A branch and bound approach for the design of decentralized supervisors in Petri net models☆

Francesco Basile[a,1], Roberto Cordone[b], Luigi Piroddi[c]

[a] Dipartimento di Ingegneria dell'Informazione, Ingegneria elettrica e Matematica applicata, Università di Salerno, Italy

[b] Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy

[c] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

Received 13 February 2014
Received in revised form 5 August 2014
Accepted 11 November 2014

1. Introduction

Supervisory control (SC) concerns the design of an agent (called the *supervisor*) that enforces forbidden state specifications on a discrete event system (DES). In the Petri net (PN) framework forbidden state specifications are often expressed in terms of linear state inequalities, called Generalized Mutual Exclusion Constraints (GMECs), which are amenable to a straightforward PN implementation, in the form of *monitor* places suitably connected to the transitions of the PN model of the plant and enforcing conservative conditions on the state evolution (through corresponding P-invariants) (Giua, DiCesare, & Silva, 1992; Moody & Antsaklis, 2000).

The supervisor design problem faces various objectives at the same time, namely the enforcement of specific properties (liveness, reversibility, controllability, etc.) in a maximally permissive way (*i.e.*, enabling as many reachable states as possible), and introducing the minimum number of monitors possible. Recent developments have shown that this problem can be optimally and efficiently solved in two steps, *i.e.* by calculating first the maximal subset of reachable states that guarantees the obtainment of the required properties, denoted $\mathcal{L}$ (the *legal* set), and then the monitor-based supervisor that restricts the reachability set of the plant net in closed loop exactly to $\mathcal{L}$.

Regarding the first step, Basile, Cordone, and Piroddi (2013) introduce a technique to calculate the legal set enforcing multiple specifications, both *static* and *behavioral*, the former being associated directly to individual states, while the latter depend on the structure of the reachability graph of the PN. Bounds on job and resource usage fall in the first category, whereas deadlock prevention (DP), liveness enforcement (LE), reversibility, controllability, etc. are behavioral specifications. The approach is particularly useful when multiple behavioral specifications, such as liveness and controllability, are formulated. Indeed, in such cases, it is inconvenient to enforce separately each behavioral property, since enforcing one may jeopardize the other.

As for the second step of the methodology, Nazeem, Reveliotis, Wang, and Lafortune (2010, 2011) provide a complete framework for the characterization of the existence of optimal supervisors and their synthesis, formulating an ILP problem where the decision variables are the GMEC parameters and the constraints are

☆ The research leading to these results has received funding from Regione Campania (Legge n.5/2002) under the preject "Automazione orientata ai servizi per linee di produzione iperflessibile". The material in this paper was not presented at any conference.

E-mail addresses: fbasile@unisa.it (F. Basile), roberto.cordone@unimi.it (R. Cordone), luigi.piroddi@polimi.it (L. Piroddi).

[1] Tel.: +39 089 964400; fax: +39 06 233227957.

expressed in terms of the legal and illegal markings. On similar lines, Chen, Li, Khalgui, and Mosbahi (2011) and Chen and Li (2011) concentrate the attention on the so-called First met Bad Markings (FBMs) and propose an iterative greedy ILP approach to find a GMEC that forbids one FBM at a time. A more efficient solution to the same problem, that systematically addresses the structural optimality of the supervisor, is suggested in Cordone and Piroddi (2013), where a simpler ILP formulation (addressing the prevention of a subset of illegal states with an individual GMEC) is used as the core element of a Branch and Bound (B&B) approach that solves the set covering problem of assigning optimally the illegal states to a minimum number of GMECs. Later developments extend these approaches to problems where a plain GMEC-based supervisor does not exist and more complex (nonlinear) supervisors are required, (Cordone, Nazeem, Piroddi, & Reveliotis, 2013, 2012; Nazeem & Reveliotis, 2012). The monitor redundancy issue has attracted much attention in the recent literature, with specific focus on the reduction of the number of control places as well as the supervisor structure. In Dideban and Alla (2008) the concept of over-state is introduced for safe PNs, and exploited to reduce the constraints for a given set of forbidden states, and this approach has been recently improved introducing the concept of quasi partial invariants and semi quasi partial invariants in Dideban, Zareiee, and Alla (2013). In Zareiee, Dideban, and Orouji (2014) ILP problems are used to obtain a small number of control places with small number of arcs. Another interesting approach for supervisor design enforcing behavioral properties, such as reversibility, is discussed in Reveliotis and Choi (2006). This work can also be extended to accommodate uncontrollable transitions.

The supervisor design problem becomes more involved in a decentralized setting. In that context, it is assumed that several local supervisors operate, each having authority only on a portion of the system (*i.e.*, on a subset of the transitions), in the absence of central coordination and with mutual communication inhibited. Such control architecture becomes of crucial importance for plants having a wide geographic extension or a large number of devices such as in modern communication systems. In these cases, communication with all plant sensors or actuators is infeasible because of economic reasons or bandwidth limitations. Even where centralized control is possible, it is of interest to study decentralized control solutions to address temporary failures that prevent communication with a certain area of the plant, in order to robustify the design.

While there is a large literature on decentralized control with formal languages and automata (Barret & Lafortune, 2000; Lin & Wonham, 1990; Rudie & Wonham, 1992), relatively fewer works address this problem in the PN framework. In Guan and Holloway (1997) global specifications are implemented by local supervisors with communication. In Chen and Hu (1991) a central coordinator is also present but specifications are given from the beginning in a distributed form. The approach of Basile, Giua, and Seatzu (2007, 2008) proposes an algorithm to optimize the permissiveness of the closed loop behavior under decentralized control by selecting with a heuristic rule the decentralized specifications that find a compromise between fairness among variables and the maximal cardinality of the set of legal markings under decentralized control. The controlled system is not guaranteed to be live or to satisfy any particular behavioral property. The mentioned works of Basile et al. (2007, 2008) employ a formalization of the decentralization specifications similar to the one adopted here, but for the fact that the control sites are expressed in terms of subsets of places rather than transitions. This design choice appears to be less intuitive and significant in practice since, while transitions are generally associated to events, places do not always have a clear physical meaning. In Iordache and Antsaklis (2006) global specifications without central coordination are considered and a sufficient condition is given for a set of GMECs to be enforced in a decentralized setting (d-admissibility). In addition, the transformation of inadmissible decentralized constraints into admissible ones is posed either in terms of the minimization of communication costs or in terms of the transformation of the constraints into a set of more restrictive – but d-admissible – ones. D-admissible constraints can be implemented by supervisors that detect and disable transitions of a single site.

The decentralized supervisor design problem is formulated here in the framework of the two-step supervisory control methodology described above. The main idea is to look for legal state sets (*i.e.*, compatible with all the requirements in the centralized setting) that are also exactly enforceable by decentralized supervisors. An optimization method is designed to find the maximal such set. Notice that, differently from Iordache and Antsaklis (2006), this paper focuses on the decentralized implementation of a set of legal markings by means of monitors, rather than the decentralization of a given set of constraints. The main difficulty in extending the two-step approach to the decentralized case lies in the fact that the two steps are interdependent. Indeed, not all sets of legal states that are compatible with a centralized supervisor implementation are also enforceable by a decentralized one. In fact, the decentralization requirement typically results in a reduction of the maximal legal set that can be actually allowed, compared to the centralized control case. Consequently, one cannot completely decouple the determination of the legal set $\mathcal{L}$ from the assessment of the existence of a decentralized supervisor that exactly enforces it.

This difficulty is here overcome by adopting a proposal-acceptance mechanism, where a candidate legal set $\mathcal{L}$ (by construction, included in or equal to the maximal set of legal states that can be allowed by a centralized supervisor), is first selected so as to guarantee the obtainment of all the desired static and behavioral requirements, and then tested for the existence of a decentralized supervisor that can exactly enforce it. In case of failure alternative smaller candidate legal sets are generated by a B&B algorithm by subsequent reductions of the global legal state set, guaranteeing a full exploration of its subsets. The B&B algorithm searches for the maximal such subset that provides all the required properties and is also enforceable in a decentralized way. Notice in passing that any existing decentralized controller can also be implemented in a centralized way, so that the existence of a centralized supervisor is in fact a pre-requisite for the existence of a decentralized one.

Two different procedures are proposed to deal with controllability from a *structural* and *behavioral* point of view, respectively. More in detail, structural controllability can be taken into account in the supervisor design phase alone by simply constraining the monitors introduced by the local supervisors not to have arcs directed towards uncontrollable transitions. On the other hand, behavioral controllability impacts on both the reachability pre-processing phase and the supervisor design. Indeed, behavioral controllability allows the existence of arcs directed from a local controller to an uncontrollable transition, as long as the latter is never disabled by an exclusive action of the former. In other words, whenever the control place of the local supervisor connected with an arc to the uncontrollable transition is insufficiently marked to enable the transition, there must always exist another place (not belonging to the local supervisor) that disables the transition. To enforce this property, a specific condition is added to the supervisor design phase, concerning every reachable marking where a partially controllable transition[2] must be disabled. This additional constraint ensures the presence of arcs disabling such a transition under the above mentioned marking only from local supervisors acting on sites where the transition is controllable. The set of such markings must be determined in the reachability pre-processing phase. Observability is also considered in the design process, but only from a structural point of view, for reasons explained in the paper.

[2] A *partially controllable* transition is a transition that can be used by multiple control sites, but is not controllable by all of them.

Finally, notice that a preliminary version of this work, considering only the simpler – and more conservative – case of structural controllability, was presented in Basile, Cordone, and Piroddi (2013). The present paper unfolds the more complex, but also more permissive, behavioral case and provides a unified framework for dealing with both approaches. It also provides a detailed presentation and analysis of the branch and bound algorithm, including a discussion on its computational complexity. Finally, a more general simulation example with the comparison between the structural and behavioral approaches is presented.

2. Preliminaries

2.1. Petri net basics

A marked PN (Murata, 1989) is a 5-tuple $N = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post}, \boldsymbol{m}_0\rangle$, where P and T are the (finite and nonempty) sets of n_p places and n_t transitions, with $P \cap T = \emptyset$, $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{N}^{n_p \times n_t}$ are the input and output matrices, and $\boldsymbol{m}_0 \in \mathbb{N}^{n_p}$ is the (initial) marking vector, $\mathbb{N}$ being the set of nonnegative integers. Places (graphically represented as circles) are connected to transitions (represented as bars) through directed weighted arcs. More precisely, $Pre(k, j)$ $[Post(k, j)]$ represents the weight of an arc going from p_k $[t_j]$ to t_j $[p_k]$ (0 if there is no such arc). In the absence of self-loops, an equivalent information is given by the incidence matrix $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$. The marking vector $\boldsymbol{m}$ defines the distribution of tokens in places. The pre-set of a set of places $\overline{P} \subseteq P$ is defined as $\bullet\overline{P} = \{t_j \in T \mid \exists p_k \in \overline{P} \text{ s.t. } Post(k, j) > 0\}$, and the post-set as $\overline{P}\bullet = \{t_j \in T \mid \exists p_k \in \overline{P} \text{ s.t. } Pre(k, j) > 0\}$.

