Supervisory Control of Timed Discrete Event Systems with Logical and Temporal Specifications

Francesco Basile, Roberto Cordone, and Luigi Piroddi

***Abstract*—A novel framework is introduced for the supervisory control (SC) of timed discrete event systems, based on Time Petri nets. The method encompasses both logical (markings to reach or avoid) and temporal specifications (arrival and departure times in specific markings). It relies on the construction of a partial forward reachability graph of the Modified State Class Graph type and the formulation of integer linear programming problems to establish suitable firing time intervals (FTIs) for the controllable transitions. The SC algorithm provides for each enabled controllable transition the largest FTI that guarantees that the specifications are met, irrespectively of the firing times of the uncontrollable transitions.**

***Index Terms*—Discrete event systems, Time Petri nets, Supervisory control, Temporal specifications, Reachability, Safety.**

I. Introduction

The most consolidated framework for the supervisory control (SC) of timed discrete event systems (TDESs) has been proposed in [1]. In detail, an event is constrained to occur between a lower and an upper time bound relative to its enabling instant, the evolution of time being accounted for by a postulated global digital clock, to which the occurrence of events is synchronized. The lower bound typically represents a delay, due, *e.g.*, to communication or control enforcement issues, while the upper bound (if finite) determines a hard deadline to the event occurrence, imposed by a legal specification or physical necessity. Events are classified as either *prospective* or *remote* if the upper bound is finite or infinite, respectively. An event can be disabled only if it is remote, by indefinitely preventing it from occurring. Prospective events cannot be permanently disabled, as this would result in a behavior incompatible with the uncontrolled system (the controller can only restrict the behavior of the uncontrolled system). The control action can also "force" an event (denoted also *forcible* event) to fire at a specific time instant, preempting a tick of the clock, provided that the time elapsed from its enabling is within the associated firing time interval (FTI) [1]. Notice that in the timed context the tick must be allowed to occur unless a forcible event is enabled. In the framework of [1], all non-prohibitible events (*i.e.*, events that cannot be prevented from occurrence) are labeled as uncontrollable, associating the notion of controllability only to events that can be disabled. We here use a broader interpretation of controllability, considering as controllable any event whose firing time can be influenced by the supervisor.

The described approach for the modeling of TDESs relies on the elementary framework of regular languages and finite automata, and represents time by the explicit enumeration of ticks. Both these features contribute to the state explosion problem, which makes this approach hardly useful in practice. Furthermore, due to this inherent complexity, only the enforcement of logical specifications is typically addressed (see, *e.g.*, [2]), although the framework can handle temporal ones as well. Typical objectives in the SC framework are reachability and safety, *i.e.* the requirements to reach or avoid specific states, respectively. These can be formulated in a purely untimed form, or be associated to time constraints, to enforce performance-related goals. In the usual practice, reachability and safety requirements are addressed separately from performance-related ones, the former at the logical control level and the other at the task planning level. This practice typically yields over-conservative control policies and ultimately leads to under-performing control systems.

It is worth recalling that the supervisor is not in charge of deciding which enabled transition must fire. Hence, if two or more enabled transitions are in conflict, another agent in an upper level of the control architecture –the dispatcher– will decide which of them must fire. This paper does not address the problem of designing a dispatching policy, which is more a scheduling-like problem than a control type one. The interested reader can refer to [3] for further details on this issue.

The state explosion problem can be alleviated to some extent by adopting Petri nets (PNs) instead of automata, in view of their intuitive and compact graphical representation and their convenient mathematical formulation. Several models based on PNs have been introduced to describe the behavior of TDESs, such as timed PNs [4] and Time PNs (TPNs) [5]. In timed PNs the enabled transitions are associated with a constant time delay, while in TPNs the enabled transitions may fire within given FTIs, similarly to the previously introduced framework of [1] for TDESs. Furthermore, TPNs are widely used in the literature for real-time system specification and verification [6], [7]. For these reasons, in this paper we focus solely on TPNs, and we address the challenge of incorporating *temporal* specifications in the control design.

F. Basile is with the Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata, Università degli Studi di Salerno, Fisciano (SA), Italy, e-mail: fbasile@unisa.it.

R. Cordone is with the Department of Computer Science "Giovanni Degli Antoni", Università degli Studi di Milano, Milano, Italy, e-mail: roberto.cordone@unimi.it.

L. Piroddi is with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano, Italy, e-mail: luigi.piroddi@polimi.it.

Postprint - Work published on IEEE Transactions on Automatic Control (http://dx.doi.org/10.1109/TAC.2021.3093618)

For untimed PNs the SC problem is generally formulated in terms of states (the net markings), but this is not particularly convenient for timed models such as TPNs, since the state concept that follows from the way time is accounted for involves both markings and clocks and ultimately results in an infinite state space. For this reason, several abstractions have been developed in the literature, such as the State Class Graph (SCG) [5], with its variants (see, *e.g.*, [8], [9]). The SCG aggregates infinite states of the net system in a single node of a graph. While it preserves linear properties, such as marking reachability and firing sequences, it does not preserve the full state space representation, which may be a problem for SC if specifications explicitly involving state and time must be met. Other abstractions, such as the strong SCG, use clocks to characterize state classes rather than firing domain constraints. Because of this, they yield a larger graph than the SCG, that does not preserve the finiteness property for all bounded TPNs. By contrast, the Modified SCG (MSCG) employs a symbolic characterization of the FTIs of the enabled transitions in a class [10], [11]. This formulation depends on the time spent in that class [11], and avoids clocks. The MSCG allows a complete representation of the evolution of a TPN system and preserves the finiteness property for bounded TPNs, provided that cyclic sequences of null duration cannot occur (which is typically a non-limiting assumption). Furthermore, it allows to explicitly formulate the duration of a sequence of transitions as the sum of the times spent in each class along the sequence. This last unique property of the MSCG is essential when addressing *temporal* specifications. For these reasons we here adopt the MSCG as a reference tool for reachability analysis.

A number of contributions in the literature concerns the application of logical SC laws to TPNs. For example, a Control Class Graph (an extension of the SCG) is proposed in [12], [13], to take into account also a non-negligible delay for control computation. This graph is equivalent to an untimed automaton, that accounts only for the ordering of activities induced by timing bounds, while the timing structure is lost, so that only logical specifications can be considered in the supervisor synthesis. An online algorithm is used in [14] to establish if the logical supervisory control law can be relaxed when applied to the TPN, thus reducing its over-conservativeness. This algorithm computes online partial MSCGs, where only the uncontrollable transitions are considered.

In [15] and [16] logical reachability and safety objectives are pursued and the SC synthesis is carried out working on state regions, employing TPNs and Timed Automata (TA), respectively. The control problem is addressed by computing the winning states of the model, *i.e.* states which will not lead, by the firing of uncontrollable transitions, to an undesired state. The computation of the winning states is based on the concept of controllable predecessors of states. Unfortunately, this turns out to be computationally demanding.

In [17] a forward on-the-fly method is proposed for the synthesis of maximally permissive controllers for TPNs guaranteeing safety. On the basis of the exploration of the SCG, all sequences containing at least a controllable transition and leading to undesired states are extracted, and FTIs to be avoided are determined. The merit of this approach is that it is the first to compute illegal FTIs for the controllable transitions, as opposed to winning states. This reduces the computational effort and emphasizes the advantage of using TPNs over TAs or finite state automata. However, the method requires that the complete SCG be computed *a priori*.

Much fewer works address the design of SC laws that enforce *temporal* reachability and safety specifications, and mostly in the context of TAs. A timed reachability problem can be studied using an augmented automaton, that includes an additional clock variable, which is never reset to zero and hence measures the time elapsed since the beginning of a run [16], [18]. A standard on-the-fly symbolic algorithm for TAs is typically employed for the reachability analysis. This involves the exploration of all paths in the automaton, considering for each path all the possible choices of times in which a state transition could occur. Thus, the algorithm has to manipulate a significant (exponential) number of zones (special polyhedra in the clock space) [18]. Indeed, the reachability algorithm was originally conceived for verification, and, consequently, it is exhaustive (it computes all possible runs of the automaton). This makes it not particularly convenient to handle temporal specifications with TAs.

In TPNs, time is implicitly represented by a real variable, which generally leads to more compact and concise models, that are more amenable to dealing with temporal specifications. For example, a basic type of temporal specification, concerning firing deadlines for the transitions of the TPN, is addressed in [19] without resorting to state exploration. However, the resulting controller is not maximally permissive, and the considered type of temporal specifications is not sufficiently general to handle complex problems. To the best of the authors' knowledge, general temporal specifications have not yet been considered in the SC synthesis for TPNs.

This paper elaborates on the results presented in [17], proposing a SC framework that can handle general temporal specifications. As in [17] (and in a number of other contributions), the control action operates by restricting the FTI of an enabled controllable transition. This paper improves the results presented in [17] from several points of view:

- A SC framework is developed for TPNs to address both logical and temporal specifications of a general form. The specifications include target markings to be reached in succession (*reachability*), and markings to avoid (*safety*). Arrival and departure times in the target markings are required to be in prescribed time intervals (*performance*).
- The MSCG is employed as a state space abstraction since it allows to address temporal requirements in a straightforward way, thanks to the parametric formulation of the firing times. The full computation of the MSCG is avoided. Instead, the firing vectors that may allow the system to fulfil the reachability and safety requirements are enumerated, and based on these a partial MSCG is constructed that encompasses all and only the TTSs that potentially meet the temporal specifications.
- The legal FTIs for the enabled controllable transitions are computed by solving a series of simple Integer Linear Programming (ILP) problems, that can be constructed in a modular way from the partial MSCG.

II. Preliminaries and background

A. Petri nets

A PN [20] is a quadruple $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$, where P is a set of n_p places (represented by circles), T is a set of n_t transitions (represented by bars), $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{N}^{n_p \times n_t}$ are the *pre-* and *post-* incidence matrices, $\mathbb{N}$ denoting the set of non-negative integers. The *incidence matrix* is $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$. A *marking* (*i.e.*, the net state) is a vector $\boldsymbol{m} \in \mathbb{N}^{n_p}$ that assigns to each place p a nonnegative integer number $m(p)$ of tokens (represented by black dots). We will sometimes use the multiset formalism $\sum_{i=1}^{n_p} m(p_i)p_i$ to represent marking $\boldsymbol{m}$. A PN *system* $\langle N, \boldsymbol{m}_0\rangle$ is a net N with an initial marking $\boldsymbol{m}_0$.

A transition t is enabled at $\boldsymbol{m}$ if $\boldsymbol{m} \geq \boldsymbol{Pre}(\cdot, t)$. The firing of an enabled transition t at a marking $\boldsymbol{m}$ yields a marking $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}(\cdot, t)$. The enabling (firing) of t in $\boldsymbol{m}$ is often denoted $\boldsymbol{m}[t\rangle$ ($\boldsymbol{m}[t\rangle\boldsymbol{m}'$). We denote as $T_e(\boldsymbol{m}) = \{t \in T \mid \boldsymbol{m} \geq \boldsymbol{Pre}(\cdot, t)\}$ the set of transitions enabled at $\boldsymbol{m}$, and as $n_e(\boldsymbol{m}) = |T_e(\boldsymbol{m})|$ and $I_e(\boldsymbol{m}) = \{j \in \{1, \ldots, n_t\} \mid t_j \in T_e(\boldsymbol{m})\}$ their number and indices, respectively. A transition sequence (TS) $s = t_{i_1}\ t_{i_2}\ \ldots\ t_{i_k}$ is enabled at a marking $\boldsymbol{m}_0$ (briefly denoted as $\boldsymbol{m}_0[s\rangle$) if $\boldsymbol{m}_0[t_{i_1}\rangle\boldsymbol{m}_1$, $\boldsymbol{m}_1[t_{i_2}\rangle\boldsymbol{m}_2$, $\ldots$, $\boldsymbol{m}_{k-1}[t_{i_k}\rangle$, and the effect of its firing can be computed in a single operation using the state equation $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}\boldsymbol{\sigma}$, where $\boldsymbol{\sigma} = \boldsymbol{\sigma}(s)$ is the firing count vector (FCV) associated to s (σ_j being the number of times that transition t_j fires in s). Accordingly, the firing of s in $\boldsymbol{m}_0$ is denoted $\boldsymbol{m}_0[s\rangle\boldsymbol{m}_k$. We will characterize an FCV as *admissible* if it admits at least one enabled TS.

A vector $\boldsymbol{y} \geq 0$ such that $\boldsymbol{C}\boldsymbol{y} = \boldsymbol{0}$ is called a T-invariant. An admissible TS whose FCV coincides with a T-invariant produces a null net marking variation ($\boldsymbol{m}' = \boldsymbol{m}$), thus taking the PN system back to the initial marking. The set of transitions $||\boldsymbol{y}|| = \{t_j \in T \mid y_j > 0\}$ is called the support of the T-invariant. A T-invariant $\boldsymbol{y}$ has minimal support if there does not exist another T-invariant $\boldsymbol{y}'$ such that $||\boldsymbol{y}'|| \subset ||\boldsymbol{y}||$. A T-invariant $\boldsymbol{y}$ is minimal if there does not exist another T-invariant $\boldsymbol{y}'$ such that $\boldsymbol{y}' \leq \boldsymbol{y}$. A *minimal-support* (MS) T-invariant has minimal support and is minimal. The set of MS T-invariants is finite and constitutes a basis: any T-invariant can be obtained by linear combination of MS T-invariants [21].

A marking $\boldsymbol{m}$ is *reachable* in $\langle N, \boldsymbol{m}_0\rangle$ if there exists a TS s such that $\boldsymbol{m}_0[s\rangle\boldsymbol{m}$. The set $R(N, \boldsymbol{m}_0)$ of all markings reachable from $\boldsymbol{m}_0$ is the *reachability set* of $\langle N, \boldsymbol{m}_0\rangle$. 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 system is bounded iff all its places are bounded.

