Maximal linear deadlock avoidance policies for sequential resource allocation systems: characterization, computation and approximation

Michael Ibrahim, Spyros Reveliotis and Ahmed Nazeem

***Abstract*—The problem of maximally permissive deadlock avoidance for sequential resource allocation systems (RAS) is a well-defined problem in the current controls literature. The corresponding supervisor is known as the maximally permissive deadlock avoidance policy (DAP), and it can be perceived as a classifier effecting the dichotomy of the underlying state space into its "safe" and "unsafe" subspaces. In the deployment of the maximally permissive DAP, an important issue is the selection of an effective and computationally efficient representation of the aforementioned dichotomy. A popular such representation is the "linear classifier", where the admissibility of any given RAS state is resolved based on its ability to satisfy a given set of linear inequalities. However, linear classifiers cannot provide effective representation of the maximally permissive DAP for all RAS instantiations. Hence, this paper provides a methodology for synthesizing linear DAPs for any given RAS instance that might not be maximally permissive in the original sense of this term, but observe a more relaxed notion of "maximality" that is defined within the particular space of the DAPs that admit a linear representation. The presented developments formally define this new DAP class, and provide the necessary algorithms for the synthesis or the systematic approximation of maximal linear DAPs for any given RAS instance.**

Keywords: sequential resource allocation systems, liveness-enforcing supervision, algebraic deadlock avoidance policies

I. Introduction

Sequential resource allocation systems (RAS) [1] is a class of discrete event systems (DES) [2] that has received extensive attention in the corresponding literature. The supervisory control problem of deadlock avoidance that underlies the operation of these systems, seeks to coordinate the sequential allocation of a finite set of reusable resources to a set of concurrently executing processes so that all these processes are able to receive the requested resources with finite delays, and eventually complete their execution and exit the system [1]. Furthermore, there is an additional request for *maximal permissiveness* for the corresponding supervisory control policies; i.e., these policies should ensure the aforementioned capability of all activated processes to run successfully to their completion, while imposing the minimum possible restriction to the original behavior that is generated by the uncontrolled system. Such a maximally permissive supervisor – also, known as a maximally permissive deadlock avoidance policy (DAP) – is well-defined and unique for the RAS instantiations studied in [1]. But it is also true that computing the maximally permissive DAP is an NP-hard problem for almost all RAS classes of interest [3].

Nevertheless, recognizing that the sought DAPs act as classifiers that dichotomize the underlying RAS state space into admissible and inadmissible subspaces, the corresponding research community has developed methodology that enables the off-line synthesis of representations for these policies that are very parsimonious, and therefore amenable for real-time control [1]. Among these DAP representations, one of the most interesting and tractable, in terms of, both, analysis and implementation, is that of a "linear" classifier [4], [5], [6]. A linear classifier determines the admissibility of any given state based on the ability of this state to satisfy a certain set of linear inequalities. In the following, we shall refer to DAPs that admit such a linear representation of their state-admissibility logic as "linear" DAPs.[1]

But as established in [7], [8], linear representation of the maximally permissive DAP is not a viable option for all RAS instantiations of practical interest. To circumvent this limitation, the works of [9], [10], [11], [12] have proposed additional representations for the sought classifiers that either employ nonlinear discriminant functions of the RAS state, or they constitute "non-parametric" classification schemes that rely on the efficient storage and processing of explicit information about the structure of the underlying state space. These alternative representations have been shown to be complete, i.e., they will always provide an effective representation of the target DAP.

Yet, in spite of the aforementioned developments, in many application contexts, DAPs that admit linear representation are still a most desirable solution, due to the analyzability of these policies, and their easy integration into broader decision-making frameworks. And, in fact, the literature avails of methodology that can synthesize correct linear (but not necessarily maximally permissive) DAPs for a large spectrum of RAS classes of practical interest. Some characteristic examples of this methodology can be found in [13], [14], [15], [16], [17], [18], while a more comprehensive treatment of these methods is provided in Chapter 6 of [1]. But the existing theory does not allow for an explicit characterization and/or control of the extent of the sub-optimality of the DAPs that are derived by

M. Ibrahim is with the Department of Computer Engineering, Cairo University, Egypt; email: michael.nawar@eng.cu.edu.eg. S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology; email: {spyros@isye}.gatech.edu. A. Nazeem is with Facebook; email: nazeem.35@gmail.com This work has been partially supported by NSF grant ECCS-1707695.

[1] A more formal definition of this concept is provided in Section II.

this theory with respect to the maximally permissive DAP.

Motivated by the last remark in the previous paragraph, in this work we seek to develop a method for the systematic deployment of linear DAPs that are appropriate for the major RAS classes defined in [1], and observe a "maximality" requirement in terms of their permissiveness that is defined in the more restricted policy space of linear DAPs. In more specific terms, the first part of this work provides a complete formal characterization of the class of linear DAPs and of the notion of "maximality" that is observed by our policy design. This part also establishes the existence of such maximal linear DAPs for any given instance from the RAS classes that are considered in this work. Furthermore, it is shown that these policies might not be unique for any given RAS instance.

The second part of the work details the necessary representations and algorithms for the effective computation of the target policies for any given instance from the considered RAS class. In principle, the presented algorithms have the potential to provide an effective enumeration of all the maximal linear DAPs for these RAS instances. In practice, however, this potential is limited by a very high computational complexity. Nevertheless, it is shown that these algorithms still can provide a very effective instrument for synthesizing high-quality approximations of the target set of policies, where, as it is typical in the corresponding literature, the quality of the different policies is compared and measured by the relative size of their admissible subspaces. A series of numerical experiments presented in the last part of the paper (i) exemplify the aforementioned developments, (ii) demonstrate and assess the scalability of the presented algorithms, and also (iii) showcase the ability of some of these algorithms to return near-optimal linear DAPs for very large RAS configurations.

From a methodological standpoint, the employed definition of the "maximal linear DAP", and also the representations and the algorithms that we propose for the effective computation of these policies, build upon the geometrical representations and the corresponding insights that are provided in [4], [6], [7] for the computation of a linear representation of the maximally permissive DAP, whenever this last DAP admits a linear representation. In fact, as it will be seen in the following, the earlier developments of [4] constitute the very first stage in the overall computation that is pursued by the algorithms that are presented in this work. In this way, it is ensured that the newly developed algorithms will return the (unique) maximally permissive DAP itself, whenever this policy admits a linear representation. Furthermore, the "anchoring" of our results to those past developments, and the particular representations that have been employed by them, is critical for ensuring the computational tractability of the pursued computations.

In view of the above positioning of the paper content and its intended contribution, the rest of it is organized as follows: The next section provides a formal characterization of the RAS class that is the primary focus for this work, and of the corresponding supervisory control problem of deadlock avoidance. This section also overviews the existing results on the computation of a parsimonious linear representation of the maximally permissive DAP, and the conditions for the existence of such a linear representation. Subsequently, Section III introduces the new class of the maximal linear DAPs, explaining the rationale that underlies the definition of these policies, and establishing their well-posedness for any given instance from the considered RAS class. Section IV addresses the more practical issues of the computation and the approximation of maximal linear DAPs for any given RAS instance. Section V presents the numerical experiments that were mentioned in the previous paragraphs. Finally, Section VI concludes the paper and provides some directions for potential future work. Furthermore, the developments that are presented in this manuscript, are complemented and supported by an electronic supplement that is accessible at `https://www2.isye.gatech.edu/~spyros/maxlinDAP-sup.pdf`. This supplement provides (i) the formal proof of a technical result that is of a more supportive nature to the main correctness analysis of the algorithms that are presented in the paper, (ii) an extensive discussion on the implementational details for the presented algorithms, and (iii) a detailed complexity analysis of these algorithms and their supporting procedures.

Closing this introductory section, we also notice, for completeness, that a preliminary version of the results that are presented in this paper, was presented in IEEE CDC 2018 [19]. That earlier version of the results focused primarily on the theoretical positioning of the class of the maximal linear DAPs along lines similar to those pursued in Section III of this document, and also provided a basic outline of the algorithm that can be used, in principle, for the enumeration of the entire set of the maximal linear DAPs. On the other hand, the theoretical analysis of this algorithm, the necessary implementational details and the modifications of this algorithm that will establish a tractable computation, and also the computational results that demonstrate and assess the efficacy of the final outcome, is material that is presented for the first time in this work.[2]

II. The considered RAS class and the corresponding problem of deadlock avoidance

Disjunctive-Conjunctive RAS: The main ideas that define the methodology to be presented in this work, are applicable to the entire spectrum of the RAS classes that are defined in [1]. But for better clarity and specificity, in the rest of this paper we focus primarily on the class of Disjunctive-Conjunctive (D/C-) RAS. This is a pretty broad RAS class that allows for (i) an arbitrary structure of the resource requests that are posed by the different processing stages, and also for (ii) the presence of routing flexibility in the supported process plans. A formal definition of the D/C-RAS class is as follows:

[2]While this work was in review, manuscript [20] appeared in the literature. The problem addressed in that work is conceptually very similar to the problem that is addressed in this paper. However, our perusal of that manuscript has revealed that it suffers from essential technical problems; the most critical of these problems is a confounding of fundamental concepts in Petri net theory [21], like those of liveness, reversibility and deadlock, during the basic positioning of the problem that is addressed in that manuscript, since this problem subsequently compromises all the technical claims that are made in the paper.

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

1) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*.
2) $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ – the set of strictly positive integers – is the system *capacity* function, characterizing the number of identical units from each resource type available in the system. Resources are assumed to be *reusable*, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore, $C(R_i) \equiv C_i$ [3] constitutes a system *invariant* for each i.
3) $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ denotes the set of the system *process types* supported by the considered system configuration. Each process type Π_j is a composite element itself, in particular, $\Pi_j = \langle \Theta_j, \mathcal{G}_j \rangle$, where: (a) $\Theta_j = \{\theta_{j,1}, \ldots, \theta_{j,l_j}\}$ denotes the set of *processing stages* involved in the definition of process type Π_j, and (b) $\mathcal{G}_j$ is an *acyclic digraph* with its node set, Q_j, being bijectively related to the set Θ_j. Denoting by $Q_j^{\nearrow}$ (resp., $Q_j^{\searrow}$) the set of *source* (resp., *sink*) nodes of $\mathcal{G}_j$, the available *process plans* for process type Π_j are represented by the *paths* leading from some node $q_s \in Q_j^{\nearrow}$ to some node $q_f \in Q_j^{\searrow}$ in digraph $\mathcal{G}_j$. Each processing stage $\theta_{j,k},\ j = 1, \ldots, n,\ k = 1, \ldots, l_j$, is an entity that is distinct from all the remaining processing stages that are supported by the considered RAS; in particular, $\forall j, q$ with $j \neq q$, $\Theta_j \cap \Theta_q = \emptyset$. Also, in the following, we shall set $\Theta \equiv \bigcup_{j=1}^{n} \Theta_j$ and $\xi \equiv |\Theta|$.
4) $D : \Theta \rightarrow \prod_{i=1}^{m} \{0, \ldots, C_i\}$ is the *resource allocation function* associating every processing stage $\theta_{j,k}$ with the *resource allocation vector* $D(\theta_{j,k})$ required for its execution; it is further assumed that $\forall j, k,\ D(\theta_{j,k}) \neq \mathbf{0}$.