A transition $t_j \in T$ is enabled in a marking $\boldsymbol{m}$ (denoted $\boldsymbol{m}[t_j\rangle$) iff $\boldsymbol{m} \geq \boldsymbol{Pre}\,\boldsymbol{e}^j$, where $\boldsymbol{e}^j$ is the jth versor of the $\mathbb{R}^{n_t}$ coordinate space (*i.e.*, $\boldsymbol{e}^j_k = 1$ if $k = j$ and 0 otherwise). A transition t_j such that $\boldsymbol{m}[t_j\rangle$ may fire at marking $\boldsymbol{m}$, yielding the marking $\boldsymbol{m}'$ (denoted $\boldsymbol{m}[t_j\rangle\boldsymbol{m}'$), where $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}\boldsymbol{e}^j$. The reachability set $R(N, \boldsymbol{m}_0)$ collects the markings reachable from $\boldsymbol{m}_0$ by way of enabled transition sequences. The reachability graph is a digraph $RG = (V, A)$, where $V = R(N, \boldsymbol{m}_0)$ is the set of vertices and $A \subseteq (V \times V)$ the set of arcs, associated to the PN transitions through a labeling function $h : A \to T$. The notation $H_{\overline{A}} = \{t \in T \mid \exists a \in \overline{A} \text{ s.t. } h(a) = t\}$ will also be used, where $\overline{A} \subseteq A$.

A strongly connected component (SCC) of a digraph is a maximal subgraph, such that any two of its nodes are connected by a directed path. An SCC may consist of a single vertex, if that vertex does not belong to any directed cycle. Let (V_S, A_S) be an SCC of RG. Then, if $|V_S| \geq 2$ the PN can evolve inside V_S for an arbitrary number of transition firings. (V_S, A_S) is characterized as a *terminal* SCC if there does not exist any $(\boldsymbol{m}_1, \boldsymbol{m}_2) \in A$ with $\boldsymbol{m}_1 \in V_S$ and $\boldsymbol{m}_2 \in V\backslash V_S$.

A place $p_i \in P$ is bounded iff $\exists k > 0$ s.t. $m_i \leq k, \forall \boldsymbol{m} \in R(N, \boldsymbol{m}_0)$. A PN is bounded iff all its places are bounded. A transition $t_j \in T$ is live iff $\forall \boldsymbol{m} \in R(N, \boldsymbol{m}_0)$, $\exists \boldsymbol{m}' \in R(N, \boldsymbol{m})$ s.t. $\boldsymbol{m}'[t_j\rangle$. N is live iff all its transitions are live. N is reversible iff $\forall \boldsymbol{m} \in R(N, \boldsymbol{m}_0)$, $\boldsymbol{m}_0 \in R(N, \boldsymbol{m})$. A marking $\boldsymbol{m} \in R(N, \boldsymbol{m}_0)$, s.t. $\nexists t_j \in T$ enabled in $\boldsymbol{m}$, is called a dead marking and represents a (total) deadlock state. Notice that it also constitutes a terminal SCC with a single vertex. A PN with no reachable dead markings is called deadlock-free.

In terms of the structure of RG, a PN is (Fumagalli, Piroddi, & Cordone, 2010):

(i) deadlock-free if all its terminal SCCs have cardinality strictly greater than 1,
(ii) live if for any terminal SCC (V_S, A_S) it holds that $|V_S| \geq 2$ and $H_{A_S} = T$, and
(iii) reversible if RG contains a single SCC (that coincides with it).

2.2. GMEC enforcement by means of monitors

Given a marked PN N with initial marking $\boldsymbol{m}_0$, a GMEC is a pair $(\boldsymbol{l}, b)$, with $\boldsymbol{l} \in \mathbb{N}^{n_p}$, $b \in \mathbb{N}$, that defines an *admissibility region* $\mathcal{M}(\boldsymbol{l}, b) = \{\boldsymbol{x} \in \mathbb{N}^{n_p} \mid \boldsymbol{l}^T\boldsymbol{x} \leq b\}$. A set of GMECs $(\boldsymbol{L}, \boldsymbol{b})$, with $\boldsymbol{L} = [\boldsymbol{l}_1^T\ \boldsymbol{l}_2^T\ \ldots\ \boldsymbol{l}_{n_c}^T]^T$ and $\boldsymbol{b} = [b_1\ b_2\ \ldots\ b_{n_c}]^T$, defines an admissibility region $\mathcal{M}(\boldsymbol{L}, \boldsymbol{b}) = \cap_{i=1}^{n_c} \mathcal{M}(\boldsymbol{l}_i, b_i)$. Provided $\boldsymbol{m}_0 \in \mathcal{M}(\boldsymbol{L}, \boldsymbol{b})$ holds, a control sub-net consisting of n_c additional places (monitors) connected to the existing PN transitions by way of the incidence matrix $\boldsymbol{C}_C = -\boldsymbol{LC}$ and marked according to $\boldsymbol{m}_{C0} = \boldsymbol{b} - \boldsymbol{Lm}_0$ enforces the said constraints (Giua, Di Cesare, & Silva, 1993; Giua et al., 1992; Yamalidou, Moody, Lemmon, & Antsaklis, 1996). The designed controller is maximally permissive, in that it prevents only transition firings leading to a violation of one of the GMECs.

Given a set $\mathcal{L} \subseteq R(N, \boldsymbol{m}_0)$ such that $\boldsymbol{m}_0 \in \mathcal{L}$, there exists a GMEC-based supervisor that exactly enforces it iff there exists $(\boldsymbol{L}, \boldsymbol{b})$ such that $\mathcal{L} \subseteq \mathcal{M}(\boldsymbol{L}, \boldsymbol{b})$ and $\mathcal{M}(\boldsymbol{L}, \boldsymbol{b})$ does not contain any other reachable marking.

The problem of restricting the reachability set of a PN within a set of legal markings $\mathcal{L}$ becomes somewhat more involved in the presence of uncontrollable transitions. In the following, it is assumed that $T = T_c \cup T_{uc}$ with $T_c \cap T_{uc} = \emptyset$, where T_{uc} is the set of uncontrollable transitions (represented as black bars), and T_c is the set of controllable transitions (represented as white bars), associated to uncontrollable and controllable events, respectively.

Definition 1. Consider a PN N with $T_{uc} \neq \emptyset$. The sub-net N_{uc} obtained from N eliminating every transition in T_c is denoted *uncontrollable sub-net of N*. ■

It is immediate to see that $R(N_{uc}, \boldsymbol{m}) \subseteq R(N, \boldsymbol{m})$.

Definition 2. A legal marking set $\mathcal{L} \subseteq \mathbb{N}^{n_p}$ is *behaviorally controllable* w.r.t. a marked PN $\langle N, \boldsymbol{m}_0\rangle$ if $\bigcup_{\boldsymbol{m}\in\mathcal{L}} R(N_{uc}, \boldsymbol{m}) \subseteq \mathcal{L}$, where N_{uc} is the uncontrollable sub-net of N. ■

In other words, $\mathcal{L}$ is controllable if no forbidden marking is reachable from any marking $\boldsymbol{m} \in \mathcal{L}$ by firing a sequence containing only uncontrollable transitions.

When the controller is modeled by a PN, a transition t enabled under the net marking can only be disabled if there is an arc from a control place to t and the control place is insufficiently marked. Therefore, to enforce a behaviorally controllable legal marking set by means of a PN controller, an arc directed from a control place to an uncontrollable transition must be avoided if there exists a reachable marking where the control place alone disables the transition, which would otherwise be enabled by way of the plant marking. A simple, but possibly restrictive condition that ensures controllability is to avoid altogether arcs directed from monitor places to uncontrollable transitions.

Definition 3 (*Moody & Antsaklis, 2000*). Let N be a PN with transition set $T = T_c \cup T_{uc}$, where $T_c \cap T_{uc} = \emptyset$. A set $\mathcal{L}$ of legal markings is said to be *structurally controllable* iff there exists a set of GMECs that exactly enforces it, such that for each monitor $p_c = 1, \ldots, n_c$ it holds that $p_c \bullet \cap T_{uc} = \emptyset$. ■

A transition is called unobservable if its firings cannot be directly detected or measured. As a consequence, a controller state change cannot be triggered by the firing of an unobservable transition. Since in a PN supervisor both input and output arcs to the plant transitions can trigger its state changes, no arcs directed towards or coming from an unobservable transition are allowed. In case of unobservable transitions, two strings of transitions (events) cannot be distinguished if they become equal deleting unobservable transitions. *Behavioral* observability requires that if it is not possible to distinguish two different strings of events in a DES, then the supervisor must produce the same action on the plant in response to their firing (Cassandras & Lafortune, 2008). Behavioral observability is necessarily tested on the reachability graph but it does not influence the supervisor implementation, that is still constrained not to have any arc connections with unobservable transitions independently from the approach adopted (either structural or behavioral). Therefore, since in this paper the focus is on the decentralized implementation of a legal marking set and not on the behavioral observability test, behavioral observability will be not considered. The reader can refer to the literature (Cassandras & Lafortune, 2008) for further details.

Denote with T_o and T_{uo} the sets of observable and unobservable transitions, respectively (correspondingly associated to observable and unobservable events), where $T = T_o \cup T_{uo}$ and $T_o \cap T_{uo} = \emptyset$.

Definition 4. Consider a PN N with $T_{uo} \neq \emptyset$. The sub-net N_{uo} obtained from N eliminating every transition in T_o is denoted as *unobservable sub-net of N*. ■

Definition 5 (*Moody & Antsaklis, 2000*). Let N be a PN with transition set $T = T_o \cup T_{uo}$, where $T_o \cap T_{uo} = \emptyset$. A set $\mathcal{L}$ of legal markings is said to be *structurally observable* iff there exists a set of GMECs that exactly enforces it, such that for each monitor $p_c = 1, \ldots, n_c$ it holds that $(p_c \bullet \cup \bullet p_c) \cap T_{uo} = \emptyset$. ■

2.3. GMEC optimization as a classification problem

A set of GMECs $(\boldsymbol{L}, \boldsymbol{k})$ can be envisaged as a *linear classifier*, separating the markings in $\mathcal{M}(\boldsymbol{L}, \boldsymbol{k})$ from those outside. Conditions for the existence of a linear classifier $(\boldsymbol{L}, \boldsymbol{k})$ that separates any two given (disjoint) marking sets, $\mathcal{L}$ (the *legal* set) and $\mathcal{U}$ (the *illegal* set), are discussed in Cordone et al. (2013, 2012).

Theorem 1 (*Cordone et al., 2012*). *There exists a linear classifier for marking sets $\mathcal{L}$ and $\mathcal{U}$ iff there does not exist a marking $\boldsymbol{m} \in \mathcal{U}$ such that $\boldsymbol{m} \in P_{\mathcal{L}}$, where $P_{\mathcal{L}}$ is the convex hull*[3] *of $\mathcal{L}$.*

Cordone and Piroddi (2011, 2013) provide an efficient method for the design of a maximally permissive GMEC-based supervisor guaranteeing the correct state classification. The design of the supervisor is reformulated as the search for an optimal covering of the illegal set $\mathcal{U}$ with suitable subsets $\mathcal{U}_i$, $i = 1, \ldots, n_c$, such that for each subset there exists a GMEC that separates it from $\mathcal{L}$. Indeed, the resulting set of GMECs provides a linear classifier that separates $\mathcal{U}$ from $\mathcal{L}$. All feasible coverings of the illegal set $\mathcal{U}$ can be systematically explored with the B&B method explained in Cordone and Piroddi (2011, 2013).

The method can be extended to nonlinear classifiers, formulated as disjunctions of linear classifiers, to deal with all cases that do not fall into Theorem 1, albeit at an increased computational cost, as discussed in Cordone et al. (2013, 2012).

