Designing optimal deadlock avoidance policies for sequential resource allocation systems through classification theory: existence results and customized algorithms

Roberto Cordone, Ahmed Nazeem, Luigi Piroddi and Spyros Reveliotis

Abstract

A recent line of work has sought the implementation of the maximally permissive deadlock avoidance policy (DAP) for a broad class of complex resource allocation systems (RAS) as a classifier that gives effective and parsimonious representation to the dichotomy of the underlying behavioral space into the admissible and inadmissible subspaces defined by that policy. The work presented in this paper complements the past developments in this area by providing (i) succinct conditions regarding the possibility of expressing the aforementioned classifier as a set of linear inequalities in the RAS state variables, and (ii) an efficient customized algorithm for the synthesis of pertinent non-linear classifiers that implement the target DAP with minimum run-time computational overhead, in the case that a linear-classifier-based representation of this policy is not possible.

I. Introduction

The work presented in this paper is part of an ongoing research endeavor to deploy effective and computationally efficient realizations of the maximally permissive deadlock avoidance policy (DAP) for various classes of resource allocation systems (RAS) [1], like those arising in the context of flexible manufacturing systems [2], [3], some traffic systems [4], [5], [6], and the multi-threaded computer programming that is currently promoted in the context of the multi-core computer architectures [7], [8]. Indicative examples of this research line can be found in [9], [10], [11], [12], [13] and [14]. From a methodological standpoint, it is known that the computation of the maximally permissive DAP can be an NP-Hard task in the context of the aforementioned RAS classes [15], [16].[1] Yet, the ongoing

R. Cordone is with the Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy. roberto.cordone@unimi.it

A. Nazeem is with United Airlines, USA. ahmed.nazeem@united.com

L. Piroddi is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy. piroddi@elet.polimi.it

S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology, USA. spyros@isye.gatech.edu

A. Nazeem and S. Reveliotis were partially supported by the NSF grant CMMI-0928231.

[1]The notion of the maximally permissive DAP and all the other technical concepts that are involved in the development of the results presented in this paper are systematically introduced in the subsequent sections.

research efforts in the directions that are discussed in this paper are motivated by the following two remarks: (i) The aforementioned NP-Hardness is a "worst-case" result, and in the context of many practical applications, the "empirical" complexity of the computation of these target policies might be (much) more benign than what is suggested by the aforementioned result. (ii) Furthermore, it is pertinent to distinguish the computation that is involved in the deployment of the aforementioned policies into an "off-line" part, that concerns all the preliminary steps that are necessary for the policy specification, and the "on-line" part that involves the computation that takes place during the real-time implementation of the policy. The considered approaches seek to isolate the high computational complexity in the off-line part of the performed computation, and streamline the on-line part of the policy implementation through a pertinent representation of the policy-defining logic. This last step has been attained by perceiving the considered policies as "classifiers" of the underlying behavioral spaces into their admissible and inadmissible subspaces, and seeking "classifier architectures" that can encode this state-space dichotomy in an effective and parsimonious manner. Additional results, that pertain to some "monotonicity" properties of the state-admissibility logic defining the target policies, have also established the tractability of the computation of the sought DAP representations, by enabling the focusing of this computation upon the classification information that is contained in a very small subset of the underlying state space.

In the context of the developments that are outlined in the previous paragraph, the policy representation that has drawn the most extensive attention is that of the "linear classifier", *i.e.*, the representation where the relevant classification logic is expressed as a set of linear inequalities on the system state. Besides its conceptual and analytical simplicity, this type of control logic is also compatible with the Petri net (PN) modeling framework [17], one of the most extensively used modeling frameworks for the considered problem. In particular, it is well known that any behavioral constraint imposed upon a given PN that is expressible as a linear inequality in the net marking, can be enforced on the net dynamics by the addition of a "control" – or "monitor" – place to the original net structure [18], [19]. Furthermore, the work of [20] has investigated the synthesis of a set of monitor places that can enforce deadlock-freedom and, under stronger conditions, the liveness of any given PN. The method presented in [20] is based on siphon analysis of the underlying PN, seeking to prevent the formation of empty siphons. It is also iterative, since the addition of the monitor places can lead to the formation of new siphons that must be analyzed for potential emptiness and controlled appropriately. However, [20] acknowledges that the proposed iteration is not necessarily finite, *i.e.*, the method can be entrapped in an infinite loop failing to return a pertinent control policy. At the same time, [20] does not provide succinct conditions under which the proposed method is expected to terminate successfully.

The aforementioned inability of the method of [20] to identify a set of monitors representing the maximally permissive policy that enforces deadlock freedom and / or liveness, even in cases where this policy is known to be well-defined and to have a finite representation in the net reachability space, is due to the fact that the policy-admissible and inadmissible subspaces might not be linearly

Preprint submitted to IEEE Transactions on Automatic Control. Received: May 10, 2013 23:51:11 PST

separable; the reader is referred to [21] for a pertinent example. Nevertheless, as remarked in the previous paragraph, monitor-based liveness enforcing supervision has been pursued extensively in the context of RAS-modeling PNs, partly due to its representational convenience and partly due to the "history" of the PN modeling framework as a prevalent modeling framework for the considered problem. Hence, the works of [22] and [23] have tried to adapt the aforementioned method of [20] to the particular class of RAS-modeling PNs, while utilizing an additional set covering [24] formulation in order to minimize the number of siphons requiring explicit control. Other similar endeavors can be traced in [25], [26], [27], [28], and [29]. In all these works, the potential inability of the monitor-based representation to express the maximally permissive DAP is addressed by compromising for a more restrictive DAP that admits, however, a linear – and therefore, monitor-based – representation. A more recent implementation of the iterative monitor-based synthesis of the maximally permissive DAP that is provably convergent and computationally efficient is presented in [30], [31]; this stronger set of results is obtained by restricting the underlying plant nets to a particular PN sub-class that is appropriate for the modeling of the allocation of mutex locks in multi-threaded software.

An alternative approach that has sought the establishment of deadlock freedom and liveness for RAS-modeling PNs through the imposition of monitor places, and therefore, a linear representation of the target DAP, has been based on the theory of regions [32]. Characteristic examples of this line of work are the results presented in [33] and [34], where the target DAP is first characterized through reachability analysis of the underlying plant PN – *i.e.*, by an automaton-based representation – and the derived results are subsequently rehashed into the PN-modeling framework through the computation of the monitor places that will confine the net behavior in the policy-admissible subspace. Clearly, this approach is also compromised by the inability of PN monitors to represent the target policy in all cases. Furthermore, it has been found that a straightforward implementation of the method will end up in a very large number of monitors, even for relatively small plant nets [35].

The problem of developing parsimonious monitor-based representations for the maximally permissive DAP of RAS-modeling PNs that can potentially admit such a representation, has been tackled successfully in the recent years through techniques that are based on mixed integer programming (MIP) [36]. More specifically, the work of [9] addressed the problem of synthesizing a linear classifier that represents the maximally permissive DAP while employing the minimum possible number of linear inequalities, for the RAS class that models the allocation of mutex locks in multithreaded programs. It was shown that, in the considered RAS class, the maximally permissive DAP will always admit a representation as a linear classifier, due to the binary nature of an appropriately defined state that traces the RAS dynamics, and the problem of the construction of a minimal linear classifier was formulated and solved as a MIP formulation on the resulting state space. That work also recognized the affinity of the considered problem to the classical set covering problem [24] that is studied in Operations Research (OR) and Computer Science (CS) literature, and exploited this relationship in order to develop an efficient heuristic for the cases that

Preprint submitted to IEEE Transactions on Automatic Control. Received: May 10, 2013 23:51:11 PST

the aforementioned MIP formulation might become intractable. Another line of work that has addressed the synthesis of minimal linear classifiers through MIP, while capitalizing upon and expanding from some original developments in the theory of regions that was discussed above, is that presented in [12], [13] and [37]. On the other hand, the work of [10], [11] has developed a more streamlined customized algorithm for the same problem, based on techniques borrowed from combinatorial optimization and taking advantage of the aforementioned affinity of this problem to the set covering problem. Finally, in order to address problem instances not admitting a maximally permissive DAP in the form of a linear classifier, the recent works of [38] and [14] have identified non-linear classification architectures that can effectively support the representation of the maximally permissive DAP for the considered RAS classes, and developed a computational methodology for the synthesis of minimal realizations of the maximally permissive DAP, by means of the selected architecture, for any instantiation of the considered RAS.

The work presented in this paper complements the aforementioned developments by (i) investigating the conditions that imply the existence of linear classifiers for the representation of the maximally permissive DAP, and (ii) extending the results of [10], [11] so that they can address the minimal-classifier-design problem in the context of some of the non-linear architectures that have been introduced in [38]. Collectively, the presented results define a complete and computationally efficient algorithm that identifies the simplest possible architecture, among the linear and the considered non-linear candidates, that can support an effective representation of the target DAP, and returns a classifier from the selected class that minimizes the run-time computational overhead for the deployed supervisor. A complementary set of results that provide a formal characterization of the relationship between the linear-classifier-design problem and the classical set covering problem, and leverage this characterization in order to develop further analytical results and insights regarding the problem feasibility and optimality, can be found in [39]. Finally, a more leisurely and less formal description of the supervisory control problem considered in this work, and of some of the proposed methods for it that were outlined in the previous paragraphs, can be found in [40].

The rest of the paper is organized as follows: Section II provides a formal characterization of the resource allocation systems and the corresponding problem of (maximally permissive) deadlock avoidance that is addressed in this work. Section III introduces the classifier architectures that are considered in this work, and establishes the necessary and sufficient conditions that must be satisfied by the state dichotomy that is effected by the maximally permissive DAP in order to be expressible by the considered classifier architectures. Section IV presents an algorithmic procedure that selects a pertinent architecture for any given RAS instance, and overviews the algorithm of [10], [11] for the efficient synthesis of a minimal linear classifier realizing the maximally permissive DAP. Section V addresses the more complex problem of the efficient synthesis of a minimal non-linear classifier that effects the sought dichotomy of the RAS state space. Section VI presents an illustrative example and a set of computational experiments that establish the efficacy and the strengths of the proposed approaches compared to relevant existing

results. Finally, Section VII concludes the paper and discusses directions for future work. Closing this introductory section, we also notice, for completeness, that a preliminary version of the results of this manuscript can be found in [41].

II. The considered RAS class and the Optimal Deadlock Avoidance Policy

This section provides a characterization of the notion of resource allocation system (RAS) and of the supervisory control problem of deadlock avoidance that arises in it. For the sake of simplicity and specificity, we present the main results of this paper in the context of the *Disjunctive / Conjunctive* (*D/C*) class of the RAS taxonomy presented in [1]. We notice, however, that the presented ideas and results are extensible to more complex classes of that taxonomy.

A. The Disjunctive / Conjunctive Resource Allocation System

A *Disjunctive / Conjunctive Resource Allocation System* (*D/C-RAS*) is formally defined by a 4-tuple $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A} \rangle$,[2] where: (i) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*. (ii) $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ – 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, $C(R_i) \equiv C_i$ constitutes a system *invariant* for each i. (iii) $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ denotes the set of the system *process types* supported by the considered system configuration. Each process type Π_j is a composite element itself, in particular, $\Pi_j =< \Delta_j, \mathcal{G}_j >$, where: (a) $\Delta_j = \{\Xi_{j1}, \ldots, \Xi_{j,l_j}\}$ denotes the set of *processing stages* involved in the definition of process type Π_j, and (b) $\mathcal{G}_j$ is an *acyclic digraph* with its node set, Q_j, being bijectively related to the set Δ_j. Let $Q_j^{\nearrow}$ (resp., $Q_j^{\searrow}$) denote the set of *source* (resp., *sink*) nodes of $\mathcal{G}_j$. Then, any *path* from some node $q_s \in Q_j^{\nearrow}$ to some node $q_f \in Q_j^{\searrow}$ defines a *process plan* for process type Π_j. Also, in the following, we shall let $\Delta \equiv \bigcup_{j=1}^{n} \Delta_j$ and $\xi \equiv |\Delta|$. (iv) $\mathcal{A} : \Delta \rightarrow \prod_{i=1}^{m} \{0, \ldots, C_i\}$ is the *resource allocation function* associating every processing stage Ξ_{jk} with the *resource allocation vector* $\mathcal{A}(\Xi_{ij})$ required for its execution; it is further assumed that $\mathcal{A}(\Xi_{ij}) \neq \mathbf{0}, \ \forall i, j$. The considered resource allocation protocol requires that a process instance executing a non-terminal stage $\Xi_{ij} \in Q_i \backslash Q_i^{\searrow}$, must first be allocated the resource differential $(\mathcal{A}(\Xi_{i,j+1}) - \mathcal{A}(\Xi_{ij}))^+$ in order to advance to (some of) its next stage(s) $\Xi_{i,j+1}$, and only then will it release the resource units $|(\mathcal{A}(\Xi_{i,j+1}) - \mathcal{A}(\Xi_{ij}))^-|$, that are not needed anymore. Furthermore, no resource type $R_i \in \mathcal{R}$ should be over-allocated with respect to its capacity C_i at any point in time. Finally, for purposes of complexity analysis, we define the *size* $|\Phi|$ of RAS Φ by $|\Phi| \equiv |\mathcal{R}| + \xi + \sum_{i=1}^{m} C_i$.

[2]The complete definition of a RAS, according to [1], involves an additional component that characterizes the time-based – or *quantitative* – dynamics of the RAS, but this component is not relevant in the modeling and analysis to be pursued in the following developments, and therefore, it is omitted.

The dynamics of the D/C-RAS $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A} \rangle$, described in the previous paragraph, can be further formalized by a *Deterministic Finite State Automaton (DFSA)* [42], $G(\Phi) = \langle S, E, f, \mathbf{s}_0, S_M \rangle$, that is defined as follows:

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

$$\forall i = 1, \ldots, m, \quad \sum_{l=1}^{\xi} \mathbf{s}[l] \cdot \mathcal{A}(\Xi_l)[i] \leq C_i \tag{1}$$

where, according to the adopted notation, $\mathcal{A}(\Xi_l)[i]$ denotes the allocation request for resource R_i that is posed by stage Ξ_l.

2) The *event set* E is the union of the disjoint event sets $E^{\nearrow}$, $\bar{E}$ and $E^{\searrow}$, which respectively collect the events corresponding to the initiation of a new process, the advancement of an initiated process by one stage, and the completion of some initiated process. More formally: $E^{\nearrow} = \{e_{rp} : r = 0,\ \Xi_p \in \bigcup_{j=1}^{n} Q_j^{\nearrow}\}$; $\bar{E} = \{e_{rp} : \exists j \in 1, \ldots, n \text{ s.t. } \Xi_p \text{ is a successor of } \Xi_r \text{ in graph } \mathcal{G}_j\}$; and $E^{\searrow} = \{e_{rp} : \Xi_r \in \bigcup_{j=1}^{n} Q_j^{\searrow},\ p = 0\}$.