B. Firing vectors and T-invariants

Given a PN system $\langle N, \boldsymbol{m}_0\rangle$, where $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ and a target marking $\boldsymbol{m}_t$, one can define a *complemented* net [22] $N_c = (P, T \cup \{t_c\}, \boldsymbol{Pre}_c, \boldsymbol{Post}_c)$, where the auxiliary transition $t_{n_t+1} = t_c$ is connected to the places of the original PN as specified by the additional last column of the *pre-* and *post-* incidence matrices, $\boldsymbol{Pre}_c = [\boldsymbol{Pre}\ \boldsymbol{m}_t]$ and $\boldsymbol{Post}_c = [\boldsymbol{Post}\ \boldsymbol{m}_0]$. As a result, the incidence matrix of N_c is given by $\boldsymbol{C}_c = [\boldsymbol{C}\ (\boldsymbol{m}_0 - \boldsymbol{m}_t)]$, which implies that if t_c is fired in $\boldsymbol{m}_t$ the system goes back to the initial marking $\boldsymbol{m}_0$. Indeed, $\boldsymbol{m}_t + \boldsymbol{C}_c\boldsymbol{\epsilon}^{(n_t+1)} = \boldsymbol{m}_t + (\boldsymbol{m}_0 - \boldsymbol{m}_t) = \boldsymbol{m}_0$, where $\boldsymbol{\epsilon}^{(j)}$ is the jth versor of the coordinate space. Then, a necessary condition for the reachability of $\boldsymbol{m}_t$ from $\boldsymbol{m}_0$ in N is the existence of a T-invariant of N_c with a *single* firing of t_c. The corresponding FCV for N is trivially obtained removing the $n_t + 1$ entry from such T-invariant.

T-invariants of N_c of this type are denoted singular complementary T-invariants (SCTs) [22]. The other T-invariants of N_c have either none or multiple firings of t_c, and are denoted non-complementary T-invariants (NCTs) and non-singular complementary T-invariants (NSCTs), respectively. Notice that the NCTs correspond to the T-invariants of N. Accordingly, we can partition the set Y of T-invariants of N_c as $Y = Y_0 \cup Y_1 \cup Y_2$, where $Y_0 = \{\boldsymbol{y} \in Y \mid y_{n_t+1} = 0\}$, $Y_1 = \{\boldsymbol{y} \in Y \mid y_{n_t+1} = 1\}$, and $Y_2 = \{\boldsymbol{y} \in Y \mid y_{n_t+1} \geq 2\}$ are the NCTs, the SCTs, and the NSCTs, respectively. Any element $\boldsymbol{y} \in Y_1$ can be constructed as a linear combination of the MS T-invariants $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(p)} \subseteq Y$ [22]:

$$\boldsymbol{y} = \sum_{j=1}^{p} \alpha_j \boldsymbol{y}^{(j)} \tag{1}$$

provided the coefficients $\alpha_j \geq 0$, $j = 1, \ldots, p$, satisfy the following conditions: i) all elements of $\boldsymbol{y}$ are integers, and ii) $y_{n_t+1} = \sum_{j=1}^{p} \alpha_j y^{(j)}_{n_t+1} = 1$. It follows from condition (i) that the α_j coefficients must be rational numbers.

The SCTs of N_c corresponding to admissible FCVs of N (*i.e.* such that there exist at least one enabled TS taking the PN system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_t$) are named *admissible* SCTs and are collected in the set $Y_1^a \subseteq Y_1$. The details of their computation are discussed in Section IV.

C. Time Petri Nets

A TPN is a pair $N_\tau = (N, Q)$, where $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ is the net structure and $Q : T \to \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$ is the set of *static* intervals associated with the transitions, where $\mathbb{R}_0^+$ is the set of non-negative real numbers. In detail, $Q(t_i) = (l_i, u_i)$ for each transition $t_i \in T$, where $0 \leq l_i < \infty$, $u_i \geq l_i$ (u_i may also be ∞). Transition t_i may (must) fire if it has remained logically enabled uninterruptedly for at least l_i time units (for u_i time units).

A pair $\langle N_\tau, \boldsymbol{m}_0\rangle$, where N_τ is a TPN and $\boldsymbol{m}_0$ is the marking of N_τ at the initial time instant $\tau_0 = 0$, is called a TPN *system*. A TPN evolution is defined by a *time-TS* (TTS), namely a sequence of pairs (transition, time instant): $s_\tau = (t_{i_1}, \tau_1)(t_{i_2}, \tau_2)\ldots(t_{i_k}, \tau_k) \in (T \times \mathbb{R}_0^+)^*$, where τ_j indicates the time when t_{i_j} fires, $j = 1, \ldots, k$, and $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_k$. The enabling (firing) of s_τ at the initial marking $\boldsymbol{m}_0$ is briefly denoted $\boldsymbol{m}_0[s_\tau\rangle$ ($\boldsymbol{m}_0[s_\tau\rangle\boldsymbol{m}_k$), $\boldsymbol{m}_k$ being the final marking reached. We denote as $l(s_\tau) = t_{i_1}t_{i_2}\ldots t_{i_k}$ the logical sequence of transitions associated with s_τ, and as $\boldsymbol{\sigma} = \boldsymbol{\sigma}(l(s_\tau)) \in \mathbb{N}^{n_t}$ the corresponding FCV.

A marking $\boldsymbol{m}$ is *reachable* in $\langle N_\tau, \boldsymbol{m}_0\rangle$ if there exists a TTS s_τ such that $\boldsymbol{m}_0[s_\tau\rangle\boldsymbol{m}$. The set of all markings reachable from $\boldsymbol{m}_0$ is the *timed reachability set* $R_\tau(N_\tau, \boldsymbol{m}_0)$. Note that $R_\tau(N_\tau, \boldsymbol{m}_0) \subseteq R(N, \boldsymbol{m}_0)$, N being the untimed version of N_τ [20].

A TPN system $\langle N_\tau, \boldsymbol{m}_0\rangle$ is *bounded* if there exists $k > 0$ such that for all $\boldsymbol{m} \in R_\tau(N_\tau, \boldsymbol{m}_0)$ $\boldsymbol{m}(p) \leq k$, $\forall p \in P$.

Notice that the boundedness of $\langle N, \boldsymbol{m}_0 \rangle$ is only a sufficient condition for the boundedness of $\langle N_\tau, \boldsymbol{m}_0 \rangle$.

In the rest of this paper we make the following assumptions.

- (A1) *Single server semantics*. Regardless of the current enabling degree, a transition may only fire once at a time. Hence, each transition represents an operation that can be executed by a single operation unit (a single server).
- (A2) *Enabling memory policy* [23], [24]. A transition has no memory of any previous enabling, *i.e.* if it is re-enabled after being disabled by the firing of other transitions, the period of time during which it has been enabled before is not considered.
- (A3) For each T-invariant of the TPN, there must be at least one transition t_i in its support with $l_i > 0$. In other words, the system cannot execute idle loops in 0 time.

The *state* of a TPN system is identified by the pair $S_k = (\boldsymbol{m}_k, \boldsymbol{\gamma}_k)$, where $\boldsymbol{m}_k \in \mathbb{N}^{n_p}$ is a reachable marking, $\boldsymbol{\gamma}_k \in (\mathbb{R}_0^+)^{n_e(\boldsymbol{m}_k)}$ is a vector associating to each enabled transition under $\boldsymbol{m}_k$ the time elapsed from its enabling. Obviously, $\gamma_{k,i} \leq u_i$. Notice that the state depends on the time τ by way of $\boldsymbol{\gamma}_k$, and, as a result, the set of states is infinite. The initial state of a TPN system at the initial time instant τ_0 is denoted by $S_0 = (\boldsymbol{m}_0, \boldsymbol{\gamma}_0)$ where $\boldsymbol{m}_0$ is the initial marking and $\gamma_{0,i} = 0$, $\forall i \in I_e(\boldsymbol{m}_0)$.

Alternatively, one can represent the state of a TPN system as $S_k = (\boldsymbol{m}_k, \boldsymbol{\Phi}_k)$, where $\boldsymbol{\Phi}_k$ is a set of $n_e(\boldsymbol{m}_k)$ inequalities $l_i^k \leq \phi_i \leq u_i^k$, $\forall i \in I_e(\boldsymbol{m}_k)$. The generic inequality $l_i^k \leq \phi_i \leq u_i^k$ means that transition t_i may (must) fire at $\boldsymbol{m}_k$ only after l_i^k (before u_i^k) time units have elapsed, unless another enabled transition has fired meanwhile, disabling t_i. In particular, $l_i^k = \max\{0, l_i - \gamma_{k,i}\}$ and $u_i^k = u_i - \gamma_{k,i}$. Obviously, $l_i^k \leq l_i$ and $u_i^k \leq u_i$, the equalities holding only at the instant when t_i is enabled.

Notice, finally, that not all the enabled transitions may actually fire in a given state, since a transition t_i cannot possibly fire before t_j if $l_i^k > u_j^k$. Denoting as $T_e(S_k) \subseteq T_e(\boldsymbol{m}_k)$ the set of transitions that can be fired in S_k, one has that:

$$T_e(S_k) = \{t_i \in T_e(\boldsymbol{m}_k) \mid l_i^k \leq \min_{j \in I_e(\boldsymbol{m}_k)} (u_j^k)\}. \qquad (2)$$

D. The Modified State Class Graph

The notion of state as introduced in the previous subsection is not amenable to the construction of state graphs representing explicitly the dynamics of the system, unless a suitable state aggregation is carried out. Indeed, when a transition fires a new state $S_k = (\boldsymbol{m}_k, \boldsymbol{\Phi}_k)$ is reached, with a new marking. From then on, the mere passing of time before a new transition firing modifies the state, by eroding the time bounds of the enabled transitions ($\boldsymbol{\Phi}_k$ is changed, whereas $\boldsymbol{m}_k$ stays the same). As a result there are infinite states associated to the same marking $\boldsymbol{m}_k$, each of which specifies the FTIs for the enabled transitions relative to the *specific* time instant in which it has been reached. The MSCG [10], [11] employs the concept of class to aggregate the infinite states of the TPN system associated to the same reachable marking. More in detail, a class $C_k = (\boldsymbol{m}_k, \boldsymbol{\Theta}_k)$ is a pair composed of a reachable marking $\boldsymbol{m}_k \in R_\tau(N_\tau, \boldsymbol{m}_0)$ and a set of inequalities $\boldsymbol{\Theta}_k$ associated to the transitions enabled at $\boldsymbol{m}_k$. Differently from $\boldsymbol{\Phi}_k$, these inequalities define the firing timing constraints relatively to the *arbitrary* time instant when the system enters in the class. For example, $l_i^k \leq \theta_i \leq u_i^k$ indicates that t_i may fire only after l_i^k time units elapse from the time instant when the system enters C_k, but must fire not later than u_i^k time units after that same instant, whatever that instant might be.

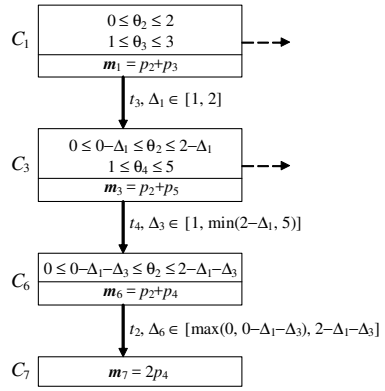

Fig. 1. MSCG example.

The MSCG is a directed graph where the nodes are the classes and the edges are associated to transitions. An outgoing edge from class $C_k = (\boldsymbol{m}_k, \boldsymbol{\Theta}_k)$ is labeled with a pair (t_i, Δ_k)[1], where $t_i \in T$ is the transition whose firing causes the exit from class C_k, and Δ_k represents the total time the system stays in C_k. Notice that Δ_k is common to *all* outgoing edges from C_k. In an edge labeled (t_i, Δ_k), Δ_k must belong to a specific FTI $I_i^k = [L_i^k, U_i^k]$, where $L_i^k = \max(0, l_i^k)$, $U_i^k = \min_{q \in I_e(\boldsymbol{m}_k)}(u_q^k)$. The notation $C_k[t_i > C_j$ is used to denote a class transition from C_k to C_j by firing t_i. Occasionally, we will denote as $\bullet C_k$ ($C_k \bullet$) the set of predecessor (successor) classes of C_k in the MSCG.

To illustrate the notation and some basic concepts regarding the MSCG, consider the portion of a MSCG depicted in Fig. 1 (taken from [10]). Each node represents a class (C_k), and contains a marking ($\boldsymbol{m}_k$) and one inequality for each enabled transition (Θ_k), expressing the allowed FTI. Each edge accounts for a class transition, as a consequence of the firing of a transition. For example, if t_3 fires in C_1 the system enters a new class, C_3. The transition firing occurs at a time Δ_1 relative to when the system entered C_1. Notice that t_3 cannot fire earlier than 1 (due to its lower bound), but must not fire later than 2, which is the upper bound for the other enabled transition t_2. In class C_3 a new transition (t_4) is enabled, while t_2 *remains* enabled. Because of this, the FTI of the latter transition is reduced by the time elapsed from its original enabling (Δ_1), and is therefore *parametric*. In general, the conditions for the firing time variables Δ_k can be parametric and nonlinear (due the presence of the min and max operators).

Thanks to the fact that the MSCG represents explicitly (through the Δ_k variables) the time passed by the system in each class, one can straightforwardly express the *duration* of

[1]In the original version of the MSCG, as introduced in [10], a label associated to t_i is also employed. This label is omitted here as we are dealing with unlabeled nets.

a sequence as the sum of the Δ_k variables associated to the classes in the corresponding path. Such duration is subject to a set of constraints on the Δ_k variables expressed on the edges of the same path. This feature of the MSCG is essential in order to address timed control specifications, which ultimately involve the duration of sequences.

III. Problem statement

A. Control specifications

Temporal control specifications can be formulated in various ways, *e.g.* by logical clauses involving markings and timing conditions. We here adopt a reachability-oriented approach, in which the specification prescribes that certain markings are to be visited in a given sequence, according to specific temporal constraints. This kind of specification is amenable to control problems that are naturally posed in a reachability-oriented perspective, as is often the case *e.g.* with manufacturing systems, robotic and logistic systems, aerospace systems.