At any point in time, RAS Φ contains a certain number of (possibly zero) instances of each process type that execute one of the corresponding processing stages. A process instance executing a non-terminal stage $\theta_{j,k} \in Q_j \backslash Q_j^{\searrow}$, in order to advance to a successor processing stage $\theta_{j,q}$ in the corresponding digraph $\mathcal{G}_j$, first must be allocated the resource differential $[D(\theta_{j,q}) - D(\theta_{j,k})]^+$ and only then will it release the resource units $|[D(\theta_{j,q}) - D(\theta_{j,k})]^-|$ that are not needed anymore.[4] This resource allocation protocol further requires that no resource type $R_i \in \mathcal{R}$ will be over-allocated with respect to its capacity C_i at any point in time. Hence, a process instance J_j can advance to one of its next processing stages only if the resource units requested for this advancement can be allocated from the slack capacity of the corresponding resources.

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

Modeling the D/C-RAS dynamics as a Finite State Automaton: The dynamics of the RAS $\Phi = \langle \mathcal{R}, C, \mathcal{P}, D \rangle$ that was described in the previous paragraph, can be further formalized by a *Deterministic Finite State Automaton (DFSA)* [2] $G(\Phi) = \langle S, E, f, \mathbf{s}_0,\ S_M \rangle$, that is defined as follows:

1) The *state set* S consists of ξ-dimensional vectors $\mathbf{s}$. The components $\mathbf{s}[l],\ l = 1, \ldots, \xi$, of $\mathbf{s}$ are in one-to-one correspondence with the RAS processing stages,[5] and they indicate the number of process instances executing the corresponding stage in the considered RAS state. Hence, S consists of all the vectors $\mathbf{s} \in (\mathbb{Z}_0^+)^{\xi}$ that further satisfy

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

where, according to the adopted notation, $D(\theta_l)[i]$ denotes the allocation request for resource R_i that is posed by stage θ_l.[6]
2) The *event set* E is the union of the disjoint event sets $E^{\nearrow}$, $\bar{E}$ and $E^{\searrow}$, where:
   a) $E^{\nearrow} = \{e_{r,p} : r = 0,\ \theta_p \in \bigcup_{j=1}^{n} Q_j^{\nearrow}\}$, i.e., event $e_{r,p}$ represents the *loading* of a new process instance that starts from stage θ_p.
   b) $\bar{E} = \{e_{r,p} : \exists j \in 1, \ldots, n$ s.t. θ_p is a successor of θ_r in graph $\mathcal{G}_j\}$, i.e., $e_{r,p}$ represents the *advancement* of a process instance executing stage θ_r to a successor stage θ_p.
   c) $E^{\searrow} = \{e_{r,p} : \theta_r \in \bigcup_{j=1}^{n} Q_j^{\searrow},\ p = 0\}$, i.e, $e_{r,p}$ represents the *unloading* of a finished process instance after executing its last stage θ_r.
3) The *state transition function* $f : S \times E \rightarrow S$ is defined by $\mathbf{s}' = f(\mathbf{s}, e_{r,p})$, where the components $\mathbf{s}'[l]$ of the resulting state $\mathbf{s}'$ are given by:

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

   We also notice that $f(\mathbf{s}, e_{r,p})$ is a *partial* function, defined only if the resulting state $\mathbf{s}'$ belongs in S (i.e., it satisfies the condition of Equation 1). For any state $\mathbf{s} \in S$, the event set $\Gamma(\mathbf{s}) \equiv \{e_{r,p} \in E : f(\mathbf{s}, e_{r,p})$ is defined$\}$ constitutes the set of *feasible events* at $\mathbf{s}$.
4) The *initial state* $\mathbf{s}_0$ is set equal to $\mathbf{0}$, i.e., the state vector with all its components equal to zero. This initial state represents the situation where the system is empty of any process instances.
5) The *set of marked states* S_M is the singleton $\{\mathbf{s}_0\}$. This specification of S_M expresses the requirement for complete process runs.

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

[3] In this document, the notation '$\equiv$' implies 'equality by definition'.

[4] We remind the reader that $\forall x \in \mathbb{R},\ [x]^+ \equiv \max\{x, 0\}$ and $[x]^- \equiv min\{x, 0\}$.

[5] We also notice that in various parts of the paper we shall use this correspondence between the state components $\mathbf{s}[l],\ l = 1, \ldots, \xi$, and the processing stages $\theta_{j,k},\ j = 1, \ldots, n,\ k = 1, \ldots, l_j$, in order to refer to (and index) the various processing stages.

[6] Following standard practice in DES literature (cf., for instance, the relevant definition in page 8 of [2]), in the rest of this document we will frequently use the terms "space" and "subspace" in order to refer to the state set S and its various subsets considered in this work.

the *reachable subspace* S_r of $G(\Phi)$ by

$$S_r \equiv \{\mathbf{s} \in S : \exists \sigma \in L(G) \text{ s.t. } \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}\} \tag{2}$$

and its *safe subspace* S_s by

$$S_s \equiv \{\mathbf{s} \in S : \exists \sigma \in E^* \text{ s.t. } \hat{f}(\mathbf{s}, \sigma) = \mathbf{s}_0\} \tag{3}$$

Also, in the following, we shall denote the complements of S_r and S_s with respect to S by $S_{\bar{r}}$ and $S_{\bar{s}}$, and we shall refer to them as the *unreachable* and *unsafe* subspaces. Finally, S_{xy}, $x \in \{r, \bar{r}\}$, $y \in \{s, \bar{s}\}$, will denote the intersection of the corresponding sets S_x and S_y.

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

$$\begin{aligned} L_m(G) &\equiv \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) \in S_M\} \\ &= \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}_0\} \end{aligned} \tag{4}$$

Equation 4, when combined with all the previous definitions, further implies that the set of states that are accessible under $L_m(G)$ is exactly equal to S_{rs}. Hence, we have the following definition of the *maximally permissive deadlock avoidance policy (DAP)* Δ^* for the considered RAS:

Definition 2: The *maximally permissive deadlock avoidance policy (DAP)* Δ^* for any instantiation Φ from the RAS class of Definition 1 is a supervisory control policy[7] that, at every state $\mathbf{s} \in S_{rs}$, admits the transition that is defined by any event $e_{r,p} \in \Gamma(\mathbf{s})$ if and only if the resulting state $\mathbf{s}' = f(\mathbf{s}, e_{r,p})$ belongs in S_s. □

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

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

As discussed in the introductory section, methodology for the effective development of such a classifier is provided in [4], [9], [10], [11], [12], [8], [6]. A result that has proven very useful in the development of the corresponding theory, is the following "monotonicity" property that is exhibited by the RAS state safety:

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

$$\forall \mathbf{s}, \mathbf{s}' \in S, \quad \mathbf{s} \leq \mathbf{s}' \iff (\forall l = 1, \ldots \xi, \quad \mathbf{s}[l] \leq \mathbf{s}'[l]) \tag{5}$$

[7]We remind the reader that in DES supervisory control theory, a supervisory control policy – or, more briefly, a supervisor – is a mapping that associates every state $\mathbf{s}$ of the underlying state space S with some subset of $\Gamma(\mathbf{s})$ that defines the set of feasible events at state $\mathbf{s}$ that are also admissible by the supervisor [2].

Then,

1) $\mathbf{s} \in S_s \ \wedge \ \mathbf{s}' \leq \mathbf{s} \implies \mathbf{s}' \in S_s$
2) $\mathbf{s} \in S_{\bar{s}} \ \wedge \ \mathbf{s} \leq \mathbf{s}' \implies \mathbf{s}' \in S_{\bar{s}}$

□

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

Linear representation of the maximally permissive policy Δ^*: As remarked in the introductory section, a desirable representation of the classification logic that is effected by the maximally permissive DAP Δ^* is that of a linear classifier. This last concept has been formally defined in [4] as follows:

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

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

$$\begin{aligned} &(\forall \mathbf{g} \in G : \ \forall i \in \{1, \cdots, k\}, \ \mathbf{a}_i^T \cdot \mathbf{g} \leq b_i) \quad \wedge \\ &\quad (\forall \mathbf{h} \in H : \exists i \in \{1, \cdots, k\}, \ \mathbf{a}_i^T \cdot \mathbf{h} > b_i) \end{aligned} \tag{6}$$

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

□

In the case of the classification that is effected by the DAP Δ^*, the roles of the sets G and H in Definition 3 are played, respectively, by the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ that contain the maximal reachable safe states and the minimal boundary unsafe states.[8] Moreover, Proposition 1 implies the following additional result for the sought classifiers [4]:

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

On the other hand, it is also well known that the maximally permissive DAP Δ^* might not admit a linear representation

[8] The astute reader will also notice that Definition 3 implies an asymmetry for the role of the the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the design of the sought classifiers. This asymmetry has been dictated by an intention for further implementation of the developed classifiers in the Petri net (PN) modeling framework [21], through the theory of monitor places [22], [23]; besides the FSA modeling framework, Petri nets have been another major formal framework for the modeling and the analysis of the dynamics of the RAS classes considered in this work.

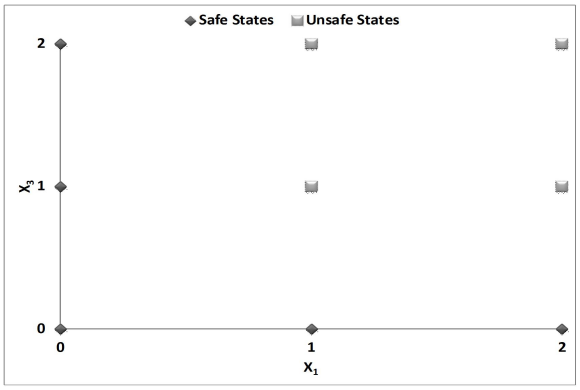

Fig. 1: Characterization of the safe and unsafe reachable states for an example D/C-RAS with two resource types, R_1 and R_2, of corresponding capacities $C(R_1) = C(R_2) = 2$, and two process types, Π_1 and Π_2. Process type Π_1 involves two processing stages, $\theta_{1,1}$ and $\theta_{1,2}$, with processing stage $\theta_{1,1}$ preceding the processing stage $\theta_{1,2}$; the corresponding resource requirements for these two processing stages are respectively defined by the vectors $(1,0)^T$ and $(0,2)^T$. Process type Π_2 also involves two processing stages, $\theta_{2,1}$ and $\theta_{2,2}$, with processing stage $\theta_{2,1}$ preceding the processing stage $\theta_{2,2}$, and the corresponding resource requirements for each processing stage of this process type being defined by the vectors $(0,1)^T$ and $(2,0)^T$. Recognizing that the terminal processing stages of these two process types will never get involved in a deadlock, we can characterize state safety for this RAS by focusing only on the state components $\mathbf{s}_1$ and $\mathbf{s}_3$, which correspond to the first processing stage of each process plan.The above figure provides this reduced, 2-dimensional representation of the underlying state space, where safe reachable states are depicted by rhombi and unsafe reachable states by squares. The reader should notice that the convex hull of the depicted safe states includes the unsafe state corresponding to point $(1,1)$, and therefore, in this case, the reachable safe states and the boundary unsafe states of the considered system are not linearly separable.

along the lines of Definition 3 [7], [8], [24], [25]. Such a case is presented in Figure 1, where it can be seen that the lack of a linear representation for the corresponding DAP Δ^* is due to the inclusion of elements of the set $\bar{S}^b_{r\bar{s}}$ in the convex hull[9] of S_{rs}. As remarked in the introductory section, this problem has been addressed through the development of additional representations for the classification logic that is effected by the target policy Δ^*. However, in Section I it was also pointed out that linear DAPs are still very desirable due to (i) the efficient representation and the analyzability of these DAPs by means of popular modeling frameworks like the PN modeling framework that is mentioned in Footnote 8, and (ii) the easy integration of these DAPs into broader decision-making frameworks concerning additional operational aspects of the considered RAS. One such possibility that is of particular interest in an ongoing research program of ours is the inclusion of the logic of the employed DAPs into some linear programming formulations that seek to complement the preventive control of deadlock avoidance with scheduling capability, and are known as "fluid relaxations" of the underlying RAS dynamics [27], [28].[10]

Motivated by the above remarks and needs, in the rest of this work, we define an approximation to the maximally permissive DAP Δ^* that (i) admits a linear representation along the lines of Definition 3 and Proposition 2, and (ii) is effectively computable for every instance Φ of the considered class of D/C-RAS. The formal definition of these policies employs a notion of "maximality" that intends to keep their permissiveness as close as possible to the permissiveness of Δ^*. We provide a formal characterization of this "maximality" concept, and the necessary algorithms for the effective computation of the corresponding policies, for any given D/C-RAS Φ.