3) The *state transition function* $f : S \times E \to S$ is defined by $\mathbf{s}' = f(\mathbf{s}, e_{rp})$, where the components $\mathbf{s}'[l]$ of the resulting state $\mathbf{s}'$ are given by:

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

Furthermore, $f(\mathbf{s}, e_{rp})$ is a *partial* function defined only if the resulting state $\mathbf{s}' \in S$.

4) Finally, the *initial state* $\mathbf{s}_0 = \mathbf{0}$, which corresponds to the situation where the system is empty of any process instances, while the *set of marked states* S_M is the singleton $\{\mathbf{s}_0\}$, a specification that expresses the requirement for complete process runs.

In the following, the set of states $S_r \subseteq S$ that is accessible from state s_0 through a sequence of feasible transitions will be referred to as the *reachable subspace* of Φ. We shall also denote by $S_s \subseteq S$ the set of states that are co-accessible to s_0, *i.e.*, S_s contains those states from which s_0 is reachable through a sequence of feasible transitions. In addition, we define $S_u \equiv S \setminus S_s$ and $S_{rx} \equiv S_r \cap S_x,\ x = s, u$. Finally, we notice that, in the deadlock avoidance literature, the sets S_{rs} and S_{ru} are respectively characterized as the (reachable) safe and unsafe subspaces, and this is the terminology that we shall also adopt in the following.

In the D/C-RAS context, the state sets S_{rs} and S_{ru} possess the following "monotonicity" properties [9], where the relations " $\leq$ " and " $\geq$ ", when applied to vectors, should be interpreted component-wise, *i.e.,* as inequalities simultaneously holding for all components:

Property 1: Let $\mathbf{x} \in S_r$ be such that $\mathbf{x} \leq \mathbf{s}$ for some $\mathbf{s} \in S_{rs}$. Then $\mathbf{x} \in S_{rs}$. ■

Property 2: Let $\mathbf{x} \in S_r$ be such that $\mathbf{x} \geq \mathbf{u}$ for some $\mathbf{u} \in S_{ru}$. Then $\mathbf{x} \in S_{ru}$. ■

For reasons that will be revealed in the following, the above properties also motivate the next definition.

Definition 1: [9] Given a RAS $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A} \rangle$,

1) a safe state $\mathbf{s} \in S_{rs}$ is *maximal* if $\nexists\, \mathbf{s}' \in S_{rs} \backslash \{\mathbf{s}\}$ such that $\mathbf{s}' \geq \mathbf{s}$;
2) an unsafe state $\mathbf{u} \in S_{ru}$ is *minimal* if $\nexists\, \mathbf{u}' \in S_{ru} \backslash \{\mathbf{u}\}$ such that $\mathbf{u}' \leq \mathbf{u}$. ■

B. Deadlock avoidance as a classification problem

A *maximally permissive deadlock avoidance policy (DAP)* for RAS Φ is a supervisory control policy that restricts the system operation exactly onto the subspace S_{rs}, guaranteeing, thus, that every initiated process can complete successfully. This definition further implies that the maximally permissive DAP is unique, in terms of the allowed and forbidden states and transitions, and can be implemented by a one-step-lookahead mechanism that can recognize and prevent transitions to unsafe states.

As remarked in the introductory section, the maximally permissive DAP for sequential RAS can be effectively implemented by means of a state classifier that separates S_{rs} from S_{ru}. Different types of classifiers may be used for this purpose depending on the representational complexity of the underlying state separation problem. This work focuses primarily on a subclass of the classifiers introduced in [38] and proposes an efficient customized algorithm for the computation of a structurally minimal classifier from this subclass that represents the maximally permissive DAP of any given D/C-RAS Φ.

III. Linear and nonlinear classifiers for deadlock avoidance

We first overview some results concerning the existence of linear and nonlinear classifiers separating two generic disjoint sets of points in a vector space, and then we specialize these results to the D/C-RAS case.

A. The linear classifier

In order to facilitate the subsequent developments, let $\mathcal{S}, \mathcal{U}$ denote two finite sets of $(\mathbb{R}_0^+)^\xi$ such that $\mathcal{S} \cap \mathcal{U} = \emptyset$.[3] The following definitions provide the notion of a linear classifier separating $\mathcal{U}$ from $\mathcal{S}$.

Definition 2: [38] Let $P_{\mathbf{A},\mathbf{b}} = \left\{\mathbf{x} \in \mathbb{R}^\xi : \mathbf{A}\mathbf{x} \leq \mathbf{b}\right\}$ be a polyhedron generated by a system of linear inequalities, where $\mathbf{A}$ is a $k \times \xi$ real-valued matrix and $\mathbf{b}$ is a vector in $\mathbb{R}^k$. The *inclusion indicator* $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{x})$ is a binary function that equals 1 if $\mathbf{x} \in P_{\mathbf{A},\mathbf{b}}$, *i.e.* $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, and 0 otherwise. ■

Definition 3: [38] A *linear classifier* separating $\mathcal{U}$ from $\mathcal{S}$ is a pair $(\mathbf{A}, \mathbf{b})$, such that $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{s}) = 1$, $\forall \mathbf{s} \in \mathcal{S}$, and $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{u}) = 0$, $\forall \mathbf{u} \in \mathcal{U}$. If at least one such classifier exists, $\mathcal{S}$ and $\mathcal{U}$ are said to be *L-separable*. We denote as $sep_L(\mathcal{S}, \mathcal{U})$ the set of linear classifiers that separate the two sets. The elements of $sep_L(\mathcal{S}, \mathcal{U})$ whose matrix $\mathbf{A}$ has a minimal number of rows k are said to be *minimal*. ■

[3] In fact, the developments of subsections III-A and III-B hold true even if the considered sets $\mathcal{S}, \mathcal{U}$ are subsets of $\mathbb{R}^\xi$. On the other hand, the developments of subsections III-C and III-D necessitate the restriction of these sets into $(\mathbb{R}_0^+)^\xi$.

The reader should notice that the above definition of the concept of the linear classifier implies an "asymmetry" for the role of the separated sets $\mathcal{S}$ and $\mathcal{U}$. This asymmetry is motivated and explained by the fact that in the eventual specialization of the results that are derived in this and the next two sections to the particular classification problem that is considered in this work, sets $\mathcal{S}$ and $\mathcal{U}$ will respectively correspond to the (sub-)sets of safe and unsafe states to be classified. Then, in more practical terms, the specification of the linear classification logic according to Definition 3 enables the eventual representation of the derived classifier through monitor places, under a PN-based representation of the resource allocation function. From a more analytical standpoint, the aforementioned asymmetry can be captured by considering the set pair $(\mathcal{S},\mathcal{U})$ as ordered. Finally, for the reasons explained above, in the subsequent discussion we shall also refer to sets $\mathcal{S}$ and $\mathcal{U}$ as the sets of *safe* and *unsafe states*.

As stated in the introductory section, a useful alternative interpretation of the linear classifier can be obtained by perceiving the corresponding separation problem as a variation of the set covering problem [24]. The following notion of *1-separability* is instrumental for this alternative interpretation.

Definition 4: Sets $\mathcal{S}$ and $\mathcal{U}$ are said to be *1-separable* if the following system of linear inequalities in $\mathbf{a}, b$, with $\mathbf{a} \in \mathbb{R}^{\xi}$ and $b \in \mathbb{R}$, is feasible:

$$\mathbf{a}^T\mathbf{x} \leq b \qquad \mathbf{x} \in \mathcal{S} \tag{2a}$$

$$\mathbf{a}^T\mathbf{x} \geq b+1 \quad \mathbf{x} \in \mathcal{U} \tag{2b}$$

In the following, the set of feasible solutions $(\mathbf{a}^T, b)$ of problem (2) will be denoted as $sep_1(\mathcal{S},\mathcal{U})$. ■

Property 3: Sets $\mathcal{S}$ and $\mathcal{U}$ are L-separable iff there exists a partition $\cup_{j=1}^{c}\mathcal{U}_j = \mathcal{U}$ such that $sep_1(\mathcal{S},\mathcal{U}_j) \neq \emptyset$, for all $j = 1,\ldots,c$.

Proof: The "only if" part of the proof results immediately from the definitions of the linear classifier and the *1-separable* sets. As for the "if" part, let $(\mathbf{a}_j^T, b_j) \in sep_1(\mathcal{S},\mathcal{U}_j)$, for $j = 1,\ldots,c$. By Definition 4, it holds that $\gamma_{\mathbf{a}_j^T,b_j}(\mathbf{s}) = 1$, $\forall \mathbf{s} \in \mathcal{S}$, and $\gamma_{\mathbf{a}_j^T,b_j}(\mathbf{u}) = 0$, $\forall \mathbf{u} \in \mathcal{U}_j$, for $j = 1,\ldots,c$. Then the linear classifier $(\mathbf{A},\mathbf{b})$, defined as $\mathbf{A}(j,\cdot) = \mathbf{a}_j^T$ and $\mathbf{b}(j) = b_j$, for $j = 1,\ldots,c$, is such that $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{s}) = 1$, $\forall \mathbf{s} \in \mathcal{S}$, and $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{u}) = 0$, $\forall \mathbf{u} \in \cup_{j=1}^{c}\mathcal{U}_j$. In other words, $(\mathbf{A},\mathbf{b})$ defines a linear classifier separating $\mathcal{S}$ from $\mathcal{U}$. ■

It is trivial to see that if $sep_1(\mathcal{S},\mathcal{U}) \neq \emptyset$, then also $sep_L(\mathcal{S},\mathcal{U}) \neq \emptyset$. Also, we remind the reader that the convex hull of the elements of set $\mathcal{S}$ defines a bounded polyhedron (*i.e.*, a polytope) $P_{\mathcal{S}} \equiv Conv(\mathcal{S})$, which can be expressed as the set of all convex combinations of the elements of $\mathcal{S}$, *i.e.*, $P_{\mathcal{S}} = \{\mathbf{x} \in \mathbb{R}^{\xi} : \mathbf{x} = \sum_{i=1}^{|\mathcal{S}|}\alpha_i\mathbf{s}_i$ with $\sum_{i=1}^{|\mathcal{S}|}\alpha_i = 1, \alpha_i \geq 0$ for $i = 1,\ldots,|\mathcal{S}|\}$. Definition 3 implies that for any linear classifier $(\mathbf{A},\mathbf{b}) \in sep_L(\mathcal{S},\mathcal{U})$ it holds that $P_{\mathcal{S}} \subseteq P_{\mathbf{A},\mathbf{b}}$.

Definition 5: An unsafe state $\mathbf{u} \in \mathcal{U}$ is called *embedded* if $\mathbf{u} \in P_{\mathcal{S}}$. The set of embedded unsafe states is denoted by $\mathcal{E}_{\mathcal{S}} = \mathcal{U} \cap P_{\mathcal{S}}$. ■

Theorem 1: For the considered sets $\mathcal{S}$ and $\mathcal{U}$,

$$sep_L(\mathcal{S},\mathcal{U}) \neq \emptyset \iff \mathcal{E}_{\mathcal{S}} = \emptyset \tag{3}$$

Proof: If there is no unsafe state $\mathbf{u} \in \mathcal{U}$ embedded in the polyhedron $P_{\mathcal{S}}$, the facets of this polyhedron provide a (generally non-minimal) linear classifier for $\mathcal{S}$ and $\mathcal{U}$. Conversely, if there is an unsafe state $\mathbf{u} = \sum_{i=1}^{|\mathcal{S}|} \alpha_i \mathbf{s}_i$, with $\sum_{i=1}^{|\mathcal{S}|} \alpha_i = 1, \alpha_i \geq 0$ for $i = 1, \ldots, |\mathcal{S}|$, then any linear inequality $\left(\mathbf{a}^T, b\right)$ satisfied by all elements of $\mathcal{S}$ is also satisfied by $\mathbf{u}$. Indeed, if $\mathbf{a}^T \mathbf{s}_i \leq b$ for all $\mathbf{s}_i \in \mathcal{S}$, then $\mathbf{a}^T \mathbf{u} = \mathbf{a}^T \sum_{i=1}^{|\mathcal{S}|} \alpha_i \mathbf{s}_i = \sum_{i=1}^{|\mathcal{S}|} \alpha_i (\mathbf{a}^T \mathbf{s}_i) \leq \sum_{i=1}^{|\mathcal{S}|} \alpha_i b = b$. Therefore, $\mathbf{u}$ is not L-separable from $\mathcal{S}$. ■

Theorem 1 implies that the existence of a linear classifier for generic disjoint vector sets is not always guaranteed, and motivates the study of more general types of classifiers; this issue is taken up in the next subsection.

Closing the discussion of the linear classifiers, we also notice that the search for a classifier of minimal size from this class is equivalent to solving to optimality a set covering problem [24], where set $\mathcal{U}$ is to be covered with a minimal number of subsets that are individually separable from $\mathcal{S}$ by means of a single linear inequality (see Property 3 and also Property 5 in Section III-C). This equivalence was introduced in [9] where it was employed for the development of an efficient heuristic for the aforementioned problem, and it was further exploited in the design of an *ad hoc* algorithm for the same problem that is presented in [10], [11]. This last algorithm is also briefly summarized in Sec. IV-B, since it provides a basis for the development of the particular procedures and algorithms that are the main theme of this work.

B. Disjunctions of linear classifiers

The following results suggest a way to tackle the issue of embedded unsafe states.

Theorem 2: Consider the sets $\mathcal{S}, \mathcal{U}$ introduced in Section III-A, and let $\mathcal{S}' \subset \mathcal{S}$. Then, $\mathcal{E}_{\mathcal{S}'} \subseteq \mathcal{E}_{\mathcal{S}}$.

Proof: This result follows immediately upon noticing that the condition $\mathcal{S}' \subset \mathcal{S}$ implies that $P_{\mathcal{S}'} \subseteq P_{\mathcal{S}}$. ■

Theorem 3: Consider the sets $\mathcal{S}, \mathcal{U}$ introduced in Section III-A, and let $\mathbf{u} \in \mathcal{E}_{\mathcal{S}}$. Then, there always exists an $\mathcal{S}' \subset \mathcal{S}$ such that $\mathbf{u} \notin \mathcal{E}_{\mathcal{S}'}$.