3. Characterization of the legal set in a decentralized framework

3.1. The decentralized setting

Let N be a (possibly unbounded) PN with initial marking $\boldsymbol{m}_0$, and with transition set $T = T_c \cup T_{uc}$ and $T = T_o \cup T_{uo}$, where $T_c \cap T_{uc} = \emptyset$, $T_o \cap T_{uo} = \emptyset$, and $T_c \subseteq T_o$ (all controllable transitions are also assumed observable). Notice that any supervisor is constrained to operate only on observable transitions, and can disable only controllable transitions. Let $\mathcal{L}$ be the maximal set of markings (denoted *legal set* in the sequel) compatible with all the static and behavioral requirements of interest and realizable with a centralized supervisor. It is here assumed that the static requirements include boundedness, so that $\mathcal{L}$ is a bounded set, and that the behavioral requirements include liveness, reversibility, and controllability. The mentioned set $\mathcal{L}$ can be determined exactly following the approach described in Basile et al. (2013) or with other supervisor design techniques that can guarantee the same properties. Let also $\mathcal{U}$ be the corresponding set of boundary illegal states (states outside $\mathcal{L}$ that can be reached from legal states with a single transition firing), briefly referred to as *illegal set*. The boundedness of $\mathcal{L}$ automatically implies that of $\mathcal{U}$ (Basile et al., 2013). Finally, the B&B methods of Cordone and Piroddi (2013) and Cordone et al. (2013) can be invoked to compute the optimal supervisor enforcing $\mathcal{L}$. In the following it is assumed that such a centralized supervisor exists, which is a pre-requisite for the existence of a decentralized supervisor.

Now, consider a decentralized setting where the sets of transitions $T_1, \ldots, T_\nu$ identify ν control sites such that any local supervisor is allowed to act only on the transitions of one site. Subsets T_i, $i = 1, \ldots, \nu$, do not necessarily form a partition nor a covering of T, so that the same transition might belong to different subsets T_i, $i = 1, \ldots, \nu$. Let $T_i = T_{c_i} \cup T_{uc_i}$, $i = 1, \ldots, \nu$, with $T_{c_i} \cap T_{uc_i} = \emptyset$, T_{c_i} collecting all (locally controllable) transitions associated to events whose firing can be disabled by the ith control site S_i. Similarly, let $T_i = T_{o_i} \cup T_{uo_i}$, $i = 1, \ldots, \nu$, with $T_{o_i} \cap T_{uo_i} = \emptyset$, where T_{o_i} represents a set of (locally observable) transitions associated to events whose firing can be detected from S_i. Further on, all transitions in $T \backslash T_i$ are assumed uncontrollable and unobservable by the ith control site. Conventionally, controllable transitions are also expected to be observable, so that $T_{c_i} \subseteq T_{o_i}$. Note that a globally controllable transition may become *partially* controllable in a decentralized setting, *i.e.* controllable only by a fraction of the sites which have authority over it.

In the decentralized setting, only a subset $\overline{\mathcal{L}} \subseteq \mathcal{L}$ of the legal markings will generally be allowed (due to the decentralization constraints). While this would still guarantee the obtainment of all the static specifications, that depend only on the individual states, the desired behavioral properties could be lost due to the contraction of the reachable space (which implies a modification of the behavioral characteristics of the system, as described by the reachability graph). For this reason, two separate notions of feasibility will be introduced, accounting for the achievement of the required behavioral properties and the realizability of a decentralized supervisor, respectively. Accordingly, an algorithm will be illustrated that finds the largest subset of $\mathcal{L}$ that achieves both properties at the same time.

3.2. B-feasibility

In the following, a set of markings $\overline{\mathcal{L}} \subseteq \mathcal{L}$ will be denoted *B-feasible* (behaviorally feasible) iff the reachability subgraph induced by $\overline{\mathcal{L}}$ on the PN possesses all the required behavioral properties. Without loss of generality, the required behavioral properties are here restricted to liveness, reversibility, and controllability. The definition of B-feasibility (and the overall approach) can be extended also to other properties (*e.g.*, DP).

Definition 6. A set $\overline{\mathcal{L}}$ such that $\{\boldsymbol{m}_0\} \subset \overline{\mathcal{L}} \subseteq \mathcal{L}$ is denoted *B-feasible* if a supervisor enforcing exactly $\overline{\mathcal{L}}$ results in a live, reversible and behaviorally controllable PN system. ■

The following result provides a necessary and sufficient condition for B-feasibility.

Lemma 1. *Let $\{\boldsymbol{m}_0\} \subset \overline{\mathcal{L}} \subseteq \mathcal{L}$ and $(\overline{\mathcal{L}}, \overline{A})$ be the subgraph induced by $\overline{\mathcal{L}}$ on the reachability graph $RG = (V, A)$, i.e. $\overline{A} = \{(\boldsymbol{m}, \boldsymbol{m}') \in A : \boldsymbol{m}, \boldsymbol{m}' \in \overline{\mathcal{L}}\}$. Then, the set $\overline{\mathcal{L}}$ is B-feasible iff*

(i) *the graph $(\overline{\mathcal{L}}, \overline{A})$ has only one SCC (coinciding with the entire graph);*
(ii) *$H_{\overline{A}} = T$;*
(iii) *$\forall a = (\boldsymbol{m}, \boldsymbol{m}') \in A$ s.t. $\boldsymbol{m} \in \overline{\mathcal{L}}$ and $h(a) \notin T_c$, where $T_c = \cup_{i=1}^{\nu} T_{c_i}$, it holds that $\boldsymbol{m}' \in \overline{\mathcal{L}}$ as well.* ■

[3] The convex hull of a set of points X in a vector space V is the minimal convex set containing X.

Proof. Assume that there exists a GMEC-based supervisor exactly enforcing $\overline{\mathcal{L}}$.[4] Such a supervisor will achieve reversibility, liveness, and behavioral controllability. Reversibility follows immediately upon observing that the reachability graph of a (bounded) reversible PN has a unique SCC (coinciding with the entire graph itself) (Fumagalli et al., 2010), so that any state is reachable from any other. Deadlock-freeness is also automatically obtained since the PN can evolve indefinitely in a SCC with cardinality greater than 1 (as implied by assumption $\overline{\mathcal{L}} \supset \{\boldsymbol{m}_0\}$). The only additional requirement for liveness is that $\forall t_j \in T$ there exists at least an arc in the reachability graph associated to the firing of t_j (Fumagalli et al., 2010). This is ensured by condition (ii). Finally, condition (iii) implies that there cannot be an arc $a = (\boldsymbol{m}, \boldsymbol{m}') \in A$ s.t. $\boldsymbol{m} \in \overline{\mathcal{L}}$ and $\boldsymbol{m}' \in \mathcal{L}\backslash\overline{\mathcal{L}}$, with $h(a) \notin T_c$. In other words, no illegal marking is reachable from within $\overline{\mathcal{L}}$ by firing only uncontrollable transitions, as required by Definition 2. ■

B-feasibility can be tested based on the reachability graph alone, and does not require the explicit calculation of the set of GMECs enforcing the given set of states.

The following Lemma provides a useful necessary condition for B-feasibility that can be employed to narrow down the search in the state space.

Lemma 2. *Let $\{\boldsymbol{m}_0\} \subset \overline{\mathcal{L}} \subseteq \mathcal{L}$ and $(\overline{\mathcal{L}}, \overline{A})$ be the subgraph induced by $\overline{\mathcal{L}}$ on the reachability graph $RG = (V, A)$. Let also $(\overline{\mathcal{L}}_S, \overline{A}_S)$ be the (unique) SCC of $(\overline{\mathcal{L}}, \overline{A})$ such that $\boldsymbol{m}_0 \in \overline{\mathcal{L}}_S$. Then, any B-feasible subset of $\overline{\mathcal{L}}$ is contained in $\overline{\mathcal{L}}_S$.* ■

Proof. By definition, $(\overline{\mathcal{L}}_S, \overline{A}_S)$ is the maximal strongly connected subgraph of $(\overline{\mathcal{L}}, \overline{A})$ containing $\boldsymbol{m}_0$. Any B-feasible set satisfying the assumption induces a strongly connected subgraph of $(\overline{\mathcal{L}}, \overline{A})$, and contains $\boldsymbol{m}_0$. Therefore, it is necessarily contained in $\overline{\mathcal{L}}_S$. ■

3.3. D-feasibility

In the following, a subset of legal states will be denoted as *D-feasible* (decentralization feasible) if it can be exactly enforced by a decentralized supervisor, *i.e.* such that each local supervisor is not connected with transitions of other control sites, and ensuring local controllability and observability. Notice that the previously recalled definition of d-admissibility given in Iordache and Antsaklis (2006) refers to a set of GMECs, while D-feasibility refers to a set of (globally) legal markings. In words, it is possible to claim that a set of markings is D-feasible if there exists a set of d-admissible GMECs that exactly enforce it. The concept of D-feasibility is considered more appropriate here, in that the developed decentralized control design method described in the sequel operates on state sets. The two local properties can be enforced in a *structural* way or *behavioral* way, the latter being more complex but allowing greater permissivity. For ease of reference, the notions of D_S- and D_B-*feasibility* are introduced to identify the specific (structural or behavioral) controllability condition considered, respectively.

The structural implementation allows outgoing arcs from monitor places only towards locally controllable transitions, and incoming arcs to monitor places only from locally observable transitions.

Definition 7. A set $\overline{\mathcal{L}} \subseteq \mathcal{L}$ is denoted D_S-*feasible* if there exists a set of GMECs that exactly enforces it, such that for each $p_c = 1, \ldots, n_{dc}$ there exists $i \in \{1, \ldots, \nu\}$ such that $p_c\bullet \subseteq T_{c_i}$ (local structural controllability) and $\bullet p_c \subseteq T_{o_i}$ (local structural observability). ■

[4] Such a supervisor always exists, though not necessarily in the form of a plain set of GMECs. However, this is irrelevant to the current lemma.

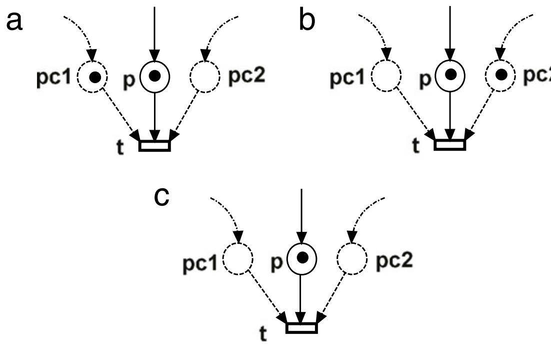

Fig. 1. Monitor places p_{c1} and p_{c2} acting on the partially controllable transition t. Monitor places and arcs are dashed.

Local behavioral controllability is enforced by ensuring that a transition t is never disabled exclusively by monitor places belonging to control sites from which t is uncontrollable. A transition disabling by a monitor place occurs if its firing takes the system from a boundary legal marking to an illegal one. In that case, the disabling monitor must belong to a control site S_i for which the associated transition is accessible and controllable, *i.e.* $t \in T_{c_i}$ (t cannot be uncontrollable from all sites, otherwise $\boldsymbol{m}$ could not have been assumed legal). Now, if such a monitor is present, there is no need to prevent arcs from *other* control places to t, even if t is locally uncontrollable from their control sites, since t is not disabled *exclusively* by those control places. This additional degree of freedom in the supervisor structure may potentially increase its permissiveness, as opposed to supervisors enforcing structural controllability conditions.

Consider the partially controllable transition t in Fig. 1, and two monitor places p_{c1} and p_{c2} associated respectively to site S_1 and S_2. Precisely, assume that t is controllable from site S_1 and uncontrollable from site S_2. From a structural point of view, the arc going from p_{c2} to t is not admissible. On the other hand, if a behavioral approach is adopted, the arc is admissible provided that any marking such as the one in Fig. 1(a) is forbidden. Indeed, in that case t turns out to be disabled precisely by p_{c2}. On the other hand, the markings in Fig. 1(b–c) are legal since t is disabled by p_{c1}. Notice that once $m(p_{c1}) = 0$, the marking of p_{c2} is indifferent: even if $m(p_{c2}) = 0$, as in Fig. 1(c), the disabling action can be attributed to p_{c1}.