III. Maximal linear DAPs

The formal definition of the maximal linear DAPs and its motivation: This section introduces the new concept of the "maximal linear DAP", as it materializes in the considered class of D/C-RAS. We shall formally define this new DAP class by providing a complete set of conditions that must be satisfied by the admissible subspaces of its constituent policies.[11]

Hence, in order to proceed with the subsequent developments, let Δ denote a tentative DAP from the considered class for some given D/C-RAS Φ, and $S_a(\Delta) \subseteq S$ denote the corresponding policy-admissible subspace. We also define $S_{\bar{a}}(\Delta) \equiv S \setminus S_a(\Delta)$. For the controlled dynamics of RAS Φ to be well-defined, clearly we need

$$\mathbf{s}_0 \in S_a(\Delta) \tag{7}$$

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

$$S_r(\Delta)^{(0)} := \{\mathbf{s}_0\} \tag{8}$$

$$S_r(\Delta)^{(k+1)} := S_r(\Delta)^{(k)} \cup \{\mathbf{s}' \in S_a(\Delta) : \exists \mathbf{s} \in S_r(\Delta)^{(k)}, e \in \Gamma(\mathbf{s}) \text{ with } f(\mathbf{s}, e) = \mathbf{s}'\} \tag{9}$$

A primary requirement in the specification of the sought policy Δ is that it does not induce any new deadlocks or livelocks; such a DAP is characterized as "correct" in the relevant literature [1]. The correctness of Δ translates into the following requirement for the corresponding set $S_r(\Delta)$ (c.f. [1], Chapter 6):

$$\forall \mathbf{s} \in S_r(\Delta) \setminus \{\mathbf{s}_0\},\ \exists e \in \Gamma(\mathbf{s}) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \in S_r(\Delta) \tag{10}$$

In more natural terms, the condition of Equation 10 requires that at every state $\mathbf{s}$ that is reachable in the considered RAS under supervision by Δ, there is a policy-admissible event e that concerns the stage advancement or the unloading of an already initiated process instance; since the digraphs $\mathcal{G}_j$ encoding the process plans of each process type Π_j, $j = 1, \ldots, n$, are acyclic, a repetitive invocation of this last condition guarantees the existence of an entire event sequence

[9] We remind the reader that the *convex hull* of a finite set of points S is the minimal convex polytope that contains all these points [26]. In this work, we shall denote the convex hull of such a point set S by $conv(S)$.

[10] Some other representative works that employ LP-based "fluid relaxations" as instruments for computing efficient scheduling policies for sequential resource allocation, can be found in [29], [30], [31], [32].

[11] We provide a systematic justification for this representational choice of ours in the closing part of this section.

that will complete every process instance J_j that is activated in some given state $\mathbf{s} \in S_r(\Delta)$.

An additional important operational requirement that must be observed by our target policies is that they should enable the execution of each process type $\Pi_j,\ j = 1, \ldots, n$, of the underlying RAS Φ. DAPs that satisfy this requirement will be characterized as "complete" in the following. The requirement for completeness can be enforced through the following condition:

$$\forall j = 1, \ldots, n,\ \exists e \in \Gamma(\mathbf{s}_0) :\ \big(\mathbf{s} \equiv f(\mathbf{s}_0, e)\big) \\ \wedge\ \big(\mathbf{s} \in S_r(\Delta)\big)\ \wedge\ \big(\mathbf{s}[i] = 1 \Longrightarrow \theta_i \in Q_j^{\nearrow}\big) \tag{11}$$

In more natural terms, the condition of Equation 11 implies that, for each process type $\Pi_j,\ j = 1, \ldots, n$, there is a policy-admissible event that activates this process type. When combined with the "correctness" requirement for the considered policies, this requirement implies the existence of an admissible process plan for every process type Π_j.

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

To formally state the conditions that will help us meet these two requirements, let $conv(S_r(\Delta))$ denote the convex hull of the set $S_r(\Delta)$, according to the corresponding notation that was introduced in Footnote 9, and also define the set $S_{\bar{r}}^b(\Delta)$ as follows:

$$S_{\bar{r}}^b(\Delta) \equiv \{\mathbf{s}' \in S_r \setminus S_a(\Delta) : \\ \exists \mathbf{s} \in S_r(\Delta),\ e \in \Gamma(\mathbf{s}) \text{ with } f(\mathbf{s}, e) = \mathbf{s}'\} \tag{12}$$

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

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

$$\forall \mathbf{s}, \mathbf{s}' \in S,\ \mathbf{s}' \leq \mathbf{s}\ \wedge \mathbf{s} \in S_a(\Delta) \Longrightarrow \mathbf{s}' \in S_a(\Delta) \tag{13}$$

$$conv(S_r(\Delta)) \cap S_{\bar{r}}^b(\Delta) = \emptyset \tag{14}$$

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

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

Definition 4: A policy Δ is a *linear* DAP for some given D/C-RAS Φ *iff* its admissible subspace $S_a(\Delta)$, together with the policy-reachable subspace, $S_r(\Delta)$, and the set of the boundary inadmissible states, $S_{\bar{r}}^b(\Delta)$, satisfy the following conditions:

Correctness: $(\mathbf{s}_0 \in S_a(\Delta))\ \wedge (\forall \mathbf{s} \in S_r(\Delta) \setminus \{\mathbf{s}_0\},$
$\exists e \in \Gamma(\mathbf{s}) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \in S_r(\Delta)\big)$

Completeness: $\forall j = 1, \ldots, n,\ \exists e \in \Gamma(\mathbf{s}_0) :$
$\big(\mathbf{s} \equiv f(\mathbf{s}_0, e)\big)\ \wedge\ \big(\mathbf{s} \in S_r(\Delta)\big)\ \wedge$
$\big(\mathbf{s}[i] = 1 \Longrightarrow \theta_i \in Q_j^{\nearrow}\big)$

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

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

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

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

Indeed, both of these policies admit the initial state $\mathbf{s}_0$ and it can be easily checked that they do not suffer from any policy-induced deadlock or livelock. Furthermore, they satisfy the "completeness" and the "monotonicity" requirements of Definition 4, and the corresponding state sets $S_r(\Delta^i), S_{\bar{r}}^b(\Delta^i),\ i = 1, 2$, will admit linear separation in the projected space that is defined by the state coordinates $\mathbf{s}_1$ and $\mathbf{s}_3$. Finally, these two policies are also maximal, since the only possible expansion of the corresponding sets $S_r(\Delta^i),\ i = 1, 2$, is by re-admitting the blocked pairs $(0, 2)$ and $(2, 0)$ in the corresponding sets $S_a^i,\ i = 1, 2$; but the policy that will result from any of these two augmentations is Δ^*, and we know that this policy is not linear.

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

Existence but non-uniqueness of maximal linear DAPs: The closing remark in the previous example further implies that, for any given D/C-RAS Φ, the maximal linear DAPs of Definition 4 will not be unique, in general. Hence, for further reference, we shall denote the set of linear DAPs for any given D/C-RAS Φ by $\mathcal{L}(\Phi)$, and its subset that contains its maximal elements by $\bar{\mathcal{L}}(\Phi)$.

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

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

Proof: For any given D/C-RAS Φ, consider the policy $\hat{\Delta}$ that admits a state $\mathbf{s} \in S$ *iff* (a) either it is the initial state $\mathbf{s}_0$, or (b) it contains only one active process instance. Then, it is easy to see that the policy $\hat{\Delta}$ is correct and complete, and satisfies the "monotonicity" requirement of Definition 4. It is also clear that the admissibility logic of this policy can

be expressed by the linear inequality

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

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

Justifying the adopted representation for the sought policies: As stated in the opening paragraph of this section, the entire characterization of the class of linear DAPs, $\mathcal{L}(\Phi)$, and the notion of "maximality" that is defined within the scope of this policy class, was based on a representation of its constituent policies Δ by their admissible subspaces, $S_a(\Delta)$, and the additional state sets $S_r(\Delta)$ and $S_{\bar{r}}^b(\Delta)$ that are induced by the set $S_a(\Delta)$ and the qualitative dynamics of the underlying RAS Φ. An alternative representation of this class of policies can be based on the sets of linear inequalities that define the corresponding classifiers (c.f. Definition 3 and Proposition 2); a compact representation of any such set of inequalities is through the corresponding pair $(A, \mathbf{b})$, where A is a $(k \times \xi)$-dimensional nonnegative real matrix, $\mathbf{b}$ is a k-dimensional nonnegative real vector, and k is the number of inequalities in this set. Let $L(\Phi)$ denote the set of pairs $(A, \mathbf{b})$ with $A \geq 0,\ \mathbf{b} \geq \mathbf{0}$, that define a correct, complete, linear DAP $\Delta(A, \mathbf{b})$ for a given RAS Φ according to the logic of Definition 4, and $\bar{L}(\Phi)$ denote the subset of $L(\Phi)$ that collects the maximal linear DAPs (always according to Definition 4). Next, we provide a formal connection between the two alternative representations of the target policy spaces, i.e., the representation that is provided by the sets $\mathcal{L}(\Phi)$ and $\bar{\mathcal{L}}(\Phi)$, and the alternative representation for the same policy spaces that is provided by the sets $L(\Phi)$ and $\bar{L}(\Phi)$.

For this, let us define the following binary relation '$\sim$' on $L(\Phi)$:

$$(A, \mathbf{b}) \sim (A', \mathbf{b}') \Longleftrightarrow S_r(\Delta(A, \mathbf{b})) = S_r(\Delta(A', \mathbf{b}')) \quad (15)$$

The relation '$\sim$' is an equivalence relation on $L(\Phi)$. The equivalence classes, $[(A, \mathbf{b})]$, that are defined in the set $L(\Phi)$ by the relation '$\sim$', correspond to the elements of the set $\mathcal{L}(\Phi)$ that was defined in the previous paragraphs of this section. Furthermore, the set $\bar{L}(\Phi)$ that collects the maximal linear DAPs under the $(A, \mathbf{b})$-based representation of these policies, can be expressed as

$$\bar{L}(\Phi) = \left\{(A, \mathbf{b}) :\ [(A, \mathbf{b})] \in \bar{\mathcal{L}}(\Phi)\right\} \quad (16)$$

The connection among the sets $\mathcal{L}(\Phi)$ (resp., $\bar{\mathcal{L}}(\Phi)$) and $L(\Phi)$ (resp., $\bar{L}(\Phi)$) that was established in the previous paragraph, essentially expresses the fact that the separation of the sets $S_r(\Delta)$ and $S_{\bar{r}}^b(\Delta)$ can be attained by an infinite number of systems of linear inequalities $(A, \mathbf{b})$, $A \geq 0,\ \mathbf{b} \geq \mathbf{0}$, when these two sets are linearly separable according to the logic of Definition 4. Hence, the sets $L(\Phi)$ and $\bar{L}(\Phi)$ will have an infinite cardinality. On the other hand, it is clear from the relevant definitions that were provided in the previous parts of this section, that the sets $\mathcal{L}(\Phi)$ and $\bar{\mathcal{L}}(\Phi)$ have finite cardinality, and therefore, they are amenable to explicit enumeration. This realization explains our focus on the two sets $\mathcal{L}(\Phi)$ and $\bar{\mathcal{L}}(\Phi)$ as the primary representation for the developments that are pursued in this work.