Proof: Let $\mathbf{s} \in \mathcal{S}$, and define $\mathcal{S}' = \{\mathbf{s}\}$. By definition, $P_{\mathcal{S}'} = \mathcal{S}'$, and since $\mathcal{S} \cap \mathcal{U} = \emptyset$, $\mathcal{E}_{\mathcal{S}'} = \mathcal{U} \cap P_{\mathcal{S}'} = \emptyset$. ■

Theorem 2 guarantees that if a separation problem is defined with respect to $\mathcal{S}' \subset \mathcal{S}$ and $\mathcal{U}$, new embedded unsafe states will not come into play. When considered together with Theorem 3, they suggest that suitable subsets of $\mathcal{S}$ might be L-separable from $\mathcal{U}$ even if the entire set $\mathcal{S}$ is not. Starting from $\mathcal{S}' = \{\mathbf{s}\}$, where $\mathbf{s} \in \mathcal{S}$, a maximal subset of this type can be obtained by iteratively adding safe states, as long as $\mathcal{U} \cap P_{\mathcal{S}'} = \emptyset$. These results pave the way for classifiers with an alternative structure, where set $\mathcal{S}$ is partitioned as

$$\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_p$$

into disjoint subsets $\mathcal{S}_i$ that are L-separable from $\mathcal{U}$, *i.e.*, such that $P_{\mathcal{S}_i} \cap \mathcal{E}_{\mathcal{S}} = \emptyset$, for $i = 1, \ldots, p$. With some abuse of the language, we shall refer to the classification structure that is facilitated by this partition as a *"disjunctive linear classifier"*, and we shall formally define it as follows:

Definition 6: A *disjunctive linear classifier* for the sets $\mathcal{S}$ and $\mathcal{U}$ that were introduced in Section III-A is defined as a set of polyhedra $\{(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}), i = 1, \ldots, p\}$, such that:

(i) For each $\mathbf{s} \in \mathcal{S}$ there exists $i \in \{1, \ldots, p\}$ such that $\gamma_{\mathbf{A}^{(i)},\mathbf{b}^{(i)}}(\mathbf{s}) = 1$.

(ii) For each $\mathbf{u} \in \mathcal{U}$ it holds that $\gamma_{\mathbf{A}^{(i)},\mathbf{b}^{(i)}}(\mathbf{u}) = 0$ for all $i = 1, \ldots, p$.

If at least one such classifier exists, $\mathcal{S}$ and $\mathcal{U}$ are said to be *D-separable*. We shall also denote by $sep_D(\mathcal{S},\mathcal{U})$ the set of disjunctive linear classifiers that separate the two sets. ■

The classification requirements expressed by Definition 6 can be expressed more compactly by the requirement for a set of polyhedra $\{(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}), i = 1, \ldots, p\}$, such that the Boolean function $\delta(\mathbf{x}) = \gamma_{\mathbf{A}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) \vee \ldots \vee \gamma_{\mathbf{A}^{(p)},\mathbf{b}^{(p)}}(\mathbf{x})$ is priced at $\delta(\mathbf{s}) = 1$, $\forall \mathbf{s} \in \mathcal{S}$, and $\delta(\mathbf{u}) = 0$, $\forall \mathbf{u} \in \mathcal{U}$. This alternative characterization of the functionality of a disjunctive linear classifier will be useful in the sequel.

We also notice that the disjunctive linear classifier is a special case of the *"generic"* classifier of [38]. The *"size"* of a disjunctive linear classifier is determined by the total number of operations required to classify a given vector [38], *i.e.*, $J = (2\xi + 1)\sum_{i=1}^{p} k^{(i)} + (p - 1)$, where $k^{(i)}$ is the row size of matrix $\mathbf{A}^{(i)}$. Also, in order to differentiate more clearly between linear and disjunctive linear classifiers, in the following we might refer to the former as *"simple"* linear classifiers. Clearly, $sep_L(\mathcal{S},\mathcal{U}) \neq \emptyset$ implies $sep_D(\mathcal{S},\mathcal{U}) \neq \emptyset$.

Property 4: The sets $\mathcal{S}$, $\mathcal{U}$ that were introduced in Section III-A are D-separable iff there exists a partition $\cup_{i=1}^{p}\mathcal{S}_i = \mathcal{S}$ such that $sep_L(\mathcal{S}_i,\mathcal{U}) \neq \emptyset$, for all $i = 1, \ldots, p$.

Proof: The "only if" part of the proof results immediately from the definitions of the disjunctive linear classifier and the *L-separable* sets. As for the "if" part, let $(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}) \in sep_L(\mathcal{S}_i,\mathcal{U})$, for $i = 1, \ldots, p$. By Definition 3, it holds that $\gamma_{\mathbf{A}^{(i)},\mathbf{b}^{(i)}}(\mathbf{s}) = 1, \forall \mathbf{s} \in \mathcal{S}_i$ and $\gamma_{\mathbf{A}^{(i)},\mathbf{b}^{(i)}}(\mathbf{u}) = 0, \forall \mathbf{u} \in \mathcal{U}$, for all $i = 1, \ldots, p$. Then, by Definition 6, $\{(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}), i = 1, \ldots, p\} \in sep_D(\mathcal{S},\mathcal{U})$. ■

Theorem 4: Let $\mathcal{S}$, $\mathcal{U}$ be two finite vector sets of $(\mathbb{R}_0^+)^\xi$ with $\mathcal{S} \cap \mathcal{U} = \emptyset$. Then $sep_D(\mathcal{S},\mathcal{U}) \neq \emptyset$.

Proof: If $\mathcal{E}_\mathcal{S} = \emptyset$, then Theorem 1 implies that $sep_L(\mathcal{S},\mathcal{U}) \neq \emptyset$, which implies the thesis. On the other hand, if $\mathcal{E}_\mathcal{S} \neq \emptyset$, there always exists a partition of $\mathcal{S} = \cup_{i=1}^{p}\mathcal{S}_i$, such that $P_{\mathcal{S}_i} \cap \mathcal{E}_\mathcal{S} = \emptyset$, for $i = 1, \ldots, p$. For example, one such partition is given by $\mathcal{S}_i = \{\mathbf{s}_i\}$, $i = 1, \ldots, |\mathcal{S}|$. Then, by Theorem 1, there exists a simple linear classifier separating $\mathcal{S}_i$ from $\mathcal{U}$, for all $i = 1, \ldots, p$, and the result is obtained from Property 4. ■

Theorem 4 implies that any pair of finite disjoint sets can be separated by a disjunctive linear classifier, and, in this way, it establishes the generality – or the "completeness" – of this type of classifiers for the classification problem that is defined by the set pair $(\mathcal{S},\mathcal{U})$.

C. Linear and disjunctive linear classifiers with non-negative coefficients

Significant reductions in the complexity of the separation problem that is considered in this work can be obtained if the coefficients of the linear inequalities employed by the simple or disjunctive linear classifiers are constrained to be non-negative (see the developments in the next subsection). It is therefore

interesting to extend the previous definitions to encompass this case, and derive appropriate conditions for the existence of simple or disjunctive linear classifiers with non-negative coefficients.

Definition 7: The sets $\mathcal{S}$, $\mathcal{U}$ introduced in Section III-A are said to be *1^+-separable* if the following system of linear inequalities in $\mathbf{a}, b$ is feasible:

$$\mathbf{a}^T\mathbf{x} \leq b \qquad \mathbf{x} \in \mathcal{S} \tag{4a}$$

$$\mathbf{a}^T\mathbf{x} \geq b+1 \quad \mathbf{x} \in \mathcal{U} \tag{4b}$$

$$\mathbf{a}, b \geq 0 \tag{4c}$$

In the following, we shall denote the set of feasible solutions $(\mathbf{a}^T, b)$ of problem (4) by $sep_1^+(\mathcal{S},\mathcal{U})$. ■

Definition 8: The sets $\mathcal{S}, \mathcal{U}$ introduced in Section III-A are said to be *L^+-separable* if $sep_L(\mathcal{S},\mathcal{U}) \neq \emptyset$ and there exists $(\mathbf{A}, \mathbf{b}) \in sep_L(\mathcal{S},\mathcal{U})$ with $\mathbf{A}, \mathbf{b} \geq 0$. The set of such non-negative simple linear classifiers will be denoted by $sep_L^+(\mathcal{S},\mathcal{U})$. ■

Definition 9: The sets $\mathcal{S}, \mathcal{U}$ introduced in Section III-A are said to be *D^+-separable* if $sep_D(\mathcal{S},\mathcal{U}) \neq \emptyset$ and there exists $\{(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}), i = 1, \ldots, p\} \in sep_D(\mathcal{S},\mathcal{U})$ with $\mathbf{A}^{(i)}, \mathbf{b}^{(i)} \geq 0$, for all $i = 1, \ldots, p$. The set of such non-negative disjunctive linear classifiers will be denoted by $sep_D^+(\mathcal{S},\mathcal{U})$. ■

Properties 3 and 4 can be trivially extended to the class of classifiers with non-negative coefficients; more specifically, we have the following:

Property 5: Consider the sets $\mathcal{S}$, $\mathcal{U}$ introduced in Section III-A. $\mathcal{S}$ and $\mathcal{U}$ are L^+-separable iff there exists a partition $\cup_{j=1}^{c}\mathcal{U}_j = \mathcal{U}$ such that $sep_1^+(\mathcal{S},\mathcal{U}_j) \neq \emptyset$, for all $j = 1, \ldots, c$.

Property 6: Consider the sets $\mathcal{S}$, $\mathcal{U}$ introduced in Section III-A. $\mathcal{S}$ and $\mathcal{U}$ are D^+-separable iff there exists a partition $\cup_{i=1}^{p}\mathcal{S}_i = \mathcal{S}$ such that $sep_L^+(\mathcal{S}_i,\mathcal{U}) \neq \emptyset$, for all $i = 1, \ldots, p$.

Clearly, $sep_L^+(\mathcal{S},\mathcal{U}) \subseteq sep_L(\mathcal{S},\mathcal{U})$ and $sep_D^+(\mathcal{S},\mathcal{U}) \subseteq sep_D(\mathcal{S},\mathcal{U})$. Therefore, the conditions for the existence of simple and disjunctive linear classifiers with non-negative coefficients are more restrictive.

Let $P_{\mathcal{S}}^+ \equiv \{\mathbf{x} \in \mathbb{R}^\xi : \exists \mathbf{y} \in P_{\mathcal{S}} \text{ s.t. } \mathbf{x} \leq \mathbf{y}\}$. By the Minkowsky-Weyl theorem [36], $P_{\mathcal{S}}^+$ is a polyhedron because each of its points is the sum of the convex combination of a finite set of points (the elements of $\mathcal{S}$) plus the conical combination of a finite set of points (those defined by the negative unit vectors $-\mathbf{e}^{(k)}$, with $k = 1, \ldots, \xi$). Let also $\mathcal{E}_{\mathcal{S}}^+ \equiv \mathcal{U} \cap P_{\mathcal{S}}^+$; this set will be characterized as the *"extended set of embedded unsafe states"*. By construction, $P_{\mathcal{S}} \subseteq P_{\mathcal{S}}^+$ and $\mathcal{E}_{\mathcal{S}} \subseteq \mathcal{E}_{\mathcal{S}}^+$.

Theorem 5: For the considered sets $\mathcal{S}$ and $\mathcal{U}$,

$$sep_L^+(\mathcal{S},\mathcal{U}) \neq \emptyset \iff \mathcal{E}_{\mathcal{S}}^+ = \emptyset \tag{5}$$

Proof: Notice that by the definition of $P_{\mathcal{S}}^+$, $\mathcal{S} \subseteq P_{\mathcal{S}}^+$, and assume that $\mathcal{U} \cap P_{\mathcal{S}}^+ = \emptyset$. Since $P_{\mathcal{S}}^+$ is a polyhedron, it admits an alternative representation as $\left\{\mathbf{x} \in \mathbb{R}^\xi : \mathbf{A}\mathbf{x} \leq \mathbf{b}\right\}$ for suitable $(\mathbf{A}, \mathbf{b})$. The latter provides a (generally non-minimal) simple linear classifier for $\mathcal{S}$ and $\mathcal{U}$. It remains to prove that $\mathbf{A}, \mathbf{b} \geq 0$. By contradiction, let us assume that the j-th scalar constraint $(\mathbf{a}^T, b)$ contains at least a negative coefficient. Furthermore, notice that if $b < 0$, there necessarily exists at least one $a_k < 0$, since

$\mathbf{s} \geq 0$. This implies that at least one a_k coefficient is negative. Now, let $\mathbf{y} \in P_{\mathcal{S}}$ and $\mathbf{x}_\delta = \mathbf{y} - \delta \mathbf{e}^{(k)}$. By definition, $\mathbf{x}_\delta \in P_{\mathcal{S}}^+$ for all $\delta \geq 0$. Therefore $\mathbf{a}^T \mathbf{x}_\delta \leq b$ should hold for all values of δ, which implies that $\mathbf{a}^T \mathbf{y} - a_k \delta \leq b$; but this last inequality is false for sufficiently high values of δ.

Conversely, suppose that there exist $\mathbf{u} \in \mathcal{U} \cap P_{\mathcal{S}}^+$ and $(\mathbf{a}^T, b) \in sep_1^+(\mathcal{S}, \{\mathbf{u}\})$, so that $\mathbf{a}^T \mathbf{s}_i \leq b$ for all $\mathbf{s}_i \in \mathcal{S}$ and $\mathbf{a}^T \mathbf{u} \geq b + 1$. Now, since $\mathbf{u} \in P_{\mathcal{S}}^+$, there exist $\mathbf{y}$, $\alpha_i \geq 0$, $i = 1, \ldots, |\mathcal{S}|$, such that $\mathbf{y} = \sum_{i=1}^{|\mathcal{S}|} \alpha_i \mathbf{s}_i$ with $\sum_{i=1}^{|\mathcal{S}|} \alpha_i = 1$, and $\mathbf{u} \leq \mathbf{y}$. Then, since $\mathbf{a}, b \geq 0$, one would obtain $b + 1 \leq \mathbf{a}^T \mathbf{u} \leq \mathbf{a}^T \mathbf{y} = \mathbf{a}^T \sum_{i=1}^{|\mathcal{S}|} \alpha_i \mathbf{s}_i = \sum_{i=1}^{|\mathcal{S}|} \alpha_i (\mathbf{a}^T \mathbf{s}_i) \leq \sum_{i=1}^{|\mathcal{S}|} \alpha_i b = b$, which is impossible. ■

Notice that a necessary condition for L^+-separability is that $\nexists (\mathbf{s}, \mathbf{u}) \in \mathcal{S} \times \mathcal{U}$, s.t. $\mathbf{u} \leq \mathbf{s}$. This condition turns out to be sufficient for the existence of a disjunctive linear classifier with non-negative coefficients.