Let $\overline{\mathcal{L}}_b = \{\boldsymbol{m} \in \overline{\mathcal{L}} \mid \boldsymbol{m}[t > \boldsymbol{m}', \boldsymbol{m}' \notin \overline{\mathcal{L}}, t \in T_c\}$ denote the set of *boundary* legal markings, *i.e.* the legal markings from which the illegal set can be reached in a single transition step. Notice that the described marking evolution can only occur through the firing of a controllable transition, otherwise $\boldsymbol{m}$ would not be legal. By selectively forbidding the mentioned controllable transitions enabled in markings belonging to $\overline{\mathcal{L}}_b$, no illegal marking will ever be reached. For each boundary legal marking $\boldsymbol{m} \in \overline{\mathcal{L}}_b$, let $\mathcal{D}(\boldsymbol{m}) = \{t \in T_c \mid \boldsymbol{m}[t > \boldsymbol{m}', \boldsymbol{m}' \notin \overline{\mathcal{L}}\}$ be the set of controllable transitions that must be disabled in $\boldsymbol{m}$. Finally, let $\Phi \in \{0, 1\}^{\nu \times n_t}$ be a binary function defining the controllability of transitions from a certain site, *i.e.* such that $\Phi(i, t) = 1$ if $t \in T_{c_i}$, and 0 otherwise. Let also $k : \{1, \ldots, n_{dc}\} \to \{1, \ldots, \nu\}$ map the individual GMECs to the control sites ($k(p_c) = i$ indicates that the p_cth GMEC operates on the ith site, *i.e.* the corresponding monitor is connected only to transitions in T_i).

Definition 8. Let $\{\boldsymbol{m}_0\} \subseteq \overline{\mathcal{L}} \subseteq \mathcal{L}$ and assume that there exists a set of GMECs that exactly enforces $\overline{\mathcal{L}}$. Set $\overline{\mathcal{L}}$ is denoted D_B-*feasible* iff

(i) For each $p_c = 1, \ldots, n_{dc}$, there exists a control site S_i such that $p_c \bullet \cup \bullet p_c \subseteq T_i$;
(ii) For each pair $(\boldsymbol{m}, t)$ with $\boldsymbol{m} \in \overline{\mathcal{L}}_b$ and $t \in \mathcal{D}(\boldsymbol{m})$ s.t. $\sum_{i=1}^{\nu} \Phi(i, t) \geq 1$, it holds that $m(p_c) \leq Pre_c(p_c, t) -$

1 (p_c disables t), at least for one $p_c \in \{1, \dots, n_{dc}\}$ with $k(p_c) = i$, where $\boldsymbol{Pre}_c$ is the input matrix associated to the supervisor. ■

Notice that D_S-feasibility implies D_B-feasibility.

4. A B&B approach for the design of an optimal decentralized supervisor

Definition 9. A set of GMECs $(\boldsymbol{L}, \boldsymbol{b})$ results in an optimal decentralized supervisor if it enforces a maximum cardinality B- and D-feasible subset of the set $\mathcal{L}$ of (globally) legal states. ■

In the preceding definition, the D-feasibility property can be declined either in the structural or behavioral version, as preferred.

Notice that, trivially, if the set $\mathcal{L}$ of legal states for the centralized supervisor design setting is D-feasible, the decentralized supervisor that enforces it is optimal, since $\mathcal{L}$ is B-feasible by definition, and there does not exist a solution (decentralized or not) that allows a larger number of states.

In the general case, in order to find the optimal decentralized supervisor (if one exists), an implicit enumeration technique over the subsets of $\mathcal{L}$ is proposed. Given a certain subset $\overline{\mathcal{L}} \subseteq \mathcal{L}$, its B-feasibility can be ascertained by analyzing the reachability subgraph induced by $\overline{\mathcal{L}}$ alongside Lemma 1. Verifying the D-feasibility of $\overline{\mathcal{L}}$ is a much more complex issue, that involves solving an ILP problem.

The systematic exploration of all possible subsets of $\mathcal{L}$ is based on a branch-and-bound approach. For this purpose, a generic node of the branching tree is associated to a specific partition of the set $\mathcal{L}$ of (globally) legal states into three pairwise disjoint subsets:

$$\mathcal{L} = \mathcal{L}_+ \cup \mathcal{L}_- \cup \mathcal{L}_\times, \tag{1}$$

where $\mathcal{L}_+$ identifies the legal states that must be included and $\mathcal{L}_-$ groups the legal markings that must not be included into any feasible solution of the current node (and all its descendants). The remaining legal states ($\mathcal{L}_\times$) are still unassigned, and essentially constitute the remaining decision variables of the current subproblem. Briefly, since Eq. (1) implies that $\mathcal{L}_\times = \mathcal{L} \setminus (\mathcal{L}_+ \cup \mathcal{L}_-)$, a node is fully characterized by the first two sets, namely:

$$\Pi = \{\mathcal{L}_+, \mathcal{L}_-\}.$$

The general outline of the B&B approach is provided below, as Algorithm 1.

The branching process is initialized with a root node defined as $\Pi_0 = \{\{\boldsymbol{m}_0\}, \emptyset\}$, implying that the initial marking must be allowed by any feasible solution (which can otherwise contain any other legal state). Then, as long as there are open nodes, one of them is extracted and processed as follows. A pre-processing phase (see Section 4.1) extends $\mathcal{L}_+$ and $\mathcal{L}_-$ based on simple logical conditions. Then, the largest B-feasible subset $\mathcal{L}_{BF}$ of $\mathcal{L}$ compatible with the node assignments ($\mathcal{L}_+ \subseteq \mathcal{L}_{BF} \subseteq \mathcal{L} \setminus \mathcal{L}_-$) is computed (see Section 4.2). Any state outside of $\mathcal{L}_{BF}$ cannot belong to a B-feasible subset, and is therefore added to $\mathcal{L}_-$. Afterwards (see Section 4.3), the largest D-feasible subset $\mathcal{L}_{DF}$ included in $\mathcal{L}_{BF}$ is found, that also abides by the node assignments ($\mathcal{L}_+ \subseteq \mathcal{L}_{DF} \subseteq \mathcal{L}_{BF}$). If $\mathcal{L}_{DF}$ is B-feasible, the decentralized supervisor that enforces it is optimal for the current node, because no larger subset can enjoy both properties. The algorithm stores in $\overline{\mathcal{L}}^*$ the best B- and D-feasible set found during the process, that provides a lower bound on the optimum of the problem. In each phase, the current node is immediately discarded if $\overline{\mathcal{L}}^*$ exceeds the size of a suitable upper bound on the size of the optimum. In that case, in fact, no solution of the node (and of its descendants) can improve the best one found so far. If $\mathcal{L}_{DF}$ is not B-feasible, $\mathcal{L}_{BF}$ could still contain B- and D-feasible subsets. Therefore, the current node is branched producing two sub-nodes that inherit the state assignments from the father node, plus additional ones that generate a partition of the solution set. The sub-nodes are finally inserted in the list of open sub-problems.

Algorithm 1 DecentralizedSupervisor

```
Require: $N, RG, \mathcal{L}, \mathcal{U}$.
Ensure: $\overline{\mathcal{L}}^*$.
  $\Pi_0 \leftarrow \{\{\boldsymbol{m}_0\}, \emptyset\}$;                          ▷ Initial problem
  $\Lambda \leftarrow \{\Pi_0\}$;                                      ▷ List of open problems
  $\overline{\mathcal{L}}^* \leftarrow \emptyset$;                                         ▷ Current best solution
  while $\Lambda \neq \emptyset$ do
    $\{\mathcal{L}_+, \mathcal{L}_-\} \leftarrow$ Get($\Lambda$);                      ▷ Pick current problem
    if $|\mathcal{L} \setminus \mathcal{L}_-| > |\overline{\mathcal{L}}^*|$ then                            ▷ Pre-processing
      $(\mathcal{L}_+, \mathcal{L}_-) \leftarrow$ Pre-processing($N, RG, \mathcal{L}, \mathcal{L}_+, \mathcal{L}_-$);
      if $\mathcal{L}_+ \cap \mathcal{L}_- \neq \emptyset$ then $\mathcal{L}_- \leftarrow \mathcal{L}$; end if
    end if
    if $|\mathcal{L} \setminus \mathcal{L}_-| > |\overline{\mathcal{L}}^*|$ then                              ▷ B-feasibility
      $\mathcal{L}_{BF} \leftarrow$ B-feasibleSubset($N, RG, \mathcal{L}, \mathcal{L}_+, \mathcal{L}_-$);
      $\mathcal{L}_- \leftarrow \mathcal{L} \setminus \mathcal{L}_{BF}$;
    end if
    if $|\mathcal{L}_{BF}| > |\overline{\mathcal{L}}^*|$ then                                   ▷ D-feasibility
      $\mathcal{L}_{DF} \leftarrow$ D-feasibleSubset($N, \mathcal{L}, \mathcal{U}, \mathcal{L}_+, \mathcal{L}_-$);
      if $\nexists \mathcal{L}_{DF}$ then $\mathcal{L}_{DF} \leftarrow \emptyset$; end if
    end if
    if $|\mathcal{L}_{DF}| > |\overline{\mathcal{L}}^*|$ then
      if $\mathcal{L}_{DF} \equiv \mathcal{L}_{BF}$ or $\mathcal{L}_{DF}$ is B-feasible then
        $\overline{\mathcal{L}}^* \leftarrow \mathcal{L}_{DF}$;                                   ▷ Best solution update
      else                                                          ▷ Branching
        Pick $\boldsymbol{m} \in \mathcal{L}_{BF} \setminus \mathcal{L}_{DF}$;
        $\Pi_- \leftarrow \{\mathcal{L}_+, \mathcal{L}_- \cup \{\boldsymbol{m}\}\}$;
        $\Pi_+ \leftarrow \{\mathcal{L}_+ \cup \{\boldsymbol{m}\}, \mathcal{L}_-\}$;
        $\Lambda \leftarrow \Lambda \cup \{\Pi_-, \Pi_+\}$;
      end if
    end if
  end while
  Return $\overline{\mathcal{L}}^*$;
```

The various phases of the algorithm are explained in detail in the next subsections.

4.1. Node pre-processing

To reduce the tree expansion and simplify the overall computation, it is convenient to exploit all available information to assign further states to either $\mathcal{L}_+$ or $\mathcal{L}_-$. This pre-processing greatly accelerates the branching process by reducing the free states of the node. The pseudo-code of the pre-processing procedure is given in Algorithm 2.

First of all, let $RG' = (V', A')$ be the subgraph induced by $V' = \mathcal{L} \setminus \mathcal{L}_-$ on RG. By Lemma 2, any B-feasible subset of V' is certainly included in the SCC of RG' that contains $\boldsymbol{m}_0$, denoted as (V_S, A_S). So, this SCC is identified and all the states not belonging to it are included into $\mathcal{L}_-$. If after this extension any state of $\mathcal{L}_+$ also belongs to $\mathcal{L}_-$, no B-feasible subset can fully include $\mathcal{L}_+$, and the branching node can be discarded. The subproblem is unfeasible also if V_S includes only $\boldsymbol{m}_0$ or if $H_{A_S} \subset T$, because under these circumstances liveness cannot be guaranteed.

The following property allows to extend subset $\mathcal{L}_+$: if a state in $\mathcal{L}_+$ has only one successor [predecessor] state, the latter must also belong to the solution. Therefore, the unique successor [predecessor] state must be included into $\mathcal{L}_+$.