Finally, with the notion of the "maximal linear DAPs" and their effective representation well-defined, next we turn to the development of the necessary algorithms that will enable us to obtain some elements from the two sets $\bar{\mathcal{L}}(\Phi)$ and $\mathcal{L}(\Phi)$ that will constitute highly permissive linear DAPs, for any given D/C-RAS Φ.

IV. Computation and approximation of the maximal linear DAPs

A. Preamble

In this section, we consider the computation of the maximal linear DAPs for any given D/C-RAS Φ. We start with the introduction of a basic algorithm that, in principle, can enumerate all the maximal linear DAPs $\Delta \in \bar{\mathcal{L}}(\Phi)$, for any given D/C-RAS Φ, and we formally prove the algorithm correctness and the finiteness of its computation. This algorithm essentially effects a search process for the target policies, using a "branch & bound (b&b)" scheme that is methodologically similar to the b&b schemes that are used in combinatorial optimization [33], [34].

But as is the case with many other combinatorial-optimization problems and the corresponding b&b schemes, the computational cost of a complete execution of the aforementioned algorithm may be prohibitively high, for most practical cases. At the same time, it is also well known from the corresponding combinatorial optimization theory, that this high computational cost can be significantly controlled through a careful selection of (i) the representations, and the corresponding data structures, that are employed by the algorithm, and (ii) the "branching logic" that drives the underlying search process. Hence, in the second part of this section, we also discuss those implementational details and modifications that we have effected upon the original algorithm in an effort to expedite its execution.

The algorithm that is obtained from the aforementioned modifications, can be executed on some pretty sizable D/C-RAS configurations within very reasonable computational times, and it will provide linear DAPs that might not be provably maximal, but they are still capable to admit a very large subset of the set S_{rs} that is admitted by the maximally permissive DAP Δ^*. Also, the relative coverage of the reference set S_{rs} by the obtained DAPs Δ is effectively measurable during the algorithm execution, and this capability enables our algorithms to effect a controlled trade-off between the extent of their computation and the quality of the derived DAPs. In this way, the algorithmic developments of this section become a practical instrument for the effective and the controlled synthesis of high-quality linear DAPs for any given D/C-RAS Φ.

The aforestated computational developments are complemented in Section V by extensive numerical experimentation that seeks to (a) demonstrate the efficacy of these developments, and (b) assess the quality of the DAPs that will be practically obtained by them. Also, as remarked in the introductory section, some further material regarding (i) the

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

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

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

Input: DFSA $G(\Phi)$, maximal-state set $\bar{S}$, pruned state $\tilde{\mathbf{s}}$
Output: $PRUNE(S, G(\Phi))$
$\hat{S}_r := \{\mathbf{s}_0\}$;
while $\hat{S} := \{\mathbf{s} \in S \setminus (\hat{S}_r \cup \{\tilde{\mathbf{s}}\}) : (\exists \mathbf{s}' \in \hat{S}_r,\ e \in \Gamma(\mathbf{s}')$ with $f(\mathbf{s}', e) = \mathbf{s})$ AND $(\exists \mathbf{s}'' \in \bar{S}$ s.t. $\mathbf{s} \leq \mathbf{s}'')\} \neq \emptyset$ **do**
  $\hat{S}_r := \hat{S}_r \cup \hat{S}$;
end while
while $\hat{S} := \{\mathbf{s} \in \hat{S}_r : \forall e \in \Gamma(\mathbf{s}) \backslash E^{\nearrow},\ f(\mathbf{s}, e) \notin \hat{S}_r\} \neq \emptyset$ **do**
  $\hat{S}_r := \hat{S}_r \setminus \hat{S}$;
end while
$\bar{S}_r := \{\mathbf{s} \in \hat{S}_r : \nexists \mathbf{s}' \in \hat{S}_r$ s.t. $\mathbf{s}' > \mathbf{s}\}$;
return $\bar{S}_r$;

establishment of the correctness of the presented algorithms, (ii) the specification of certain implementational details for these algorithms, and (iii) a detailed complexity analysis of the algorithms themselves and their supporting procedures, can be found in an electronic supplement to this paper that is accessible at: `https://www2.isye.gatech.edu/~spyros/maxlinDAP-sup.pdf`.

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

Description of the proposed algorithm: The basic structure of the algorithm that we propose for the complete enumeration of the policy set $\bar{\mathcal{L}}(\Phi)$, for any given D/C-RAS Φ, is presented in the pseudo-code of Algorithm 1. This algorithm starts with the computation of the set of reachable and safe states, S_{rs}, that defines the reachable subspace under the maximally permissive DAP Δ^*, and seeks to detect all the maximal subsets of this set – including the set S_{rs} itself – that will define correct linear DAPs.

In more specific terms, Algorithm 1 starts by computing the set S_{rs} of the reachable and safe states of the considered D/C-RAS Φ; this computation can be performed straightforwardly through basic reachability and co-reachability analyses of the underlying state space with respect to the empty state $\mathbf{s}_0$ [2]. Subsequently, the algorithm tests whether the obtained set S_{rs} is linearly separable from the set of the boundary reachable unsafe states, $S^b_{r\bar{s}}$, and if this test is positive, then the algorithm infers that, for the considered RAS instance Φ, the maximally permissive DAP Δ^* is linearly representable; hence, it exits returning the set S_{rs} as the single element of the set $\bar{\mathcal{L}}(\Phi)$. On the other hand, if the maximally permissive DAP Δ^* is not linearly representable, the algorithm will run a search process for all those proper subsets of S_{rs} that constitute the reachable subspace for a maximal linear DAP $\Delta \in \bar{\mathcal{L}}(\Phi)$.

Each proper subset of S_{rs} that is considered by this search process, is obtained from a "parent" subset in the generated "search graph" by (i) first removing a single maximal element of the "parent" set, and subsequently (ii) pruning any additional states that need to be removed in order to restore the correctness of the induced DAP Δ. Furthermore, the DAP Δ that is obtained through this processing, must be tested for membership in $\mathcal{L}(\Phi)$.

Some possible ways to perform the linearity test for the maximally permissive DAP Δ^* – and also for all the other DAPs Δ that will be considered by Algorithm 1 during its search process – are through the algorithms that are provided in [4] for the construction of a linear separator for the DAP Δ^* whenever such a linear separator is available. However, according to our numerical experimentation that is reported in Section V, the most efficient way to conduct the DAP-linearity tests that are performed in this work, is by computing the sets $\bar{S}_r$ and $\bar{S}^b_{\bar{r}}$, containing, respectively, the maximal reachable states and the minimal boundary inadmissible states under the considered policy Δ, and, subsequently testing the linear separability of each state $\mathbf{u} \in \bar{S}^b_{\bar{r}}$ from the states $\mathbf{s} \in \bar{S}_r$, by formulating and solving the following linear program (LP) for each $\mathbf{u} \in \bar{S}^b_{\bar{r}}$:

$$\max_{\mathbf{a} \geq \mathbf{0}, b \geq 0} 0 \tag{17}$$

s.t.

$$\mathbf{a}^T \mathbf{s}_i \leq b\ ,\ \forall \mathbf{s}_i \in \bar{S}_r \tag{18}$$

$$\mathbf{a}^T \cdot \mathbf{u} \geq b + \epsilon \tag{19}$$

In the above LP formulation, ϵ is a preselected parameter such that $\epsilon \to 0^+$. This LP is essentially a feasibility test for the constraints that appear in it. A negative outcome for this test implies that the considered minimal boundary inadmissible state $\mathbf{u}$ is not linearly separable from the set of the Δ-maximal

admissible states, $\bar{S}_r$, and therefore, $\Delta \notin \mathcal{L}(\Phi)$. Furthermore, all those states $\mathbf{u} \in \bar{S}_{\bar{r}}^b$ that will fail the test of Equations 17-19, will be used by Algorithm 1 in order to spawn from policy Δ some new candidate policies Δ', according to the "state-elimination" scheme that was mentioned in the previous paragraph.[12]

Another salient point for the complete understanding of the pseudo-code that is presented in Algorithm 1, is that the aforementioned subsets of S_{rs} that are generated during the search process, will be represented by means of their maximal elements; in the presented pseudo-code, this fact is indicated by "barring" or "tilding" the corresponding sets. In the statement of Algorithm 1, we also denote the policy Δ that is induced by such a set of maximal states, $\bar{S}_r$, by $\Delta(\bar{S}_r)$. Furthermore, in accordance with our previously defined notation, the entire reachable subspace for the policy $\Delta(\bar{S}_r)$ is denoted by $S_r(\Delta(\bar{S}_r))$. A detailed discussion on the buildup and the maintenance of the particular representations that are employed by Algorithm 1 throughout its entire execution, is provided in the electronic supplement of this paper.

The search process that was described in the previous paragraphs, is facilitated in Algorithm 1 through the employment of the two lists $STORE$ and $EXPLORE$. The list $STORE$ holds the subsets of S_{rs} that correspond to linear DAPs and are maximal among the currently detected such sets. The list $EXPLORE$ holds those subsets of S_{rs} that have been generated as potential candidates for specifying maximal linear DAPs, but have not been assessed and processed yet. The detailed processing of a set $\bar{S}_r$ extracted from the list $EXPLORE$ is defined in Lines 3–16 of Algorithm 1, and it consists of the following steps: First it is checked whether this set defines a linear DAP, through the test that was presented in the previous paragraphs. If this is the case, and, furthermore, the processed set $\bar{S}_r$ is not dominated by any set already in $STORE$, it is entered in $STORE$ as the reachable subspace of a tentative maximal linear DAP. During this stage, $STORE$ is also cleared by any already stored sets that are dominated by the new entrance. If, on the other hand, the considered set does not specify a linear DAP, then it spawns a number of entries for the list $EXPLORE$. As already explained, each of these entries is generated through (i) the removal of a maximal element from the "parent" set, and (ii) the further pruning of the resulting set in order to ensure that it specifies a correct DAP. The function that performs this pruning is listed in Algorithm 2, and it constitutes a convergent iterative computation that seeks to establish the correctness condition of Definition 4. Finally, the policy $\Delta(\bar{S}_r)$ that corresponds to the state set $\bar{S}_r$ that is returned by Algorithm 2, is tested for completeness, and the set $\bar{S}_r$ will enter the list $EXPLORE$ only if $\Delta(\bar{S}_r)$ is complete and not dominated by any of the current entries in $STORE$.