Theorem 6: For the considered sets $\mathcal{S}$ and $\mathcal{U}$,

$$sep_D^+(\mathcal{S}, \mathcal{U}) \neq \emptyset \iff \nexists (\mathbf{s}, \mathbf{u}) \in \mathcal{S} \times \mathcal{U} \text{ s.t. } \mathbf{u} \leq \mathbf{s} \tag{6}$$

Proof: If the assumption holds, a valid disjunctive linear classifier is obtained by setting $p = |\mathcal{S}|$, $\mathbf{A}^{(i)} = \mathbf{I}_\xi$, $\mathbf{b}^{(i)} = \mathbf{s}_i$. Conversely, assume that there exists $(\mathbf{s}, \mathbf{u}) \in \mathcal{S} \times \mathcal{U}$, s.t. $\mathbf{u} \leq \mathbf{s}$. Then $sep_1^+(\{\mathbf{s}\}, \{\mathbf{u}\}) = \emptyset$, since any linear inequality $(\mathbf{a}^T, b)$, with $\mathbf{a}, b \geq 0$, satisfied by $\mathbf{s}$ is also satisfied by $\mathbf{u}$, given that $\mathbf{a}^T \mathbf{u} \leq \mathbf{a}^T \mathbf{s} \leq b$. In view of Properties 5 and 6, this also implies that $sep_D^+(\mathcal{S}, \mathcal{U}) = \emptyset$. ■

D. Linear and disjunctive linear classifiers expressing the optimal DAP in D/C-RAS

As discussed in the previous sections, the maximally permissive DAP for the considered RAS can be implemented by means of a state classifier that separates S_{rs} from S_{ru}. More precisely, we can restrict the latter set to S_{ru}^b, which collects the "boundary" unsafe states, *i.e.*, those unsafe states that can be reached from S_{rs} in a single transition (and are the critical unsafe states that must be effectively recognized in order to prevent transitions leading outside the reachable and safe subspace S_{rs}). Within this setting, a problem of particular practical interest is that of finding a classifier of *minimal* size within the considered classifier class; such a classifier will reduce to a minimum the computational overhead required for the on-line resolution of the admissibility of any contemplated transition, according to the one-step-lookahead scheme discussed in Section II. The aforementioned problem is a very difficult combinatorial problem, and, in the past works, it has been effectively solved through its simplification to another separation problem concerning two sets of (significantly) reduced cardinality and dimensionality, that are obtained by a "thinning" process described in [9]. However, this simplification scheme will work only if the sought classifiers are restricted to have non-negative coefficients. More specifically, denoting by $\bar{S}_{rs}$ and $\bar{S}_{ru}^b$ the set of *maximal* reachable safe states and the set of *minimal* reachable boundary unsafe states, respectively, it can be shown that any simple or disjunctive linear classifier with non-negative coefficients separating $\bar{S}_{rs}$ and $\bar{S}_{ru}^b$ also separates S_{rs} and S_{ru}^b [9], [38]. Furthermore, additional computational gains can be achieved by eliminating from the vectors of $\bar{S}_{rs}$ and $\bar{S}_{ru}^b$ the coordinates in which the elements of $\bar{S}_{ru}^b$ are dominated uniformly by each of the elements of $\bar{S}_{rs}$ [14]. Let $Proj(\bar{S}_{rs})$ and $Proj(\bar{S}_{ru}^b)$

denote respectively the images of the sets $\bar{S}_{rs}$ and $\bar{S}^b_{ru}$ under the projection $Proj$ that eliminates these coordinates. Finally, the sets $Proj(\bar{S}_{rs})$ and $Proj(\bar{S}^b_{ru})$ can be further reduced to the sets $\overline{Proj(\bar{S}_{rs})}$ and $\overline{Proj(\bar{S}^b_{ru})}$, that consist, respectively, of the maximal and the minimal elements of the corresponding superset.

In view of Properties 1-2 and of the logic that defines the applied projection, it holds that $\nexists(\mathbf{s},\mathbf{u}) \in \overline{Proj(\bar{S}_{rs})} \times \overline{Proj(\bar{S}^b_{ru})}$, s.t. $\mathbf{u} \leq \mathbf{s}$. But then, Theorem 6 guarantees that there always exists a disjunctive linear classifier with non-negative coefficients that separates $\overline{Proj(\bar{S}_{rs})}$ from $\overline{Proj(\bar{S}^b_{ru})}$. If the more restrictive condition of Theorem 5 is met as well, then the separation of the sets $\overline{Proj(\bar{S}_{rs})}$ and $\overline{Proj(\bar{S}^b_{ru})}$ can also be effected by a simple linear classifier with non-negative coefficients. Finally, from the discussion provided in the previous paragraph, the classifiers obtained for the smaller sets $\overline{Proj(\bar{S}_{rs})}$ and $\overline{Proj(\bar{S}^b_{ru})}$ effect also the separation of the original sets $\bar{S}_{rs}$ and $\bar{S}^b_{ru}$.

Next we show that for the classification problem defined by the aforementioned sets $\bar{S}_{rs}$ and $\bar{S}^b_{ru}$ (and also for its simplified version that is defined by the reduced sets $\overline{Proj(\bar{S}_{rs})}$ and $\overline{Proj(\bar{S}^b_{ru})}$), the restriction of the sought classifiers to the subclass of classifiers with non-negative coefficients does not compromise the existence of a simple linear classifier. In other words, if there exists a simple linear classifier expressing the maximally permissive DAP for a given RAS configuration, then there will also exist a simple linear classifier with non-negative coefficients. The next two lemmas constitute stepping stones towards the aforementioned result.[4]

Lemma 1: Consider an instance $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A} \rangle$ of the D/C-RAS class defined in Section II. Then, the state set S_{rs} contains all the integer vectors that are included between the origin $\mathbf{0}$ and the maximal elements $\mathbf{s} \in \bar{S}_{rs}$.

Proof: To prove Lemma 1, consider the partial order that is defined upon the set S_{rs} by the ordering relationship " $\leq$ ", where the inequality is applied component-wise. Next, we argue that all possible "chains" of integer vectors between the origin $\mathbf{0}$ and the elements of $\bar{S}_{rs}$, that are defined w.r.t. the aforementioned partial order, belong to S_r. Indeed, consider a state $\mathbf{s}$ belonging to an integer vector chain from state $\mathbf{0}(\equiv \mathbf{s}_0)$ to an element $\mathbf{s}' \in \bar{S}_{rs}$. Since $\mathbf{s}' \in \bar{S}_{rs}$, it is a reachable state, and therefore, there exists an event sequence σ' such that $\mathbf{s}' = \hat{f}(\mathbf{s}_0, \sigma')$. Furthermore, from the specification of the states $\mathbf{s}$ and $\mathbf{s}'$, it also holds that $\mathbf{s} \leq \mathbf{s}'$, and therefore, the set of process instances contained in $\mathbf{s}$ is a subset of the process instances contained in $\mathbf{s}'$. But since, as implied by the RAS definition, the various process instances do not interact with each other except for the sharing of the system resources, it is easy to see that the reachability of state $\mathbf{s}'$ implies also the reachability of state $\mathbf{s}$; a corresponding event sequence σ can be obtained from the event sequence σ' mentioned above by removing from it all the

[4]An alternative proof for this result can be found in [9]. In fact, the work of [9] establishes the stronger result that a minimal linear classifier with non-negative coefficients will possess the same number of inequalities as a minimal linear classifier with no sign restrictions for its defining coefficients, and therefore, the imposed sign restriction does not compromise the structural minimality of the derived classifiers.

events concerning process instances not belonging to $\mathbf{s}$.[5]

The fact that the considered state $\mathbf{s}$ belongs also to S_s, and therefore, to S_{rs}, results from Property 1 and the aforestated fact that $\exists \mathbf{s}' \in \bar{S}_{rs}$ s.t. $\mathbf{s} \leq \mathbf{s}'$. ■

Lemma 2: Let $\mathbb{N}$ denote the set of natural numbers, and consider a set $\mathcal{X} \subseteq \mathbb{N}^\xi$ such that

$$\forall \mathbf{x} \in \mathcal{X}, \forall \mathbf{x}' \in \mathbb{N}^\xi, \mathbf{x}' \leq \mathbf{x} \implies \mathbf{x}' \in \mathcal{X} \tag{7}$$

Furthermore, let $\mathbf{x}_1 \in Conv(\mathcal{X})$. Then, $Conv(\mathcal{X})$ also contains any other vector $\mathbf{x}_2 \in \mathbb{R}^\xi$ such that $\mathbf{0} \leq \mathbf{x}_2 \leq \mathbf{x}_1$.

Proof: Assume that $\mathbf{x}_2 = diag(t_1, t_2, \ldots, t_\xi) \cdot \mathbf{x}_1$, where $diag(\cdot)$ implies a diagonal matrix with its principal diagonal consisting of the quoted elements, and, without loss of generality, set

$$0 \leq t_1 \leq t_2 \leq \ldots \leq t_\xi \leq 1 \tag{8}$$

Let $\mathbf{y}_1, \ldots, \mathbf{y}_l$ be a set of vectors in $\mathcal{X}$ such that

$$\mathbf{x}_1 = \sum_{i=1}^{l} \lambda_i \cdot \mathbf{y}_i; \ \sum_{i=1}^{l} \lambda_i = 1; \ \lambda_i > 0, \ \forall i = 1, \ldots, l \tag{9}$$

Define the vectors $\mathbf{y}_{ij}, \ i = 1, \ldots, l, \ j = 1, \ldots, \xi$, such that:

$$\mathbf{y}_{ij}[k] \equiv \begin{cases} 0 & \text{if } 1 \leq k \leq j \\ \mathbf{y}_i[k] & \text{if } j < k \leq \xi \end{cases} \tag{10}$$

Equation (7) implies that all vectors $\mathbf{y}_{ij}$ belong to $\mathcal{X}$. Also, note that $\mathbf{y}_{i\xi} = \mathbf{0}, \ \forall i$. Furthermore, for notational convenience, define $t_0 = 0$, $t_{\xi+1} = 1$, and $\mathbf{y}_{i0} = \mathbf{y}_i, \ \forall i$. Then, we can see that:

$$\begin{aligned} \sum_{j=1}^{\xi+1} (t_j - t_{j-1}) \cdot \mathbf{y}_{i,j-1} &= t_1 \cdot \mathbf{y}_i + (t_2 - t_1) \cdot \mathbf{y}_{i1} + (t_3 - t_2) \cdot \mathbf{y}_{i2} + \ldots + (1 - t_\xi) \cdot \mathbf{y}_{i\xi} = \\ &= t_1 \cdot (\mathbf{y}_i[1], \mathbf{y}_i[2], \ldots, \mathbf{y}_i[\xi])^T + (t_2 - t_1) \cdot (0, \mathbf{y}_i[2], \ldots, \mathbf{y}_i[\xi])^T + \ldots \\ &\quad + (t_\xi - t_{\xi-1}) \cdot (0, \ldots, 0, \mathbf{y}_i[\xi])^T + (1 - t_\xi) \cdot (0, 0, \ldots, 0)^T \\ &= (t_1 \cdot \mathbf{y}_i[1], t_2 \cdot \mathbf{y}_i[2], \ldots, t_\xi \cdot \mathbf{y}_i[\xi])^T = diag(t_1, t_2, \ldots, t_\xi) \cdot \mathbf{y}_i \end{aligned}$$

[5]A more formal, but maybe less intuitive argument for the reachability of $\mathbf{s}$ can be obtained by considering the "reversal" of the DFSA modeling the dynamics of the considered RAS that is defined by reversing all the stage transitions in the RAS process subnets while maintaining the same resource allocation vector for each processing stage. Then, it can be easily checked that in this reversed automaton, state $\mathbf{s}'$ is co-reachable; an event sequence $\hat{\sigma}'$ leading from $\mathbf{s}'$ to state $\mathbf{0}$ is readily obtained by reversing σ'. Since $\mathbf{s} \leq \mathbf{s}'$, Property 1 implies that state $\mathbf{s}$ is co-reachable in the reversed DFSA, as well; *i.e.*, in the reversed DFSA there exists an event sequence $\hat{\sigma}$ leading from $\mathbf{s}'$ to $\mathbf{0}$. But then, reversing the event sequence $\hat{\sigma}$ we obtain an event sequence σ that leads from state $\mathbf{0}$ to state $\mathbf{s}$ in the original DFSA.

Furthermore,

$$\sum_{i=1}^{l} \lambda_i \cdot (\sum_{j=1}^{\xi+1} (t_j - t_{j-1}) \cdot \mathbf{y}_{i,j-1}) =$$

$$= \sum_{i=1}^{l} \lambda_i \cdot (diag(t_1, t_2, \ldots, t_\xi) \cdot \mathbf{y}_i) = diag(t_1, t_2, \ldots, t_\xi) \cdot \sum_{i=1}^{l} \lambda_i \cdot \mathbf{y}_i = \mathbf{x}_2 \tag{11}$$

From the expansion of the telescopic series below, it can be seen that:

$$\sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{\xi+1} (t_j - t_{j-1}) = \sum_{i=1}^{l} \lambda_i \cdot 1 = 1 \tag{12}$$

Equations (8–9) also imply that:

$$\lambda_i \cdot (t_j - t_{j-1}) \geq 0, \quad \forall i = 1, \ldots, l, \ \forall j = 1, \ldots, \xi + 1 \tag{13}$$

But then, Equations (11–13) and the fact that $\mathbf{y}_{ij} \in \mathcal{X}, \ \forall i = 1, \ldots, l, \ \forall j = 0, \ldots, \xi$, imply that $\mathbf{x}_2 \in Conv(\mathcal{X})$. ■

Next, let $P_{S_{rs}} = Conv(S_{rs})$ and set $P^{+}_{S_{rs}} \equiv \{\mathbf{x} \in \mathbb{R}^\xi \mid \exists \mathbf{y} \in P_{S_{rs}} \text{ s.t. } \mathbf{x} \leq \mathbf{y}\}$. Then, the following proposition is an immediate implication of Lemmas 1 and 2.

Proposition 1: In the considered RAS class, $P^{+}_{S_{rs}} \cap (\mathbb{R}^{+}_{0})^\xi = P_{S_{rs}}$.

When combined with the non-negativity of the RAS states, Proposition 1 further implies that, in the context of the classification problem that is considered in this work, the existence conditions of Theorems 1 and 5 are equivalent.

Having established the conditions for the existence of simple and disjunctive linear classifiers for the problem at hand, the rest of this document focuses upon the computation of minimum-size classifiers from the two considered classes. Furthermore, in order to alleviate the notation, in the sequel we shall set $\mathcal{S} \equiv \overline{Proj(\bar{S}_{rs})}$ and $\mathcal{U} \equiv \overline{Proj(\bar{S}^{b}_{ru})}$; this substitution will also allow a more explicit connection of the subsequent discussion to the concepts and the results that were developed in Section III.

