that activates the same set of untimed transitions. This restriction reduces substantially the decision variables that appear in the final NLP formulation. At the same time, it can be easily seen that static random switches can express the entire class of "static priority" policies that are frequently used in many practical settings, and therefore, this set of policies is very relevant to the realities of the current industrial practice (c.f., for instance, Kumar, 1994).

Furthermore, both classes of scheduling policies, based either on dynamic or on static random switches, can be further refined, and their member policies can be rendered more parsimonious, by a pertinent assessment of the actual conflicts that might exist among the enabled untimed transitions at any given marking, and the elimination of any unnecessary effort to coordinate non-conflicting transitions. The corresponding analysis can be performed on the subgraphs of the underlying state transition diagram (STD) that characterize the net transition between two tangible markings, i.e., markings that enable only timed transitions and therefore not involving any further scheduling decisions. These subgraphs are rather "local" structures in the underlying STD, and therefore, the aforementioned analysis for (non-)conflict can be effectively integrated in the simulation logic that is used for the solution of the corresponding NLP formulation. Practical experience has shown that such a refinement usually results in very extensive reductions of the decision variables employed by the final NLP formulation (Li & Reveliotis, 0000).

We should also notice that the aforementioned screening process develops at the interface of the qualitative / behavioral and the quantitative / performance-oriented dynamics of the underlying RAS, and therefore, it relies substantially upon the ability of the GSPN modeling framework to provide an integrated representation of these two types of dynamics of the considered RAS.

An additional important remark is that static random switches essentially define an aggregation scheme on the underlying RAS state space (Bertsekas, 2005). Hence, it is plausible to attempt the enhancement of the performance of a scheduling policy that is optimal within the class of policies expressed by the static random switches, through a "refinement" process that seeks the partial disaggregation of the underlying state aggregates. Such a disaggregation scheme can be driven by information that is conveyed in the structure of the current policy, and it constitutes an effective mechanism for managing the underlying trade-off between the operational efficiency of the obtained policies and their representational and operational complexity.

Finally, another important feature of the scheduling methodology that is outlined in the above paragraphs, is the ability of this methodology to integrate additional operational requirements by expressing them as "behavioral" requirements for the underlying RAS. More specifically, these requirements can be imposed on the considered RAS by the applied logical control policy, and effectively encoded by the GSPN structure that models the admissible RAS behavior. On the other hand, quite conveniently, such an approach leaves unaltered the computational procedures that will compute the final scheduling policies. As a more concrete example of this capability we mention the potential enforcement of throughput-ratio (or other similar) constraints regulating the relevant throughputs of the various process types in any given RAS; these constraints essentially constitute "fairness" constraints (Hu, Zhou, & Li, 2012; Murata, 1989) for the corresponding processes and they can be encoded by superimposing additional structure on the RAS modeling GSPN.

4.4. Some additional results pertaining to the RAS scheduling problem

Concluding this discussion on the performance control of the considered RAS, we also want to notice a set of further developments that pertain to this problem, although not necessarily presented by means of the corresponding terminology.

Such an example is provided by the ongoing endeavors to develop deadlock-free, cyclic / repetitive schedules for the cluster tools that are used in the contemporary semiconductor fabs. These cluster tools can be modeled as Linear-Single-Unit RAS with deterministic processing times, and the aforementioned endeavors to the scheduling of these environments essentially constitute a specialization of the existing theory of max-plus algebra, that has been developed for the PN class of timed (weighted) marked graphs, to the particular structures that are encountered in the cluster tool configurations (Qiao & Zhou, 2014).

Additional examples of Petri net-based modeling of complex resource allocation for scheduling purposes can be found in the application domain of health-care delivery systems (Augusto & Xie, 2014). Finally, there have been some sporadic endeavors to address some RAS scheduling problems with deterministic processing times through integer programming and the generic search-based methods that are used in combinatorial optimization; some examples along these lines can be found in Ramaswamy and Joshi (1996) and Luo, Xing, Zhou, Li, and Wang (2015).

5. Open challenges

The previous sections have outlined the extensive progress that has been made by the presented research program towards the development of a systematic body of results able to support the efficient and expedient resource allocation that is necessary in many of the contemporary technological applications, including those that were highlighted in the introductory part of this paper. Yet, there are important remaining challenges that must be addressed for the culmination of these results to a complete theory for the corresponding RAS, and the effective migration of this theory to the relevant industrial practice. We conclude this article by highlighting some of these challenges and the opportunities that they define for the corresponding research communities.

5.1. Strengthening the RAS performance analysis and control

It is evident from the previous discussion that a major remaining challenge for the emergent RAS theory highlighted in this paper, is the strengthening and the extension of the corresponding scheduling theory. This strengthening can be pursued at the modeling and the algorithmic level, by seeking the definition, the effective computation and the deployment of richer classes of parsimonious scheduling policies than the set of classes that was outlined in Section 4.

But even more importantly, this set of scheduling problems requires a more profound understanding of the corresponding notions of "feasibility" and "optimality", from a qualitative / analytical standpoint. Characteristically, it is important to obtain a clearer understanding of the structure of the "value functions" for the MDPs that model the considered scheduling problems, and the particular factors that are most influential in the determination of the "value" of any given RAS state. The availability of such a perspective can subsequently guide the endeavors towards the development of pertinent and computationally efficient approximations of the optimal scheduling policies in the context of the burgeoning theory of the Approximate Dynamic Programming (ADP) (Bertsekas, 2005; Konda & Tsitsiklis, 2003; Powell, 2007).

5.2. Coping with non-stationarity

An additional line of research that can complement our current understanding and capability w.r.t. the real-time RAS control problem that is addressed in this paper is the accommodation of non-stationary behavior, maybe along the lines that are outlined in Footnote [6]. In spite of the high complexity and even the apparent ill-posedness of the corresponding RAS control problem that were discussed in Footnote [6], this problem is of significant practical interest, since many of the targeted applications can exhibit considerable non-stationarity in their underlying dynamics.

5.3. Extending the RAS logical control theory

On the RAS logical control side, we can consider the extension of the existing results to RAS with even more complex structure and behavior than those addressed by most of the current developments. The taxonomy of Section 2 offers a systematic base for organizing this extension.

Perhaps, an even more important extension of the current RAS SC theory is in a direction that will enable it to address more challenging operational environments, like those that offer only partial observability of the underlying resource allocation function, or necessitate the distribution of the control function to a number of coordinating controllers due to scaling, communication or other operational constraints. These developments can be referred to results borrowed from the existing DES theory (Cassandras & Lafortune, 2008; Ramadge & Wonham, 1989; Seatzu, Silva, & van Schuppen, 2013), but it is also expected that the special and rich structure of the RAS concept will enable customized analyses and solutions for this set of problems, as well.

Finally, in a vein similar to the aforementioned extensions, one can consider the problem of reactive or proactive accommodation of resource capacity losses, in a way that minimizes the experienced disruption and/or ensures a certain minimal functionality for the degraded system; some limited work along these lines can be found in Reveliotis (1999) and Lawley and Sulistyono (2002).

5.4. Coping with new behavioral requirements

Another possible extension of the existing RAS theory is its enrichment in order to encompass additional operational constraints beyond the fundamental problem of deadlock avoidance. Along these lines, we already saw the potential need to observe certain "fairness" requirements across the RAS process types. This notion of "fairness" can be extended to more general "coordination" requirements among the running processes. In all these cases, the new operational requirements must be formally expressed in the employed DES-modeling frameworks of FSA and PNs, and the resulting formal models must be further analyzed for their properties w.r.t. blocking; the existing deadlock avoidance theory must be extended to this new class of systems, as well. A first set of results along these lines can be found in Hu et al. (2012).

5.5. Transferring the emerging theory to the industrial practice

Finally, as the presented RAS theory grows and strengthens its methodology along the lines indicated in the previous paragraphs, additional endeavor must be expended towards the development of the human capital and of the technological and computational base that will enable the constructive migration of this theory to the future engineering practice. This endeavor certainly involves the eventual undertaking of some "pilot" large-scale applications that will highlight the technical strength of the theory and the competitive advantage that can be supported by it.

But even more importantly, the aforementioned endeavors must also seek the effective integration of the existing and the emerging results into the relevant engineering curricula, and the organization of these results in a series of computational platforms that will enable their robust and expedient utilization by the field engineers. In fact, this last activity can be part of a broader initiative concerning the further promotion of DES theory and of the emerging formal methods in the engineering curriculum and practice. It is expected that, collectively, all these endeavors will define a spectrum of fundamental developments and trends with profound and transformative repercussions for the related fields of control and automation engineering.

References

Ajmone Marsan, M., Balbo, G., & Conte, G. (1986). *Performance models of multiprocessor systems*. Cambridge, MA: The MIT Press.

Araki, T., Sugiyama, Y., & Kasami, T. (1977). Complexity of the deadlock avoidance problem. In *Proceedings of the 2nd IBM symposium on mathematical foundations of computer science* (pp. 229–257). IBM.

Asmussen, S., & Glynn, P. W. (2007). *Stochastic simulation: Algorithms and analysis*. NY: Springer.

Augusto, V., & Xie, X. (2014). Model synthesis, planning, scheduling and simulation of health-care delivery systems using Petri nets. In J. Campos, C. Seatzu, & X. Xie (Eds.), *Formal methods in manufacturing* (pp. 571–598). CRC Press / Taylor & Francis.

Badouel, E., & Darondeau, P. (1998). Theory of regions. In W. Reisig, & G. Rozenberg (Eds.), *Advances in Petri nets: Basic models*. In *1491 of LNCS* (pp. 529–586). Springer-Verlag.

Bellman, R. (1957). *Applied dynamic programming*. Princeton, NJ: Princeton University.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*: Vol. 1. Belmont, MA: Athena Scientific.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control (3rd ed.)*. Belmont, MA: Athena Scientific.

Cao, X. (2007). *Stochastic learning and optimization*. NY: Springer.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems (2nd ed.)*. NY: Springer.

Chen, Y. F., & Li, Z. W. (2011). Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems. *Automatica, 47*, 1028–1034.

Choi, J. Y., & Reveliotis, S. A. (2003). A generalized stochastic Petri net model for performance analysis and control of capacitated re-entrant lines. *IEEE Transaction on Robotics and Automation, 19*, 474–480.

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

Chu, F., & Xie, X.-L. (1997). Deadlock analysis of Petri nets using siphons and mathematical programming. *IEEE Transaction on R&A, 13*, 793–804.

Coffman, E. G., Elphick, M. J., & Shoshani, A. (1971). System deadlocks. *Computing Surveys, 3*, 67–78.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., & Schrijver, A. (1998). *Combinatorial optimization*. NY: Wiley-Interscience.

Cordone, R., Nazeem, A., Piroddi, L., & Reveliotis, S. (2013). Designing optimal deadlock avoidance policies for sequential resource allocation systems through classification theory: existence results and customized algorithms. *IEEE Transaction Automatic Control, 58*, 2772–2787.

Curry, G. L., & Feldman, R. M. (2011). *Manufacturing systems modeling and analysis, 2nd ed.*. Berlin, Germany: Springer-Verlag.

Dijkstra, E. W. (1965). Cooperating sequential processes. *Technical Report*. Eindhoven, Netherlands: Technological University.

Ezpeleta, J., Colom, J. M., & Martinez, J. (1995). A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transaction on R&A, 11*, 173–184.

Fanti, M. P., Maione, B., Mascolo, S., & Turchiano, B. (1997). Event-based feedback control for deadlock avoidance in flexible production systems. *IEEE Transaction on Robotics and Automation, 13*, 347–363.

Fanti, M. P., Maione, B., & Turchiano, B. (1998). Event control for deadlock avoidance in production systems with multiple capacity resources. *Studies in Informatics and Control, 7*, 343–364.

Fei, Z., Reveliotis, S., Miremadi, S., & Akesson, K. (2015). A BDD-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems. *IEEE Transaction on Automation Science and Engineering, 12*, 990–1006.

Ghaffari, A., Rezg, N., & Xie, X. (2003). Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Transaction on Robotics & Automation, 19*, 137–141.

Giua, A., DiCesare, F., & Silva, M. (1992). 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.

Giua, A., Fanti, M. P., & Seatzu, C. (2006). Monitor design for colored Petri nets: an application to deadlock prevention in railway networks. *Control Engineering Practice, 10*, 1231–1247.

Glasserman, P., & Yao, D. (1994). *Monotone structure in discrete-event systems*. NY: John Wiley & Sons, Inc.

Gold, E. M. (1978). Deadlock prediction: Easy and difficult cases. *SIAM Journal of Computing, 7*, 320–336.

Groover, M. P. (1996). *Fundamentals of modern manufacturing: Materials, processes and systems*. Englewood Cliffs, NJ: Prentice Hall.

Habermann, A. N. (1969). Prevention of system deadlocks. *Communications of the ACM, 12*, 373–377.

Havender, J. W. (1968). Avoiding deadlock in multi-tasking systems. *IBM Systems Journal, 2*, 74–84.

Heizer, J., & Render, B. (2010). *Operations management, (10th ed.)*. Upper Saddle River, NJ: Pearson.

Holt, R. D. (1972). Some deadlock properties of computer systems. *ACM Computing Surveys, 4*, 179–196.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages and computation*. Reading, MA: Addison-Wesley.

Hu, H., Zhou, M., & Li, Z. (2012). Liveness and ratio-enforcing supervision of automated manufacturing systems using Petri nets. *IEEE Transaction on Systems, Man and Cybernetics – Part A: Systems and Humans, 42*, 392–403.

Jeng, M., Xie, X., & Peng, M. Y. (2002). Process nets with resources for manufacturing modeling and their analysis. *IEEE Transaction on Robotics & Automation, 18*, 875–889.

Kelly, T., Wang, Y., Lafortune, S., & Mahlke, S. (2009). Eliminating concurrency bugs with control engineering. *IEEE Computer, 42*(12), 52–60.

Konda, V. R., & Tsitsiklis, J. N. (2003). Actor-critic algorithms. *SIAM Journal on Control and Optimization, 42*, 1143–1166.

Kumar, P. R. (1994). Scheduling semiconductor manufacturing plants. *IEEE Control Systems Magazine, 14–6*, 33–40.

Kushner, H. J., & Yin, G. G. (2003). *Stochastic approximation and recursive algorithms and applications*. NY: Springer.

Lawley, M., Reveliotis, S., & Ferreira, P. (1997). FMS Structural Control and The Neighborhood Policy, Part 1: Correctness and Scalability. *IIE Transaction, 29*, 877–887.

Lawley, M., Reveliotis, S., & Ferreira, P. (1997). FMS structural control and the neighborhood policy, part 2: Generalization, optimization and efficiency. *IIE Transaction, 29*, 889–899.

Lawley, M., Reveliotis, S., & Ferreira, P. (1998). The application and evaluation of Banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *The International Journal of Flexible Manufacturing Systems, 10*, 73–100.

Lawley, M., Reveliotis, S., & Ferreira, P. (1998). A correct and scalable deadlock avoidance policy for flexible manufacturing systems. *IEEE Transaction on Robotics & Automation, 14*, 796–809.

Lawley, M., & Sulistyono, W. (2002). Robust supervisory control policies for manufacturing systems with unreliable resources. *IEEE Transaction on R&A, 18*, 346–359.

Lawley, M. A., & Reveliotis, S. A. (2001). Deadlock avoidance for sequential resource allocation systems: hard and easy cases. *International Journal of FMS, 13*, 385–404.

Lawrence, P. (1997). *Workflow handbook*. NY: John Wiley & Sons, Inc.

Li, R., & Reveliotis, S. Designing parsimonious scheduling policies for complex resource allocation systems through concurrency theory. Discrete Event Dynamic Systems: Theory and Applications, (to appear).

Li, R., & Reveliotis, S. (2015). Performance optimization for a class of generalized stochastic Petri nets. *Discrete Event Dynamic Systems: Theory and Applications, 25*, 387–417.

Liao, H., Lafortune, S., Reveliotis, S., Y. , W., & Mahlke, S. (2013). Optimal liveness-enforcing control of a class of Petri nets arising in multithreaded software. *IEEE Transactions on Automatic Control, 58*, 21–28.

Liao, H., Wang, Y., Stanley, J., Lafortune, S., Reveliotis, S., Kelly, T., & Mahlke, S. (2013). Robust and adaptive supervisory control of discrete event systems. *IEEE Transaction Control, Systems Technology, 21*, 2067–2082.

Luo, J. C., Xing, K. Y., Zhou, M. C., Li, X. L., & Wang, X. N. (2015). Deadlock-free scheduling of automated manufacturing systems via Petri nets and hybrid heuristic search. *IEEE Transaction on SMC: Systems, 45*, 530–541.

Marbach, P., & Tsitsiklis, J. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transaction on Automatic Control, 46*, 191–209.

Moody, J. O., & Antsaklis, P. J. (1998). *Supervisory control of discrete event systems using Petri nets*. Boston, MA: Kluwer Academic Pub.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77*, 541–580.

Nazeem, A., & Reveliotis, S. (2011). A practical approach for maximally permissive liveness-enforcing supervision of complex resource allocation systems. *IEEE Transaction on Automation Science and Engineering, 8*, 766–779.

Nazeem, A., & Reveliotis, S. (2012). Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the non-linear case. *IEEE Transaction on Automatic Control, 57*, 1670–1684.

Nazeem, A., & Reveliotis, S. (2014). Efficient enumeration of minimal unsafe states in complex resource allocation systems. *IEEE Transaction on Automation Science & Engineering, 11*, 111–124.

Nazeem, A., & Reveliotis, S. (2015). Maximally permissive deadlock avoidance for resource allocation systems with R/W-locks. *Discrete Event Dynamic Systems: Theory and Applications, 25*, 31–63.

Nazeem, A., Reveliotis, S., Wang, Y., & Lafortune, S. (2011). Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the linear case. *IEEE Transaction on Automatic Control, 56*, 1818–1833.

Nilsson, N. J. (1990). *The mathematical foundations of learning machines*. San Mateo, CA: Morgan Kaufmann.

Park, J., & Reveliotis, S. (2000). Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems. *IEEE Transaction on R&A, 16*, 190–195.

Park, J., & Reveliotis, S. A. (2001). Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings. *IEEE Transaction on Automatic Control, 46*, 1572–1583.

Perros, H. G. (1994). *Queuing networks with blocking: Exact and approximate solutions*. NY: Oxford University Press.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality*. NY: Wiley.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons.

Qiao, F., & Zhou, M. (2014). Scheduling of semiconductor manufacturing systems using Petri nets. In J. Campos, C. Seatzu, & X. Xie (Eds.), *Formal methods in manufacturing* (pp. 553–569). CRC Press / Taylor & Francis.

Ramadge, P. J. G., & Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of the IEEE, 77*, 81–98.

Ramaswamy, S. E., & Joshi, S. B. (1996). Deadlock-free schedules for automated manufacturing workstations. *IEEE Transaction on R&A, 12*, 391–400.

Reveliotis, S., & Nazeem, A. (2013). Optimal linear separation of the safe and unsafe subspaces of sequential RAS as a set-covering problem: algorithmic procedures and geometric insights. *SIAM Journal on Control and Optimization, 51*, 1707–1726.

Reveliotis, S., & Nazeem, A. (2014). Deadlock avoidance policies for automated manufacturing systems using finite state automata. In J. Campos, C. Seatzu, & X. Xie (Eds.), *Formal methods in manufacturing* (pp. 169–195). CRC Press / Taylor & Francis.

Reveliotis, S., & Roszkowska, E. (2010). On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems. *IEEE Transaction on Automatic Control, 55*, 1646–1651.

Reveliotis, S. A. (1999). Accommodating fms operational contingencies through routing flexibility. *IEEE Transaction on R&A, 15*, 3–19.

Reveliotis, S. A. (2000). An analytical investigation of the deadlock avoidance vs. detection & recovery problem in buffer-space allocation of flexibly automated production systems. *IEEE Transaction on SMC: Part B, 30*, 799–811.
Reveliotis, S. A. (2000). Conflict resolution in AGV systems. *IIE Transaction, 32*(7), 647–659.
Reveliotis, S. A. (2003). On the siphon-based characterization of liveness in sequential resource allocation systems. In *Proceedings of the applications and theory of Perti nets 2003* (pp. 241–255).
Reveliotis, S. A. (2005). *Real-time management of resource allocation systems: A discrete event systems approach*. NY: Springer.
Reveliotis, S. A., Lawley, M. A., & Ferreira, P. M. (1997). Polynomial complexity deadlock avoidance policies for sequential resource allocation systems. *IEEE Transaction on Automatic Control, 42*, 1344–1357.
Ross, S. M. (1983). *Stochastic processes*. NY: Wiley.
Roszkowska, E., & Reveliotis, S. (2008). On the liveness of guidepath-based, zoned–controlled, dynamically routed, closed traffic systems. *IEEE Transaction on Automatic Control, 53*, 1689–1695.
Roszkowska, E., & Reveliotis, S. (2013). A distributed protocol for motion coordination in free-ranging vehicular systems. *Automatica, 49*, 1639–1653.
Seatzu, C., Silva, M., & van Schuppen, J. H. (2013). *Control of discrete event systems: Automata and Petri net perspectives*. London, UK: Springer.
Tolio, T. (2009). *Design of flexible production systems: Methodologies and tools*. Berlin, Germany: Springer-Verlag.
Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). *Facilities planning (4th ed.)*. Hoboken, NJ: John Wiley & Sons, Inc..
Tricas, F., Garcia-Valles, F., Colom, J. M., & Ezpeleta, J. (2005). A Petri net structure-based deadlock prevention solution for sequential resource allocation systems. In *Proceedings of the ICRA 2005* (pp. 271–277). IEEE.
Wang, Y., Cho, H. K., Liao, H., Nazeem, A., Kelly, T., Lafortune, S., … Reveliotis, S. (2010). Supervisory control for software execution for failure avoidance: experience from the Gadara project. In *Proceedings of the Wodes 2010*.

Spyros Reveliotis is a Professor in the School of Industrial & Systems Engineering, at the Georgia Institute of Technology. He holds a Diploma in Electrical Engineering from the National Technical University of Athens, Greece, an M.Sc. degree in Computer Systems Engineering from Northeastern University, Boston, and a Ph.D. degree in Industrial Engineering from the University of Illinois at Urbana-Champaign. Dr. Reveliotis' research interests are in the area of Discrete Event Systems theory and its applications. He is a Fellow of IEEE and a member of INFORMS. Dr. Reveliotis currently serves as a Senior Editor for the IEEE Trans. on Automation Science and Engineering, as a Department Editor for IIE Transactions, and as an Associate Editor for the Journal of Discrete Event Dynamic Systems. He has also served as an Associate Editor for the IEEE Transaction on Automatic Control, the IEEE Trans. on Robotics & Automation, and the IEEE Trans. on Automation Science and Engineering, and as a Senior Editor on the Conference Editorial Board for the IEEE Intl. Conference on Robotics & Automation. In 2009 he was the Program Chair for the IEEE Conference on Automation Science and Engineering, and currently he serves as a member of the Steering Committee for this conference. Dr. Reveliotis has been the recipient of a number of awards, including the Kayamori Best Paper Award for the 1998 IEEE Intl. Conference on Robotics & Automation and the 2014 Best Paper Award of the IEEE Trans. on Automation Science and Engineering.