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

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

$$S_r(\Delta(\bar{S}_r)) \supseteq S_r(\Delta(\bar{S}'_r)) \iff \forall \mathbf{s}' \in \bar{S}'_r,\ \exists \mathbf{s} \in \bar{S}_r :\ \mathbf{s} \geq \mathbf{s}' \tag{20}$$

Similarly, the "completeness" test that is conducted by Algorithm 1 for the generated policies $\Delta(\bar{S}_r)$, can be based on Equation 11, where, however, the condition $\mathbf{s} \in S_r(\Delta(\bar{S}_r))$ is replaced by the equivalent condition $\exists \mathbf{s}' \in \bar{S}_r :\ \mathbf{s}' \geq \mathbf{s}$.

Proving the correctness of Algorithm 1 and the finiteness of its computation: Next we proceed to prove the correctness of Algorithm 1 and the finiteness of its computation. In order to derive these results, we shall start with a more technical proposition that will ensure that the policies Δ induced by the sets $\bar{S}_r$ that are stored in the lists $EXPLORE$ and $STORE$ of Algorithm 1, satisfy the "monotonicity" requirement of Definition 4. In order to establish this result, we must also specify more explicitly the sets $S_a(\Delta)$ that consist of all the admissible states by any such policy Δ. Hence, for the needs of the subsequent discussion, we shall define the set $S_a(\Delta(\bar{S}_r))$, for any DAP $\Delta(\bar{S}_r)$ that is induced by a set $\bar{S}_r$ generated in the search process of Algorithm 1, as follows:

$$S_a(\Delta(\bar{S}_r)) \equiv \left\{\mathbf{s} \in S_r : \exists \mathbf{s}' \in \bar{S}_r \text{ s.t. } \mathbf{s} \leq \mathbf{s}'\right\} \cup S_{\bar{r}s} \tag{21}$$

The first set of states in the right-hand-side of Equation 21 contains all the reachable states of the FSA $G(\Phi)$ that are dominated by some element of the policy-defining set $\bar{S}_r$, and therefore, are admissible by the corresponding policy $\Delta(\bar{S}_r)$ according to the logic of Algorithm 1 that was discussed in the previous part of this subsection. The second set is the set of all the safe but unreachable states of $G(\Phi)$. Since these states are unreachable in the original dynamics of the FSA $G(\Phi)$, they will never materialize when this FSA is controlled under the considered policy $\Delta(\bar{S}_r)$. But their inclusion in the set $S_a(\Delta(\bar{S}_r))$ is technically necessary in order to ensure that this set will contain the entire sub-lattices of $(\mathbb{Z}_0^+)^\xi$ that are dominated by each of its elements (since some elements of these sub-lattices might be unreachable).

With the sets $S_a(\Delta(\bar{S}_r))$ well-defined through Equation 21, now we can state the following result that is necessary for the eventual establishment of the correctness of Algorithm 1.

Proposition 4: The sets $S_a(\Delta(\bar{S}_r))$, corresponding to the policies $\Delta(\bar{S}_r)$ that are generated by Algorithm 1 through Equation 21, satisfy the "monotonicity" condition of Definition 4. □

[12]The efficiency of the DAP-linearity test that is defined by Equations 17-19, compared to the corresponding tests that can be induced from the material of [4], is due to the fact that this test does not try to construct a *parsimonious* linear separator for the considered DAP Δ, in the case that this DAP is indeed linear. On the other hand, the corresponding algorithms of [4] can be used, after the execution of Algorithm 1, for the construction of parsimonious linear separators for each DAP Δ that will be returned by this algorithm.

The proof of Proposition 4 can be based on a double induction, where the outer induction runs on the sets that enter list $EXPLORE$, and the inner induction is defined on the state sets that are pruned during the iterations that take place in Lines 5–7 of Algorithm 2. But due to the technical nature and the length of the arguments that are involved in this proof, we have opted to provide it in the electronic supplement of this paper.

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

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

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

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

On the other hand, if Δ^* does not admit a linear representation, and the algorithm must search for alternative policies, then, Line 11 of the algorithm, together with the definition of function $PRUNE$ in Algorithm 2, ensure that every set $\bar{S}_r$ that enters the lists $EXPLORE$ and (possibly) $STORE$, induces a policy $\Delta(\bar{S}_r)$ that satisfies the "correctness" condition of Definition 4. Also, Proposition 4 ensures that all these policies satisfy the "monotonicity" condition of Definition 4. Similarly, the first condition in the "if" statement in Line 6 of the algorithm, together with the first condition in the "if" statement in Line 12, ensure that a policy Δ will enter list $STORE$ only if it is linear and complete. Finally, the way that the sets $\bar{S}_r$ are generated by pruning their "parent" sets by one maximal element at a time, together with the set-inclusion tests that are performed in Lines 6 and 7 of the algorithm, ensure that the eventual content of list $STORE$ will be the maximal subsets of the set S_{rs} that pass the aforementioned tests.

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

Finally, the claimed non-emptiness of the $STORE$ list that is returned by Algorithm 1, follows from (i) the algorithm correctness, that was established in the previous paragraphs, and (ii) Proposition 3, which established that $\mathcal{L}(\Phi) \neq \emptyset$ for every D/C-RAS Φ.

C. Complexity considerations

From a computational standpoint, Algorithm 1 is a very expensive operation. The primary sources of this high complexity can be identified as follows:

1) *The representation and the processing of the considered policies Δ by means of their reachable subspaces $S_r(\Delta)$.* As shown in the electronic supplement of this paper, the procedures that are employed by Algorithm 1 for the construction and the manipulation of these sets, are of (a low-degree) polynomial worst-case computational complexity with respect to the cardinality of these sets. But it is also well-known that the cardinality of the reachable state space S_r of any given D/C-RAS Φ, that is the starting point of the algorithm computation and the "parent" set for all the generated sets $S_r(\Delta)$, is a super-polynomial function of the RAS size $|\Phi|$ [1]. Characteristically, in Section V.A of the electronic supplement of the paper, it is shown that, for any D/C-RAS Φ where every processing stage requires only a single unit from a single resource type for its execution, the cardinality of the corresponding state space S is given by $|S| = \Pi_{i=1}^{m} \frac{(C_i+|\Theta(R_i)|)!}{C_i!\cdot|\Theta(R_i)|!}$; in this expression, C_i is the capacity of resource R_i, for $i = 1, \ldots, m$, according to the notation introduced in Definition 1, and $|\Theta(R_i)|$ denotes the number of processing stages that are supported by resource R_i. Also, even though we do not have a closed-form expression for the cardinality of S for any arbitrary RAS Φ coming from the broader D/C-RAS class, it is expected that the growth of this quantity with respect to the defining elements of the considered RAS Φ will present a similar behavior.
2) *The dynamics of the search process that is conducted by Algorithm 1.* In the electronic supplement of the paper it is also shown that (i) the worst-case computational complexity of the main iteration of Algorithm 1 is $O(|S|^2)$, (ii) the total number of iterations for this algorithm is $O(|S| \cdot 2^{|S|})$, and (iii) the combination of these two results implies a worst-case computational complexity for Algorithm 1 of $O(|S|^3 \cdot 2^{|S|})$. This result highlights very clearly the practical intractability of Algorithm 1 and its underlying sources.

As is also the case with our earlier works of [4], [9], the computational challenge that is described in the first item of the above list, can be effectively addressed by the possibility to represent and process the considered policies Δ using only the maximal elements of their admissible subspaces. This possibility is established by the "monotonicity" property that characterizes the targeted policies, and it enables the reduction of the size of the employed representations by many orders of magnitude. The definition of the necessary data structures, and

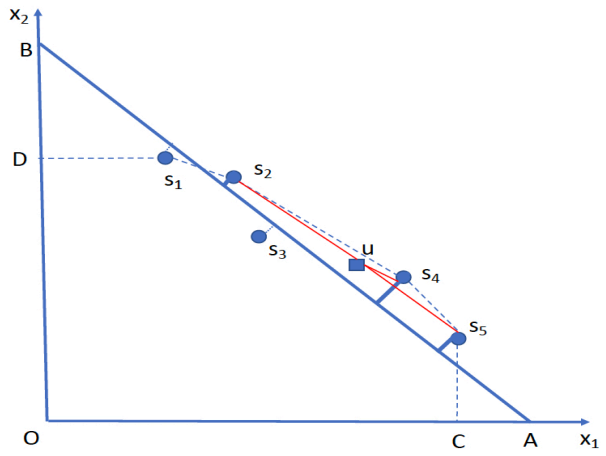

Fig. 2: A schematic demonstration of the "branching" logic that is employed by Algorithm 3.

the organization of the various operations that are performed by Algorithm 1 in a way that takes full advantage of these compressed representations, is a non-trivial issue; but given the more procedural nature of the corresponding material, and the imposed page limits for this document, we provide the corresponding developments in the electronic supplement of this manuscript.

When it comes to the second item of the above list, first we want to clarify that the exponential $2^{|S|}$ in the expression $O(|S| \cdot 2^{|S|})$ arises from the facts that (i) Algorithm 1 must search within the power-set of the set S_{rs} for those maximal subsets of this set that define complete, linear DAPs for the underlying RAS Φ, and (ii) the cardinality of this power-set is $2^{|S_{rs}|}$, which is $O(2^{|S|})$. In most practical cases, the number of subsets that will be explicitly considered by Algorithm 1 will be much smaller that $2^{|S_{rs}|}$, and it will depend on (a) the relative placement of the minimal boundary unsafe states with respect to the boundary of the convex hull of the set S_{rs}, and (b) the ability of Algorithm 1 to detect and separate the most problematic of these boundary unsafe states in an effective and expedient manner. But it will also remain a significant percentage of $2^{|S_{rs}|}$, and therefore, it will still grow exponentially with respect to $|S_{rs}|$. In this work, we have sought to address the computational challenge that is defined by this exponential dependency, by confining further the search process that is conducted by Algorithm 1. More specifically, acknowledging the intractability of the exhaustive search over the power-set of S_{rs} that is attempted by Algorithm 1, we propose to settle for a more partial search process that will seek the identification of a single linear DAP; but this new search process will be biased in a way that will guarantee a very high state-admissibility by the resulting policy. We provide the details of the resulting algorithm in the next subsection.

D. Efficient computation of near-optimal linear DAPs through a pertinent structuring of the underlying search process

We start this subsection by considering in further detail the "branching logic" of Algorithm 1 that will drive the generation and the processing of new candidate policies Δ', any time that a processed policy Δ fails the linearity test of Algorithm 1. We remind the reader that in the basic definition of Algorithm 1 (c.f. Section IV-B), each of the aforementioned policies Δ' is generated from the original policy Δ by removing a maximal state $\mathbf{s}'$ from the corresponding set $S_r(\Delta)$, and further processing the resulting state set through the $PRUNE$ function of Algorithm 2 in order to establish the correctness of the new DAP.

However, as pointed out in the previous subsection, the spawning of a new policy Δ' for every single element of the set $\bar{S}_r(\Delta)$ implies a very rapid expansion of the underlying search space and a prohibitively high computational complexity for most instantiations of the considered RAS. Hence, it is imperative to try to control the extent of "branching" that will take place at each node of the directed acyclic graph (DAG) that represents the underlying search process. At the same time, this restriction must be performed in a way that will not compromise considerably the quality of the DAPs that will be obtained by the resulting algorithm. In particular, we would like the policies that are returned by the modified algorithm to present a state-admissibility level that is pretty close to the state-admissibility level that is attained by the policies in the target set $\bar{\mathcal{L}}(\Phi)$; in the following, we shall characterize those linear DAPs that attain this objective as *"near-optimal"*. Finally, we would like to obtain these near-optimal linear DAPs in an expedient manner.