IV. Computing a linear classifier of minimal size

In this section we provide a specific algorithm for the synthesis of minimal (simple) linear classifiers that will represent effectively and parsimoniously the maximally permissive DAP for any D/C-RAS instance Φ that admits such a classifier. We will restrict attention to classifiers with non-negative coefficients for the reasons explained in the previous section; however, all the presented results can be easily extended to the general case. Also, in the following discussion, the classified sets $\mathcal{S}$ and $\mathcal{U}$ can be respectively interpreted either as the reachable safe and the reachable boundary unsafe subspaces of the considered RAS, or as the images of these two subspaces that are obtained by the thinning process of [9], depending on the applying occasion.

A. Determining the appropriate structure for the sought classifier

The first step in the synthesis of the sought classifiers is the determination of the classifier structure that is suitable for the classification problem at hand. According to Theorem 5, this issue can be resolved by determining the existence of unsafe states embedded in $P_{\mathcal{S}}^{+}$. Indeed, it follows from Theorem 5 (applied to the individual elements of $\mathcal{U}$) that the set $\mathcal{E}_{\mathcal{S}}^{+}$ contains exactly all the unsafe states that cannot be individually separated from $\mathcal{S}$ by a linear inequality with non-negative coefficients. Testing whether $\mathbf{u} \in \mathcal{E}_{\mathcal{S}}^{+}$, for any $\mathbf{u} \in \mathcal{U}$, is equivalent to checking if $sep_1^{+}(\mathcal{S}, \{\mathbf{u}\}) = \emptyset$, *i.e.*, checking the infeasibility of the corresponding problem (4).

If no embedded unsafe state has been found after this test has been iterated over the entire set $\mathcal{U}$, then the algorithm described in Section IV-B can be applied to find an optimal simple linear classifier. Otherwise, one may seek an optimal disjunctive linear classifier using the algorithm described in Section V.

B. A B&B algorithm for the computation of a linear classifier of minimal size

This section briefly overviews an algorithm for computing a structurally minimal linear classifier, originally developed in [10], [11]. The presented material is provided for completeness, but it also defines a basis for the development of the algorithm in Section V, that computes an optimal disjunctive linear classifier and constitutes one of the main contributions of this work.

Following Definition 8, a simple linear classifier for sets $\mathcal{S}$ and $\mathcal{U}$ consists of a set of c linear inequalities, each of which separates $\mathcal{S}$ from a subset $\mathcal{U}_j \subseteq \mathcal{U}$, for $j = 1, \ldots, c$, while $\cup_{j=1}^{c} \mathcal{U}_j = \mathcal{U}$. Hence, the problem of finding a minimal simple linear classifier for $\mathcal{S}$ and $\mathcal{U}$ can be reformulated as that of finding a minimal set cover of $\mathcal{U}$ by 1^{+}-separable subsets of it. A Branch & Bound (B&B) scheme [36] can be applied to efficiently explore all feasible coverings of set $\mathcal{U}$ and return the best one. The B&B mechanism reduces the problem to a number of smaller sub-problems, such that the solution sets of the sub-problems form a partition of the overall solution set. The branching process defines a tree structure with its nodes (known as "branching nodes") corresponding to the sub-problems.

Each sub-problem is associated with a different ordered collection $\langle \mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_c \rangle$ of disjoint subsets of $\mathcal{U}$. If $sep_1^{+}(\mathcal{S}, \mathcal{U}_j) = \emptyset$ for some $j = 1, \ldots, c$, the sub-problem is infeasible and the node is pruned. If, on the contrary, $sep_1^{+}(\mathcal{S}, \mathcal{U}_j) \neq \emptyset$ for all $j = 1, \ldots, c$, a system of inequalities $(\mathbf{A}, \mathbf{b})$ such that $(\mathbf{A}(j, \cdot), b_j) \in sep_1^{+}(\mathcal{S}, \mathcal{U}_j)$, for $j = 1, \ldots, c$, defines a linear classifier separating $\mathcal{S}$ from $\cup_{j=1}^{c} \mathcal{U}_j$.

Now, let $\Sigma = \mathcal{U} \setminus \cup_{j=1}^{c} \mathcal{U}_j$. If $\Sigma = \emptyset$ or $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{u}) = 0$, $\forall \mathbf{u} \in \Sigma$, the linear classifier $(\mathbf{A}, \mathbf{b})$ separates $\mathcal{S}$ from $\mathcal{U}$, thus providing a feasible solution of the problem. On the other hand, if $\exists \mathbf{u} \in \Sigma$ such that $\gamma_{\mathbf{A},\mathbf{b}}(\mathbf{u}) = 1$, the algorithm will attempt to accommodate it by branching the current node into $c + 1$ children nodes with the following assignments:

- $\langle \mathcal{U}_1 \cup \{\mathbf{u}\}, \mathcal{U}_2, \ldots, \mathcal{U}_c \rangle$
- $\ldots$
- $\langle \mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_c \cup \{\mathbf{u}\} \rangle$

- $\langle \mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_c, \{\mathbf{u}\} \rangle$

The rationale of the above branching scheme is that in order to find a full solution, either one of the scalar linear inequalities can be adjusted to exclude $\mathbf{u}$ (as well as the corresponding $\mathcal{U}_j$) or a new inequality must be introduced for this purpose. Notice that the first c children nodes have a collection of c subsets, as the parent node, while the last one has one more.

The branching process starts with $c = 1$ and $\mathcal{U}_1$ containing a single unsafe state, and progressively augments the number and contents of the $\mathcal{U}_j$ subsets to explore all possible partitions. As a result, the set of feasible solutions can only be reduced going down the branching tree, and the nodal costs, that are defined by the size of the corresponding solutions, are non-decreasing, since more inequalities are employed by the (partially) constructed classifiers. Obviously, an infeasible node can be safely eliminated, since none of its descendants can provide feasible solutions. Furthermore, adopting a node visit strategy that considers the unprocessed nodes by non-decreasing values of c guarantees that the first complete solution found is minimal (see also Theorem 1 in [11]).

The core problem at each node is to establish whether a given $\mathcal{U}_j$ subset is linearly separable from $\mathcal{S}$ with a single linear inequality. This can be done by checking if $sep_1^+(\mathcal{S} \cup \mathcal{U}_j^*, \mathcal{U}_j) \neq \emptyset$, where $\mathcal{U}_j^* = \bigcup_{l=j+1}^{c} \mathcal{U}_l$ is introduced in order to avoid variable permutations corresponding to the same solution; for further details, the reader is referred to [11].

As demonstrated in [11], the algorithm outlined above can be more efficient, in terms of the computational time that is required to obtain an optimal solution, than the alternative algorithms that are based on a monolithic Mixed Integer Programming (MIP) formulation of the underlying separation problem (like, *e.g.*, those in [12], [9]).

V. Computing a disjunctive linear classifier of minimal size

The B&B algorithm of Section IV-B can be adapted to the more complex problem of finding an optimal disjunctive linear classifier for the separation of the state sets $\mathcal{S}$ and $\mathcal{U}$, in the case that $\mathcal{E}_{\mathcal{S}}^+ \neq \emptyset$, by exploiting Theorems 2-3. We remind the reader that these theorems establish that set $\mathcal{S}$ can be partitioned into subsets that are individually separable from $\mathcal{U}$ using a different simple linear classifier for each of them. The basic requirement for the partition of $\mathcal{S}$ is that $\mathcal{E}_{\mathcal{S}_i}^+ = \emptyset$ for $i = 1, \ldots, p$. To find an optimal partition of $\mathcal{S}$, we can apply an incremental scheme similar to that described in the previous section for both sets $\mathcal{S}$ and $\mathcal{U}$, accounting for a greater number of states for the underlying search process, as the levels of the B&B tree increase. As in the previous algorithm, we separate the combinatorial problem of exploring the possible partitions of $\mathcal{S}$ and $\mathcal{U}$ from that of looking for the set of constraints that achieve separation.

The resulting algorithm, to be referred to as (the) *ExactClassifier*, is described by the pseudo-code of Figure 1, whose various procedures are explained in detail in the following subsections. In the sequel, each node of the underlying search tree is associated to a sub-problem Π, defined as follows.

Definition 10: Let $\mathcal{S}$ and $\mathcal{U}$ be the sets introduced in Section III-A. Let also $\mathcal{S}_i$, $i = 1, \ldots, p$, be such that $\bigcup_{i=1}^{p} \mathcal{S}_i \subseteq \mathcal{S}$, and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, $\forall i \neq j$. Finally, let $\mathcal{W}_i$, $i = 1, \ldots, p$, be such that $W_i = \langle \mathcal{U}_{i,1}, \ldots, \mathcal{U}_{i,c_i} \rangle$ is an ordered collection of disjoint subsets of $\mathcal{U}$, *i.e.*, for $i = 1, \ldots, p$, $\bigcup_{j=1}^{c_i} \mathcal{U}_{i,j} \subseteq \mathcal{U}$ and $\mathcal{U}_{i,j} \cap \mathcal{U}_{i,k} = \emptyset$, $\forall k \neq j$. Then, the ordered collection $\Pi = \langle (\mathcal{S}_1, W_1), \ldots, (\mathcal{S}_p, W_p) \rangle$ defines a sub-problem w.r.t. the classification of $\mathcal{S}$ and $\mathcal{U}$. ■

procedure EXACTCLASSIFIER($\mathcal{S}$, $\mathcal{U}$)
 $(\mathbf{s}, \mathbf{u}) := PickInitialStatePair(\mathcal{S}, \mathcal{U})$;
 $\Pi := \langle (\{\mathbf{s}\}, \langle \{\mathbf{u}\} \rangle) \rangle$;
 $C_L := Solve(\Pi)$; $LB := Size(C_L)$;
 $C_U^* := GreedyClassifier(\Pi)$; $UB^* := Size(C_U^*)$;
 $List := \{(\Pi, LB, UB^*)\}$;
 while $List \neq \emptyset$ **do**
  $(\Pi, LB, UB) := Extract(List)$;
  if $LB < UB^*$ **then**
   $(\mathbf{x}, k) := ChooseBranchingState(\Pi)$;
   $\{\Pi_1, \ldots, \Pi_n\} := Branch(\Pi, \mathbf{x}, k)$;
   for $i := 1$ to n **do**
    $C_L := Solve(\Pi_i)$; $LB := Size(C_L)$;
    $C_U := GreedyClassifier(\Pi_i)$; $UB := Size(C_U)$;
    if $UB < UB^*$ **then**
     $C_U^* := C_U$; $UB^* := UB$;
    end if
    if $LB < UB^*$ **then**
     $List := List \cup \{(\Pi_i, LB, UB)\}$;
    end if
   end for
  end if
 end while
 return C_U^*;
end procedure

Figure 1. Pseudo-code of the algorithm *ExactClassifier*

A. Node processing

Each node is processed as follows. First of all, function *Solve*(Π) determines whether the assignments in Π allow feasible solutions. This function returns *NULL* if Π is infeasible, or a disjunctive linear classifier C_L solving Π otherwise. In more detail, each pair $(\mathcal{S}_i, \mathcal{U}_{i,j})$, for $i = 1, \ldots, p$, $j = 1, \ldots, c_i$, defines an elementary separation problem involving a single linear inequality, whose feasibility can be easily checked with the test of Eq. (4). If there exists (i, j) such that $sep_1^+(\mathcal{S}_i \cup \mathcal{U}_{i,j}^*, \mathcal{U}_{i,j}) = \emptyset$, where $\mathcal{U}_{i,j}^* = \bigcup_{l=j+1}^{c_i} \mathcal{U}_{i,l}$, the sub-problem Π is infeasible. On the other hand, if $sep_1^+(\mathcal{S}_i \cup \mathcal{U}_{i,j}^*, \mathcal{U}_{i,j}) \neq \emptyset$ for all

$i = 1, \ldots, p$, $j = 1, \ldots, c_i$, then a disjunctive linear classifier $\{(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}), i = 1, \ldots, p\}$ can be defined for $\bigcup_{i=1}^{p} \mathcal{S}_i$ and $\bigcap_{i=1}^{p} \left(\bigcup_{j=1}^{c_i} \mathcal{U}_{i,j} \right)$, with $\mathbf{A}^{(i)} = [\mathbf{a}_1^{(i)} \ldots \mathbf{a}_{c_i}^{(i)}]^T$, and $b^{(i)} = [b_1^{(i)} \ldots b_{c_i}^{(i)}]^T$, for $i = 1, \ldots, p$, where $({\mathbf{a}_j^{(i)}}^T, b_j^{(i)}) \in sep_1^+(\mathcal{S}_i \cup \mathcal{U}_{i,j}^*, \mathcal{U}_{i,j})$.

If Π is infeasible, the node is immediately pruned, since its descendants in the branching tree will also be infeasible. Otherwise, the size $J = (2\xi + 1) \sum_{i=1}^{p} c_i + (p - 1)$ of C_L (returned by function $Size$) provides a lower bound (LB) to the size of any feasible solution belonging to $sep_D^+(\mathcal{S}, \mathcal{U})$ and compatible with the node assignments described by Π. Such a solution can be obtained by extending the assignments in Π to accommodate the currently unassigned states, possibly introducing additional linear inequalities. This is precisely what the heuristic algorithm *GreedyClassifier*, described in Section V-C, does. The solution C_U provided by the heuristic algorithm determines an upper bound (UB) on the size of the optimal classifier. The smallest encountered UB is stored into UB*, and the corresponding solution C_U is stored in C_U^*. Notice that a node with LB greater than or equal to UB* can be pruned, because it does not admit solutions better than C_U^*.

After its processing, the node is stored in an ordered list ($List$), which defines the node visit strategy. Nodes are extracted by function $Extract$ in order of increasing LB, then by increasing UB (when available), and finally by decreasing number of assigned states, calculated as $\sum_{i=1}^{p} \left(|S_i| + \frac{1}{p} \sum_{j=1}^{c_i} |U_{i,j}| \right)$. This ordering guarantees that as soon as the currently processed node has an LB greater than or equal to UB*, the algorithm can be terminated and the solution stored in C_U^* is optimal.

B. The branching process

The branching process evolves as follows. The initial node is defined by $\{(\mathcal{S}_1, W_1)\}$ with $W_1 = \langle \mathcal{U}_{1,1} \rangle$ and $|\mathcal{S}_1| = |\mathcal{U}_{1,1}| = 1$, and it corresponds to a separation problem that involves only one safe and one unsafe state; these two states are picked out with function $(\mathbf{s}, \mathbf{u}) = PickInitialStatePair(\mathcal{S}, \mathcal{U})$, described below. Notice that the feasibility of the initial sub-problem is ensured by Properties 1-2, which imply that $\nexists (\mathbf{s}, \mathbf{u})$ such that $\mathbf{u} \leq \mathbf{s}$, as required by Theorem 6. The heuristic solution obtained for the initial node is stored as the current best, and its size defines the initial UB*. Also, the node itself is inserted in the (initially empty) node list.