Lemma 3. *Let $G = (V, A)$ be a digraph and $S = (V_S, A_S)$ an SCC of G such that $|V_S| \geq 2$. Let also $\boldsymbol{w} \in V_S$ and denote $\boldsymbol{w}\bullet = \{\boldsymbol{v} \in V \mid (\boldsymbol{w}, \boldsymbol{v}) \in A\}$ [$\bullet\boldsymbol{w} = \{\boldsymbol{v} \in V \mid (\boldsymbol{v}, \boldsymbol{w}) \in A\}$] be the set of successor [predecessor] nodes. Then, if $|\boldsymbol{w}\bullet| = 1$ [$|\bullet\boldsymbol{w}| = 1$], it holds that $\boldsymbol{w}\bullet \subseteq V_S$ [$\bullet\boldsymbol{w} \subseteq V_S$].* ■

Proof. A node belonging to an SCC of cardinality greater than 1 has at least one predecessor node and a successor node that also belong

Algorithm 2 Pre-processing

```
Require: $N, RG = (V, A), \mathcal{L}, \mathcal{L}_+, \mathcal{L}_-$.
Ensure: $\mathcal{L}_+, \mathcal{L}_-$.
  $V' \leftarrow \mathcal{L} \backslash \mathcal{L}_-$;
  $A' \leftarrow \{(\boldsymbol{m}, \boldsymbol{m}') \in A \mid \boldsymbol{m}, \boldsymbol{m}' \in V'\}$;
  Let $(V_S, A_S)$ be the SCC of $(V', A')$ s.t. $\boldsymbol{m}_0 \in V_S$;
  $\mathcal{L}_- \leftarrow \mathcal{L} \backslash V_S$;                                  ▷ Extend $\mathcal{L}_-$
  if $\mathcal{L}_+ \cap \mathcal{L}_- \neq \emptyset$ or $|\mathcal{L} \backslash \mathcal{L}_-| = 1$ or $H_{A_S} \subset T$ then
    $\mathcal{L}_+ \leftarrow \emptyset$; $\mathcal{L}_- \leftarrow \mathcal{L}$;                    ▷ Discard the node
  end if
  for all $\boldsymbol{m} \in \mathcal{L}_+$ s.t. $\bullet\boldsymbol{m} = \{\boldsymbol{m}'\}$, with $\boldsymbol{m}' \notin \mathcal{L}_+$ do
    $\mathcal{L}_+ \leftarrow \mathcal{L}_+ \cup \{\boldsymbol{m}'\}$;                           ▷ Extend $\mathcal{L}_+$
  end for
  for all $\boldsymbol{m} \in \mathcal{L}_+$ s.t. $\boldsymbol{m}\bullet = \{\boldsymbol{m}'\}$, with $\boldsymbol{m}' \notin \mathcal{L}_+$ do
    $\mathcal{L}_+ \leftarrow \mathcal{L}_+ \cup \{\boldsymbol{m}'\}$;                           ▷ Extend $\mathcal{L}_+$
  end for
  if $\mathcal{L}_+ \cap \mathcal{L}_- \neq \emptyset$ then
    $\mathcal{L}_+ \leftarrow \emptyset$; $\mathcal{L}_- \leftarrow \mathcal{L}$;                    ▷ Discard the node
  end if
  Return $(\mathcal{L}_+, \mathcal{L}_-)$;
```

to the same SCC. Therefore, if it has only one predecessor/successor node, then that node is necessarily included in the SCC. ■

Once again, if after the extension $\mathcal{L}_+$ and $\mathcal{L}_-$ intersect, the node is unfeasible and can be discarded.

4.2. *Finding the largest B-feasible subset*

After the pre-processing, the reachability subgraph has been reduced to a single SCC, (V_S, A_S), but it is not necessarily B-feasible, due to the uncontrollable transitions. The largest B-feasible subset of V_S is then determined using Algorithm 3, which is readapted from Basile et al. (2013).

Algorithm 3 B-feasibleSubset

```
Require: $N, RG = (V, A), \mathcal{L}, \mathcal{L}_+, \mathcal{L}_-$.
Ensure: $\mathcal{L}_{BF}$.
  $V' \leftarrow \mathcal{L} \backslash \mathcal{L}_-$;
  repeat
    $V'_{old} \leftarrow V'$;
    $A' \leftarrow \{(\boldsymbol{m}, \boldsymbol{m}') \in A \mid \boldsymbol{m}, \boldsymbol{m}' \in V'\}$;
    Let $(V_S, A_S)$ be the SCC of $(V', A')$ s.t. $\boldsymbol{m}_0 \in V_S$;
    if $\mathcal{L}_+ \not\subseteq V_S$ or $|V_S| = 1$, or $H_{A_S} \subset T$ then
                    ▷ Incompatibility with $\mathcal{L}_+$ or B-feasibility violation
      Return $\mathcal{L}_{BF} \leftarrow \emptyset$;
    else
      $V' \leftarrow V_S$;
    end if
    if $\exists \boldsymbol{m} \in V', \exists t \in T_{uc}, \exists \boldsymbol{m}' \notin V'$ s.t. $\boldsymbol{m}[t\rangle\boldsymbol{m}'$ then
      if $\boldsymbol{m} \in \mathcal{L}_+$ then
        Return $\mathcal{L}_{BF} \leftarrow \emptyset$;                   ▷ Incompatibility with $\mathcal{L}_+$
      else
        $V' \leftarrow V' \backslash \{\boldsymbol{m}\}$;
      end if
    end if
  until $V' = V'_{old}$
  Return $V'$;
```

Briefly, Algorithm 3 recursively prunes the SCC of any states violating one of the required behavioral properties, *i.e.* states from which an illegal state can be reached by firing only uncontrollable transitions, terminal cyclic SCCs where not all transitions can be fired, deadlock states, etc. This task is necessarily iterative since any graph reduction may jeopardize one or more of such properties. If during the process a state belonging to $\mathcal{L}_+$ is pruned, Algorithm 3 is stopped, and the node is eliminated, since no feasible solution exists. Otherwise, let $\mathcal{L}_{BF}$ be the B-feasible set returned by the algorithm. If $|\mathcal{L}_{BF}| \leq |\overline{\mathcal{L}}^*|$, the node is discarded, because none of its solutions can improve the best one found so far.

4.3. *Finding the largest D-feasible subset*

At this point one has obtained a B-feasible subset $\mathcal{L}_{BF} \supseteq \mathcal{L}_+$ and such that $|\mathcal{L}_{BF}| > |\overline{\mathcal{L}}^*|$. Therefore such set could provide an improving solution, should it be proved D-feasible as well.

The procedure *D-feasibleSubset* consists in solving an ILP problem, described in detail in the next section, that is designed to find the largest D-feasible subset $\mathcal{L}_+ \subseteq \mathcal{L}_{DF} \subseteq \mathcal{L}_{BF}$. Now, if the ILP problem is unfeasible, the node can be discarded. Otherwise, $\mathcal{L}_{DF}$ is a D-feasible subset and, being the largest one compatible with the node assignments, its value $|\mathcal{L}_{DF}|$ provides an upper bound on the optimum of the current sub-problem. If $|\mathcal{L}_{DF}| \leq |\overline{\mathcal{L}}^*|$, the node can be discarded because no (B- and D-feasible) solution of the current problem can improve over $\overline{\mathcal{L}}^*$. Otherwise, better solutions could exist, provided that they are also B-feasible. Therefore, $\mathcal{L}_{DF}$ is analyzed to verify *a posteriori* whether it is B-feasible (note that B-feasibility automatically holds if $\mathcal{L}_{DF} \equiv \mathcal{L}_{BF}$). In the affirmative case $\mathcal{L}_{DF}$ provides a feasible solution for the overall problem, which is more permissive than the current best. Therefore, it is stored in its stead. Finally, if $\mathcal{L}_{DF}$ is not B-feasible, the node is branched, because it could still contain a feasible solution with fewer states.

4.4. *Branching*

A binary branching policy is adopted. More precisely, two children nodes are generated from Π_i where a free marking $\boldsymbol{m}$ is extracted from $\mathcal{L}_\times$ and added to $\mathcal{L}_-$ for the first child node and to $\mathcal{L}_+$ for the second one, respectively:

$$\Pi_- = \{\mathcal{L}_+, \mathcal{L}_- \cup \{\boldsymbol{m}\}\},$$

$$\Pi_+ = \{\mathcal{L}_+ \cup \{\boldsymbol{m}\}, \mathcal{L}_-\}.$$

The free marking $\boldsymbol{m}$ is chosen among those included in $\mathcal{L}_{BF}$ and not belonging to $\mathcal{L}_{DF}$. In this way, the first child node will necessarily yield a B-feasible subset different from $\mathcal{L}_B$, or none. Conversely, the second child node will necessarily yield a D-feasible subset different from $\mathcal{L}_D$, or none.

5. An ILP approach to find the decentralized supervisor

As discussed in the previous section, the proposed approach operates by means of a proposal-acceptance mechanism, where a B-feasible candidate legal set $\mathcal{L}_{BF}$ is first computed, and then tested for the existence of a decentralized supervisor that can exactly (or partially) enforce it (D-feasibility). Such a test requires solving an optimization problem formulated as an ILP problem, that maximizes the number of markings in $\mathcal{L}_{BF}$ that can be allowed by a decentralized supervisor.

The optimization problem solved in node $\Pi = \{\mathcal{L}_+, \mathcal{L}_-\}$ aims to find the decentralized set of GMECs (if one exists) that allows all the states in $\mathcal{L}_+$ and as many free legal states as possible, while forbidding all the states in $\mathcal{L}_- \cup \mathcal{U}$.

5.1. *ILP formulation*

5.1.1. *Variables*

The variables used in the ILP are listed below:

$$\gamma(\boldsymbol{m}) \in \{0,1\}, \quad \boldsymbol{m} \in \mathcal{L} \cup \mathcal{U} \tag{2}$$

$$L(p_c, p) \text{ integer}, \quad p_c \in P_c,\ p \in P \tag{3}$$

$$b(p_c) \text{ integer}, \quad p_c \in P_c \tag{4}$$

$$\tilde{L}(p_c, p) \text{ integer}, \quad p_c \in P_c,\ p \in P \tag{5}$$

$$\tilde{b}(p_c) \text{ integer}, \quad p_c \in P_c \tag{6}$$

$$Post_c(p_c, t), Pre_c(p_c, t) \geq 0, \quad p_c \in P_c,\ t \in T \tag{7}$$

$$\delta(p_c, \boldsymbol{m}) \in \{0,1\}, \quad p_c \in P_c,\ \boldsymbol{m} \in \mathcal{L} \cup \mathcal{U} \tag{8}$$

$$X^{Post}(p_c, t), X^{Pre}(p_c, t) \in \{0,1\}, \quad p_c \in P_c,\ t \in T \tag{9}$$

$$k(p_c, i) \in \{0,1\}, \quad p_c \in P_c,\ i \in I \tag{10}$$

where $P = \{1, \ldots, n_p\}$, $T = \{1, \ldots, n_t\}$, $P_c = \{1, \ldots, n_{dc}\}$, $I = \{1, \ldots, \nu\}$, for ease of readability.