The objectives that were stated in the previous paragraph, can be addressed in a pertinent manner when considering the reason for the nonlinearity of any processed DAP Δ; i.e., the presence of elements of the set $\bar{S}_r^b(\Delta)$ in the convex hull of the set $S_r(\Delta)$. This situation is schematically demonstrated through the 2-dimensional example of Figure 2. The states $\mathbf{s}_i, i = 1, \ldots, 5$, depicted in Figure 2 denote the maximal states of the set $S_r(\Delta)$ in the considered situation, while state $\mathbf{u}$ is a minimal boundary Δ-inadmissible state that is in the convex hull of $S_r(\Delta)$.

From the geometry that is depicted in Figure 2, it is easy to see that the selection of the elements of the set $\bar{S}_r(\Delta)$ for the generation of the new policies Δ' can be restricted to the set $\bar{S}'_r(\Delta) \subseteq \bar{S}_r(\Delta)$ that contains those elements of $\bar{S}_r(\Delta)$ that constitute vertices – or, more formally, extreme points – of the convex hull of the set $S_r(\Delta)$, without compromising the correctness of Algorithm 1. The possibility for this restriction is formally stated and justified in the electronic supplement of the paper. In fact, the corresponding part of the electronic supplement shows that the set $\bar{S}'_r(\Delta)$ can be thinned even further to another set $\bar{S}''_r(\Delta) \subseteq \bar{S}'_r(\Delta)$, which will lead to a further reduction of the "branching" that is effected by the underlying search process.

However, in spite of the reduction of the extent of "branching" that is attained by the remarks that were provided in the previous paragraph, the computational requirements of the corresponding implementation of Algorithm 1 remain very high in terms of, both, time and memory. Hence, next we present a more "heuristic" algorithm that can be perceived as an execution of Algorithm 1 over a single thread of the underlying search process for some linear DAP. Central in the

Algorithm 3 A heuristic algorithm for computing a near-optimal linear DAP

Input: DFSA $G(\Phi)$
Output: $\bar{S}_r$

/* INITIALIZE */
1: $\bar{S}_r := \bar{S}_{rs}$;
2: $S^b_{\bar{r}} := \{\mathbf{s} \in S : \big(\exists \mathbf{s}' \in S,\ \mathbf{s}'' \in \bar{S}_r,\ e \in \Gamma(\mathbf{s}')$ s.t. $f(s', e) = \mathbf{s}$ AND $\mathbf{s}' \leq \mathbf{s}''\big)$ AND $\big(\nexists\ \mathbf{s}''' \in \bar{S}_r$ s.t. $\mathbf{s} \leq \mathbf{s}'''\big)\}$;
3: $\bar{S}^b_{\bar{r}} := \{\mathbf{s} \in S^b_{\bar{r}} : \nexists \mathbf{s}' \in S^b_{\bar{r}}$ s.t. $\mathbf{s}' \neq \mathbf{s}$ AND $\mathbf{s}' \leq \mathbf{s}\}$;
/* MAIN ITERATION */
4: **while** $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ not linearly separable **do**
5: Select $\mathbf{u} \in \bar{S}^b_{\bar{r}}$ s.t. $(\bar{S}_r, \mathbf{u})$ not linearly separable;
6: $\hat{S}(\mathbf{u}) := \{\mathbf{s} \in \bar{S}_r : I^*_{\mathbf{s}} > 0\}$, where $I^*_{\mathbf{s}}$ is obtained through the solution of the linear program defined by Equations 22 – 24;
7: **for all** $\mathbf{s} \in \hat{S}(\mathbf{u})$, in increasing order of $||\mathbf{s} - \mathbf{u}||_2$ **do**
8: $\bar{S}_r := PRUNE(\bar{S}_r,\ \mathbf{s},\ G(\Phi))$;
9: **if** $(\bar{S}_r, \mathbf{u})$ linearly separable **then**
10: Break from this for-loop;
11: **end if**
12: **end for**
13: Compute the sets $S^b_{\bar{r}}$ and $\bar{S}^b_{\bar{r}}$ for the newly obtained set $\bar{S}_r$, as in Steps 2 and 3 above;
14: **end while**
/* TERMINATE */
15: **return** $\bar{S}_r$;

design of this heuristic algorithm is the specification of the selection process of the maximal states to be removed from the various sets $S_r(\Delta)$ that are generated during the execution of the algorithm, in a way that the obtained linear DAP in near-optimal.[13]

A formal statement of the heuristic algorithm that is proposed in this subsection, is provided in Algorithm 3. Similar to Algorithm 1, when applied on a given D/C-RAS Φ, Algorithm 3 first sets $\bar{S}_r := \bar{S}_{rs}$, and tests whether this set is linearly separable from the corresponding set of minimal boundary reachable unsafe states, i.e., the corresponding set $\bar{S}^b_{\bar{r}}$. If this is true, Algorithm 3 returns the current set $\bar{S}_r$ as the set of the maximal states that are admitted by the computed DAP Δ, and therefore, the returned policy Δ is the maximally permissive DAP Δ^*.

If, on the other hand, $\Delta^* \notin \mathcal{L}(\Phi)$, then $\bar{S}^b_{\bar{r}} \cap conv(S_r) \neq \emptyset$.[14] Algorithm 3 selects arbitrarily a state $\mathbf{u}$ from this set, and formulates and solves the following LP for some $\epsilon \to 0^+$:

$$\min_{\substack{\mathbf{a} \geq \mathbf{0},\ b \geq 0,\\ \forall \mathbf{s} \in \bar{S}_r, I_{\mathbf{s}} \geq 0}} \sum_{\mathbf{s} \in \bar{S}_r} I_{\mathbf{s}} \tag{22}$$

s.t.

$$\mathbf{a}^T \cdot \mathbf{s} \ \leq \ b + I_{\mathbf{s}}\ ,\ \forall \mathbf{s} \in \bar{S}_r \tag{23}$$

$$\mathbf{a}^T \cdot \mathbf{u} \ \geq \ b + \epsilon \tag{24}$$

Let $\langle \mathbf{a}^*;\ b^*;\ I^*_{\mathbf{s}},\ \mathbf{s} \in \bar{S}_r \rangle$ denote the obtained optimal solution for the above LP. Then, the set $\hat{S}(\mathbf{u})$ containing the elements of $\bar{S}_r$ that will constitute the "seeds" for the spawned policies Δ', is defined as follows:

$$\hat{S}(\mathbf{u}) := \big\{\mathbf{s} \in \bar{S}_r :\ I^*_{\mathbf{s}} > 0\big\} \tag{25}$$

More specifically, Algorithm 3 will order the elements of the set $\hat{S}(\mathbf{u})$ in increasing Euclidean distance from the considered state $\mathbf{u}$, and it will keep removing these states from the set $\bar{S}_r$, one at a time, until it obtains a subset of $\bar{S}_r$, let's say $\bar{S}'_r$, that induces a correct DAP Δ' and is linearly separable from the considered state $\mathbf{u}$. At this point, the algorithm computes the minimal boundary inadmissible states for the DAP Δ', and uses this information in order to assess its linearity. If Δ' is found to be linear, then the algorithm terminates returning the newly obtained set $\bar{S}'_r$ as the representation of this DAP. Otherwise, the algorithm picks a minimal boundary inadmissible state $\mathbf{u}' \in \bar{S}^b_{\bar{r}}(\Delta') \cap conv(S_r(\Delta'))$ and performs another iteration along the lines that were described in the previous paragraphs, using the pair $(\bar{S}'_r, \mathbf{u}')$.

In more conceptual terms, the LP of Equations 22 – 24 that is employed by Algorithm 3 in order to compute the set $\hat{S}(\mathbf{u})$, essentially computes a hyperplane that is defined by the pair $(\mathbf{a}, b)$ and minimizes the total violation of the inequalities

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

that should be satisfied by this hyperplane, if it acted as a separatrix of the set $\bar{S}_r$ from the selected state $\mathbf{u}$. A positive value for the variable $I^*_{\mathbf{s}}$ corresponding to state $\mathbf{s} \in \bar{S}_r$, indicates a state $\mathbf{s}$ that is not easily separable from the considered inadmissible state $\mathbf{u}$. Therefore, this state should be in the set $\hat{S}(\mathbf{u})$ consisting of the elements of $\bar{S}_r$ that will constitute the "seeds" for the newly spawned policies Δ'; more specifically, these policies will be obtained through the iterative elimination of the elements of $\hat{S}(\mathbf{u})$ from the set $\bar{S}_r$, as discussed in the previous paragraph.

Example: The "branching" logic that was defined in the previous paragraphs is exemplified in Figure 2 by means of the straight line AB. Assuming that this line constitutes an optimal solution for the corresponding LP of Equations 22 – 24, we can see that, in the depicted case, the states that will constitute "seed" states for the generation of the new DAPs Δ', are the states $\mathbf{s}_2$, $\mathbf{s}_4$ and $\mathbf{s}_5$, i.e., the states $\mathbf{s}_i$ that are above the line AB. These states will be processed by the 'FOR'-loop of Algorithm 3 in the sequence $\langle \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_2 \rangle$, which is determined by the Euclidean distances of these states from the depicted state $\mathbf{u}$; these distances are provided by the red lines in Figure 2. □

The previous example demonstrates schematically the ability of the "branching logic" that is employed by Algorithm 3 to generate a pertinent subset of the set $\bar{S}_r(\Delta)$ that will advance the conducted search for a good linear DAP. Furthermore, in Section V of the electronic supplement it is shown that

[13] We also want to notice that, even though it is not scalable to larger D/C-RAS instantiations, the detailed implementation of Algorithm 1 that is presented in the electronic supplement of the paper is still very useful for this study, since its application on some smaller RAS configurations (i) enables an experimental investigation of the structure of the set $\mathcal{L}(\Phi)$, and (ii) provides a benchmark for the assessment of the heuristic algorithm; both of these possibilities are pursued in Section V.

[14] We remind the reader that the state set $\bar{S}^b_{\bar{r}}(\Delta) \cap conv(S_r(\Delta))$ will be among the outcomes of the linearity test that is defined by Equations 17-19, for those policies Δ that turn out to be nonlinear.

the worst-case computational complexity of Algorithm 3 is $O(|S|^{1+a} \cdot \xi)$, for some $a \in (1.25, 2.5)$, which is substantially lower than the worst-case computational complexity of Algorithm 1. Indeed, extensive numerical experimentation that is reported in Section V, will show that by using the aforementioned "branching" scheme, Algorithm 3 can consistently identify near-optimal linear DAPs, within a very reasonable running time, even for some very large D/C-RAS configurations.

Finally, as it is also the case with other heuristic algorithms that involve some arbitrary choice in their defining logic, the quality of the DAP that will be obtained by means of Algorithm 3 for any given D/C-RAS configuration, can be further enhanced through the repetitive execution of the algorithm, either in a sequential or in a parallel mode, while randomizing the selection of the state $\mathbf{u}$ that will be employed during the various iterations of each of these executions. The finally implemented DAP will be selected among the DAPs that will be returned by each of these executions.