As long as the node list is not empty, the first node is extracted for branching. Branching consists in picking out a – safe or unsafe – unassigned state and assigning it to the subsets of the current node in all possible ways. More precisely, let $\mathcal{X}_0 = \mathcal{S} \setminus \bigcup_{i=1}^{p} \mathcal{S}_i$ be the set of the unassigned safe states and $\mathcal{X}_i = \mathcal{U} \setminus \bigcup_{j=1}^{c_i} \mathcal{U}_{i,j}$ be the set of the unassigned unsafe states with respect to $\mathcal{S}_i$, $i = 1, \ldots, p$. Let $\mathbf{x} \in \mathcal{X}_k$, for some $k \in \{0, 1, \ldots, p\}$, be the selected branching state. Then, procedure $Branch(\Pi, \mathbf{x}, k)$ returns a collection of children nodes generated according to the following scheme:

1) If a safe state $\mathbf{s} \in \mathcal{X}_0$ is selected for branching, $p + 1$ children nodes are generated, defined by
   - $\langle (\mathcal{S}_1 \cup \{\mathbf{s}\}, W_1), \ldots, (\mathcal{S}_p, W_p) \rangle$
   - $\ldots$

- $\langle(\mathcal{S}_1, W_1), \ldots, (\mathcal{S}_p \cup \{\mathbf{s}\}, W_p)\rangle$
- $\langle(\mathcal{S}_1, W_1), \ldots, (\mathcal{S}_p, W_p), (\{\mathbf{s}\}, \langle\{\mathbf{u}\}\rangle)\rangle$

where the unsafe state $\mathbf{u}$ is picked out with function $PickUnsafeState(\mathbf{s}, \mathcal{U})$, described in the sequel.

2) If an unsafe state $\mathbf{u} \in \mathcal{X}_k$ is selected ($k > 0$), $c_k + 1$ children nodes are generated that are identical to the parent node except from $W_k = \langle \mathcal{U}_{k,1}, \ldots, \mathcal{U}_{k,c_k} \rangle$ which is modified, respectively, into:
   - $\langle \mathcal{U}_{k,1} \cup \{\mathbf{u}\}, \ldots, \mathcal{U}_{k,c_k} \rangle$
   - $\ldots$
   - $\langle \mathcal{U}_{k,1}, \ldots, \mathcal{U}_{k,c_k} \cup \{\mathbf{u}\} \rangle$
   - $\langle \mathcal{U}_{k,1}, \ldots, \mathcal{U}_{k,c_k}, \{\mathbf{u}\} \rangle$

   After branching, the parent node is eliminated, and each child node is processed.

Different rationales can be employed to select the branching state (function $ChooseBranchingState$). Ideally, the branching state should be selected as the most critical state to accommodate within the current node's assignments. This should tend to drive the branching process to quickly identify the appropriate number of the $\mathcal{S}_i$ and $\mathcal{U}_{i,j}$ subsets needed to solve the overall problem. Along this line, the ideal branching state would be one that forces an increase in the node cost (due to the generation of a new subset). To save computational time, this policy is approximated by choosing the first state that triggers a cost increment when executing the heuristic algorithm at the current node.

Function $(\mathbf{s}, \mathbf{u}) = PickInitialStatePair(\mathcal{S}, \mathcal{U})$ is applied at the root node, and aims at providing the algorithm with the "most difficult" pair of safe-unsafe states to accommodate initially. Intuitively, this pair of states should be such that the set $sep_1^+(\mathbf{s}, \mathbf{u})$ of all separating hyperplanes is the smallest possible. For this purpose, $\mathbf{u}$ is chosen in $\mathcal{E}_{\mathcal{S}}^+$ and $\mathbf{s}$ in $\mathcal{S}$ so that $|\mathbf{s}| \geq |\mathbf{u}|$ and the angle between $\mathbf{s}$ and $\mathbf{u}$ is minimum. A similar idea is reapplied whenever branching is performed with respect to a safe state $\mathbf{s}$, and a child node is generated with $\mathcal{S}_{p+1} = \{\mathbf{s}\}$. In this case, function $\mathbf{u} = PickUnsafeState(\mathbf{s}, \mathcal{U})$ picks out an embedded unsafe state among those with $|\mathbf{u}| \leq |\mathbf{s}|$ that has the smallest angle with $\mathbf{s}$.

C. Computation of the upper bound

This section describes the heuristic algorithm *GreedyClassifier*, which tries to complete the current node's assignments, by fitting in them the unassigned states with a greedy policy (the corresponding linear inequalities are updated, if necessary). We remind the reader that, in the context of the considered algorithm, the assignments instantiated by the current node are represented by $\Pi = \langle(\mathcal{S}_1, W_1), \ldots, (\mathcal{S}_p, W_p)\rangle$ where $W_i = \langle \mathcal{U}_{i,1}, \ldots, \mathcal{U}_{i,c_i} \rangle$. States that do not fit in any of the current assignments are placed into new subsets. The heuristic always provides a complete, but generally sub-optimal solution, that can be used to improve the UB. Figure 2 reports the pseudo-code of the heuristic.

Each step of the algorithm picks out an index k from $\{0, \ldots, p\}$ and a state $\mathbf{x} \in \mathcal{X}_k$ according to a suitable policy, defined by function $(k, \mathbf{x}) = PickStateAndSubset(\Pi)$. Then, it performs one of the

Preprint submitted to IEEE Transactions on Automatic Control. Received: May 10, 2013 23:51:11 PST

procedure GREEDYCLASSIFIER(Π)
while $\bigcup_{i=0}^{p} \mathcal{X}_i \neq \emptyset$ **do**
$(k, \mathbf{x}) := PickStateAndSubset(\Pi)$;
if $k = 0$ **then**
$i := 1$; FOUND := FALSE;
while NOT FOUND AND $i \leq p$ **do**
if $sep_1^+(\mathcal{S}_i \cup \{\mathbf{x}\}, \mathcal{U}_{i,j}) \neq \emptyset, \forall j = 1, \ldots, c_i$ **then**
FOUND := TRUE;
else
$i := i + 1$;
end if
end while
if FOUND **then**
$\mathcal{S}_i := \mathcal{S}_i \cup \{\mathbf{x}\}$;
else
$p := p + 1$; $\mathcal{S}_p := \{\mathbf{x}\}$;
$\mathbf{u} := PickUnsafeState(\mathbf{x}, \mathcal{U})$;
$\mathcal{U}_{p,1} := \{\mathbf{u}\}$;
end if
else
$j := 1$; FOUND := FALSE;
while NOT FOUND AND $j \leq c_k$ **do**
if $sep_1^+(\mathcal{S}_k, \mathcal{U}_{k,j} \cup \{\mathbf{x}\}) \neq \emptyset$ **then**
FOUND := TRUE;
else
$j := j + 1$;
end if
end while
if FOUND **then**
$\mathcal{U}_{k,j} := \mathcal{U}_{k,j} \cup \{\mathbf{x}\}$;
else
$c_k := c_k + 1$; $\mathcal{U}_{k,c_k} := \{\mathbf{x}\}$;
if $sep_1^+(\mathcal{S}_k, \mathcal{U}_{k,c_k}) = \emptyset$ **then**
Exit("Node infeasible");
end if
end if
end if
end while
C := $Solve(\Pi)$;
return C;
end procedure

Figure 2. Pseudo-code of the heuristic algorithm *GreedyClassifier*

following assignments, depending on the type of state chosen (safe or unsafe) and on the value of k. If $k = 0$ (unassigned safe state):

a) add $\mathbf{x}$ to the first subset $\mathcal{S}_i$ (if one exists), such that $sep_1^+(\mathcal{S}_i \cup \{\mathbf{x}\}, \mathcal{U}_{i,j}) \neq \emptyset, \forall j = 1, \ldots, c_i$;

b) otherwise, set $p := p+1$ and insert $\mathbf{x}$ into a newly created empty subset $\mathcal{S}_p$; then, define $\mathcal{U}_{p,1} = \{\mathbf{u}\}$, where $\mathbf{u} = PickUnsafeState(\mathbf{x}, \mathcal{U})$.

If $k > 0$ (unassigned unsafe state):

c) add $\mathbf{x}$ to the first subset $\mathcal{U}_{k,j}$ (if one exists), such that $sep_1^+(\mathcal{S}_k, \mathcal{U}_{k,j} \cup \{\mathbf{x}\}) \neq \emptyset$;

d) otherwise, set $c_k := c_k + 1$ and insert $\mathbf{x}$ into a newly created empty subset $\mathcal{U}_{k,c_k}$.

The algorithm terminates when all states have been assigned, that is when $\mathcal{X}_i = \emptyset$ for all $i = 0, \ldots, p$.

The reader should notice that the starting assignment is always feasible, and that each additional assignment of a further state must preserve feasibility. The latter can be rapidly evaluated with the following sufficient condition, or, when this condition fails, by using the criterion given in Definition 7.

Property 7: Let $\bar{\mathcal{S}}$ and $\bar{\mathcal{U}}$ be two 1^+-separable state sets, and $(\mathbf{a}^T, b) \in sep_1^+(\bar{\mathcal{S}}, \bar{\mathcal{U}})$. Also, let $\mathbf{x} \in (\mathbb{R}_0^+)^\xi$. Then:

(i) $\mathbf{a}^T\mathbf{x} \leq b \implies (\mathbf{a}^T, b) \in sep_1^+(\bar{\mathcal{S}} \cup \{\mathbf{x}\}, \bar{\mathcal{U}})$

(ii) $\mathbf{a}^T\mathbf{x} \geq b + 1 \implies (\mathbf{a}^T, b) \in sep_1^+(\bar{\mathcal{S}}, \bar{\mathcal{U}} \cup \{\mathbf{x}\})$

Proof: These results follow immediately from the separability test of Definition 7. ■

When p is incremented and a new set $\mathcal{S}_p$ is introduced to accommodate an unassigned safe state $\mathbf{s}$ that does not fit into any of the previous $\mathcal{S}_i$ subsets, then, whatever (embedded) unsafe state $\mathbf{u}$ is chosen by function $PickUnsafeState$, it holds that $sep_1^+(\mathbf{s}, \mathbf{u}) \neq \emptyset$. This follows immediately from the following property.

Property 8: Let $\mathbf{s} \in (\mathbb{R}_0^+)^\xi$ and $\bar{\mathcal{U}}$ be a vector set in $(\mathbb{R}_0^+)^\xi$ such that $\nexists \mathbf{u} \leq \mathbf{s}$ in $\bar{\mathcal{U}}$. Then, $sep_L^+(\{\mathbf{s}\}, \bar{\mathcal{U}}) \neq \emptyset$.

Proof: This result follows from Theorem 5, upon noticing that $P_{\{\mathbf{s}\}}^+ = \{\mathbf{x} \in \mathbb{R}^\xi | \mathbf{x} \leq \mathbf{s}\}$, which further implies that $P_{\{\mathbf{s}\}}^+ \cap \bar{\mathcal{U}} = \emptyset$. ■

On the other hand, when c_i is incremented and a new set $\mathcal{U}_{i,c_i}$ is introduced to accommodate an unassigned unsafe state $\mathbf{u}$ that does not fit into any of the previous $\mathcal{U}_{i,j}$ subsets, then it is not automatically guaranteed that $\mathcal{S}_i$ will be separable from $\mathcal{U}_{i,c_i}$. In particular, the separability condition of Theorem 5 will not hold if $\mathbf{u} \in \mathcal{E}_{\mathcal{S}_i}^+$. In that case, the algorithm terminates without generating a solution. However, by adopting a suitable policy in the selection of the state to be assigned at each step, it is possible to guarantee that this condition never occurs. This selection policy will enforce that an embedded unsafe state is assigned to $\mathcal{U}_{i,c_i}$ only if the corresponding $\mathcal{S}_i$ has cardinality 1. To achieve this effect, *all* embedded unsafe states must be accommodated with respect to $\mathcal{S}_i$ before incrementing the cardinality of this set to any value higher than 1.

In line with the state assignment policy that was described in the previous paragraph, the heuristic algorithm *GreedyClassifier* proceeds as follows. Once a new subset $\mathcal{S}_p$ is created (with a single safe

state), all embedded unsafe states are accommodated with respect to it, possibly generating more than one $\mathcal{U}_{p,j}$ subset. This is feasible in view of Property 8. Furthermore, from this point on, any further addition of a safe state to $\mathcal{S}_p$, or of an unsafe state to any of the corresponding $\mathcal{U}_{p,j}$ subsets, will remain feasible. This is formally proven in Proposition 2 below; however, in order to establish this proposition, we need to introduce some further basic properties.

The next lemma implies that a linear classifier separating a set of safe states from a set of unsafe states can always be extended to account for further unsafe states (possibly requiring the increase in the number of necessary inequalities), provided that they are not embedded in the polyhedron of safe states.

Lemma 3: Let $\bar{\mathcal{S}}$ and $\bar{\mathcal{U}}$ be two disjoint vector sets in $(\mathbb{R}_0^+)^\xi$ such that $sep_L^+(\bar{\mathcal{S}},\bar{\mathcal{U}}) \neq \emptyset$. Let also $\mathbf{u} \in (\mathbb{R}_0^+)^\xi \setminus \bar{\mathcal{U}}$. Then, if $\mathbf{u} \notin P_{\bar{\mathcal{S}}}^+$, it also holds that $sep_L^+(\bar{\mathcal{S}},\bar{\mathcal{U}} \cup \{\mathbf{u}\}) \neq \emptyset$.

Proof: By Theorem 5, there is no state in $\bar{\mathcal{U}}$ inside the polyhedron $P_{\bar{\mathcal{S}}}^+$. By assumption, $\mathbf{u} \notin P_{\bar{\mathcal{S}}}^+$ also. Then, Theorem 5 holds also for sets $\bar{\mathcal{S}}$ and $\bar{\mathcal{U}} \cup \{\mathbf{u}\}$. ■

The next lemma ensures that a disjunctive linear classifier separating a set of safe states from a set of unsafe states can always be extended to account for further safe states (possibly requiring the increase in the number of necessary simple linear classifiers involved in the disjunction).

Lemma 4: Let $\bar{\mathcal{S}}$ and $\bar{\mathcal{U}}$ be two disjoint vector sets in $(\mathbb{R}_0^+)^\xi$ such that $(\mathbf{A}^{(1)}, \mathbf{b}^{(1)}) \in sep_L^+(\bar{\mathcal{S}},\bar{\mathcal{U}})$. Let also $\mathbf{s} \in (\mathbb{R}_0^+)^\xi \setminus \bar{\mathcal{S}}$, such that $\nexists \mathbf{u} \leq \mathbf{s}$ in $\bar{\mathcal{U}}$. Then, it also holds that $sep_D^+(\bar{\mathcal{S}} \cup \{\mathbf{s}\},\bar{\mathcal{U}}) \neq \emptyset$.

Proof: By Property 8, there exists a simple linear classifier $(\mathbf{A}^{(2)}, \mathbf{b}^{(2)})$ with $\mathbf{A}^{(2)}, \mathbf{b}^{(2)} \geq 0$ separating the (single) safe state $\mathbf{s}$ from $\bar{\mathcal{U}}$. Then the disjunctive linear classifier $\{(\mathbf{A}^{(1)}, \mathbf{b}^{(1)}), (\mathbf{A}^{(2)}, \mathbf{b}^{(2)})\}$ achieves the thesis. ■

Proposition 2: Let $\mathcal{S}$ and $\mathcal{U}$ be two disjoint vector sets in $(\mathbb{R}_0^+)^\xi$ such that $\nexists(\mathbf{s}, \mathbf{u}) \in \mathcal{S} \times \mathcal{U}$ with $\mathbf{u} \leq \mathbf{s}$. Then, the heuristic algorithm *GreedyClassifier* will always find a feasible solution $C_U \in sep_D^+(\mathcal{S},\mathcal{U})$ to the corresponding separation problem.

Proof: The algorithm initially defines set $\mathcal{S}_1 \subseteq \mathcal{S}$, such that $|\mathcal{S}_1| = 1$. Then, it finds a simple linear classifier separating $\mathcal{S}_1$ from $\mathcal{E}_{\mathcal{S}}^+$, whose existence is guaranteed by Property 8. Afterwards, it fits all unassigned safe and unsafe states, one by one, in the existing $\mathcal{S}_i$ and $\mathcal{U}_{i,j}$ subsets, or in newly created ones, if necessary. Each extension is feasible thanks to Lemmas 3 and 4. ■

The incremental process implemented by the heuristic algorithm is guaranteed to terminate, since only a finite number of safe state subsets are necessary (at most $|\mathcal{S}|$, each containing a single safe state).

Finally, we note that the heuristic can be stopped prematurely to save computational time, once the solution size UB reaches the current best UB*; at that point it is clear that the heuristic will not produce a better solution than the current best, and it is pointless to continue further. So, if such a case is encountered, UB is set to $+\infty$.

D. Function PickStateAndSubset

Function $(k, \mathbf{x}) = PickStateAndSubset(\Pi)$ establishes the policy according to which states are progressively included in the heuristic solution. Based on Proposition 2, the function enforces the following

order of state inclusion, while operating on the remaining unassigned states:

1) embedded unsafe states,
2) safe states,
3) remaining unsafe states.

The state selection in each of these categories can be performed either on a purely sequential basis, to reduce the computational load, or by looking first for states that are incompatible with the current assignments, so as to force an increase in the solution size. This second policy is generally costlier, but may provide a more appropriate state for branching purposes (recall that the branching state is selected as the first that triggers a size increase with respect to the LB of the current node). As a consequence, in the following experimental section, the first policy has been used to compute the upper bound, while the selection of the branching state has been performed by re-executing *GreedyClassifier* with the second policy, truncating the execution at the first increase of the solution size.

E. Heuristics for the estimation of sub-optimal solutions

The algorithm *GreedyClassifier* applied at the initial node rapidly provides a sub-optimal solution. However, extensive computational experience shows two important facts:

- The initial solution is generally significantly improved after the processing of a few further nodes.
- The order of examination of the states in the application of the heuristic algorithm *GreedyClassifier* at each node greatly influences the size of the solution found (yet, in the basic scheme, states are examined in a fixed order, namely, the order suggested by the storage of these states in the relevant data structure).

Building on these remarks, a modified scheme has been devised for an efficient estimation of a sub-optimal solution. This scheme is characterized as Randomized Truncated B&B (RT-B&B) in the following, and it is defined by a restart of the overall B&B algorithm several times, after the underlying search has generated a prescribed small number of nodes. At each restart, the best UB^* found so far is used as initial information (the greedy heuristic applied at each node exploits this information by stopping when UB^* is reached), and the state sets $\mathcal{S}$ and $\mathcal{U}$ are reshuffled to modify randomly the order of examination.

F. A brief remark on the computational complexity of the method

The minimal classifier design problem is essentially a combinatorial optimization problem [43]. It is expected that, from a worst-case point of view, the problem computational complexity will be super-polynomial w.r.t. the sizes of the two classified sets $\mathcal{S}$ and $\mathcal{U}$.[6] The proposed algorithm is based on a reduction of the considered problem to a variation of the Set Covering problem, and it constitutes an

[6]We want to clarify, however, that the super-polynomial complexity of the considered classifier design problem w.r.t. the classified sets $\mathcal{S}$ and $\mathcal{U}$ is only a conjecture; currently we lack formal complexity results for this problem.

implicit enumeration approach over the relevant solution space, that explores the space of partitions of the sets $\mathcal{S}$ and $\mathcal{U}$. In each of these partitions, first the safe set $\mathcal{S}$ is divided into subsets, and subsequently a partition of the unsafe set $\mathcal{U}$ is associated to each of these safe subsets. Each elementary pair of a safe and an unsafe subset defines a linear separation problem for which a single linear inequality is sought. Therefore, the computational cost of the sub-problem that is defined by each of the aforementioned partitions is given by the number of the pairs of subsets involved times the complexity of the LP problem that is defined by the separability test of Definition 7. Both of these factors are polynomial w.r.t. the cardinalities of the sets $\mathcal{S}$ and $\mathcal{U}$. Hence, the overall algorithm complexity is primarily determined by the total number of sub-problems that will be solved by the proposed B&B scheme; in the worst case, this number is exponential even for the B&B scheme that is used for the construction of minimal linear classifiers, as discussed in detail in [11].[7]

Different implicit enumeration methods, that take place on different subproblem spaces, can be applied to solve the classifier design problem that is considered in this work. The theoretical analysis of [11] points out the intrinsic superiority of the approach based on Set Covering with respect to alternative MIP-based approaches, due to the facts that the number of the (explicitly) explored sub-problems, while potentially exponential, is typically smaller for this method, and the subproblems themselves present a lower complexity. The computational results presented in Section VI-B confirm this advantage and reveal that the proposed method is tractable for problem instances of practical size.

VI. Computational developments

A. An illustrative example

We first consider a small illustrative example to demonstrate the flow of the computation and the effectiveness of the proposed algorithm. Let $\mathcal{S} = \{\mathbf{s}_i, i = 1, \ldots, 6\}$ and $\mathcal{U} = \{\mathbf{u}_i, i = 1, \ldots, 3\}$, where
$\mathbf{s}_1$ = [3 3 0 0 0 3], $\mathbf{s}_2$ = [1 3 0 0 1 3], $\mathbf{s}_3$ = [0 3 1 2 0 0],
$\mathbf{s}_4$ = [0 3 0 3 0 3], $\mathbf{s}_5$ = [0 3 0 2 1 3], $\mathbf{s}_6$ = [0 2 1 2 0 3],
$\mathbf{u}_1$ = [0 3 1 0 0 1], $\mathbf{u}_2$ = [0 0 0 3 1 0], $\mathbf{u}_3$ = [1 0 0 1 0 0].

The extended set of embedded unsafe states is not empty: $\mathcal{E}_{\mathcal{S}}^{+} = \{\mathbf{u}_3\}$. Indeed, it can be easily checked that $\mathbf{u}_3 < \frac{1}{3}\mathbf{s}_1 + \frac{2}{3}\mathbf{s}_3$. Hence, we are seeking a minimal-size disjunctive linear classifier for these two sets, using the algorithm that was presented in Section V. Initially, a pair of safe and unsafe states is picked out to generate the root node using function $PickInitialStatePair$. Not surprisingly, $\mathbf{u}_3$ is selected as the unsafe state of the pair. The selected safe state is $\mathbf{s}_1$, since, among the elements of $\mathcal{S}$ with $|\mathbf{s}| \geq |\mathbf{u}_3|$, $\mathbf{s}_1$ makes the smallest angle with $\mathbf{u}_3$. The algorithm generates 10 nodes as represented in Figure 3. The figure caption also reports the sub-problem definition at each node.

The heuristic *GreedyClassifier* applied at the initial node produces a feasible solution $\mathcal{S}_1 = \{\mathbf{s}_1, \mathbf{s}_2\}$, $\mathcal{U}_{1,1} = \mathcal{U}$, $\mathcal{S}_2 = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$, $\mathcal{U}_{2,1} = \{\mathbf{u}_1, \mathbf{u}_3\}$, and $\mathcal{U}_{2,2} = \{\mathbf{u}_2\}$, costing $UB = 40$. The solution is

[7]The number of possible partitions of a set can be characterized using the Stirling number of the second kind.

saved. Then, node 1 is branched over $\mathbf{s}_3$ (*i.e.*, the first safe marking that could not fit into $\mathcal{S}_1$), producing nodes 2 and 3. Node 2 is infeasible (any "$\leq$" inequality with non-negative coefficients satisfied by $\mathbf{s}_1$ and $\mathbf{s}_3$ is also satisfied by $\mathbf{u}_3$), and thus discarded. Node 3 is feasible, but the solution found by the heuristic *GreedyClassifier* does not improve on the current best. Therefore, the solution is not saved and the node is branched over $\mathbf{u}_2$ (for $p = 2$) producing nodes 4 and 5. At node 4, the heuristic *GreedyClassifier* produces a feasible solution costing $UB = 40$, equal to the best one found so far. Therefore, again, the solution is not saved and node 4 is branched over $\mathbf{u}_1$ (for $p = 2$) producing nodes 6 and 7. Node 5 is discarded since it has an LB equal to the best UB so far. For node 6 the heuristic *GreedyClassifier* obtains a solution costing $UB = 42$ (not improving the current best). The node is branched over $\mathbf{s}_5$ producing nodes 8, 9, and 10. Node 7 is discarded since LB is equal to the current best. Nodes 8 and 9 are infeasible, and thus discarded. Finally, node 10 is discarded since LB is greater than the current best. The heuristic solution found at node 1, and never improved later, is now certified to be optimal. This solution corresponds to the disjunctive linear classifier:

$$\mathbf{A}^{(1)} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix},\ \mathbf{b}^{(1)} = 0,\ \mathbf{A}^{(2)} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 13 & 3 & 3 & 0 & 0 & 1 \end{bmatrix},\ \mathbf{b}^{(2)} = \begin{bmatrix} 3 \\ 12 \end{bmatrix}.$$

Furthermore, since $\gamma_{\mathbf{A}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) = 1$, $\forall \mathbf{x} \in \{\mathbf{s}_1, \mathbf{s}_2\}$, and $\gamma_{\mathbf{A}^{(2)},\mathbf{b}^{(2)}}(\mathbf{x}) = 1$, $\forall \mathbf{x} \in \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$, and both functions are equal to 0 in all other states, it is trivial to verify that $\delta(\mathbf{x}) = \gamma_{\mathbf{A}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) \vee \gamma_{\mathbf{A}^{(2)},\mathbf{b}^{(2)}}(\mathbf{x})$ is such that $\delta(\mathbf{s}) = 1$, $\forall \mathbf{s} \in S$, and $\delta(\mathbf{u}) = 0$, $\forall \mathbf{u} \in U$. The size of the obtained classifier is equal to $(2 \cdot 6 + 1) \cdot 3 + 1 = 40$; *i.e.*, this classifier must perform 40 elementary operations in order to classify any given vector.

B. Computational experiments

In this section we present some further results that demonstrate the efficiency of the proposed algorithm. More specifically, in Tables I-IV, we report results concerning the performance of our B&B algorithm on three groups of benchmark RAS instances of increasing size and complexity that were previously considered in [38]. Both the B&B algorithm and the RT-B&B heuristic were tested on an Intel 3.4 GHz processor with 16 GB of RAM, and the LP formulations addressed by each subproblem were solved with CPLEX 11.1. In Tables I, II and IV, Columns $2-7$ describe the problem instance characteristics in terms of the cardinalities of the original state sets S_{rs} and S_{ru}, and their trimmed versions $\mathcal{S}$ and $\mathcal{U}$, the dimension ξ of the trimmed state space, and the number of embedded unsafe states $|\mathcal{E}_S^+|$. Notice that, as in [38], we have applied the trimming of the original state sets to identify classifiers with non-negative coefficients, which is extremely powerful and provides a decisive step towards the practical solvability of the classification problem. The performance of the tested algorithms is described in terms of the following indices: $c = \sum_{i=1}^{p} c_i$ is the overall number of inequalities employed by the obtained solution, p is the number of subsets of $\mathcal{S}$ that are employed by the obtained solution, and J is the size of the obtained solution, *i.e.*, a measure of the computational cost for any (on-line) state classification that is attempted

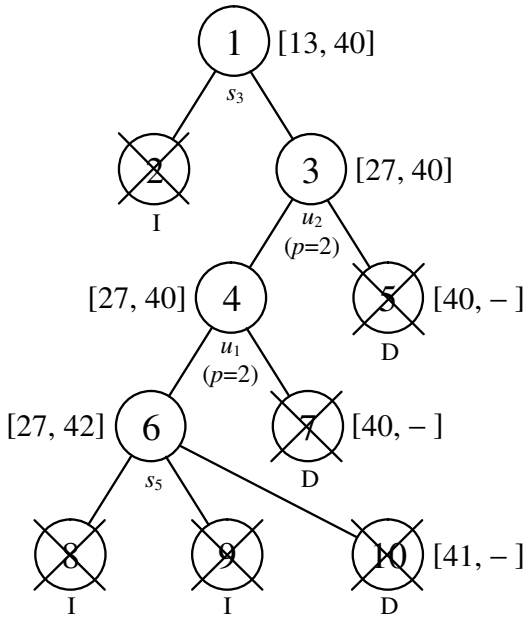

Figure 3. The "branching tree" for the illustrative example of Section VI-A. Each node is labeled with $[LB, UB]$ (UB is omitted when LB exceeds the best solution so far). Labels I and D identify nodes discarded due to infeasibility or dominance, respectively. Branched nodes are also labeled with the branching state. Using the notation of Def. 10, the depicted nodes are defined by the following sub-problems: 1) $\langle(\{s_1\},\{u_3\})\rangle$, 2) $\langle(\{s_1,s_3\},\{u_3\})\rangle$, 3) $\langle(\{s_1\},\{u_3\}),(\{s_3\},\{u_3\})\rangle$, 4) $\langle(\{s_1\},\{u_3\}),(\{s_3\},\{u_2,u_3\})\rangle$, 5) $\langle(\{s_1\},\{u_3\}),(\{s_3\},(\{u_2\},\{u_3\}))\rangle$, 6) $\langle(\{s_1\},\{u_3\}),(\{s_3\},\{u_1,u_2,u_3\})\rangle$, 7) $\langle(\{s_1\},\{u_3\}),(\{s_3\},(\{u_1\},\{u_2,u_3\}))\rangle$, 8) $\langle(\{s_1,s_5\},\{u_3\}),(\{s_3\},\{u_1,u_2,u_3\})\rangle$, 9) $\langle(\{s_1\},\{u_3\}),(\{s_3,s_5\},\{u_1,u_2,u_3\})\rangle$, 10) $\langle(\{s_1\},\{u_3\}),(\{s_3\},\{u_1,u_2,u_3\}),(\{s_5\},\{u_3\})\rangle$.

by the derived classifier. In addition, the tables provide the number of branching nodes BN and the CPU time, in seconds, that is required to solve the problem by each of the two methods proposed herein.

Table I compares the results of the exact method of [38], obtained by applying a MIP solver with a time limit of 120 minutes, with those of the proposed B&B method. Exact solutions with respect to J are reported in bold face. For the remaining cases, where the time limit was reached, the optimality of the result is not guaranteed and the best available result is reported. Tables II and III compare the heuristic algorithm of [38] with the proposed B&B method (with a 15 minute time limit) and with the RT-B&B heuristic, respectively. The particular implementation of the latter algorithm employed 10 runs of the B&B algorithm truncated after the first 10 branching nodes. Finally, the RT-B&B heuristic has also been tested on some larger instances (Table IV).

All the smaller instances (Table I) could be solved to optimality in a few seconds by the B&B method, confirming the exact results and improving the heuristic ones obtained in [38].

Regarding the medium-size instances, Table II indicates that an exact solution could be obtained in two cases within the time limit. In the remaining cases, the combined action of the branching process and the heuristic *GreedyClassifier* in the B&B algorithm allows strong improvements with respect to

Table I
COMPUTATIONAL RESULTS ON THE SMALL-SIZE BENCHMARK INSTANCES

	Problem characteristics						Exact method of [38]			Prop. B&B method				
#	$\|S_{rs}\|$	$\|S_{ru}\|$	$\|\mathcal{S}\|$	$\|\mathcal{U}\|$	ξ	$\|\mathcal{E}_S^+\|$	c	p	J	c	p	J	BN	CPU
222	109	1 379	34	35	17	2	3	2	**107**	3	2	**107**	21	0.295
224	593	2 756	31	32	14	2	6	2	176	4	2	**118**	72	0.710
247	623	1 133	69	40	26	2	3	2	**161**	3	2	**161**	76	1.811
272	342	857	30	23	11	5	5	2	117	4	2	**94**	191	1.608
309	758	2 611	71	31	19	5	4	2	**158**	4	2	**158**	206	2.910
381	222	1 244	56	68	27	1	3	2	**167**	3	2	**167**	110	4.373
1010	196 021	11 451	46	15	8	1	7	2	121	3	2	**53**	86	0.739

the heuristic of [38] (occasionally, more than one order of magnitude). On the other hand, the RT-B&B algorithm (Table III), thanks to the combined influence of randomization and branching, improves (or, in the worst case, equals) the best results obtained by the B&B algorithm, in a much smaller time. Interestingly enough, the set $\mathcal{S}$ is partitioned in 2 in most cases, and there is only one instance where a high number of subsets is required in the heuristic solution. This suggests that the proposed approaches are particularly effective in identifying and exploiting the underlying geometric structure of the problem.

Table II
COMPUTATIONAL RESULTS ON THE MEDIUM-SIZE BENCHMARK INSTANCES

	Problem characteristics						Heur. method of [38]			Prop. B&B method				
#	$\|S_{rs}\|$	$\|S_{ru}\|$	$\|\mathcal{S}\|$	$\|\mathcal{U}\|$	ξ	$\|\mathcal{E}_S^+\|$	c	p	J	c	p	J	BN	CPU
445	94 431	72 238	849	71	17	52	140	29	4 929	55	21	1 946	880	900.000
447	115 766	28 510	95	12	11	3	17	11	402	5	2	**117**	399	3.079
474	10 913	96 361	404	163	41	10	69	43	5 770	10	2	832	1 495	900.000
476	6 596	129 036	527	221	39	6	90	42	7 152	11	2	871	1 075	900.000
605	104 550	49 620	1 937	67	15	42	8	7	255	5	3	**158**	2 029	357.086
621	42 571	69 016	3 754	188	30	1	116	22	7 098	12	2	734	72	900.000
1026	757 699	700 781	2 120	89	21	13	170	19	7 329	13	2	561	640	900.000

Table III
RESULTS OF THE RT-B&B ALGORITHM ON THE MEDIUM-SIZE BENCHMARK INSTANCES

	RT-B&B ($BN = 10 \times 10$)			
#	c	p	J	CPU
445	50	14	1 764	60.105
447	5	2	**117**	3.707
474	9	2	749	36.978
476	10	2	792	48.387
605	5	3	**158**	64.771
621	12	2	734	742.441
1026	11	2	475	214.415

Table IV illustrates the results obtained on the larger instances by the RT-B&B heuristic. The latter proves able to tackle these instances in no more than a few hours, providing solutions that are not

Preprint submitted to IEEE Transactions on Automatic Control. Received: May 10, 2013 23:51:11 PST

necessarily optimal, but which are quite compact, especially with respect to the employed number of subsets p.

Table IV
COMPUTATIONAL RESULTS ON THE LARGE-SIZE BENCHMARK INSTANCES

	Problem characteristics						RT-B&B ($BN = 10 \times 10$)			
#	$\lvert S_{rs}\rvert$	$\lvert S_{ru}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{U}\rvert$	ξ	$\lvert\mathcal{E}_S^+\rvert$	c	p	J	CPU
558	21 099	906 478	741	225	48	13	14	3	1 361	84.377
565	65 992	2 105 590	2 314	708	56	55	56	8	6 336	831.935
642	34 695	1 773 193	1 697	427	55	19	22	4	2 446	392.763
643	118 470	1 253 425	3 432	612	54	7	42	3	4 581	1 764.604
897	53 080	1 410 311	5 560	1 046	74	13	30	2	4 472	5 029.219
1034	1 622 861	2 600 349	15 783	257	37	15	17	4	1 279	7 554.122
1040	396 931	1 146 292	13 742	559	51	13	36	3	3 711	19 149.563

VII. CONCLUSIONS

The work presented in this paper has complemented some recent endeavors that seek the implementation of the maximally permissive deadlock avoidance policy (DAP) for various complex resource allocation systems as a classifier that encodes the policy-induced dichotomy of the underlying behavioral space into admissible and inadmissible subspaces. The presented results have provided (i) succinct conditions regarding the possibility of expressing the aforementioned classifier as a set of linear inequalities in the state variables, and (ii) an efficient customized algorithm for the synthesis of pertinent non-linear classifiers that implement the target DAP with minimum run-time computational overhead, in the case that a linear classifier-based representation of this policy is not possible.

Our future work will seek the further enhancement of the presented algorithm, through a more profound understanding of the geometry that impacts the performance of the algorithm and the structure of the optimal solution, and the development of heuristics for the employed B&B method that take better advantage of these qualitative results and insights. The development of improved algorithms to address more efficiently the reachability analysis that precedes the classifier synthesis problem is also part of our current investigations. Some results along this line can be found in [44], [45]. Finally, as remarked in Section V-F, a remaining open issue is the formal characterization of the complexity of the classifier design problem that was addressed in this work.

REFERENCES

[1] S. A. Reveliotis, *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. NY, NY: Springer, 2005.

[2] M. Fanti and M. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Trans. Syst. Man Cybern. Part A, Syst. Humans*, vol. 34, no. 1, pp. 5–22, 2004.

[3] M. Zhou and M. P. Fanti (editors), *Deadlock Resolution in Computer-Integrated Systems*. Singapore: Marcel Dekker, Inc., 2004.

[4] S. A. Reveliotis, "Conflict resolution in AGV systems," *IIE Trans.*, vol. 32(7), pp. 647–659, 2000.

[5] N. Wu and M. Zhou, "Resource-oriented Petri nets in deadlock avoidance of AGV systems," in *Proceedings of the ICRA'01*. IEEE, 2001, pp. 64–69.

[6] E. Roszkowska and S. Reveliotis, "On the liveness of guidepath-based, zoned-controlled, dynamically routed, closed traffic systems," *IEEE Trans. on Automatic Control*, vol. 53, pp. 1689–1695, 2008.

[7] Y. Wang, H. Liao, S. Reveliotis, T. Kelly, S. Mahlke, and S. Lafortune, "Gadara nets: Modeling and analyzing lock allocation for deadlock avoidance in multithreaded software," in *Proc.* 48^{th} *IEEE Conference on Decision and Control*, Shanghai, China, 2009, pp. 4971–4976.

[8] T. Kelly, Y. Wang, S. Lafortune, and S. Mahlke, "Eliminating concurrency bugs with control engineering," *IEEE Computer*, vol. 42, no. 12, pp. 52–60, 2009.

[9] A. Nazeem, S. Reveliotis, Y. Wang, and S. Lafortune, "Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: The linear case," *IEEE Trans. Autom. Control*, vol. 56, no. 8, pp. 1818–1833, 2011.

[10] R. Cordone and L. Piroddi, "Monitor optimization in Petri net control," in *Proc.* 7^{th} *IEEE Conf. on Automation Science and Engineering*, Trieste, Italy, 2011, pp. 413–418.

[11] ——, "Parsimonious monitor control of Petri net models of FMS," *IEEE Trans. Syst. Man Cybern: Systems*, vol. 43, no. 1, pp. 215–221, Jan. 2013.

[12] Y. Chen, Z. Li, M. Khalgui, and O. Mosbahi, "Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 2, pp. 374–393, 2011.

[13] Y. Chen and Z. 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.

[14] A. Nazeem and S. Reveliotis, "Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: The nonlinear case," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1670–1684, 2012.

[15] T. Araki, Y. Sugiyama, and T. Kasami, "Complexity of the deadlock avoidance problem," in *2nd IBM Symp. on Mathematical Foundations of Computer Science*. IBM, 1977, pp. 229–257.

[16] 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.

[17] T. Murata, "Petri nets: properties, analysis and application," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[18] 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.

[19] 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.

[20] M. Iordache, J. Moody, and P. Antsaklis, "Synthesis of deadlock prevention supervisors using Petri nets," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 59–68, 2002.

[21] S. Reveliotis, E. Roszkowska, and J. Y. Choi, "Generalized algebraic deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. on Automatic Control*, vol. 52, pp. 2345–2350, 2007.

[22] L. Piroddi, R. Cordone, and I. Fumagalli, "Selective siphon control for deadlock prevention in Petri nets," *IEEE Trans. Syst. Man Cybern. Part A, Syst. Humans*, vol. 38, no. 6, pp. 1337–1348, 2008.

[23] ——, "Combined siphon and marking generation for deadlock prevention in Petri nets," *IEEE Trans. Syst. Man Cybern. Part A, Syst. Humans*, vol. 39, no. 3, pp. 650–661, 2009.

[24] V. Vazirani, *Approximation Algorithms*. NY,NY: Springer, 2003.

[25] 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.

[26] 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.

[27] Y. Huang, M. Jeng, X. Xie, and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Intl. Journal of Production Research*, vol. 39, pp. 283–305, 2001.

[28] 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.

[29] 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.

[30] H. Liao, S. Lafortune, S. Reveliotis, Y. Wang, and S. Mahlke, "Optimal liveness-enforcing control of a class of Petri nets arising in multithreaded software," *IEEE Trans. Autom. Control*, vol. 58, no. 5, pp. 1123–1138, 2013.

[31] H. Liao, Y. Wang, J. Stanley, S. Lafortune, S. Reveliotis, T. Kelly, and S. Mahlke, "Eliminating concurrency bugs in multithreaded software: A new approach based on discrete-event control," *IEEE Trans. Control, Systems Technology* (to appear – DOI 10.1109/TCST.2012.2226034).

[32] E. Badouel, L. Bernardinello, and P. Darondeau, "Polynomial algorithms for the synthesis of bounded nets," in *Proceedings of CAAP'95*. Springer-Verlag LNCS 915, 1995, pp. 364–378.

[33] A. Ghaffari, N. Rezg, and X. Xie, "Design of a live and maximally permissive Petri net controller using the theory of regions," *IEEE Trans. on Robotics & Automation*, vol. 19, pp. 137–141, 2003.

[34] M. Uzam, "An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions," *Int. J. Adv. Manuf. Technol.*, vol. 19, pp. 192–208, 2002.

[35] Z. Li, M. Zhou, and N. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Systems, Man and Cybernetics – Part C: Applications and Reviews*, vol. 38, pp. 173–188, 2008.

[36] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: John Wiley and Sons, 1988.

[37] Y. Chen and Z. Li, "On structural minimality of optimal supervisors for flexible manufacturing systems," *Automatica*, vol. 48, no. 10, pp. 2647–2656, 2012.

[38] A. Nazeem and S. Reveliotis, "Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory," in *Proc. 7^{th} IEEE Conf. on Automation Science and Engineering*, Trieste, Italy, 2011, pp. 405–412.

[39] S. A. 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 J. Control Optim.*, vol. 51, no. 2, pp. 1707–1726, 2013.

[40] S. Reveliotis and A. Nazeem, "Deadlock Avoidance Policies for Automated Manufacturing Systems using Finite State Automata," in *Formal Methods in Manufacturing*, J. Campos, C. Seatzu, and X. Xie, Eds. CRC Press / Taylor & Francis (to appear).

[41] R. Cordone, A. Nazeem, L. Piroddi, and S. Reveliotis, "Maximally permissive deadlock avoidance for sequential resource allocation systems using disjunctions of linear classifiers," in *Proc. 51^{st} IEEE Conf. on Decision and Control*, Maui (HI), USA, 2013, pp. 7244–7251.

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

[43] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. New York (NY), USA: Wiley Interscience, 1998.

[44] A. Nazeem and S. Reveliotis, "An efficient algorithm for the enumeration of the minimal unsafe states in complex resource allocation systems," in *Proc. 48^{th} IEEE Intl. Conference on Automation Science and Engineering*, Seoul, Korea, 2012.

[45] ——, "Efficient enumeration of minimal unsafe states in complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. (to appear).