The binary variables $\boldsymbol{\gamma} \in \{0,1\}^{|\mathcal{L}|+|\mathcal{U}|}$ identify the states allowed by the decentralized supervisor: $\gamma(\boldsymbol{m}) = 1$ if $\boldsymbol{m}$ is allowed and 0 otherwise. The γ variables are preset to 0 for illegal markings ($\boldsymbol{m} \in \mathcal{L}_- \cup \mathcal{U}$), and to 1 for markings in $\mathcal{L}_+$.

$$\gamma(\boldsymbol{m}) = 0, \quad \boldsymbol{m} \in \mathcal{L}_- \cup \mathcal{U} \tag{11}$$

$$\gamma(\boldsymbol{m}) = 1, \quad \boldsymbol{m} \in \mathcal{L}_+. \tag{12}$$

The integer variables $\boldsymbol{L} \in \mathbb{N}^{n_{dc} \times n_p}$, $\boldsymbol{b} \in \mathbb{N}^{n_{dc}}$, define the GMECs $(\boldsymbol{L}, \boldsymbol{b})$ to be determined (the maximum number of GMECs n_{dc} is a design parameter). $\tilde{L}(p_c, p)$ and $\tilde{b}(p_c)$ are real-valued auxiliary variables which coincide with the absolute values of the GMEC coefficients (see below). $\boldsymbol{Post}_c$ and $\boldsymbol{Pre}_c$ are the output and input matrices of the control subnet that define the supervisor net topology (*i.e.*, the weights of the arcs connecting the monitors to the PN transitions). The binary variables $\delta(p_c, \boldsymbol{m})$ associate each forbidden marking $\boldsymbol{m}$ to the control place p_c that forbids it. The binary variables $X^{Post}(p_c, t)$ [resp., $X^{Pre}(p_c, t)$] state whether there is an arc from transition t to monitor p_c [from monitor p_c to transition t] or not. The binary variable $k(p_c, i)$ states whether monitor p_c belongs to the ith site ($k(p_c, i) = 1$) or not ($k(p_c, i) = 0$).

5.1.2. The objective function

The objective function implements a hierarchy of objectives:

$$\max f = \sum_{\boldsymbol{m} \in \mathcal{L}} \gamma(\boldsymbol{m}) - \epsilon \sum_{p_c \in P_c} \left[\sum_{p \in P} \tilde{L}(p_c, p) + \tilde{b}(p_c) \right]. \tag{13}$$

The primary objective maximizes the number of markings that are allowed by the decentralized supervisor, while the secondary objective minimizes the absolute values of the GMEC coefficients. Coefficient ϵ provides the desired weighting of the two objectives (in the following, $\epsilon = 0.01$). The purpose of the secondary objective is to prevent ill-conditioning of the optimization problem (GMECs are defined up to a multiplicative constant). Furthermore, if there exists a solution with fewer monitors than n_{dc}, one or more of the obtained GMECs will have null parameters, allowing the designer to easily discard them *a posteriori*.

To ensure that the auxiliary variables $\tilde{L}(p_c, p)$ and $\tilde{b}(p_c)$ coincide with the absolute values of the GMEC coefficients, the following constraints are added:

$$\tilde{L}(p_c, p) \geq L(p_c, p), \quad p_c \in P_c,\ p \in P \tag{14}$$

$$\tilde{L}(p_c, p) \geq -L(p_c, p), \quad p_c \in P_c,\ p \in P \tag{15}$$

$$\tilde{b}(p_c) \geq b(p_c), \quad p_c \in P_c \tag{16}$$

$$\tilde{b}(p_c) \geq -b(p_c), \quad p_c \in P_c. \tag{17}$$

5.1.3. GMEC control policy constraints

The following constraints define the control policy exerted by each GMEC on the states:

$$\sum_{p \in P} L(p_c, p) m(p) - b(p_c) \leq (1 - \gamma(\boldsymbol{m})) M,$$
$$p_c \in P_c,\ \boldsymbol{m} \in \mathcal{L}_B \tag{18}$$

$$\sum_{p \in P} L(p_c, p) m(p) - b(p_c) \geq 1 - (1 - \delta(p_c, \boldsymbol{m})) M,$$
$$p_c \in P_c,\ \boldsymbol{m} \in \mathcal{L} \cup \mathcal{U} \tag{19}$$

$$\gamma(\boldsymbol{m}) + \sum_{p_c \in P_c} \delta(p_c, \boldsymbol{m}) \geq 1, \quad \boldsymbol{m} \in \mathcal{L} \cup \mathcal{U}. \tag{20}$$

Constraint (18) ensures that all the legal states with $\gamma(\boldsymbol{m}) = 1$ are allowed by all monitors. Constraint (19) takes care of all markings forbidden by some GMEC. By constraint (20) a marking is either allowed ($\gamma(\boldsymbol{m}) = 1$ and $\delta(p_c, \boldsymbol{m}) = 0$ for all control places) or forbidden ($\gamma(\boldsymbol{m}) = 0$ and $\delta(p_c, \boldsymbol{m}) = 1$ for at least one control place). Notice that, by way of constraint (19), when $\delta(p_c, \boldsymbol{m}) = 1$, $\boldsymbol{m}$ violates the GMEC associated to monitor p_c. The constant M is set to a sufficiently large value (big-M parameter), so that constraint (18) is always satisfied for $\gamma(\boldsymbol{m}) = 0$ and constraint (19) automatically holds if $\delta(p_c, \boldsymbol{m}) = 0$. The big-M parameter is set to $M = 10$ in the examples documented in the paper.

5.1.4. GMEC implementation constraints

Eq. (21) relates the weights of the arcs to the corresponding GMEC parameters $\boldsymbol{L}$, and (22) requires that the GMECs are satisfied in the initial marking.

$$Post_c(p_c, t) - Pre_c(p_c, t) = -\sum_{p \in P} L(p_c, p) C(p, t),$$
$$p_c \in P_c,\ t \in T \tag{21}$$

$$\sum_{p \in P} L(p_c, p) \boldsymbol{m}_0(p) - b(p_c) \leq 0, \quad p_c \in P_c. \tag{22}$$

The GMEC implementation must also account for the given decentralization conditions (each monitor can operate only on the transitions associated to its control site).

$$Post_c(p_c, t) \leq X^{Post}(p_c, t) M, \quad p_c \in P_c,\ t \in T \tag{23}$$

$$Pre_c(p_c, t) \leq X^{Pre}(p_c, t) M, \quad p_c \in P_c,\ t \in T \tag{24}$$

$$X^{Post}(p_c, t) + k(p_c, i) \leq 1, \quad p_c \in P_c,\ i \in I, t \notin T_i \tag{25}$$

$$X^{Pre}(p_c, t) + k(p_c, i) \leq 1, \quad p_c \in P_c,\ i \in I, t \notin T_i \tag{26}$$

$$\sum_{i \in I} k(p_c, i) = 1, \quad p_c \in P_c. \tag{27}$$

Conditions (23) and (24) set to zero the weights of missing arcs. Constraints (25) and (26) remove the arcs between a monitor p_c belonging to the ith site ($k(p_c, i) = 1$) and all transitions $t \notin T_i$. Finally, condition (27) specifies that each monitor must be assigned exactly to one module.

5.1.5. Controllability and observability constraints

To enforce controllability and observability according to the *structural* definition, it suffices to extend conditions (25)–(26) also to the transitions that are not controllable or observable from a particular site:

$$X^{Post}(p_c, t) + k(p_c, i) \leq 1, \quad p_c \in P_c,\ i \in I, t \in T_{uo_i} \tag{28}$$

$$X^{Pre}(p_c, t) + k(p_c, i) \leq 1, \quad p_c \in P_c,\ i \in I, t \in T_{uc_i}. \tag{29}$$

Behavioral controllability cannot be ensured in the decentralization framework by analysis of the reachability graph alone, but requires additional conditions on the structure of the supervisor. Such conditions do not prevent the use of arcs from monitor places to transitions that they cannot control, as long the disabling of such transitions does not occur exclusively due to an insufficient marking of such places (behavioral controllability).

Accordingly, to enforce *behavioral* controllability the following equation is introduced into the ILP in place of (29).

$$\gamma(\boldsymbol{m}_2) + \sum_{p_c \in P_c} \sum_{i \in I} \Phi(i,t) k(p_c,i) \delta(p_c, \boldsymbol{m}_2) \geq \gamma(\boldsymbol{m}_1),$$
$$\boldsymbol{m}_1, \boldsymbol{m}_2 \in \mathcal{L} \cup \mathcal{U},\ t \in T \text{ s.t. } \boldsymbol{m}_1[t\rangle \boldsymbol{m}_2. \quad (30)$$

Notice that the constraint is automatically satisfied for all possible values of the decision variables $k(p_c, i)$ and $\delta(p_c, \boldsymbol{m}_2)$ if $\gamma(\boldsymbol{m}_2) = 1$ or $\gamma(\boldsymbol{m}_1) = 0$ (for example when $\boldsymbol{m}_1 \in \mathcal{L}_- \cup \mathcal{U}$ or $\boldsymbol{m}_2 \in \mathcal{L}_+$). In detail, the constraint requires that if $\boldsymbol{m}_1$ is allowed by the solution ($\gamma(\boldsymbol{m}_1) = 1$) and $\boldsymbol{m}_2$ is not ($\gamma(\boldsymbol{m}_2) = 0$), the firing of a transition leading from $\boldsymbol{m}_1$ to $\boldsymbol{m}_2$ must be forbidden at least by one control place acting on a site from which t is controllable. More precisely, $\boldsymbol{m}_2$ is forbidden by p_c ($\delta(p_c, \boldsymbol{m}_2) = 1$), p_c belongs to site S_i ($k(p_c, i) = 1$) and site S_i controls transition t ($\Phi(i,t) = 1$). For an uncontrollable transition ($\sum_{i \in I} \Phi(i,t) = 0$), allowing $\boldsymbol{m}_1$ directly implies that also $\boldsymbol{m}_2$ must be allowed.

Notice that constraint (30) is nonlinear, since both k and δ are decision variables. However, it can be easily linearized by replacing product $k(p_c, i)\delta(p_c, \boldsymbol{m}_2)$ in expression (30) with an auxiliary binary variable $\psi(p_c, i, \boldsymbol{m}_2)$, and writing:

$$\psi(p_c, i, \boldsymbol{m}) \leq k(p_c, i)$$
$$\psi(p_c, i, \boldsymbol{m}) \leq \delta(p_c, \boldsymbol{m})$$
$$\psi(p_c, i, \boldsymbol{m}) \geq k(p_c, i) + \delta(p_c, \boldsymbol{m}) - 1$$

for $p_c \in P_c$, $i \in I$, and $\boldsymbol{m} \in \mathcal{L} \cup \mathcal{U}$. For this reason, with a slight abuse of terminology, the IP employing constraint (30) is referred to as an ILP problem.

In the following, the ILP (2)–(29) is denoted as ILP$_S$ to emphasize that it addresses the structural setting of the supervisor design. Accordingly, ILP$_B$ is used to denote the problem (2)–(28), (30).

5.2. Supervisor optimality and final considerations

Any solution of ILP$_S$ identifies a D$_S$-feasible subset, as proved by the following Lemma.

Lemma 4. *A set $\overline{\mathcal{L}} \subseteq \mathcal{L}$ is D$_S$-feasible if problem ILP$_S$ initialized with $\mathcal{L}_- = \mathcal{L} \setminus \overline{\mathcal{L}}$ admits a feasible solution such that $\gamma(\boldsymbol{m}) = 1$, $\forall \boldsymbol{m} \in \overline{\mathcal{L}}$.* ■

Proof. Thanks to constraints (23)–(27), any feasible solution of the ILP$_S$ problem abides by the decentralization requirements, *i.e.* $(\bullet p_c \cup p_c \bullet) \cap T_i = \emptyset$ for all p_c not belonging to control site S_i. The optimal solution of ILP$_S$ will allow a subset of the states in $\mathcal{L} \setminus \mathcal{L}_-$. However, since by assumption the obtained solution has $\gamma(\boldsymbol{m}) = 1$ for each $\boldsymbol{m} \in \overline{\mathcal{L}}$, the obtained decentralized supervisor enforces exactly $\overline{\mathcal{L}}$.

In addition, condition (28) forces each monitor place assigned to control site S_i ($k(p_c, i) = 1$) not to receive arcs from unobservable transitions. Similarly, constraint (29) forbids arcs from the locally uncontrollable (and unobservable, since $T_{c_i} \subseteq T_{o_i}$ implies $T_{uc_i} \supseteq T_{uo_i}$) transitions to the monitor. ■

The following result ensures that a solution to the ILP$_B$ problem guarantees D$_B$-feasibility.