We formalize the control specifications by means of a Generalized Timed State Sequence (GTSS), defined as an ordered list of 4-tuples of the form $(\mathscr{L}_i, \mathscr{F}_i, I_i^A, I_i^D)$, where:

- the *target marking set* $\mathscr{L}_i \subseteq R_\tau(N_\tau, \boldsymbol{m}_0)$ is a non-empty set of reachable markings of the TPN system;
- the *forbidden marking set* $\mathscr{F}_i \subseteq R_\tau(N_\tau, \boldsymbol{m}_0)$ is a (possibly empty) set of reachable markings that must be avoided in the process of reaching $\mathscr{L}_i$;
- the *absolute arrival time interval* $I_i^A = [l_i^A, u_i^A]$ defines the absolute time constraints within which $\mathscr{L}_i$ must be reached, with $l_i^A \in \mathbb{R}_0^+$, $u_i^A \in \mathbb{R}_0^+ \cup \{\infty\}$;
- the *absolute departure time interval* $I_i^D = [l_i^D, u_i^D]$ defines the absolute time constraints within which $\mathscr{L}_i$ must be left, with $l_i^D \in \mathbb{R}_0^+ \cup \{\infty\}$, $u_i^D \in \mathbb{R}_0^+ \cup \{\infty\}$.

Accordingly, a general form of an n-step GTSS is

$$g = (\mathscr{L}_0, \emptyset, \tau_0, I_0^D)(\mathscr{L}_1, \mathscr{F}_1, I_1^A, I_1^D) \dots (\mathscr{L}_n, \mathscr{F}_n, I_n^A, I_n^D) \tag{3}$$

where the initial 4-tuple indicates the initial marking ($\mathscr{L}_0 = \{\boldsymbol{m}_0\}$), the initial time ($\tau_0$), and specifies a departure time interval. For the consistency of the sequence, the timing constraints must obey the following conditions: $l_i^A \leq l_i^D$, $u_i^A \leq u_i^D$, $i = 1, \dots, n$, and $l_{i-1}^D \leq l_i^A$, $u_{i-1}^D \leq u_i^A$, $i = 1, \dots, n$. In the following, we will say that a GTSS is *time-bounded* if u_n^A is finite. It is also possible to express through a GTSS the purely logical requirement that a marking $\boldsymbol{m}_i$ is reached (at any time instant) by setting[2]: $g = (\boldsymbol{m}_0, \varnothing, \tau_0, \cdot)(\boldsymbol{m}_i, \varnothing, \cdot, \cdot)$.

A legal trajectory must be coherent with the sequence of marking sets to be reached (and avoided) according to the associated temporal constraints. To enforce a GTSS there must exist at least one TTS that results in a legal trajectory. We will denote a GTSS as *complete* if a TTS is executed such that all temporal and logical requirements of the GTSS are fulfilled.

[2]With a slight abuse of notation we denote a singleton marking set $\{\boldsymbol{m}\}$ simply as $\boldsymbol{m}$, and a single-point absolute time interval $[\tau, \tau]$ as τ. We also use the notation $I_i^A = \cdot$ ($I_i^D = \cdot$) in the absence of temporal constraints on the arrival in (departure from) a marking set, for brevity.

B. The control action

The control action does not consist merely in the full disabling of a controllable transition, but may also result in the disabling of a transition for a fraction of its allowed FTI. To express this partial disabling action, we introduce a time-varying control function, which is implementable as a restriction of the firing regions associated to the controllable transitions enabled in the current marking.

***Definition** 1:* Let $S_k = (\boldsymbol{m}_k, \boldsymbol{\Phi}_k)$ be a state of the TPN. The control action is a time-dependent function of the system state $\mathcal{F}(S_k, \tau_k) = \hat{\boldsymbol{\Phi}}_k$, where

$$\hat{\boldsymbol{\Phi}}_k = \{\hat{l}_i^k \leq \boldsymbol{\phi} \leq \hat{u}_i^k, \forall i \in I_e(S_k)\},$$

with $\hat{l}_i^k \geq l_i^k$ and $\hat{u}_i^k \leq u_i^k$ ($\hat{l}_i^k = l_i^k$ and $\hat{u}_i^k = u_i^k$ for the transitions in $T_e(S_k)$ that are not controllable). ■

This formulation encompasses both control actions proposed in [1]:

- Prohibitible events: if $\hat{l}_i^k = \hat{u}_i^k = \infty$ for a transition t_i, then the control action completely disables t_i.
- Forcible events: if $\hat{l}_i^k = \hat{u}_i^k = \delta$ for a transition t_i, then transition t_i must fire exactly at time $\tau_k + \delta$.

The net system obtained by applying the control function $\mathcal{F}$ to $\langle N_\tau, \boldsymbol{m}_0 \rangle$ is denoted by $\langle N_\tau, \boldsymbol{m}_0, \mathcal{F} \rangle$. It is obvious that $R_\tau(N_\tau, \boldsymbol{m}_0, \mathcal{F}) \subseteq R_\tau(N_\tau, \boldsymbol{m}_0)$.

C. Control synthesis overview

When the control action is computed in a state S_k, a worst-case scenario must be assumed, *i.e.* the FTIs of the enabled controllable transitions must be defined so that the GTSS can be satisfied for all possible firings of uncontrollable transitions. This ensures that the system can be made to follow a legal trajectory at all times, provided that the controllable transitions enabled in S_k are fired within the prescribed FTIs, and that subsequent firings of controllable transitions satisfy the conditions calculated at τ_k.

This implies that all possible evolutions of the system are accounted for in the computation of the control action. Accordingly, the search for all TTSs moving the system from one marking to another assigned one plays a key role and must be performed as efficiently as possible. This information can be retrieved from the MSCG of the TPN, but the computation of the full MSCG is often a demanding task (impossible, in the unbounded case). It is also unnecessary, as the part of the MSCG that is relevant to the given GTSS is generally much smaller. For this reason, we build a *partial* MSCG (denoted PMCSG in the following), that encompasses all legal TTSs. This greatly reduces the computational load associated to the generation and processing of the MSCG. The allowed FTIs of the controllable transitions are calculated based on this graph.

More in detail, a three-step procedure will be introduced to calculate the control function in a given state:

- Step 1) Structural analysis is applied (to the untimed net) to find the set Y_i^* of admissible FCVs corresponding to TSs leading from $\boldsymbol{m}_{i-1}$ to $\boldsymbol{m}_i$, for each pair $(\boldsymbol{m}_{i-1}, \boldsymbol{m}_i)$ in $\mathscr{L}_{i-1} \times \mathscr{L}_i$, $i = 1, \dots, n$.
- Step 2) The PMSCG is constructed, describing only the TTSs compatible with the admissible FCVs found

at the previous step. Paths leading to forbidden states or incompatible with the temporal specifications of the GTSS are pruned *a posteriori*.

- Step 3) The PMSCG is used to determine the necessary restrictions to the FTIs of the controllable transitions enabled in the current state, in order to guarantee the timed specifications in the worst case. To this aim, each path of the PMSCG is analyzed individually to calculate the time constraints it yields on the controllable transitions to ensure that a) the arrival and departure time interval specifications for each way point are met, and b) transitions taking the state of the TPN outside the designed PMSCG cannot fire. Then, all path-specific time constraints for a controllable transition are merged to determine its allowed FTI.

A crucial point concerns the *timing* of the control algorithm, *i.e.* when the outlined procedure should be invoked. Since the control function cannot affect the firing of uncontrollable transitions, it is only computed when the system enters a class where at least one controllable transition is enabled. While the system remains in the same class, time passes, but the constraints on the controllable transitions do not change. Therefore, there is no need to invoke the procedure again until a transition fires. While in principle it could make sense to anticipate the computation of the control function before the actual enabling of a controllable transition, this option is not pursued here, since it generally yields over-conservative conditions. Indeed, *before* an uncontrollable transition fires, one has to account for the possibility that this event will occur at any time in the corresponding FTI. However, *after* it has fired, the remaining part of its FTI can be disregarded. In view of this, the constraints considered previously in the computation of the control action can be relaxed, possibly resulting in larger FTIs for the controllable transitions that must yet be fired to fulfil the GTSS, compared to what computed previously. This justifies the recalculation of the control action after the firing of any uncontrollable transition (provided that at least one controllable transition is enabled in the resulting state).

Algorithm 1 gives the details of the online implementation, the actual computation of $\mathcal{F}(S_{curr}, \tau_{curr})$ being addressed in the following sections. The procedure requires a TPN (with its initial state and time) and a GTSS. If a solution exists, the algorithm follows the process evolution and recalculates the time firing constraints on the enabled controllable transitions at every class change up to the completion of the GTSS. These constraints are fed back to the controller, that will actually decide what transition to fire and when.

Remark 1: In the considered timed framework, multiple events can fire simultaneously at time τ_{obs} in a state $S_{prev} = (\boldsymbol{m}_{prev}, \boldsymbol{\Gamma}_{prev})$. Denoting with $obs = (T_{obs}, \tau_{obs})$ the set of simultaneous event occurrences, where T_{obs} is the set of fired transitions, the resulting system state is $S_{curr} = (\boldsymbol{m}_{curr}, \boldsymbol{\Gamma}_{curr})$, where $\boldsymbol{m}_{curr} = \boldsymbol{m}_{prev} + \boldsymbol{C}\boldsymbol{\sigma}_{obs}$, $\boldsymbol{\sigma}_{obs} = \boldsymbol{\sigma}_{obs}(T_{obs})$ denoting a firing vector having an entry equal to 1 for each transition in T_{obs}. As for $\boldsymbol{\Gamma}_{curr}$, it is important to remark that the timer of each transition enabled under the previous marking $\boldsymbol{m}_{prev}$ and fired at τ_{obs} must be reset if the transition is still enabled under the current marking $\boldsymbol{m}_{curr}$, since it has been immediately re-enabled after zero time. ■

Algorithm 1: Online control algorithm

Input: $\langle N_\tau, \boldsymbol{m}_0\rangle$, $S_0 = (\boldsymbol{m}_0, \boldsymbol{\Gamma}_0)$, τ_0, g
$S_{curr} = (\boldsymbol{m}_{curr}, \boldsymbol{\Gamma}_{curr}) := S_0$; $\tau_{curr} := \tau_0$;
forall $t_i \in T_e(\boldsymbol{m}_{curr})$ **do**
  if $t_i \in T_c$ **then**
    compute $\mathcal{F}_t(S_{curr}, \tau_{curr}) = [\hat{l}_i^{curr}, \hat{u}_i^{curr}]$;
    if *a solution does not exist* **then exit;**
  else $\hat{l}_i^{curr} = l_i^{curr}$; $\hat{u}_i^{curr} = u_i^{curr}$;
Feed back $\mathcal{F}(S_{curr}, \tau_{curr})$ to the controller;
while g *is not completed* **do**
  if $obs = (T_{obs}, \tau_{obs}) \neq \emptyset$ **then**
    // new set of events observed
    $S_{prev} = (\boldsymbol{m}_{prev}, \boldsymbol{\Gamma}_{prev}) := S_{curr}$;
    $\tau_{prev} := \tau_{curr}$;
    $\boldsymbol{\sigma}_{obs} := \boldsymbol{\sigma}(T_{obs})$;
    $\boldsymbol{m}_{curr} := \boldsymbol{m}_{prev} + \boldsymbol{C}\boldsymbol{\sigma}_{obs}$;
    $\tau_{curr} := \tau_{obs}$;
    forall $t \in T_e(\boldsymbol{m}_{curr})$ **do**
      if $t \in T_e(\boldsymbol{m}_{prev}) \setminus T_{obs}$ **then**
        $\boldsymbol{\Gamma}_{curr}(t) := \boldsymbol{\Gamma}_{prev}(t) + (\tau_{curr} - \tau_{prev})$;
      else $\boldsymbol{\Gamma}_{curr}(t) := 0$;
    $S_{curr} := (\boldsymbol{m}_{curr}, \boldsymbol{\Gamma}_{curr})$;
    forall $t_i \in T_e(\boldsymbol{m}_{curr})$ **do**
      if $t_i \in T_c$ **then**
        compute $\mathcal{F}_t(S_{curr}, \tau_{curr}) = [\hat{l}_i^{curr}, \hat{u}_i^{curr}]$;
      else $\hat{l}_i^{curr} = l_i^{curr}$; $\hat{u}_i^{curr} = u_i^{curr}$;
    Feed back $\mathcal{F}(S_{curr}, \tau_{curr})$ to the controller;

IV. Step 1: Computation the FCVs

The first step of the procedure consists in enumerating the admissible FCVs, associated to sequences leading from a marking $\boldsymbol{m}_{i-1} \in \mathscr{L}_{i-1}$ to a marking $\boldsymbol{m}_i \in \mathscr{L}_i$, for $i = 1, \ldots, n$. It is trivial to show that the following property holds:

Property 1: Let s_τ be a legal TTS according to a given GTSS g. Then, the TS $s = l(s_\tau)$ satisfies the untimed version of the reachability and safety requirements expressed by g. ■

This provides a necessary condition for the TTSs of interest, whereby the analysis of the timed system can be restricted to sequences following the mentioned FCVs. Conveniently, the latter can be found by structural analysis of the untimed net, as outlined in Section II-B.

A. Finiteness of the FCVs

The following results establish the conditions for which the number of FCVs that must be computed is finite, and provide an upper bound for the number of transitions firings they involve.