V. Computational Results

In this section, we present the results from a number of numerical experiments that intend to (i) assess the scalability of Algorithms 1 and 3, (ii) identify certain factors that will impact this scalability, and also (iii) get some insights regarding the structure of the target policy spaces $\bar{\mathcal{L}}(\Phi)$. The considered algorithms were coded using the JAVA programming language, where the formulated LPs were solved using the CPLEX 12.8 package, and all the experiments were executed on a Windows 10 computational platform with an Intel Core i7, 2.8 GHz processor, and 64GB memory. The results obtained from these experiments are organized in Tables I, II and III.

More specifically, Table I reports the execution of Algorithms 1 and 3 on 15 Conjunctive (C-)RAS configurations on which Algorithm 1 was able to execute completely within a provided time budget of 48 hours. The table is organized in four major sections, with the first section providing some information on the size and the structure of each configuration. More specifically, column "Res" reports the number of the resource types involved, column "Cap" reports the (common) capacity level for these resource types, and column "Proc." reports the number of processing stages for each process type that is uspported by the corresponding RAS. The second section of Table I reports the size of the reachable safe and unsafe subspaces of the 15 configurations, and also the cardinalities of the sets of the maximal safe states and the minimal boundary unsafe states, that are the primary sets eventually used by the considered algorithms. The third section of Table I reports (i) the number of the maximal linear DAPs identified by Algorithm 1 for each configuration, (ii) the number of the states that are in the admissible spaces of all the returned DAPs, (iii) the minimal and the maximal state-admissibility levels among these DAPs, and finally (iv) the execution time of the algorithm. The fourth section of Table I reports the execution of Algorithm 3 on the considered configurations. More specifically, this section reports (i) the execution time of the algorithm, (ii) the number of the admissible states by the returned DAP Δ, and (iii) the ratio of the cardinality of the Δ-admissible subspace to the maximal cardinality among the subspaces that are admitted by the maximal linear DAPs.

Table II reports the execution of Algorithms 1 and 3 on 15 Conjunctive (C-)RAS configurations on which Algorithm 1 was not able to execute completely within the provided time budget of 48 hours. The structure of this Table is similar to that of Table I. However, the numbers reported in the third section of this table refer to the policies that were obtained by Algorithm 1within the 48 hours of its execution, and not to the entire sets of the maximal linear DAPs for the corresponding configurations. Also, recognizing that Algorithm 1 might have failed to identify the maximal linear DAP with the maximum possible cardinality for its admissible state space, in the fourth section of this table we take a rather conservative approach in the evaluation of the relative performance of the policy Δ that is returned by Algorithm 3; more specifically, the column "ratio" in this section reports the ratio $|S_r(\Delta)|/|S_{rs}|$. Since $|S_{rs}|$ is an upper bound of the maximal cardinality among the subspaces that are admissible by the maximal linear DAPs, the algorithm performance index that is reported in column "ratio" is an under-estimate of the actual performance that is attained by the algorithm.

Table III reports primarily the performance of Algorithm 3 when applied on some larger C-RAS configurations that – as expected – will challenge the execution of Algorithm 1. The primary scaling of the employed RAS configurations that has taken place in this case, is in terms of the number of processes that are supported by each of these configurations; the corresponding notation '$n \times m$' in column "proc" implies n processes with each process possessing m processing stages. In the second section of this table, it can also be seen that the aforementioned scaling results in significantly larger state (sub-)spaces compared to the corresponding sizes that are reported in Tables I and II. The third section of Table III is similar to the fourth section of Table II; in particular, this section focuses on the execution of Algorithm 3 on the corresponding configurations and the quality of the returned policies. Also, as in the case of Table II, the column "ratio" reports the ratio $|S_r(\Delta)|/|S_{rs}|$, for each considered configuration. Finally, the fourth section of Table III reports the times that it took Algorithm 1 to identify the first linear DAP when applied on the considered configurations, and an evaluation of these DAPs that are obtained by Algortihm 1 along lines similar to those used in the third section for the evaluation of the DAPs that are returned by Algorithm 3. The indication "no policy" for the last two configurations, in this section, implies that Algorithm 1 was not able to identify a linear DAP within the allowed 48 hours for its execution.

Next we provide some remarks that highlight certain aspects of the results that are reported in Tables I–III, and communicate our current understanding of these results.

We start this discussion by noticing that, for those configurations that are amenable to a complete enumeration of the corresponding policy spaces $\bar{\mathcal{L}}(\Phi)$ by Algorithm 1, these policy spaces have turned out to be of quite low cardinality. Also, for these configurations, the maximal linear DAPs provide an extensive coverage of the corresponding sets S_{rs} that are admitted by the maximally permissive DAP Δ^*. Finally,

TABLE I: Results from the execution of Algorithms 1 and 3 on some RAS configurations for which Algorithm 1 completes with 48 hours.

#	Configuration			State Space				Algorithm 1					Algorithm 3		
	Res	Cap	Proc.	Safe	Uns.	Max. Safe	Min. Uns.	Cnt	Com	Min	Max	Time (sec.)	Time (sec.)	$\|S_r\|$	ratio
1	8	4	$\{8,8,8\}$	109	1379	57	35	2	94	98	105	26	0.448	105	1
2	7	4	$\{8,8,9\}$	144	961	53	32	2	122	125	141	42	0.414	141	1
3	7	4	$\{7,8,10\}$	128	2215	69	41	2	114	118	124	123	0.525	124	1
4	8	4	$\{7,7,8\}$	87	490	42	29	2	75	77	85	2	0.352	85	1
5	6	4	$\{6,8,8\}$	99	1371	49	35	2	84	88	95	98	0.461	95	1
6	8	4	$\{8,8,10\}$	99	1218	61	44	2	92	95	96	3	0.554	95	0.989
7	7	4	$\{8,8,8\}$	101	1461	52	28	2	87	91	97	23	0.448	97	1
8	6	4	$\{7,8,8\}$	139	1688	76	23	3	119	123	135	1062	0.459	135	1
9	7	4	$\{6,7,8\}$	105	910	59	31	3	83	87	101	137	0.425	101	1
10	6	4	$\{6,7,7\}$	100	906	55	31	3	78	82	96	31	0.369	96	1
11	7	4	$\{7,7,9\}$	104	1686	55	37	2	90	94	100	2	0.448	100	1
12	6	4	$\{6,7,8\}$	91	772	44	31	2	77	81	87	109	0.405	87	1
13	6	4	$\{7,7,8\}$	121	1481	66	21	3	101	105	117	1163	0.419	117	1
14	7	4	$\{7,8,8\}$	89	962	44	26	2	77	79	87	3	0.328	87	1
15	8	4	$\{7,8,10\}$	94	1214	57	44	2	87	90	91	15	0.456	90	0.989

TABLE II: Results from the execution of Algorithms 1 and 3 on some RAS configurations for which Algorithm 1 does not complete with 48 hours.

#	Configuration			State Space				Algorithm 1				Algorithm 3			
	Res	Cap	Proc.	Safe	Uns.	Max. Safe	Min. Uns.	Cnt	Com	Min	Max	Time (sec.)	$\|S_r\|$	ratio
1	10	4	$\{8,8,8\}$	593	2756	81	32	3	395	399	527	1.106	542	0.913
2	9	4	$\{7,7,8\}$	175	885	46	20	2	122	126	165	0.426	163	0.931
3	7	4	$\{6,7,7\}$	455	2160	53	25	2	268	269	410	0.758	409	0.898
4	10	4	$\{7,8,8\}$	569	2744	80	32	2	384	385	508	1.015	519	0.912
5	9	4	$\{7,8,8\}$	554	2550	75	31	2	444	445	507	0.998	506	0.913
6	7	4	$\{8,8,8\}$	602	2751	79	32	2	491	492	553	1.342	548	0.910
7	7	4	$\{7,8,8\}$	634	3087	82	32	2	486	487	573	1.277	574	0.905
8	7	4	$\{7,8,8\}$	1181	6483	112	38	2	804	805	1073	2.191	1059	0.896
9	7	4	$\{6,7,8\}$	1031	5175	98	32	3	723	724	927	1.823	912	0.884
10	8	4	$\{6,8,8\}$	610	3075	81	32	2	501	502	554	1.143	551	0.903
11	8	4	$\{6,7,8\}$	577	3018	72	32	2	382	383	520	1.056	520	0.901
12	9	4	$\{6,7,7\}$	1112	4294	178	23	3	829	831	1043	1.128	1064	0.956
13	8	4	$\{6,6,7\}$	453	1925	56	24	3	269	272	397	0.759	408	0.901
14	7	4	$\{5,7,8\}$	444	1749	53	23	3	328	331	396	2.345	396	0.892
15	7	4	$\{5,7,9\}$	1682	13608	194	36	3	1173	1174	1499	3.341	1473	0.875

as revealed by the columns "Com" in Tables I and II, there is also a very large intersection among the subspaces that are admitted by the maximal linear DAPs.

Another interesting remark can be based on the juxtaposition of the first two sections of Table I with their counterparts in Table II. In particular, the RAS configurations, Φ, that are involved in the experiments that are reported in these two tables, are quite comparable in terms of the corresponding RAS sizes, $|\Phi|$. Yet, the running times of Algorithm 1 on the configurations of Table I will not exceed more than a few minutes, while, for all of the configurations of Table II, Algorithm 1 will fail to complete in 48 hours. A plausible explanation for this dramatic difference in the performance of Algorithm 1 with respect to these two RAS sets, might be provided by the ratio (Max Safe / Safe) for the involved RAS instances: this ratio is significantly higher for the RAS instances of Table I, and this fact might imply that the searches that are conducted by Algorithm 1 over the subsets of the set S_{rs} for the RAS instances of Table I, might be more "shallow", and therefore, less expensive, than the corresponding searches that are conducted by this algorithm for the RAS instances of Table II. More generally, however, this difference in the execution times of Algorithm 1 with respect to the RAS instances in Tables I and II implies that it is not possible to characterize and predict the (in-)tractability of this algorithm merely through the structural elements that define any given D/C-RAS Φ according to Definition 1.

An additional observation that pertains to the execution of Algorithm 1 on the RAS configurations of Table II, is that Algorithm 1 made all of its progress, in terms of obtaining some very good DAPs, within less than 24 hours, and it spent the remaining time of its execution essentially "validating" the quality of the policies that were derived in the first part of its computation. Such a behavior is typical of the broader class of the b&b algorithms that are used on some very hard combinatorial optimization problems, and, from a more practical standpoint, it implies that, in many cases, these algorithms might be able to turn out some good-quality results even if they are terminated prematurely.

When it comes to Algorithm 3, all three Tables I–III demonstrate very clearly the ability of this algorithm to return DAPs that are very competitive in terms of the size of their admissible state spaces, in short computational times. In particular, Table III demonstrates that this capability extends to RAS with quite sizable state spaces. It is also interesting to notice that in the cases that are reported in Table III, Algorithm 3 is able to come up with a linear DAP that has a higher state admissibility than the linear DAP that is computed

TABLE III: Results from the execution of Algorithms 1 and 3 on some larger RAS configurations.