Lemma 5. *A set $\overline{\mathcal{L}} \subseteq \mathcal{L}$ is D$_B$-feasible if problem ILP$_B$ initialized with $\mathcal{L}_- = \mathcal{L} \setminus \overline{\mathcal{L}}$ admits a feasible solution such that $\gamma(\boldsymbol{m}) = 1$, $\forall \boldsymbol{m} \in \overline{\mathcal{L}}$.* ■

Proof. As in Lemma 4, any feasible solution of the ILP$_B$ problem will respect the decentralization requirements (Definition 8.i), and the feasible solution considered in the assumption enforces exactly $\overline{\mathcal{L}}$.

Further on, constraint (30) enforces condition (ii) of Definition 8. Indeed, observe that it is a non trivial constraint only if $\boldsymbol{m}_1$ is allowed by the solution ($\gamma(\boldsymbol{m}_1) = 1$) and the firing of t in $\boldsymbol{m}_1$ leads to a marking $\boldsymbol{m}_2$, that is forbidden ($\gamma(\boldsymbol{m}_2) = 0$) by the solution. This makes $\boldsymbol{m}_1$ a *boundary* legal marking ($\boldsymbol{m}_1 \in \overline{\mathcal{L}}_b$ and t a transition belonging to $\mathcal{D}(\boldsymbol{m}_1)$). Now, the monitor p_c forbidding $\boldsymbol{m}_2$ ($\delta(p_c, \boldsymbol{m}_2) = 1$) will be actually disabling t in $\boldsymbol{m}_1$. For this to occur, its marking in correspondence to $\boldsymbol{m}_1$ will necessarily have to be less than the weight of the arc (p_c, t), as expressed by condition (ii) of Definition 8. Also, by constraint (30), p_c must operate on a control site S_i for which t is controllable ($\Phi(i,t) = 1$). ■

Lemma 6. *For any feasible solution of the ILP$_S$ [ILP$_B$] problem, the values of variables $L(p_c, p)$ and $b(p_c)$ identify a set of GMECs $(\boldsymbol{L}, \boldsymbol{b})$ which exactly enforce $\overline{\mathcal{L}} = \{\boldsymbol{m} \in \mathcal{L} \mid \gamma(\boldsymbol{m}) = 1\}$ in a decentralized way.* ■

Proof. By construction, constraint (18) guarantees that $\boldsymbol{Lm} \leq \boldsymbol{b}$, $\forall \boldsymbol{m} \in \overline{\mathcal{L}}$. Conversely, by conditions (19)–(20), there exists at least one GMEC that forbids a marking outside $\overline{\mathcal{L}}$. ■

Theorem 2. *Let $\overline{\mathcal{L}}^*$ be the solution returned by Algorithm 1. Then, the corresponding set of GMECs $(\boldsymbol{L}, \boldsymbol{b})$ obtained in the solution of the ILP problem identifies a maximally permissive decentralized GMEC-based supervisor.* ■

Proof. The set $\overline{\mathcal{L}}^*$ is the maximum cardinality B- and D-feasible subset of legal states that can be enforced by a decentralized GMEC-based supervisor. Indeed, the branching mechanism guarantees that all possible subsets of $\mathcal{L}$ are explored. The enumeration scheme is implicit, since the subsets that provably cannot yield feasible or optimal solutions are excluded from analysis. All the others are evaluated and the maximum cardinality one is returned. Finally, in view of Lemma 6, the associated ILP solution provides the supervisor enforcing $\overline{\mathcal{L}}^*$, which is therefore maximally permissive. ■

Remark 1. For simplicity reasons, the size optimization of the supervisor (in terms of the number n_{dc} of GMECs) is not carried out in this work, and n_{dc} is a fixed design parameter. This implies that to obtain a solution it might be necessary to increase n_{dc} and repeat the whole procedure. Notice also, that, thanks to the secondary objective function, if n_{dc} is selected larger than necessary, null GMECs will be obtained. By repeating the procedure for increasing values of n_{dc} one can, in principle, ascertain the structural optimality of the supervisor. ■

The problem can be further simplified, if, besides providing a tentative size for the supervisor, the designer should also predefine its structure, by assigning a priori the individual GMECs to the control sites (*i.e.*, pre-setting k). Notice that, in this case, constraint (30) automatically reduces to a linear one.

Remark 2. A plain GMEC supervisor is not guaranteed to exist not even in the centralized problem. However, it is shown in Cordone et al. (2013) that a nonlinear supervisor (obtained as a disjunction of GMECs) can always separate any two arbitrary legal and illegal sets (with no state in common), in the centralized case with full controllability and observability. To the authors' knowledge there are no equivalent results for the setting studied here (decentralized case with partial controllability and observability), that could be invoked to suggest a different class of supervisors than the plain GMEC ones. For this reason, besides computational complexity, the focus is here exclusively on this class of supervisors. ■

6. Computational complexity

B&B methods are typically characterized by a large difference between the theoretical worst-case complexity, which is intrin-

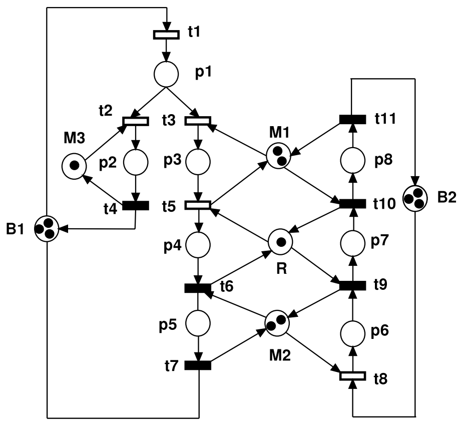

Fig. 2. Petri net of example.

sically exponential, and the practical average-case performance, which can be reasonably efficient if the algorithm is endowed with smart pre-processing mechanisms and with tight bounds on the objective function. In the following, a rough over-estimation of the worst-case complexity of the presented algorithm is given, to complement the experimental results discussed in Section 7.

The external cycle of the method (see Algorithm 1) is a B&B over the legal markings. If the branching tree were explored exhaustively, the overall complexity would be of order $O(2^{n_V} \cdot (c_2 + c_3 + c_{ILP}))$, where c_2, c_3, and c_{ILP} are the worst-case complexities of the pre-processing (Algorithm 2), B-feasibility (Algorithm 3) and D-feasibility phases (see ILP problem of Section 5).

The pre-processing phase requires a full exploration of the reachability graph, to build the initial SCC and to perform other minor operations (Cormen, Leiserson, Rivest, & Stein, 2009). Therefore, its complexity is $O(n_A)$. The B-feasibility phase requires repeated examinations of the reachability graph. This step has been fully analyzed in Basile et al. (2013) and found to be in $O(n_V n_A)$. Finally, the ILP problem can be solved in polynomial time $p(n_c, n_p)$ for each possible value of the binary variables, which are $n_{bin} = (n_c + 1)n_V + n_c(2n_t + n_i)$, where $n_i = |I|$. Therefore, its complexity is $O(2^{n_{bin}} p(n_c, n_p))$. Obviously, the complexity of the ILP problem largely dominates that of the previous phases, so that the overall complexity is of $O(2^{n_V + n_{bin}} p(n_c, n_p))$.

In practice, neither the external B&B, nor the ILP problem is solved exhaustively. In particular, the rules implemented in Algorithms 2–3, as well as the use of an upper bound for the solution in Algorithm 1, allow to discard most of the branching nodes without solving them explicitly. Similar mechanisms are also exploited by the ILP solver to reduce the computational effort. The computational results discussed in Section 7 confirm that the number of explored nodes is a small fraction of the theoretical estimate.

7. Simulation example

7.1. Centralized supervisor design

Consider the PN represented in Fig. 2 taken from Ghaffari, Rezg, and Xie (2003), for which one wants to design a GMEC-based supervisor that guarantees liveness, reversibility and controllability. Resource (M_1, M_2, M_3, and R) and idle (B_1 and B_2) places are considered part of the process. The PN has 331 reachable markings, only 300 of which are included in the initial SCC.

Consider first the centralized supervisor design problem and assume that $T_c = \{t_1, t_2, t_3, t_5, t_8\}$ and $T_{uc} = T \backslash T_c$ (all transitions are assumed observable). The resulting optimal solution using the behavioral approach (*i.e.* employing ILP$_B$ for D$_B$-feasibility) has 2 GMECs:

$$m_4 + m_6 \leq 2 \tag{31}$$

$$3m_3 + m_6 + m_7 \leq 6 \tag{32}$$

and allows 295 states of the 300 maximum possible (5 states are rejected for behavioral controllability reasons). The obtained solution is also structurally controllable (control places have outgoing arcs only towards transitions in T_c): $p_{c1}\bullet = \{t_5, t_8\}$ and $p_{c2}\bullet = \{t_3, t_8\}$. Accordingly, reapplying the method with ILP$_S$ yields the same solution. It is interesting to note that if t_5 is not assumed controllable, the optimal (behavioral) solution of the problem allows 280 markings only, using again (31) plus the following GMEC:

$$3m_3 + 2m_6 + m_7 \leq 6. \tag{33}$$

As already commented, the monitor implementing (31) has an arc towards t_5, which is now an uncontrollable transition, but this is fine in the behavioral setting, since it is never exclusively responsible of its disabling. Indeed, there are 45 markings in which two or more places (among which the mentioned monitor) disable t_5, but none in which only the monitor disables it. As for GMEC (33), it is a restriction of (32) that introduces arcs from the monitor place to the controllable transitions t_3 and t_8. Using the structural approach a different supervisor is obtained that allows the same 280 states. It employs and (33) plus the following GMEC:

$$m_3 + m_4 + 2m_6 \leq 4. \tag{34}$$

In all the analyzed cases, the optimal solution is found already at the first node of the B&B procedure. Indeed, the largest B-feasible subset contained in $\mathcal{L}$ has 295 and 280 markings, depending on the controllability of t_5. Since it also solves the ILP problem, it is the maximal B- and D-feasible subset, which corresponds to the optimal supervisor.

7.2. Decentralized supervisor design: part 1

Now, consider the same problem in a decentralized setting, where one can employ monitors of two control sites, defined in 3 alternative scenarios (differing only for the role of t_5) as follows:

$$\begin{aligned}
\text{case (a) } & S_1 : T_1 = [t_5\ t_6\ t_8\ t_9], \quad \text{with } T_{c_1} = [t_8],\\
& S_2 : T_2 = [t_3\ t_5\ t_7\ t_8\ t_9\ t_{10}], \quad \text{with } T_{c_2} = [t_3\ t_8],\\
\text{case (b) } & S_1 : T_1 = [t_5\ t_6\ t_8\ t_9], \quad \text{with } T_{c_1} = [t_5\ t_8],\\
& S_2 : T_2 = [t_3\ t_5\ t_7\ t_8\ t_9\ t_{10}], \quad \text{with } T_{c_2} = [t_3\ t_8],\\
\text{case (c) } & S_1 : T_1 = [t_5\ t_6\ t_8\ t_9], \quad \text{with } T_{c_1} = [t_8],\\
& S_2 : T_2 = [t_3\ t_5\ t_7\ t_8\ t_9\ t_{10}], \quad \text{with } T_{c_2} = [t_3\ t_5\ t_8].
\end{aligned}$$

It is further assumed that $T_{o_i} = T_i, i = 1, 2$.

The B&B algorithm has been tested in all three scenarios using both the structural and behavioral approaches. The algorithm performance is summarized in Table 1. Apparently, two GMECs are sufficient to achieve the same performance of the centralized supervisor (*i.e.*, 295 or 280 allowed states, depending on the controllability of t_5), but the two controllability notions have a different impact on the efficacy of the control sites. For example in scenario (a) the behavioral approach obtains the maximally permissive solution exploiting both control sites, whereas the structural approach cannot find any use for S_1, given that it cannot control any transition of the left side of the process. The behavioral solution has a monitor in site S_1 with an arc going to t_5, which is acceptable, since it is never responsible for an exclusive inhibitory action on the firing of the (uncontrollable) transition. In scenario (c) both the behavioral and structural approaches find it more convenient to use S_2 alone. Finally, the different degree of permissivity of the two approaches is apparent when they are compared in identical conditions (same scenario and equal distribution of GMECs to the control sites).

Table 1
Algorithm performance on the example: part 1.

Scenario	GMECs in S_1	GMECs in S_2	$\lvert\overline{\mathcal{L}}\rvert$	B&B nodes
Structural approach				
a	1	1	259	1
	0	2	280	1
b	1	1	295	1
c	1	1	259	1
	0	2	295	1
Behavioral approach				
a	1	1	280	1
b	1	1	295	1
c	1	1	280	1
	0	2	295	1

Table 2
Algorithm performance on the example: part 2.

Scenario	GMECs in S_1	GMECs in S_2	$\lvert\overline{\mathcal{L}}\rvert$	B&B nodes
Structural approach				
a	0	1	93	35
b	1	1	128	99
c	0	2	130	69
Behavioral approach				
a	1	2	128	25
b	1	2	138	67
c	1	2	142	299

7.3. *Decentralized supervisor design: part 2*

An even more interesting case unfolds if one removes place R from the PN, and adds the corresponding static constraint:

$$m_4 + m_7 \leq 1 \tag{35}$$

to the supervisor design requirements. In other words, our aim here is to evaluate the cost of imposing the requirement corresponding to place R (together with the behavioral properties of liveness, reversibility, and controllability), which was previously centralized, in a decentralized way.

Constraint (35) appears to be particularly hard to enforce in a decentralized way, both with the structural and behavioral approaches, resulting in a non-trivial branching process with several nodes examined, and a significantly lower permissivity compared to the centralized case (see Table 2).

Increasing n_{dc} to allow more GMECs per control site does not provide solution improvements. In particular, the structural approach cannot fully exploit all the given degrees of freedom and is only capable of producing quite conservative solutions (for which few GMECs are sufficient). For example, the structural solution to case (a) consists of a single GMEC:

$$m_3 + m_4 + m_5 + m_6 + m_7 \leq 1,$$

that essentially allows only one of the two processes (the left downward sequence or the right upward sequence) to be active at a time.

To give the reader a feel of the branching process, Fig. 3 provides a picture of the branching tree relative to case (a), addressed with the behavioral approach. Nodes are numbered in order of generation, and are accompanied by either the upper bound information ($|\mathcal{L}_{DF}|$) or the reason for node elimination (violation of either B- or D-feasibility). When a lower bound equal to the upper bound is obtained, the latter is graphically emphasized with a square. The initial problem has $|\mathcal{L}_+| = 1$, $|\mathcal{L}_-| = 98$, and $|\mathcal{L}_\times| = 149$. When a node is branched, two children nodes are generated, the first with $\mathcal{L}_-$ augmented by one state, and the second with $\mathcal{L}_+$ augmented by one state. Further assignments are sometimes added in the preprocessing phase. The branching process initially goes down the left side, where a solution to the ILP$_B$ problem is obtained with 142 states for several nodes. More in detail, the states added to

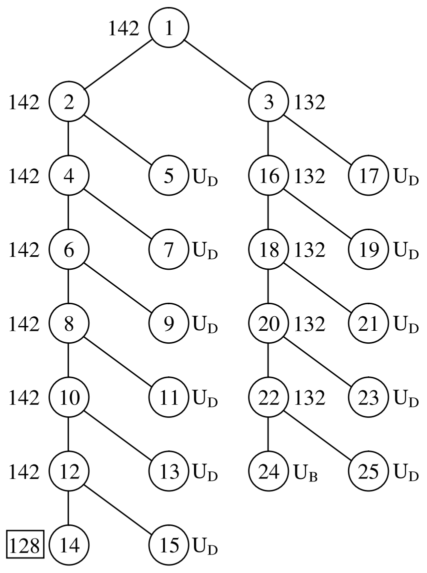

Fig. 3. Branching tree: each node with UB or unfeasibility information (U_B = violation of B-feasibility, U_D = violation of D-feasibility).

$\mathcal{L}_-$ at nodes 2, 4, 6, 8, 10, 12, progressively reduce the size of $\mathcal{L}_{BF}$, whereas the same $\mathcal{L}_{DF}$ is obtained. Conversely, at the nodes 5, 7, 9, 11, 13, 15, a state is added to $\mathcal{L}_+$ which is not allowed by the D$_B$-feasible solution obtained at the father node. Apparently no other D$_B$-feasible solution is compatible with the new state assignments. Finally, at node 14 an important reduction of the size of $\mathcal{L}_{BF}$ occurs, which forces a different solution of the ILP$_B$ problem as well. Since $\mathcal{L}_{DF} = \mathcal{L}_{BF}$, a candidate solution for optimality is obtained. However, since there is still an open node (3) with an upper bound higher than 128, a better solution could still exist and the B&B continues the exploration for an additional 10 nodes, before the optimality of the solution can be definitively assessed.

8. Conclusions

A novel approach has been presented for the synthesis of compact and decentralized supervisors for PN systems. Both static and behavioral control specifications can be considered in the method, but the focus is here mainly on the latter, which pose the greater difficulties in the supervisor design. Particular emphasis is attributed to the controllability property, which is enforced in two different ways, based on structural and behavioral arguments. The method operates on the state space of the PN, searching for the maximal set of reachable markings that configures a subgraph of the reachability graph with all the required behavioral properties and that is also enforceable by a decentralized supervisor. For this reason, the two separate notions of B- and D-feasibility have been introduced, as well as conditions for their obtainment. In particular, B-feasibility is ascertained by graph theory tools on the reachability subgraph, whereas D-feasibility is established by solving an ILP which provides the supervisor GMECs. A branch and bound method has been developed to systematically and efficiently explore all possible subsets of the legal states of the centralized case to find the maximally permissive one that meets the constraints.

References

Barret, G., & Lafortune, S. (2000). Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control*, *45*(9), 1620–1638.
Basile, F., Cordone, R., & Piroddi, L. (2013). Compact and decentralized supervisors for general constraint enforcement in Petri net models. In *52nd IEEE conference on decision and control, CDC'13* (pp. 7279–7284). Florence, Italy.

Basile, F., Cordone, R., & Piroddi, L. (2013). Integrated design of optimal supervisors for the enforcement of static and behavioral specifications in Petri net models. *Automatica*, *49*, 3432–3439.

Basile, F., Giua, A., & Seatzu, C. (2007). Supervisory control of Petri nets with decentralized monitor places. In *26th American control conference, ACC'07* (pp. 4957–4962). New York, NY, USA.

Basile, F., Giua, A., & Seatzu, C. (2008). Some new results on supervisory control of Petri nets with decentralized monitor places. In *17th IFAC world congress* (pp. 531–536). Seoul, Korea. July.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems* (II ed.). Springer.

Chen, H., & Hu, B. (1991). Distributed control of discrete event systems described by a class of controlled Petri nets. In *IFAC int. symposium on distributed intelligence systems*.

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

Chen, Y. F., Li, Z. W., Khalgui, M., & Mosbahi, O. (2011). Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems. *IEEE Transactions on Automatic Science Engineering*, *8*(2), 374–393.

Cordone, R., Nazeem, A., Piroddi, L., & Reveliotis, S. A. (2013). Designing optimal deadlock avoidance policies for sequential resource allocation systems through classification theory: Existence results and customized algorithms. *IEEE Transactions on Automatic Control*, *58*(11), 2772–2787.

Cordone, R., Nazeem, A., Piroddi, L., & Reveliotis, Spyros (2012). Maximally permissive deadlock avoidance for sequential resource allocation systems using disjunctions of linear classifiers. In *51st IEEE conf. on decision and control* pp. (7244–7251). Maui, HI, USA.

Cordone, R., & Piroddi, L. (2011). Monitor optimization in Petri net control. In *7th IEEE conf. on automation science and engineering* (pp. 413-418). Trieste, Italy.

Cordone, R., & Piroddi, L. (2013). Parsimonious monitor control of Petri net models of FMS. *IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans)*, *43*(1), 215–221.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. The MIT Press.

Dideban, A., & Alla, H. (2008). Reduction of constraints for controller synthesis based on safe Petri nets. *Automatica*, *44*(7), 1697–1706.

Dideban, A., Zareiee, M., & Alla, H. (2013). Controller synthesis with highly simplified linear constraints. *Asian Journal of Control*, *15*(1), 80–94. Fumagalli, I., Piroddi, L., & Cordone, R. (2010). A reachability graph partitioning technique for the analysis of deadlock prevention methods in bounded Petri nets. In *American control conference, ACC2010* (pp. 3365–3370). Baltimore, MD, USA.

Ghaffari, A., Rezg, N., & Xie, Xiaolan (2003). Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Transactions on Robotics and Automation*, *19*(1), 137–141.

Giua, A., Di Cesare, F., & Silva, M. (1993). Petri net supervisors for generalized mutual exclusion constraints. In *IFAC world congress* (pp. 267–270). Sydney, Australia. July.

Giua, A., DiCesare, F., & Silva, M. (1992). Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *IEEE int. conf. on systems, man and cybernetics* (pp. 974–979). Chicago, IL, USA.

Guan, X., & Holloway, L.E. (1997). Control of distributed discrete event systems modeled as Petri nets. In *1997 American control conference* (pp. 2342–2347). Albuquerque, NM, USA. June.

Iordache, M. V., & Antsaklis, P. J. (2006). Decentralized control of Petri nets with constraint tansformation. *IEEE Transactions on Automatic Control*, *51*(2), 376–381.

Lin, F., & Wonham, W. M. (1990). Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, *35*(12), 1330–1337.

Moody, J. O., & Antsaklis, P. J. (2000). Petri net supervisors for DES with uncontrollable and unobservable transitions. *IEEE Transactions on Automatic Control*, *45*(3), 462–476.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Nazeem, A., & Reveliotis, S. (2012). Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: The nonlinear case. *IEEE Transactions on Automatic Control*, *57*(7), 1670–1684.

Nazeem, A., Reveliotis, S., Wang, Y., & Lafortune, S. (2010). Optimal deadlock avoidance for complex resource allocation systems through classification theory. In *10th IFAC int. workshop on discrete event systems* (pp. 277–284). Berlin, Germany.

Nazeem, A., Reveliotis, S., Wang, Y., & Lafortune, S. (2011). Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: The linear case. *IEEE Transactions on Automatic Control*, *56*(8), 1818–1833.

Reveliotis, S. A., & Choi, J. Y. (2006). Designing reversibility-enforcing supervisors of polynomial complexity for bounded Petri nets through the theory of regions. In S. Donatelli, & P. S. Thiagarajan (Eds.), *Lecture notes in computer science*: *volume 4024. Petri nets and other models of concurrency — ICATPN 2006* (pp. 322–341). Berlin, Heidelberg: Springer.

Rudie, K., & Wonham, W. M. (1992). Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, *37*(11), 1692–1708.

Yamalidou, K., Moody, J. O., Lemmon, M. D., & Antsaklis, P. J. (1996). Feedback control of Petri nets based on place invariants. *Automatica*, *32*(1), 15–28. Zareiee, M., Dideban, A., & Orouji, A. A. (2014). Safety analysis of discrete event systems using a simplified Petri net controller. *ISA Transactions*, *53*(1), 44–49.