Theorem 1: Let $\langle N_\tau, \boldsymbol{m}_0\rangle$ be a TPN system abiding by assumptions A1-A3. Let also S_τ be the set of TTSs leading from $\boldsymbol{m}_0$ to a target marking $\boldsymbol{m}_t$ within a time τ_{max}. Then the set $S = \{s \mid s = l(s_\tau), \forall s_\tau \in S_\tau\}$ is finite. ■

Proof: If the set $F = \{\boldsymbol{\sigma} \mid \boldsymbol{\sigma} = \boldsymbol{\sigma}(s), \forall s \in S\}$ of FCVs corresponding to sequences in S is finite, the thesis holds. Let N be the untimed version of N_τ. Then, $F = \{\boldsymbol{\sigma} \mid \sigma(j) = y(j), j = 1, \ldots, n_t, \forall \boldsymbol{y} \in Y_1\}$, where Y_1 is the set of SCTs of the complemented net N_c. As a consequence, $||F|| = ||Y_1||$. As explained in Section II-B, any element $\boldsymbol{y} \in Y_1$ is obtained as a non-negative linear combination $\boldsymbol{y} = \boldsymbol{y}' + \boldsymbol{y}''$ of the MS T-invariants [22], where $\boldsymbol{y}' = \sum_{j_0=1}^{p_0} \alpha_{j_0} \boldsymbol{y}^{(j_0)}$ and

$$\boldsymbol{y}'' = \sum_{j_1=p_0+1}^{p_0+p_1} \alpha_{j_1} \boldsymbol{y}^{(j_1)} + \sum_{j_2=p_0+p_1+1}^{p_0+p_1+p_2} \alpha_{j_2} \boldsymbol{y}^{(j_2)} \tag{4}$$

where $\alpha_j \geq 0$, $j = 1, \ldots, p$, with $p = p_0 + p_1 + p_2$, $Y^{\text{ms}} = \{\boldsymbol{y}^{(j)}, j = 1, \ldots, p\}$ is the (finite) set of MS T-invariants of N_c, and $\boldsymbol{y}^{(j)} \in Y_0$, for $j = 1, \ldots, p_0$, $\boldsymbol{y}^{(j)} \in Y_1$, for $j = p_0 + 1, \ldots, p_0 + p_1$, $\boldsymbol{y}^{(j)} \in Y_2$, for $j = p_0 + p_1 + 1, \ldots, p$. The coefficients in (4) must also satisfy the condition $\sum_{j=1}^{p} \alpha_j y_{n_t+1}^{(j)} = 1$, which implies that $\alpha_j \leq 1/y_{n_t+1}^{(j)}$, $j = p_0 + 1, \ldots, p$. As a consequence $\boldsymbol{y}''$ is bounded. A bound $\bar{\boldsymbol{y}}''$ can be computed as:

$$\bar{y}_t'' = \max_{j=p_0+1,\ldots,p} \frac{y_t^{(j)}}{y_{n_t+1}^{(j)}}, \quad t = 1, \ldots, n_t \tag{5}$$

$$\bar{y}_{n_t+1}'' = 1 \tag{6}$$

On the other hand, the coefficients α_j, $j = 1, \ldots, p_0$ *can* exceed 1. This implies that there exist virtually infinite non-minimal SCTs, since the coefficients α_j associated to NCTs are not bounded. Consider, *e.g.* the $j^\star$th NCT, with $1 \leq j^\star \leq p_0$, and assume that the marking in the net is sufficient to enable once the execution of $\boldsymbol{y}^{(j^\star)}$. Since, by Assumption A3 the TPN does not admit T-invariants with all transitions fireable in zero time, each sequence involving the complete firing of $\boldsymbol{y}^{(j^\star)}$ requires a minimum time[3], say $\bar{\tau}_{j^\star}$. Similarly, each sequence involving the complete firing of $h \cdot \boldsymbol{y}^{(j^\star)}$, with $h \in \mathbb{N}$, will require at least $h \cdot \bar{\tau}_{j^\star}$ time units. Then, $\alpha_{j^\star} \leq \tau_{max}/\bar{\tau}_{j^\star}$. If the net marking allows a k-enabling of the NCT, the bound is correspondingly increased by a factor k. Now, since all α_j, $j = 1, \ldots, p_0$ are bounded, $\boldsymbol{y}'$ is bounded as well. A bound on $\boldsymbol{y}'$ can be calculated as

$$\bar{\boldsymbol{y}}' = \sum_{j_0=1}^{p_0} k_j \frac{\tau_{max}}{\bar{\tau}_j} \boldsymbol{y}^{(j_0)}, \tag{7}$$

where k_j is the enabling degree of the jth NCT.

Finally, since $\boldsymbol{y}$ is bounded, there exist only a finite number of vectors in F, and the thesis follows. ■

***Theorem** 2:* Let g be a GTSS, and $\langle N_\tau, \boldsymbol{m}_0 \rangle$ a TPN system abiding by assumptions A1-A3. Let also S_τ be the set of legal TTSs according to the GTSS. Then, if g is time-bounded, the set $S = \{s \mid s = l(s_\tau), \forall s_\tau \in S_\tau\}$ is finite. ■

Proof: The boundedness of g implies that u_i^A is bounded for $i = 1, \ldots, n$. Then, by the previous Thm. 1, the number of sequences taking the net from a marking $\boldsymbol{m}_{i-1} \in \mathscr{L}_{i-1}$ to a marking $\boldsymbol{m}_i \in \mathscr{L}_i$, $i = 1, \ldots, n$, is finite. ■

[3]A conservative value for this bound can be obtained by taking the maximum of the lower bounds associated to the transitions in the support of the T-invariant. Depending on the structure of the net, less conservative bounds may be defined.

In view of Thm. 1, the maximum number of transition firings required to get from a marking $\boldsymbol{m}_{i-1} \in \mathscr{L}_{i-1}$ to a marking $\boldsymbol{m}_i \in \mathscr{L}_i$ is bounded by $K = ||\bar{\boldsymbol{y}}'||_1 + ||\bar{\boldsymbol{y}}''||_1 - 1$, where $\bar{\boldsymbol{y}}'$ is calculated with $\tau_{max} = u_i^A$. Then, one can apply an efficient enumeration scheme to find all the relevant FCVs, based on the following two features: i) an ILP formulation to characterize the admissible FCVs of maximum length K, and ii) a Branch & Bound (B&B) procedure that partitions the solution space once a solution is found, introducing suitable constraints to exclude previously found solutions. Notice that, differently from the approach of [25], where the existence of fireable TSs compatible with the calculated FCVs is verified *a posteriori*, here the admissibility check is integrated in the enumeration procedure, to avoid carrying over useless FCVs to the –more costly– timed analysis.

B. An ILP formulation to characterize admissible FCVs

An admissible FCV $\boldsymbol{\sigma}$ is necessarily associated to an SCT of the complemented net $\boldsymbol{y} \in Y_1$, by way of the relation $\boldsymbol{y} = [\boldsymbol{\sigma}\ 1]^T$. Now, consider a generic TS $(t_{j_1}, \ldots, t_{j_K})$ of length K, where $j_1, \ldots, j_K \in \{1, \ldots, n_t\}$. The corresponding FCV can be written as $\boldsymbol{\sigma} = \sum_{k=1}^{K} \boldsymbol{\epsilon}^{(j_k)}$. Expressing $\boldsymbol{\sigma}$ as a sum of versors automatically ensures the integrality of vector $\boldsymbol{y}$. Furthermore, it can be used to enforce the fireability of the corresponding sequence by setting the following constraints:

$$\boldsymbol{m}_0 + \boldsymbol{C} \sum_{k=1}^{i-1} \boldsymbol{\epsilon}^{(j_k)} \geq \boldsymbol{Pre}\ \boldsymbol{\epsilon}^{(j_i)}, \quad i = 1, \ldots, K. \tag{8}$$

Assembling the previously developed expressions we can obtain an ILP formulation characterizing the admissible FCVs involving not more than K transition firings solving the reachability problem from $\boldsymbol{m}_0$ to $\boldsymbol{m}_t$:

$$\text{minimize} \sum_{k=1}^{K} \sum_{j=1}^{n_t} e_j^{(k)}$$

subject to

$$\boldsymbol{m}_t = \boldsymbol{m}_0 + \boldsymbol{C}\boldsymbol{\sigma}$$

$$\boldsymbol{\sigma} = \sum_{k=1}^{K} \boldsymbol{e}^{(k)}$$

$$\begin{bmatrix} \boldsymbol{\sigma} \\ 1 \end{bmatrix} = \sum_{j=1}^{p} \alpha_j \boldsymbol{y}^{(j)}$$

$$\boldsymbol{m}_0 + \boldsymbol{C} \sum_{k=1}^{i-1} \boldsymbol{e}^{(k)} \geq \boldsymbol{Pre}\ \boldsymbol{e}^{(i)} \qquad i = 1, \ldots, K$$

$$\sum_{j=1}^{n_t} e_j^{(k+1)} \leq \sum_{j=1}^{n_t} e_j^{(k)} \leq 1 \qquad k = 1, \ldots, K-1$$

$$e_j^{(k)} \in \{0, 1\} \qquad j = 1, \ldots, n_t,\ k = 1, \ldots, K$$

$$\alpha_j \geq 0 \qquad j = 1, \ldots, p$$

In the previous formulation, the sequence $\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(K)}$ identifies the sequence of transition firings ($\boldsymbol{e}^{(k)}$ corresponds to the kth firing in the sequence). To allow for sequences shorter than K, some of these vectors can be null (no transition

fired), while those corresponding to actual transition firings are versors. To avoid having multiple equivalent solutions (corresponding to the same firing sequence), null vectors are forced to be at the end of the sequence. The cost function ensures that the solution $\boldsymbol{\sigma}$ will be the admissible FCV with less transition firings among all the solutions. In the sequel, we will denote the previous ILP formulation as $\Pi_0 = \Pi(f, \mathcal{C})$, f denoting the cost function and $\mathcal{C}$ the set of constraints. The set of vectors $\boldsymbol{y} = [\boldsymbol{\sigma}^T \; 1]^T$ satisfying $\mathcal{C}$ equals $Y^\star = Y_1^a \cap \{\boldsymbol{y} \in Y \mid \sum_{j=1}^{n_t} y_j \leq K\}$.

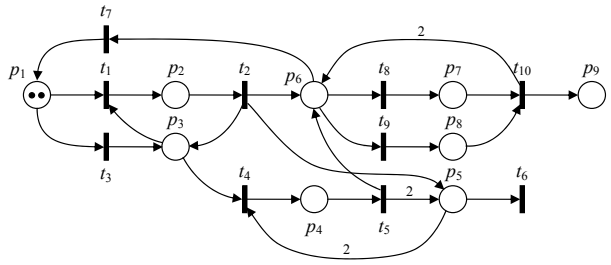

Fig. 2. PN of Example 1.

Consider *e.g.* the PN system depicted in Figure 2 [25], for which we want to solve the reachability problem from $\boldsymbol{m}_0 = 2p_1$ to $\boldsymbol{m}_t = 2p_1 + p_9$ with $K = 25$. The set of MS T-invariants of the complemented net is given by $Y^{\text{ms}} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3\}$, where $\boldsymbol{y}_1 = [0\,0\,1\,1\,1\,0\,1\,0\,0\,0\,0]^T$, $\boldsymbol{y}_2 = [1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0]^T$, and $\boldsymbol{y}_3 = [0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1]^T$. The first two are NCTs (their last element is 0) and correspond to the T-invariants of the original PN. The last one is a SCT, corresponding to a (non-admissible) FCV $\boldsymbol{\sigma}_3 = [0\,0\,0\,0\,0\,0\,0\,1\,1\,1]^T$ satisfying the required marking variation. Solving the resulting Π_0 yields the non-MS SCT $\boldsymbol{y}_4 = \boldsymbol{y}_1 + 2\boldsymbol{y}_2 + \boldsymbol{y}_3$. Indeed, the shortest TS achieving the wanted result contains 15 transition firings, *e.g.* $(t_3\,t_1\,t_2\,t_7\,t_1\,t_2\,t_4\,t_9\,t_5\,t_8\,t_{10}\,t_7\,t_7\,t_6\,t_6)$. In other words, to reach $\boldsymbol{m}_t$ one has to follow $\boldsymbol{\sigma}_3$, but in order to enable the sequence the necessary tokens must be *borrowed* through suitable executions of the T-invariants of the original PN [25].

C. The branching scheme

Using the previous ILP formulation as a cornerstone we can employ a B&B method to enumerate the elements of $Y^\star$. The B&B method operates by partitioning the solution space into smaller regions that exclude all previously found solutions.

***Property** 2:* Assume that $Y_1^a \neq \varnothing$ and let $\overline{Y}_1^a \subseteq Y_1^a$ be a generic non-empty subset. Let also $\boldsymbol{y}^\star \in \overline{Y}_1^a$ be such that there does not exist $\boldsymbol{y} \in \overline{Y}_1^a \setminus \{\boldsymbol{y}^\star\}$ such that $||\boldsymbol{y}||_1 < ||\boldsymbol{y}^\star||_1$. Then $\overline{Y}_1^a$ can be partitioned as:

$$\overline{Y}_1^a = \{\boldsymbol{y}^\star\} \cup \overline{Y}_1^{a,1} \cup \ldots \cup \overline{Y}_1^{a,n_t} \cup \overline{Y}_1^{a,n_t+1}$$

where

- $\overline{Y}_1^{a,j} = \{\boldsymbol{y} \in \overline{Y}_1^a \mid (y_l \geq y_l^\star, l = 1, \ldots, j-1) \wedge (y_j < y_j^\star)\}$, $j = 1, \ldots, n_t$.
- $\overline{Y}_1^{a,n_t+1} = \{\boldsymbol{y} \in \overline{Y}_1^a \mid (y_j \geq y_j^\star, j = 1, \ldots, n_t) \wedge (\sum_{j=1}^{n_t} y_j > \sum_{j=1}^{n_t} y_j^\star)\}$. ■

Notice that:

- $\overline{Y}_1^{a,j} \cap \overline{Y}_1^{a,l} = \varnothing$, $j, l = 1, \ldots, n_t + 1$, $j \neq l$.
- None of the sets $\overline{Y}_1^{a,j}$, $j = 1, \ldots, n_t + 1$ includes $\boldsymbol{y}^\star$ (since either one element of $\boldsymbol{y}$ or the sum of its elements are forced to be different) or a solution with a smaller sum of elements (by assumption).
- Only $\overline{Y}_1^{a,n_t+1}$ can include solutions that are element-wise greater than or equal to $\boldsymbol{y}^\star$ (non-minimal solutions).

In view of the previous partition, if a solution to the ILP problem $\Pi_0 = \Pi(f, \mathcal{C})$ is found, corresponding to a firing vector $\boldsymbol{\sigma}^\star$, further solutions can be sought for by addressing the following modified versions of the same problem:
$\Pi_1 = \Pi(f, \mathcal{C} \wedge (\sigma_1 < \sigma_1^\star))$,
$\Pi_2 = \Pi(f, \mathcal{C} \wedge (\sigma_1 \geq \sigma_1^\star) \wedge (\sigma_2 < \sigma_2^\star))$,
...
$\Pi_{n_t} = \Pi(f, \mathcal{C} \wedge (\sigma_j \geq \sigma_j^\star, j = 1, \ldots, n_t - 1) \wedge (\sigma_{n_t} < \sigma_{n_t}^\star))$.
$\Pi_{n_t+1} = \Pi(f, \mathcal{C} \wedge (\sigma_j \geq \sigma_j^\star, j = 1, \ldots, n_t) \wedge (\sum_{j=1}^{n_t} \sigma_j > \sum_{j=1}^{n_t} \sigma_j^\star))$.

To complete the enumeration, the partitioning procedure of Property 2 is applied again for each of the generated sub-problems that admit a solution, thereby configuring a tree of sub-problems stemming from Π_0, each one associated to a specific subset of Y_1^a. Notice that all nodes of the tree that include a constraint of the type $\sigma_j < \sigma_j^\star$ can be immediately classified as infeasible if $\sigma_j^\star = 0$. When all leaves of the branching tree correspond to infeasible sub-problems, the enumeration is complete. The enumeration procedure is summarized in Algorithm 2[4].

Algorithm 2: Algorithm EAFCV: Enumeration of admissible FCVs from $\boldsymbol{m}_0$ to $\boldsymbol{m}_t$ with at most K firings.

Input: N, $\boldsymbol{m}_0$, $\boldsymbol{m}_t$, K
Output: $Y^\star$
Define Π_0;
$\Lambda := (\Pi_0)$; `// Init. list of open problems`
$Y^\star := \varnothing$; `// Init. solution set`
while $\Lambda \neq ()$ **do**
  $\Pi = (f, \mathcal{C}) := \text{pop}(\Lambda)$;
  $\boldsymbol{y}^\star = [\boldsymbol{\sigma}^{\star T} \; 1]^T := \text{solve}(\Pi)$;
  if $\boldsymbol{y}^\star \neq$ *null* **then** `// A solution exists`
    $Y^\star := Y^\star \cup \{\boldsymbol{y}^\star\}$;
    for $j := 1$ **to** n_t **do** `// Branching`
      $\mathcal{C}' := \mathcal{C} \wedge (\sigma_k \geq \sigma_k^\star, k = 1, \ldots, j-1) \wedge (\sigma_j < \sigma_j^\star)$;
      $\Lambda := (\Lambda, (f, \mathcal{C}'))$;
    $\mathcal{C}' := \mathcal{C} \wedge (\sigma_j \geq \sigma_j^\star, j = 1, \ldots, n_t) \wedge (\sum_{j=1}^{n_t} \sigma_j > \sum_{j=1}^{n_t} \sigma_j^\star)$;
    $\Lambda := (\Lambda, (f, \mathcal{C}'))$;
return $Y^\star$;

Algorithm 3 uses the EAFCV procedure to calculate the admissible firing vectors for a given GTSS.

V. Step 2: Computation of the PMSCG

The second step of the procedure outlined in Section III-C consists in the computation of the PMSCG representing only

[4]A list Λ is an ordered set of elements $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$. The `pop` function extracts the first element of a list, reducing it to $\Lambda = (\lambda_2, \ldots, \lambda_k)$. Two lists $\Lambda = (\lambda_1, \ldots, \lambda_k)$ and $M = (\mu_1, \ldots, \mu_j)$ can be appended as follows: $(\Lambda, M) = (\lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_j)$. Finally, an empty list is denoted as $()$.

Algorithm 3: Algorithm FCVS: Calculation of FCV sets.

Input: N, $\boldsymbol{m}_0$, K, g
Output: $Y_i^\star$, $i = 1, \ldots, n$
for $i := 1$ **to** n **do**
  $Y_i^\star := \emptyset$;
  forall $\boldsymbol{m}_{ini} \in \mathscr{L}_{i-1}$ **do**
    forall $\boldsymbol{m}_{end} \in \mathscr{L}_i$ **do**
      $Y_{i,add}^\star := EAFCV(N, \boldsymbol{m}_{ini}, \boldsymbol{m}_{end}, K)$;
      $Y_i^\star := Y_i^\star \cup Y_{i,add}^\star$;
  if $Y_i^\star = \emptyset$ **then exit**;
  // g not realizable
return $Y_i^\star$, $i = 1, \ldots, n$;

Algorithm 4: Algorithm PMSCG: Computation of the PMSCG

Input: $\langle N_\tau, \boldsymbol{m}_0 \rangle$, $S_0 = (\boldsymbol{m}_0, \boldsymbol{\Gamma}_0)$, g, $Y_i^\star$, $i = 1, \ldots, n$
Output: G, $\mathscr{C}_{unsafe}$
$\Theta_0 := \{\max\{0, l_i - \boldsymbol{\Gamma}_0(t_i)\} \leq \phi_i \leq u_i - \boldsymbol{\Gamma}_0(t_i), \forall i \in I_e(\boldsymbol{m}_0)\}$;
$C_0 := (\boldsymbol{m}_0, \Theta_0)$; // initial class
$\mathscr{C} := \{C_0\}$; // classes (nodes of the MSCG)
$\mathscr{A} := \emptyset$; // arcs (edges of the MSCG)
$\Lambda_F := \emptyset$; // list of forbidden classes
$\mathscr{C}_0 := \{C_0\}$, $i := 0$;
// Graph generation
while $\mathscr{C}_i \neq \emptyset \wedge i < n$ **do**
  $i := i + 1$;
  $\mathscr{C}_i := \emptyset$; // terminal classes ith step
  forall $C = (\boldsymbol{m}, \Theta) \in \mathscr{C}_{i-1}$ **do**
    forall $s_i = t_{i_1} \ldots t_{i_K}$ *s.t.* $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_i(s_i) \in Y_i^\star$ **do**
      if $\boldsymbol{m}[s_i\rangle$ **then**
        $C_{j_0} := C$;
        for $k := 1$ **to** K **do**
          Compute (following the MSCG rules) the class $C_{j_k} = (\boldsymbol{m}_{j_k}, \Theta_{j_k})$ "reachable" by firing t_{i_k} from $C_{j_{k-1}}$;
          $\mathscr{C} := \mathscr{C} \cup \{C_{j_k}\}$;
          $\mathscr{A} := \mathscr{A} \cup \{(C_{j_{k-1}}, C_{j_k})\}$;
          if $\boldsymbol{m}_{j_k} \in \mathscr{F}_i$ **then**
            $\Lambda_F := (\Lambda_F, C_{j_k})$; **break**;
        // full legal sequence
        $\mathscr{C}_i := \mathscr{C}_i \cup \{C_{j_K}\}$;
  // Test of temporal conditions
forall $\pi = C_{j_1} \cdots C_{j_{n_n+1}} \in G$ **do**
  if *system* (11) *unfeasible* **then** $\Lambda_F := (\Lambda_F, C_{j_{n_n+1}})$;
$G := (\mathscr{C}, \mathscr{A})$;
$(G, \mathscr{C}_{unsafe}) = \text{prune}(G, \Lambda_F)$;
return G, $\mathscr{C}_{unsafe}$;

the legal trajectories of the system according to the GTSS. Algorithm 4 summarizes the procedure.

The generation phase explores all and only the TTSs compatible with the admissible FCVs found in Step 1 for each source-target marking pair envisaged by g, and collected in the sets $Y_i^\star$, $i = 1, \ldots, n$. All these TTSs have the structure $s_\tau = s_{\tau,1}\, s_{\tau,2}\, \ldots\, s_{\tau,n}$, where $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_i(l(s_{\tau,i})) \in Y_i^\star$, $i = 1, \ldots, n$. This automatically ensures that all included paths will pass through $\mathscr{L}_1, \ldots, \mathscr{L}_n$. The expansion of a TTS is stopped if at the ith step of the sequence a forbidden marking belonging to $\mathscr{F}_i$ is encountered. The corresponding class is marked as forbidden. Each full path is also tested for the admissibility of the temporal constraints of the GTSS and its terminal class marked as forbidden in case of infeasibility. When all possible paths have been computed, the PMSCG is pruned from all loose ends (see Algorithm 5), by iteratively removing the forbidden classes and their exclusive predecessors. Finally, to facilitate the final step of the procedure, Algorithm 4 tags as "unsafe" any class in the PMSCG enabling a transition that moves the system outside the graph.

***Remark** 2:* Notice that each FCV can be associated to many fireable sequences. However, the computation of a sequence from a FCV can be efficiently performed (and consequently Algorithm 4 can be significantly accelerated) starting from a prefix of length one and adding a new transition at a time or discarding the whole prefix if one of the following conditions holds: a) the transition results not (logically or temporally) enabled, b) a class associated to a forbidden marking is reached, c) the temporal conditions are violated. ■

While most of Algorithm 4 is straightforward, it remains to explain how to check the existence of an admissible firing schedule for each path of the MSCG. For this purpose we will formulate an ILP problem that models all the necessary timing constraints. A generic MSCG path compatible with a GTSS g of type (3) is the concatenation of n sub-paths:

$$\pi = \pi_1 \ldots \pi_n \tag{9}$$

where π_i accounts for the ith step of g. Precisely,

$$\begin{aligned} &\pi_1 = C_{j_1} \to C_{j_2} \to \cdots \to C_{j_{n_1+1}} \\ &\pi_2 = C_{j_{n_1+1}} \to C_{j_{n_1+2}} \to \cdots \to C_{j_{n_2+1}} \\ &\ldots \\ &\pi_n = C_{j_{n_{n-1}+1}} \to \cdots \to C_{j_{n_n+1}} \end{aligned} \tag{10}$$

where C_{j_1} is the unique class in the path associated to $\boldsymbol{m}_0 \in \mathscr{L}_0$, and $C_{j_{n_i+1}}$ is the unique class in the path associated to $\boldsymbol{m}_i \in \mathscr{L}_i$, with $i = 1, \ldots, n$. The generic arc from C_{j_k} to $C_{j_{k+1}}$ is endowed with the transition firing in the former class as well as the allowed FTI: $C_{j_k} \xrightarrow{t_{i_k}, \Delta_{j_k} \in I_{i_k}^{j_k}} C_{j_{k+1}}$. Parameter n_i denotes the number of transition firings from C_{j_1} to the end of π_i: denoting with K_i the number of transition firings of the ith sub-path, it holds that $n_1 = K_1$, $n_2 = n_1 + K_2$, …, $n_n = n_{n-1} + K_n$.

The existence of legal FTIs for each transition in π can be ascertained by checking the feasibility of the following set of

constraints for the transition firing times $\Delta_{j_1}, \ldots, \Delta_{j_{n_n}}$:

$$\begin{cases} l_i^A \le \sum_{k=1}^{n_i} \Delta_{j_k} \le u_i^A, \quad i = 1, \ldots, n & (a_i) \\ l_0^D \le \Delta_{j_1} \le u_0^D & (b_0) \\ l_i^D \le \sum_{k=1}^{n_i} \Delta_{j_k} + \Delta_{j_{n_i+1}} \le u_i^D, \quad i = 1, \ldots, n-1 & (b_i) \\ \sum_{k=1}^{n_n} \Delta_{j_k} + \bar{u}_{n_n} \ge l_n^D & (b_{n,1}) \\ \sum_{k=1}^{n_n} \Delta_{j_k} + \bar{l}_{n_n} \le u_n^D & (b_{n,2}) \\ L_{i_k}^{j_k} \le \Delta_{j_k} \le U_{i_k}^{j_k}, \quad k = 1, \ldots, n_n & (c) \end{cases} \tag{11}$$

where $\bar{u}_{n_n} = \min_{q \in I_e(\boldsymbol{m}_{j_{n_n+1}})} u_q^{j_{n_n+1}}$ and $\bar{l}_{n_n} = \min_{q \in I_e(\boldsymbol{m}_{j_{n_n+1}})} l_q^{j_{n_n+1}}$, $\boldsymbol{m}_{j_{n_n+1}}$ being the marking in class $C_{j_{n_n+1}}$. Variables $\bar{u}_{n_n}$ and $\bar{l}_{n_n}$ account for the worst case time limitations for the firing of the enabled transitions in the terminal class of the path.

Algorithm 5: Algorithm prune: Pruning of the PMSCG

```
Input: G, Λ_F
Output: G, C_unsafe
(C, A) := G;
C_unsafe := ∅;
while Λ_F ≠ ∅ do
    C := pop(Λ_F);
    forall C_pre ∈ •C do
        if C_pre• ⊆ C_F then Λ_F := (Λ_F, C_pre);
        else C_unsafe := C_unsafe ∪ {C_pre};
    // remove C from PMSCG
    C := C \ {C};
    forall a = (C', C") ∈ A | (C' = C) ∨ (C" = C) do
        A := A \ {a};
G := (C, A);
return G, C_unsafe;
```

Constraints (a_i) impose the necessary limitations on the length of each step of the sequence, to abide by the arrival time requirements. Indeed, $\sum_{k=1}^{n_i} \Delta_{j_k}$ is equal to the time spent to reach class $C_{j_{n_i+1}}$. Such time must be inside the allowed interval $I_i^A = [l_i^A, u_i^A]$.

Constraints (b_i) formalize the departure time requirements of each step of the sequence, including those for the starting class of the path, C_{j_1} (constraint (b_0)). The last class $C_{j_{n_i+1}}$ of the ith step, with $i = 1, \ldots, n-1$, must be left within the allowed time interval $I_i^D = [l_i^D, u_i^D]$. Notice that $\Delta_{j_{n_i+1}}$ represents the time spent in $C_{j_{n_i+1}}$. The last of such constraints (*i.e.*, (b_n)) is slightly different, as the actual transition firing in $C_{j_{n_n+1}}$ is not specified by g. Parameters $\bar{l}_{n_n}$ and $\bar{u}_{n_n}$ are used to express the most stringent timing conditions associated to the firing of the enabled transitions in $C_{j_{n_n+1}}$. Indeed, if $(b_{n,1})$ is violated, any enabled transition that fires will be forced to do so before l_n^D, so that $C_{j_{n_n+1}}$ will be left too early. Similarly, if $(b_{n,2})$ is violated, no enabled transition can fire earlier than u_n^D, and the class will be left too late. Finally, constraint (c) ensures that the variables Δ_{j_k}, $k = 1, \ldots, n_n$ stay inside the time intervals given in the MSCG.

Notice that constraints (b) and (c) in system (11) are nonlinear, by way of $\bar{l}_{n_n}$, $\bar{u}_{n_n}$, $L_{i_k}^{j_k}$ and $U_{i_k}^{j_k}$, which are defined using the min or max operator. They can be linearized using the following rules (similar rules apply for the max operator):

Rule 1. $\min(a_1, a_2, \ldots, a_q) \ge b$ is equivalent to the set of q constraints $a_i \ge b$, $i = 1, \ldots, q$. ■

Rule 2. $\min(a_1, a_2, \ldots, a_q) \le b$ is equivalent to the following system of conditions:

$$\begin{cases} w \le b \\ a_i - (1 - y_i)M \le w \le a_i, \quad i = 1, \ldots, q \\ \sum_{i=1}^{q} y_i \ge 1 \\ y_i \in \{0, 1\}, \quad i = 1, \ldots, q \end{cases}$$

where w is an auxiliary (real) variable, $y_1, \ldots, y_q$ are auxiliary binary variables, and M is a sufficiently large number. ■

VI. Step 3: Computation of the control function

The PMSCG computed with Algorithm 4 includes all the paths for which there exists at least a transition firing schedule that configures a legal TTS, according to the given GTSS g. A control solution, if any exists, has to follow one of the paths of this PMSCG. We denote as $\Pi(C)$ the set of such paths stemming from the class C corresponding to the current state S (where one or more controllable transitions are enabled). Notice that the admissibility of a path with respect to the timing constraints does not guarantee their actual obtainment, because uncontrollable transitions cannot be forced to fire at specific times. In the third step of the procedure we verify if there exists a feasible restriction of the FTIs of the controllable transitions that guarantees that the temporal specifications of the GTSS are obtained in the worst case (*i.e.*, for all possible firing time schedules of the uncontrollable transitions).

Furthermore, the control function must also ensure that no firing of a transition is allowed in an unsafe class if it causes the system to exit from the PMSCG (which accounts for all paths that can fulfil the GTSS). We will characterize as "unsafe" such transitions. A controllable unsafe transition can be delayed from firing by increasing its lower bound up to the value of its upper bound. If this modification makes the lower bound of the unsafe transition greater than the upper bound of the safe transitions in the same class, then the firing of the undesired transition is prevented. Conversely, if the unsafe transition is uncontrollable, its firing can be delayed only if its lower bound is parametrically dependent on the firing times of any controllable transitions along the path, as occurs if the transition has been enabled in a previous class.

Notice that, in the absence of controllable transitions along a path, one can only perform a legitimacy check on the set of the corresponding TTSs, but can take no corrective action. Accordingly, in the sequel we will assume that each path includes at least one controllable transition.

The computation of the control function is based on the analysis of all the paths in the set $\Pi(C)$. Any path $\pi \in \Pi(C)$

has the structure (9), where C_{j_1} equals the current class C. Each path imposes different restrictions on the FTIs of the controllable transitions, to guarantee that all the specifications are met in the worst case (*i.e.* for any possible firing time schedule of the uncontrollable transitions). All these restrictions must be taken into account by the control function. Indeed, the control function provides for each controllable transition the largest FTI that is compatible with the full set of restrictions.

Consider a generic path $\pi \in \Pi(C)$ not including unsafe classes, and assume that it contains one controllable transition, say t_{i_1} without loss of generality. The calculation of its allowed FTI $\mathcal{F}^{\pi}_{t_{i_1}}(S,\tau) = [L^{i_1}_{\pi}, U^{i_1}_{\pi}]$ with respect to π can be carried out by solving two ILPs, with the following structure:

$$\begin{aligned} & L^{i_1}_{\pi} = \min \Delta_{j_1} \quad [U^{i_1}_{\pi} = \max \Delta_{j_1}] \\ & \text{subject to:} \\ & C(\pi) \\ & \Delta_{j_k} = L^{j_k}_{i_k}\, [\Delta_{j_k} = U^{j_k}_{i_k}], \quad \forall i_k \text{ s.t. } t_{i_k} \in T_u \end{aligned} \tag{12}$$

where $C(\pi)$ represents the set of constraints (11). Notice that in problems (12) the timing variables associated to uncontrollable transitions are fixed (to the worst case), while those associated to other controllable transitions besides t_{i_1} are free. Indeed, in a worst case scenario the minimum (maximum) firing time for the controllable transition is limited by the minimum (maximum) firing time of all uncontrollable transitions.

Assume now that path π includes an unsafe class C_{j_k} enabling an unsafe transition t_{i_f}. If t_{i_f} is uncontrollable, the following constraint must be added to problem (12) to prevent its firing in C_{j_k}:

$$l^{j_k}_{i_f} > u^{j_k}_{i_k}. \tag{13}$$

Constraint (13) implies that the lower bound of t_{i_f} is strictly greater than the upper bound of t_{i_k}, *i.e.* the transition that should fire in C_{j_k} according to the considered path. This guarantees that t_{i_k} will fire, and t_{i_f} will not. Conversely, if t_{i_f} is controllable, the constraint takes the form:

$$u^{j_k}_{i_f} > u^{j_k}_{i_k}, \tag{14}$$

since the controllability of t_{i_f} allows us to defer its firing at most until its upper bound.

Clearly, the control design problem for path π has a solution only if the resulting FTI is not empty, *i.e.* if $L^{i_1}_{\pi} \leq U^{i_1}_{\pi}$. In the negative case, there is no feasible FTI of the controllable transitions that meets g, and π must be further pruned from the PMSCG using Algorithm 5 and removed from $\Pi(C)$ (notice that this also implies that a further constraint of type (13) or (14) must be introduced to account for the additional unsafe class). Once all unfeasible paths have been removed, then the overall control function for t is simply obtained as $\mathcal{F}_t(S,\tau) = \bigcap_{\pi\in\Pi(C)} \mathcal{F}^{\pi}_t(S,\tau)$. Then, either the intersection of the obtained FTIs is non-empty, in which case the overall control design problem has a solution and $\Pi(C)$ contains only paths satisfying the given GTSS, or the intersection is empty and there is no solution.

Finally, if more than one controllable transitions are enabled in the current class, one can calculate the control function for each of them, and then leave to the controller the decision regarding which transition to fire and when.

***Definition** 2:* Let $\mathcal{F}^1_t(S,\tau)$ and $\mathcal{F}^2_t(S,\tau)$ be two control functions for the transition t at current state S. Then, $\mathcal{F}^1_t(S,\tau)$ is more permissive than $\mathcal{F}^2_t(S,\tau)$, if $\mathcal{F}^1_t(S,\tau) \supseteq \mathcal{F}^2_t(S,\tau)$. ■

***Theorem** 3:* Given a TPN system $\langle N_\tau, \boldsymbol{m}_0\rangle$ and a GTSS g, let $\mathcal{F}_t(S,\tau)$ be the non-empty control function for the transition t at the current state S, computed as $\mathcal{F}_t(S,\tau) = \bigcap_{\pi\in\Pi(C)} \mathcal{F}^{\pi}_t(S,\tau)$, where C is the class associated to S and $\mathcal{F}^{\pi}_t(S,\tau) = [L^t_\pi, U^t_\pi]$, L^t_π and U^t_π being obtained from (12). Then, it holds that

i) $\langle N_\tau, \boldsymbol{m}_0, \mathcal{F}_t\rangle$ satisfies g;
ii) $\mathcal{F}_t(S,\tau)$ is maximally permissive. ■

Proof: i) By construction, each path in the PMSCG is associated to a fireable sequence passing by each way point and leading the system to a target marking. It follows that if $\mathcal{F}^{\pi}_t(S,\tau) = [L^t_\pi, U^t_\pi]$ and L^t_π and U^t_π are obtained from (12), it is guaranteed that the controlled system satisfies g when moving along the path π. Now, $\mathcal{F}_t(S,\tau)$ is obtained as the intersection of the FTIs $\mathcal{F}^{\pi}_t(S,\tau)$, $\forall\pi \in \Pi(C)$. Consequently, $\mathcal{F}_t(S,\tau) \subseteq \mathcal{F}^{\pi}_t(S,\tau)$, $\forall\pi \in \Pi(C)$, which implies that the permissivity along each path is further restricted, so that the satisfaction of g is all the more guaranteed.

ii) Assume, *ad absurdum*, that there exists a control function $\mathcal{F}'_t(S,\tau)$ that is more permissive than $\mathcal{F}_t(S,\tau)$. Then, the net system $\langle N_\tau, \boldsymbol{m}_0, \mathcal{F}'_t\rangle$ obtained by applying this control function must admit at least a TTS satisfying g that cannot be generated by $\langle N_\tau, \boldsymbol{m}_0, \mathcal{F}_t\rangle$. Denote this TTS as $s_\tau = (t_{i_1},\tau_1)(t_{i_2},\tau_2)\dots(t_{i_k},\tau_k)$, and let $S_1\, S_2\, \dots\, S_k$ be the corresponding sequence of states reached by the system at each transition firing. By construction, S_i, $i = 1,\dots,k$, is associated to a class of the MSCG of the given net system, but also to a class C_{j_i}, $j = 1,\dots,k$ of the PMSCG, since s_τ follows a path that satisfies g (for some schedule of the transition firings). Moreover, by firing s_τ the PMSCG would move along the path $\pi = C\, C_{j_1}\, C_{j_2}\, \dots\, C_{j_k}$.

Then, it must be possible to move along π for *any* possible firing time schedule of the uncontrollable transitions without violating g, and this would imply that the $\mathcal{F}'^{\pi}_t(S,\tau) = [L'^t_\pi, U'^t_\pi] \supseteq [L^t_\pi, U^t_\pi]$, which is a contradiction, as L^t_π and U^t_π are obtained from (12). ■

VII. Complexity and flexibility of the proposed approach

The proposed approach guarantees the optimality of the solution (in terms of maximal permissiveness), but at the same time it is flexible enough to offer the user a whole range of trade-offs between its computational load and the quality of the solution, in order to deal with tight-timed problems. The next two subsections discuss the complexity of the approach and possible heuristic compromises, respectively.

A. A discussion on complexity

The SC design requires a reachability analysis in a timed context, that can be restricted, but not avoided. This is the main source of complexity of the methodology, which can be partially mitigated if a heuristic trade-off between complexity

and optimality (*i.e.* permissiveness) is adopted, as suggested in the next subsection.

Let us consider Algorithm 1 first, regarding the computation of the control law. This computation is carried out when a new class is reached that has at least an enabled controllable transition (so that a decision has to be made concerning its firing). The calculation of the control law determines the allowed FTIs for the controllable transitions. To this aim two ILP problems are solved (see equation (12)) for each path in the PMSCG going from the initial class to the target class (or classes). The number of such paths is an intrinsic feature of the considered instance. These ILP problems have a worst-case number of constraints and binary variables in $O(n_n n_t)$, that is bilinear with respect to the length of the path (n_n) and the size of the net (n_t is the number of transitions). Therefore, their solution is in general a manageable task.

The construction of the PMSCG with Algorithm 4 corresponds to the actual reachability analysis. As it is inherently exponential, it can pay huge dividends to limit its extension as much as possible, avoiding the generation of the full MSCG. To this purpose, two pre-processing steps restrict the length of the potential sequences (based on the time limit imposed by the specifications), and the set of firing vectors consistent with the potential sequences (based on structural analysis and on the reachability objective). The former step enumerates the MS T-invariants of the complemented net, the latter the non-MS SCTs, and consequently the FCVs: only the sequences compatible with these FCVs are included in the PMSCG. While the former step is relatively easy and admits efficient algorithms, the latter deserves some further remarks. Since the non-MS SCTs are finite (as explained in Section IV-A), the second step could be sped up by neglecting the admissibility of the FCVs and obtaining the non-MS SCTs by simple algebraic combinations of the MS T-invariants (see equation (1) in Section II-B and Thm. 1 in Section IV-A).

However, the resulting PMSCG would include spurious paths, increasing the cost of the *timed* reachability analysis, which is much more computationally intensive. We therefore prefer to apply a B&B process to check that there exists at least one admissible transition sequence for each FCV used to build the PMSCG. This corresponds to anticipating part of the reachability analysis to the second pre-processing step, increasing its computational load to reduce that of the third one. The rationale is that a purely logical reachability analysis is generally faster than the corresponding timed task.

The computational advantage guaranteed by the two limitations (sequence length and structural analysis) is difficult to quantify precisely, as it is case-dependent. It is certainly significant for control problems with a very large reachability graph and a short time horizon. Notice that, for unbounded systems, a full reachability analysis would even be awkward, if not impossible.

B. Complexity mitigation and heuristics

The control law need not actually be computed multiple times. Indeed, as its computation adopts a worst-case perspective, the specifications are guaranteed for any possible future firing schedule of the uncontrollable transitions, simply following the computed path without further changes. The purpose of recalculating the control when entering new classes is to enlarge the firing time intervals (FTIs) of the controllable transitions with respect to the initial solution, taking advantage of the system evolution after the firing of an uncontrollable transition.

The computation of the control law itself involves the solution of a pair of ILPs for each path in the PMSCG. If computational time is an issue, the solution of an ILP problem can be prematurely arrested as soon as the solver returns a feasible solution.

The most impactful heuristic, however, consists in adopting a value for K smaller than the bound computed in Section II-B. This results in a reduced PMSCG, which can be employed to find a feasible solution more quickly. Of course, as long as some legal trajectories are neglected the solution found will not be necessarily maximally permissive. This paves the way for computation-saving strategies, where the PMSCG is constructed incrementally (even one path at a time) with the aim to find gradually improving feasible solutions. Obviously, if protracted to the point of including all legal trajectories in the PMSCG, this strategy will eventually provide the same optimality guarantees of the presented approach.

VIII. Case study

Consider the TPN system depicted in Figure 3 modelling a simple material handling system with two AGV systems (subnets on the left) and a workstation (subnet on the right).

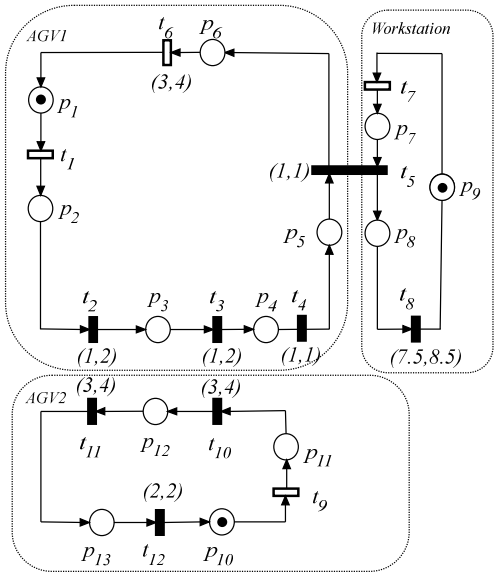

Fig. 3. TPN system of the case study.

The AGVs perform missions along their respective guided paths. Precisely, p_1 and p_{10} represent their home positions, and the remaining places of each AGV subnet are associated to the presence of the vehicle in certain sections of its guided path. Accordingly, transitions t_1 and t_9 are associated to the commands to start the respective missions, and the remaining transitions in each AGV subnet (with the exception of t_5)

represent the movement of the vehicle from one section to another. Finally, transition t_5 describes the unloading of a part in the input buffer of the workstation. As for the workstation, t_7 represents the command to start a working cycle and t_8 the processing of a part, while the marking of p_7 indicates that the workstation is waiting for a part, and p_8 and p_9 represent the working and the idle state, respectively.

Transitions t_1, t_7 and t_9 are associated to remote and controllable events (the commands can be enabled at any time), t_6 is associated to a prospective and controllable event (the speed of the movement in the respective section can be tuned within the static interval shown in Figure 3), while the other transitions are associated to prospective and uncontrollable events and their static intervals are shown in Figure 3.

The following GTSS summarizes the logical and temporal specifications of the control problem:

$$g = \quad (\boldsymbol{m}_0, \varnothing, [0,0], [\,])(\mathscr{L}_1, \mathscr{F}_1, [3,5.5], [3,6.5]) \\ (\mathscr{L}_2, \mathscr{F}_2, [9,13.5], [\,]),$$

where $\mathscr{L}_1 = \{\boldsymbol{m}_1\}$, with $\boldsymbol{m}_1 = p_5 + p_7 + p_{10}$, $\mathscr{F}_1 = \{\boldsymbol{m} |\ \boldsymbol{m}(p_3) + \boldsymbol{m}(p_{12}) > 1\}$, $\mathscr{L}_2 = \{\boldsymbol{m}_{21}, \boldsymbol{m}_{22}\}$, with $\boldsymbol{m}_{21} = p_1 + p_8 + p_{13}$, $\boldsymbol{m}_{22} = p_6 + p_9 + p_{13}$, and $\mathscr{F}_2 = \{\boldsymbol{m} |\ \boldsymbol{m}(p_2) = 1 \vee \boldsymbol{m}(p_9) = 1\}$. In practice, g requires that:

a) The system, initially in marking $\boldsymbol{m}_0$ at the initial time instant $\tau_0 = 0$, leaves it to reach the marking $\boldsymbol{m}_1$, representing that a part is ready to be delivered by AGV1 to the input buffer of the workstation while AGV2 is (or remains) idle. This must occur not earlier than 3 and not later than 5.5 time units, and without passing by any marking such that $\boldsymbol{m}(p_3) + \boldsymbol{m}(p_{12}) > 1$, representing a shared zone where only one vehicle at a time can stay.

b) The system leaves $\boldsymbol{m}_1$ not earlier than 3 and before 6.5 to reach either of the two markings in $\mathscr{L}_2$, thus bringing back to the home position either AGV1 or the workstation, while the other two agents remain one step from their respective home positions. In the process, any marking with $\boldsymbol{m}(p_2) = 1$ or $\boldsymbol{m}(p_9) = 1$ must be avoided, indicating that only one agent between AGV1 and the workstation is allowed to reach its initial state. Finally, $\mathscr{L}_2$ must be reached between 9 and 13.5 time units.

A. Step 1

The TPN has two T-invariants, namely, $\boldsymbol{TI}^{(1)} = [1\,1\,1\,1\,1\;1\,1\,1\,0\,0\,0\,0]^T$, $\boldsymbol{TI}^{(2)} = [0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1]^T$, the first accounting for the operation cycle of AGV1 and the workstation, and the second for the cycle of AGV2. A simple inspection of the two loops reveals that there is no way that, starting from the current marking, they can be completed in less than $\bar{\tau}_1 = 8.5$ and $\bar{\tau}_2 = 8$ time units, respectively.

The first step of the GTSS requires to go from $\boldsymbol{m}_0$ to $\boldsymbol{m}_1$. The corresponding complemented net N_c has only 3 MS T-invariants, *i.e.*: $\boldsymbol{y}^{(1,1)} = [1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0]^T$, $\boldsymbol{y}^{(1,2)} = [0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0]^T$, $\boldsymbol{y}^{(1,3)} = [1\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,1]^T$, the first two being NCTs (they correspond to $\boldsymbol{TI}^{(1)}$ and $\boldsymbol{TI}^{(2)}$) and the third being an SCT. As there is only one SCT, the only possible way to obtain other SCTs consists in combining $\boldsymbol{y}^{(1,3)}$ with integer multiples of the two NCTs. However, the first mission has to be completed within 5.5 time units, which is more than what required to complete the corresponding cycles. Therefore, the only SCT that may satisfy the timing constraints is precisely $\boldsymbol{y}^{(1,3)}$. Accordingly, $\boldsymbol{Y}_1^* = \{\boldsymbol{y}^{(1,3)}\}$. This is confirmed by Algorithm 3, with a maximum sequence length of $K = 5$, and this guarantees that the SCT is associated to at least one fireable sequence of the untimed net.

The second step of g requires to go from $\boldsymbol{m}_1$ to $\boldsymbol{m}_{21}$ or $\boldsymbol{m}_{22}$. The mission has to be completed in no more than 10.5 time units, which allows for a single execution of the two system cycles. This results in a bound of 17 for K in both cases, according to Thm. 1. Regarding the reachability problem from $\boldsymbol{m}_1$ to $\boldsymbol{m}_{21}$, the complemented net has 3 T-invariants, *i.e.* the two previous NCTs ($\boldsymbol{y}^{(2a,1)} = \boldsymbol{y}^{(1,1)}$ and $\boldsymbol{y}^{(2a,2)} = \boldsymbol{y}^{(1,2)}$), plus the SCT $\boldsymbol{y}^{(2a,3)} = [0\,0\,0\,0\,1\,1\,0\;0\,1\,1\,1\,0\,1]^T$. The following solution set was found with Algorithm 2: $\boldsymbol{Y}_{2a}^* = \{\boldsymbol{y}^{2a,3}, \boldsymbol{y}^{(2a,3)} + \boldsymbol{y}^{(2a,2)}, \boldsymbol{y}^{(2a,3)} + \boldsymbol{y}^{(2a,1)}, \boldsymbol{y}^{(2a,3)} + \boldsymbol{y}^{(2a,1)} + \boldsymbol{y}^{(2a,2)}, \boldsymbol{y}^{(2a,3)} + 2\boldsymbol{y}^{(2a,2)}\}$. A similar reasoning applies for the alternative mission to $\boldsymbol{m}_{22}$, for which $\boldsymbol{y}^{(2b,1)} = \boldsymbol{y}^{(1,1)}$, $\boldsymbol{y}^{(2b,2)} = \boldsymbol{y}^{(1,2)}$, and the unique MS SCT is $\boldsymbol{y}^{(2b,3)} = [0\,0\,0\,0\,1\,0\,0\,1\,1\,1\,1\,0\,1]^T$. The corresponding set of SCTs is given by $\boldsymbol{Y}_{2b}^* = \{\boldsymbol{y}^{(2b,3)}, \boldsymbol{y}^{(2b,3)} + \boldsymbol{y}^{(2b,2)}, \boldsymbol{y}^{(2b,3)} + \boldsymbol{y}^{(2b,1)}, \boldsymbol{y}^{(2b,3)} + \boldsymbol{y}^{(2b,1)} + \boldsymbol{y}^{(2b,2)}\}$. In conclusion, $\boldsymbol{Y}_2^* = \boldsymbol{Y}_{2a}^* \cup \boldsymbol{Y}_{2b}^*$.

B. Step 2

The second step consists in constructing the PMSCG by expanding all sequences that follow the computed FCVs and that can be fired in the TPN, and subsequently pruning those that cannot be possibly executed without violating g.

As discussed in Remark 2, the computation of all fireable sequences compatible with an FCV does not require to check all possible permutations of the involved transitions (which could be an extremely large number), but can be tackled as a simplified (temporal) reachability problem. To give an idea of the potential reduction of the computational complexity, consider *e.g.* the non-minimal SCT $\boldsymbol{y}^{(3)} + \boldsymbol{y}^{(2)}$ obtained for the first mission. There are 9! possible sequence permutations compatible with the associated FCV, but only 302 fireable sequences from a logical point of view, and only 42 of these are fireable also from a timed point of view. Finally, none of them admits a value for the transition firing times that meets the temporal conditions expressed by g, which is a necessary condition for the solvability of the control problem.

Repeating this analysis on all the provided FCVs one can determine merely 19 sequences that are fireable according to all the requirements of g, each corresponding to a different path of the PMSCG in Figure 4, obtained by applying Algorithm 4. The set of 19 paths of the PMSCG is denoted $\Pi(C_0)$, C_0 being the class corresponding to the initial state. Recall that this is not sufficient for the solvability of the control problem, in that when solving the latter one has to take into account that uncontrollable transitions cannot be forced to fire at will.

Notice that the firing of any enabled transition, that would cause the system to exit the PMSCG, must be explicitly forbidden, by applying condition (13) or (14). For example, t_9 cannot be allowed to fire in C_0, since there is no admissible

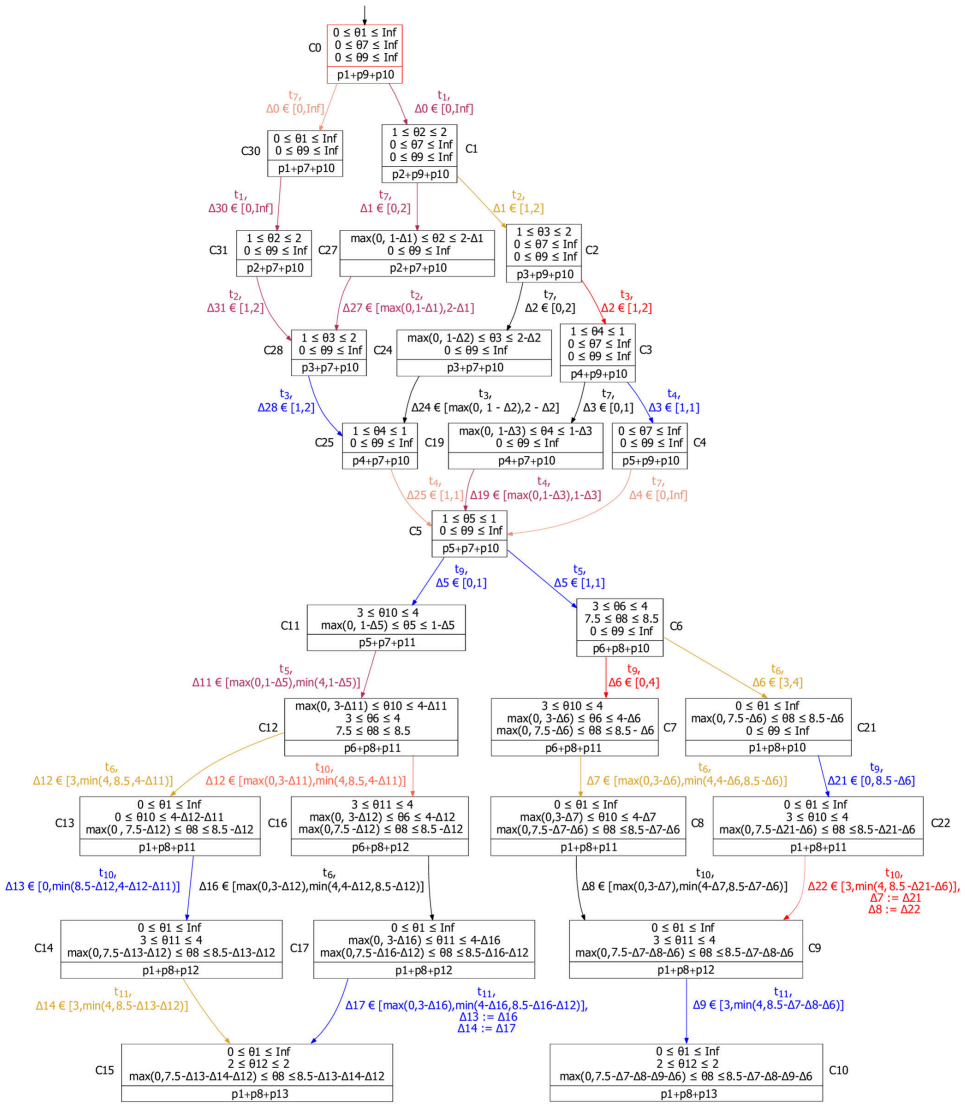

Fig. 4. PMSCG for the case study example.

sequence that can start with t_9. Since t_9 is controllable, its firing can be delayed *ad libitum*, which is virtually equivalent to prohibiting it. Similarly, t_8 should be prevented from firing in C_9, if g is to be met, but since this transition is uncontrollable, this results in a more compelling constraint. Indeed, Δ_9 must be smaller than the lower firing bound of t_8.

C. Step 3

By solving problem (12) for the 19 paths of the PMSCG, it follows that only the paths where t_9 fires before t_5 are compliant with g, in a worst case scenario (*i.e.*, for all possible time firing patterns of the uncontrollable transitions). Precisely, let $\Pi_1 = \{\pi_2, \pi_3, \pi_6, \pi_7\ \pi_{10}, \pi_{11}, \pi_{14}, \pi_{15}\}$ and $\Pi_2 = \{\pi_{18}, \pi_{19}\}$. Then, if the system were to follow any path in the set $\Pi(C_0)\backslash(\Pi_1 \cup \Pi_2)$ there would be no way to guarantee that g is met, by action of the controllable transitions. Conversely, the specifications can be guaranteed if one of the paths in $\Pi_1 \cup \Pi_2$ is chosen. Accordingly, any transition causing the system to follow a different path must also be prevented from firing by the supervisor.

Now, considering only the paths in $\Pi_1 \cup \Pi_2$, and taking into account all the necessary constraints required to keep the system on track, it results that $\mathcal{F}^{\pi}_{t_1}(S_0, \tau_0) = [0, 0.5]$, $\forall \pi \in \Pi_1$ and $\mathcal{F}^{\pi}_{t_7}(S_0, \tau_0) = [0, \infty]$, $\forall \pi \in \Pi_1$, while $\mathcal{F}^{\pi}_{t_1}(S_0, \tau_0) = [0, \infty]$, $\forall \pi \in \Pi_2$, $\mathcal{F}^{\pi}_{t_7}(S_0, \tau_0) = [0, 0.5]$, $\forall \pi \in \Pi_2$. Hence, by application of the intersection operator, one can conclude that $\mathcal{F}_{t_1}(S_0, \tau_0) = [0, 0.5]$ and $\mathcal{F}_{t_7}(S_0, \tau_0) = [0, 0.5]$ hold. This result implies that, provided that t_1 or t_7 fires before 0.5 time units, there always exists a firing time pattern of the controllable transitions that will be enabled afterwards that guarantees that g is met, notwithstanding any legitimate firing time pattern of the uncontrollable transitions. We next analyze a possible state (and class) evolution of the TPN system, illustrating the control evaluations along the sequence (see also Table I).

Notice that under a SC approach, both t_1 and t_7 are enabled (not forced) to occur at S_0. A controller (not the supervisor) will ultimately decide which of the two controllable transitions to fire and force its occurrence. Once this control decision is taken, the system will evolve from the initial state. The procedure need not be invoked again, as long as the system

TABLE I
STATE EVOLUTION AND CONTROL FUNCTION EVALUATION OF NET SYSTEM IN FIGURE 4 FROM THE INITIAL STATE ALONG THE SEQUENCE $(t_1, 0.5)(t_7, 1.5)(t_2, 2.5)(t_3, 3.5)(t_4, 4.5)(t_9, 4.5)(t_5, 5.5)(t_6, 8.5)$

(t, τ)	C	S	$\boldsymbol{m}$	$\boldsymbol{\Phi}$	$\hat{\boldsymbol{\Phi}}$
(-,0)	C_0	S_0	$p_1 + p_9 + p_{10}$	$0 \leq \theta_1 \leq \infty$ $0 \leq \theta_7 \leq \infty$ $0 \leq \theta_9 \leq \infty$	[0,0.5] [0,0.5] $\emptyset$
$(t_1, 0.5)$	C_1	S_1	$p_2 + p_9 + p_{10}$	$1 \leq \theta_2 \leq 2$ $0 \leq \theta_7 \leq \infty$ $0 \leq \theta_9 \leq \infty$	$[1, 2]$ $[0, 2]$ $\emptyset$
$(t_7, 1.5)$	C_{27}	S_2	$p_2 + p_7 + p_{10}$	$0 \leq \theta_2 \leq 1.5$ $0 \leq \theta_9 \leq \infty$	$[0, 1.5]$ $\emptyset$
$(t_2, 2.5)$	C_{28}	S_3	$p_3 + p_7 + p_{10}$	$1 \leq \theta_3 \leq 2$ $0 \leq \theta_9 \leq \infty$	$[1, 2]$ $\emptyset$
$(t_3, 3.5)$	C_{25}	S_4	$p_4 + p_7 + p_{10}$	$1 \leq \theta_4 \leq 1$ $0 \leq \theta_9 \leq \infty$	$[1, 1]$ $\emptyset$
$(t_4, 4.5)$	C_5	S_5	$p_5 + p_7 + p_{10}$	$1 \leq \theta_5 \leq 1$ $0 \leq \theta_9 \leq \infty$	$[1, 1]$ $[0, 1)$
$(t_9, 4.5)$	C_{11}	S_6	$p_5 + p_7 + p_{11}$	$1 \leq \theta_5 \leq 1$ $3 \leq \theta_{10} \leq 4$	$[1, 1]$ $[3, 4]$
$(t_5, 5.5)$	C_{12}	S_7	$p_6 + p_8 + p_{11}$	$3 \leq \theta_6 \leq 4$ $7.5 \leq \theta_8 \leq 8.5$ $2 \leq \theta_{10} \leq 3$	$[3, 3]$ $[7.5, 8.5]$ $[2, 3]$
$(t_6, 8.5)$	C_{13}	S_8	$p_1 + p_8 + p_{11}$	$0 \leq \theta_1 \leq \infty$ $0 \leq \theta_{10} \leq 0$ $3.5 \leq \theta_8 \leq 4.5$	$\emptyset$ $[0, 0]$ $[3.5, 4.5]$

follows one of the legitimate timed paths (the controllable transitions fired in the sequence must adhere to the solution of problem (12) for that path). However, the firing of uncontrollable transitions may relax the timing constraints on the control problem as the system evolves, enlarging the solution space.

Consider for example the case that the controller decides to fire t_1 at $\tau = 0.5$. Now, if we run again Algorithm 1 for t_7 in the new state S_1 we get $\mathcal{F}_{t_7}(S_1, 0.5) = [0, 2]$, which indicates that there is now a larger discretion on the time of firing of t_7 with respect to the initial state, where the allowed FTI for t_7 was just $[0, 0.5]$.

Assume now that t_7 is fired at $\tau = 1.5$, taking the system in the state S_2, included in the class C_{27}. Here, the only enabled controllable transition is t_9, but its firing must still be prevented to keep the system in the PMSCG. This is true also when t_2 and t_3 fire. Assume that $(t_2, 2.5)(t_3, 3.5)$ is observed. However, with the subsequent firing of t_4 at time 4.5, the system enters state S_5 which belongs to the class C_5, and the firing of t_9 would now keep the evolution inside the PMSCG. Then, Algorithm 1 returns $\mathcal{F}_{t_9}(S_5, 4.5) = [0, 1)$. This ensures that t_9 will fire before t_5, therefore keeping the system along the paths belonging to $\Pi_1 \cup \Pi_2$. After the firing of t_9 and t_5, a state is reached where a controllable transition is again enabled. Indeed, assume that the sequence $(t_9, 4.5)(t_5, 5.5)$ is observed, and the system reaches S_7, included in the class C_{12}. Algorithm 1 returns $\mathcal{F}_{t_6}(S_7, 5.5) = [3, 3]$ for transition t_6, implying that it must be fired exactly at $\tau = 8.5$ to guarantee g. If this is indeed what happens, then the system will travel safely to the target provided that t_1 is disabled (to avoid exiting the MSCG), whatever the firing times of the uncontrollable transitions t_{10} and t_{11}, as guaranteed by Algorithm 1. Indeed, t_8 will not fire before reaching the target. On the other hand, if t_{10} fires first in C_{12}, say at time $\tau = 7.5$, the system enters a state S_8 included in C_{16} where t_6 is still enabled. Provided that it is fired within the FTI $\mathcal{F}_{t_6}(S_8, 7.5) = [0, 1]$ (and t_1 is disabled), the target will be safely reached avoiding the firing of t_8.

IX. CONCLUSIONS AND FUTURE WORK

In this work we outline a framework for the supervisory control of TDESs, represented in the form of TPNs. This framework is specifically designed to deal with temporal reachability-type specifications, where the user prescribes a certain sequence of markings to be reached (and other markings to avoid in the process), with timing constraints. Specifications of this kind are typical *e.g.* in trajectory planning problems, but are also common in various other applications. Notice that the proposed framework can deal with time-varying specifications and static FTIs as well.

Since the supervisory control approach described in this work is based on a worst case analysis, it sometimes occurs that the problem does not admit a solution at the current state (the firing time window for the enabled controllable transition that guarantees the achievement of the specifications is empty). This does not necessarily mean that the specifications cannot be met, but only that this outcome cannot be guaranteed (it requires that some of the uncontrollable transitions fire with a certain specific timing pattern). Repeating the procedure later on (for another enabled controllable transition in another reached state), may provide guaranteed solutions. However, there remains the open problem of deciding what policy to enact in the current state, where the procedure did not provide an answer. In this respect, future research activity will focus on the design of an optimization-based procedure that suggests the firing time for the enabled controllable transition that is most likely to yield positive results in the future.

REFERENCES

[1] B. Brandin and W. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 39, no. 2, pp. 329–342, 1994.

[2] B. Zhao, F. Lin, C. Wang, X. Zhang, M. P. Polis, and L. Y. Wang, "Supervisory control of networked timed discrete event systems and its applications to power distribution networks," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 146–158, June 2017.

[3] F. Basile and P. Chiacchio, "On the implementation of supervised control of discrete event systems," *IEEE Transactions Control System Technology*, vol. 15, no. 4, pp. 725–739, 2007.

[4] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Massachusetts Inst. of Technology, Cambridge, MA, USA, Tech. Rep., 1974.

[5] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE Trans. on Software Eng.*, vol. 17, no. 5, pp. 259–273, 1991.

[6] J. C. Wang, Y. Deng, and G. Xu, "Reachability analysis of real-time systems using time Petri nets," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, no. 5, pp. 725–736, 2000.

[7] S. Bernardi and J. Campos, "Computation of performance bounds for real-time systems using time Petri nets," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 168–180, 2009.

[8] R. Hadjidj and H. Boucheneb, "Efficient reachability analysis for time Petri nets," *IEEE Trans. Computers*, vol. 60, no. 8, pp. 1085–1099, 2011.

[9] G. Jiroveanu and R. Boel, "A distributed approach for fault detection and diagnosis based on time Petri nets," *Math. Comp. Simul.*, vol. 70, no. 5, pp. 287–313, 2006.

[10] F. Basile, M. P. Cabasino, and C. Seatzu, "State estimation and fault diagnosis of labeled time Petri net systems with unobservable transitions," *IEEE Trans. Autom. Control*, vol. 60, no. 4, pp. 997–1009, 2015.

[11] Z. He, Z. Li, A. Giua, F. Basile, and C. Seatzu, "Some remarks on "State Estimation and Fault Diagnosis of Labeled Time Petri Net Systems with Unobservable Transitions"," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 5253–5259, Dec 2019.
[12] A. S. Sathaye and B. H. Krogh, "Synthesis of real-time supervisors for controlled time Petri nets," *Proc. 32nd IEEE Conf. Decision Control, San Antonio, TX, USA*, pp. 235–236, 1993.
[13] ——, "Supervisor synthesis for real-time discrete event systems," *Discrete Event Dynamic Systems*, vol. 8, no. 1, pp. 5–35, 1998.
[14] L. Li, F. Basile, and Z. Li, "An approach to improve permissiveness of supervisors for GMECs in Time Petri Net systems," *IEEE Transactions on Automatic Control*, vol. 65, no. 1, pp. 237–251, Jan 2020.
[15] G. Gardey, O. F. Roux, and O. H. Roux, "Safety control synthesis for time Petri nets," in *Proc. IEEE Int. Workshop Discrete Event Syst.*, Ann Arbor, MI, USA, 2006, pp. 222–228.
[16] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *CONCUR 2005 – Concurrency Theory*, M. Abadi and L. de Alfaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 66–80.
[17] P. Heidari and H. Boucheneb, "Maximally permissive controller synthesis for time Petri nets," *International Journal of Control*, vol. 86, no. 3, pp. 493–511, 2013.
[18] Y. Abdeddaïm, E. Asarin, and O. Maler, "Scheduling with timed automata," *Theoretical Computer Science*, vol. 354, no. 2, pp. 272 – 300, 2006.
[19] H. Wang, L. Grigore, U. Buy, M. Lehene, and H. Darabi, "Enforcing periodic transition deadlines in time Petri nets with net unfoldings," *IEEE Trans. Syst., Man, Cybern. A*, vol. 41, no. 3, pp. 522–539, 2011.
[20] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. of the IEEE*, vol. 77, no. 4, pp. 541–580, April 1989.
[21] G. Memmi and G. Roucairol, "Linear algebra in net theory," in *Proceedings of the Advanced Course on General Net Theory of Processes and Systems: Net Theory and Applications*. London, UK, UK: Springer-Verlag, 1980, pp. 213–223.
[22] A. E. Kostin, "Reachability analysis in T-invariant-less Petri nets," *IEEE Trans. on Automatic Control*, vol. 48, no. 6, pp. 1019–1024, June 2003.
[23] C. Seatzu, M. Silva, and J. van Schuppen, "Control of discrete-event systems. Automata and Petri net perspectives," *Lecture Notes in Control and Information Sci.*, vol. 433, 2012.
[24] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. Roux, "Comparison of different semantics for time Petri nets," in *Proc. 3rd Int. Conf. Automated Technol. Verification Anal.*, 2005.
[25] A. E. Kostin, "Using transition invariants for reachability analysis of Petri nets," in *Petri Net,Theory and Applications*, V. Kordic, Ed. I-Tech Education and Publishing, Feb. 2008, ch. 19, pp. 435–458.

Francesco Basile (SM'11) received the Laurea degree in electronic engineering and the Ph.D. degree in electronic and computer engineering from the University of Naples, Naples, in 1995 and 1999, respectively. In 1999, he was a Visiting Researcher with the Departamento de Ingenieria Informatica y Systemas, University of Zaragoza, Zaragoza, Spain, for six months. He is currently Full Professor of Automatic Control with the Dipartimento di Ingegneria dell'informazione ed elettrica e matematica applicata, Università of Salerno, Fisciano, Italy. He has published over 120 papers on international journals and conferences. His current research interests include modeling and control of discrete event systems, automated manufacturing, and robotics. Prof. Basile has been Associate Editor of the International Journal of Robotics and Automation, IEEE Transactions on Control Systems Technology and IEEE Transactions on Automation Science and Engineering. He has been member of IEEE Control System Society Conference Editorial Board. He is Associate Editor of IEEE Control Systems Letters. He has been General Chair of 14th International Workshop on Discrete Event Systems (WODES 2018).

Roberto Cordone was born in Milan, Italy, in 1969. He received the "Laurea" degree in Electronic Engineering and the Ph.D. degree in Computer Science and Control Theory from the Politecnico di Milano, Milano, Italy, in 1996 and 2000, respectively. From 2002 to 2017 he was an Assistant Professor with the Universitá degli Studi di Milano. Since 2017 he is an Associate Professor with the Universitá degli Studi di Milano, where he holds courses on heuristic algorithms and decisions methods and models. He is a member of the editorial board of IEEE Transactions on Automation Science and Engineering. His research interests include operations research and algorithm design and analysis.

Luigi Piroddi (M'07) was born in London, U.K., in 1966. He received his laurea degree in Electrical Engineering and the Ph.D. degree in Computer Science and Control Theory from the Politecnico di Milano, Milano, Italy, in 1990 and 1995, respectively. Between 1994 and 1999, he was a Professor of fundamentals of automation with the Universitá degli Studi di Bergamo, Bergamo, Italy. From 1999 to 2004, he was an Assistant Professor with the Politecnico di Milano. From 2004 to 2015 he has been an Associate Professor, and from 2016 he is Full Professor with the same institution, where he holds various courses in the systems and control area. His research interests include nonlinear model identification, Petri nets, modeling, and control of manufacturing processes. He has served on the editorial board of the IEEE Transactions on Automation Science and Engineering from 2014 to 2017.