#	Configuration			State Space				Algorithm 3			Algorithm 1		
	Res	Cap	Proc.	Safe	Uns.	Max. Safe	Min. Uns.	Time (minutes)	$\|S_r\|$	ratio	Time (hours)	$\|S_r\|$	ratio
1	8	4	7×8	21099	906478	1271	225	2.364	20480	0.971	1.09	17884	0.85
2	9	4	8×5	42571	69016	6148	188	0.573	42466	0.997	0.326	42448	0.997
3	9	4	6×4	94431	72238	1694	71	29.396	85608	0.906	11.53	85196	0.90
4	11	4	6×4	115766	28510	1401	12	1.336	114731	0.991	0.608	112891	0.975
5	8	4	6×8	10913	96361	766	163	0.660	10499	0.962	0.112	10164	0.93
6	10	4	8×3	104550	49620	3840	67	3.805	103569	0.991	1.52	103753	0.99
7	8	4	8×8	34695	1773193	2860	427	8.153	33363	0.961	2.59	32376	0.93
8	9	4	8×8	118470	1253425	4335	612	6.962	118150	0.997	0.743	118158	0.98
9	11	4	11×8	53080	1410311	7716	1046	21.494	51894	0.977	8.62	51835	0.98
10	4	4	6×4	196021	11451	13998	15	0.376	195983	0.999	2.84	195878	0.999
11	11	4	13×3	757699	700781	36267	89	50.512	753652	0.995	no policy		
12	8	4	13×5	396931	1146292	29545	559	43.661	393966	0.993	no policy		

by Algorithm 1, and this DAP is also obtained in a much shorter time.

The superior quality of the policy that is obtained by Algorithm 3 testifies to the pertinence of the "branching logic" that is employed by this algorithm. On the other hand, the expediency of the corresponding computation is the result of (i) the aforementioned quality of the "branching logic" and the pertinent guidance that it provides to the underlying search process, but also (ii) the fact that the implementation of Algorithm 3 was further streamlined by considering the fact that this algorithm essentially executes a single thread of the broader search that is conducted by Algorithm 1. More specifically, the adherence to a single path in the search that is conducted by Algorithm 3, enables a much lighter memory "footprint" for this algorithm, and, therefore, the execution of the algorithm on pretty large configurations without resorting to the secondary memory. This remark also indicates that, at the very end, a considerable "bottleneck" in many of the more difficult executions of the algorithms that are presented in this work, might not be the required computational time per se, but an explosion of the memory requirements involved that necessitates the usage of the secondary memory and slows down the algorithm very dramatically. Furthermore, it might be possible to resolve this sort of "bottlenecks" through a more pertinent definition of the data structures that are employed by the algorithms, and a closer attention to issues like the "memory leakages" that might occur during their execution.

Finally, we also used the data that are provided in Tables I–III in order to compute the correlation – or, in more technical, statistical terms, the R-value [35] – between (i) the number of states that are pruned from the sets S_{rs} that are processed by Algorithm 3 when applied on the considered RAS configurations, and (ii) the corresponding running times of this algorithm. The obtained R-value was $R \approx 0.8$, which, according to [35], is an indicator of "fairly strong positive linear correlation" between the two considered quantities. This result substantiates, in a strong quantitative sense, our claims in the earlier parts of this paper that the empirical computational complexity of the presented algorithms is determined significantly by the extent of the state-pruning that must be performed during the computation of the target policies.

VI. Conclusions

The first part of this paper introduced the new class of maximal linear DAPs for the sequential RAS that have been studied in [1], providing a complete, formal characterization of this class of policies, and enabling, thus, a more systematic measure for assessing the various heuristic methodologies that have sought to provide linear DAPs for those RAS. The second part of the work focused on the more operational problems of the computation and the approximation of the target set of the maximal linear DAPs, for any given (D/C-)RAS Φ. More specifically, this part of the work first presented a b&b algorithm that, in principle, can provide a complete enumeration of the corresponding set of the maximal linear DAPs, $\bar{\mathcal{L}}(\Phi)$, but acknowledging the very high computational complexity of this algorithm, subsequently it developed a heuristic algorithm that is motivated by the original algorithm and can return a high-quality linear DAP for the considered RAS Φ in a consistent and expedient manner. Finally, the numerical experimentation that accompanies the more theoretical developments of the paper, has demonstrated the efficacy of the aforementioned results, and the ability of the heuristic algorithm to generate correct linear DAPs that are very close to the elements of the target set $\bar{\mathcal{L}}(\Phi)$ in terms of the state admissibility that is attained by these policies.

From a methodological standpoint, our heuristic algorithm is able to achieve the aforementioned performance by taking an extensive view of the "geometry" of the underlying subspaces, and not focusing only on the structural and the behavioral characteristics of the controlled RAS that are usually traced by some of the most popular methodologies for the development of the sought linear DAPs. In this way, this new algorithm complements the earlier efforts that have pursued the development of linear DAPs for the RAS classes considered in this work, in a substantial manner. It will be interesting to assess more systematically the potential gains in state admissibility that are attained by this new approach, and the underlying trade-off between the computational intensity of this new method and the quality of the returned DAPs. This task will be part of our future investigations in this area.

References

[1] S. Reveliotis, "Logical Control of Complex Resource Allocation Systems," *NOW Series on Foundations and Trends in Systems and Control*, vol. 4, pp. 1–224, 2017.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems (2nd ed.)*. NY, NY: Springer, 2008.

[3] S. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Trans. on Automatic Control*, vol. 55, pp. 1646–1651, 2010.

[4] A. Nazeem, S. Reveliotis, Y. Wang, and S. Lafortune, "Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the linear case," *IEEE Trans. on Automatic Control*, vol. 56, pp. 1818–1833, 2011.

[5] Y. F. Chen and Z. W. Li, "Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems," *Automatica*, vol. 47, pp. 1028–1034, 2011.

[6] R. Cordone and L. Piroddi, "Parsimonious monitor control of petri net models of flexible manufacturing systems," *IEEE Trans. on SMC: Systems*, vol. 43, pp. 215–221, 2013.

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

[8] R. Cordone, A. Nazeem, L. Piroddi, and S. Reveliotis, "Designing optimal deadlock avoidance policies for sequential resource allocation systems through classification theory: existence results and customized algorithms," *IEEE Trans. Autom. Control*, vol. 58, pp. 2772–2787, 2013.

[9] A. Nazeem and S. Reveliotis, "Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the non-linear case," *IEEE Trans. on Automatic Control*, vol. 57, pp. 1670–1684, 2012.

[10] ——, "A practical approach for maximally permissive liveness-enforcing supervision of complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. 8, pp. 766–779, 2011.

[11] ——, "Efficient enumeration of minimal unsafe states in complex resource allocation systems," *IEEE Trans. on Automation Science & Engineering*, vol. 11, pp. 111–124, 2014.

[12] Z. Fei, S. Reveliotis, S. Miremadi, and K. Akesson, "A BDD-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. 12, pp. 990–1006, 2015.

[13] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. on R&A*, vol. 11, pp. 173–184, 1995.

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

[15] M. Lawley, S. Reveliotis, and P. Ferreira, "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," *IEEE Trans. on Robotics & Automation*, vol. 14, pp. 796–809, 1998.

[16] J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Trans. on Automatic Control*, vol. 46, pp. 1572–1583, 2001.

[17] Z. Li and M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 34, no. 1, pp. 38–51, 2004.

[18] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A Petri net structure-based deadlock prevention solution for sequential resource allocation systems," in *Proceedings of the ICRA 2005*. IEEE, 2005, pp. 271–277.

[19] M. Ibrahim and S. Reveliotis, "Maximal linear deadlock avoidance policies for complex resource allocation systems," in *Proc. of the 57th IEEE Conference on Decision and Control*. IEEE, 2018, pp. –.

[20] X. Cong, A. Wang, Y. Chen, N. Wu, T. Qu, M. Khalgui, and Z. Li, "Most permissive liveness-enforcing Petri net supervision for discrete event systems via linear monitors," *ISA Trans.*, vol. 92, pp. 145–154, 2019.

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

[22] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *Proceedings of the 1992 IEEE Intl. Conference on Systems, Man and Cybernetics*. IEEE, 1992, pp. 974–979.

[23] K. Yamalidou, J. Moody, M. D. Lemmon, and P. J. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, pp. 15–28, 1996.

[24] M. V. Iordache and P. J. Antsaklis, "Synthesis of deadlock prevention supervisors using Petri nets," *IEEE Trans. on Robotics & Automation*, vol. 18, pp. 59–68, 2002.

[25] ——, "Design of t-liveness enforcing supervisors in petri nets," *IEEE Trans. on Automatic Control*, vol. 48, pp. 1962–1974, 2003.

[26] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows (2nd ed.)*. NY, NY: John Wiley & Sons, 1990.

[27] M. Ibrahim and S. Reveliotis, "Throughput maximization of capacitated re-entrant lines through fluid relaxation," in *Proc. of ACC'18*. APS, 2018, pp. –.

[28] ——, "Throughput maximization of capacitated re-entrant lines through fluid relaxation," *IEEE Trans. on Automation Science and Engineering*, vol. 16, pp. 792–810, 2019.

[29] G. Weiss, "On the optimal draining of re-entrant fluid lines," Georgia Tech and Technion, Tech. Rep., 1994.

[30] J. G. Dai and G. Weiss, "Stability and instability of fluid models for certain re-entrant lines," *Math. of Op. Res.*, vol. 21, pp. 115–134, 1996.

[31] S. Meyn, *Control Techniques for Complex Networks*. Cambridge, UK: Cambridge University Press, 2008.

[32] D. Bertsimas, E. Nasrabadi, and I. C. Paschalidis, "Robust fluid processing networks," *IEEE Trans. on Automatic Control*, vol. 60, pp. 715–728, 2015.

[33] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algortihms and Complexity*. Mineola, NY: Dover, 1998.

[34] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. NY, NY: Wiley-Interscience, 1998.

[35] E. R. Dougherty, *Probability and Statistics for the Engineering, Computing and Physical Sciences*. Englewood Cliffs, NJ: Prentice Hall, 1990.

Michael Ibrahim received the B.Sc. and M.Sc. degrees from the Cairo University, Faculty of Engineering, Department of Computer Engineering, in 2012 and 2015, and the Ph.D. degree in Industrial Engineering from the School of Industrial & Systems Engineering at the Georgia Institute of Technology, in 2019.

Currently, he is an Assistant Professor in the Department of Computer Engineering at the Cairo University.

His research interests include discrete event systems, operations research, and machine learning.

Spyros Reveliotis received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, the M.Sc. degree in computer systems engineering from Northeastern University in Boston, and the Ph.D. degree in industrial engineering from the University of Illinois at Urbana-Champaign.

He is a Professor with the School of Industrial and Systems Engineering, Georgia Institute of Technology, in Atlanta, GA. His main research interests are in discrete-event systems theory and its applications.

Dr. Reveliotis is an IEEE Fellow and a member of INFORMS. He has served on the editorial boards of many journals and conferences in his areas of interest; some of his most recent positions are Senior Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, Associate Editor for the Journal of Discrete Event Dynamic Systems, and the Editor-in-Chief of the Editorial Board at the IEEE Conference on Automation Science and Engineering (CASE). He has also served as the Program Chair for the 2009 IEEE CASE Conference, and the General Co-Chair for the 2014 edition of the same conference. Finally, he has been a recipient of a number of awards, including the 2014 Best Paper Award of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

Ahmed Nazeem is an Operations Research Scientist at Facebook in Menlo Park, CA. In his current position, he develops analytical models for enterprise services and he conducts applied research in machine learning and optimization. Dr. Nazeem received his Ph.D. in Industrial Engineering from Georgia Tech. He has his work published in several academic journals and conferences, and he has also filed multiple patent disclosures. He has been a recipient of a number of awards, including the CASE 2011 Kayamori Best Paper Award, and the 2014 Best Paper Award of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING