State estimation and fault diagnosis of Time labeled Petri net systems with unobservable transitions

Francesco Basile*, Maria Paola Cabasino** and Carla Seatzu**

* Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica applicata
Università di Salerno, Italy
fbasile@unisa.it

** Dipartimento di Ingegneria Elettrica ed Elettronica
Università di Cagliari, Italy.
{cabasino,seatzu}@diee.unica.it

***Abstract*—In this paper we present a procedure for the state estimation and fault diagnosis of a labeled Time Petri net system. Starting from the *State Class Graph* defined by Berthomieu and Diaz, we introduce a new graph called *Modified State Class Graph* (MSCG) that allows an exhaustive representation of the evolution of the timed system. Then, we present a procedure that, given a timed observation, i.e., a sequence of labels with their firing time instants, and a time instant τ, allows one to determine in which states the system can be at time τ by using the MSCG and solving a certain number of linear programming problems. Finally, we present a procedure to perform fault diagnosis using the MSCG.**

M.P. Cabasino, gratefully acknowledges Sardinia Regional Government for the financial support of her Post Doc fellowship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2007-2013 - Axis IV Human Resources, Objective l.3, Line of Activity l.3.1.). This work also falls under the Cyprus Research Promotion Foundation (CRPF) Framework Programme for Research, Technological Development and Innovation 2009–2010 (CRPF's FP 2009–2010), co-funded by the Republic of Cyprus and the European Regional Development Fund, and specifically under Grant $T\Pi E/OPIZO/0609(BE)/08$. This work has also been partially supported by RAS project (L.R. n. 7/2007, Year 2010). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of CRPF or RAS.

Postprint - Work published on IEEE Transactions on Automatic Control (http://dx.doi.org/10.1109/TAC.2014.2363916)

I. Introduction

A. Paper contribution

The explicit consideration of time is crucial for the specification and the verification of systems such as communication protocols, circuits, or real-time systems as well as to study a series of extremely important problems such as state estimation, state feedback control and fault diagnosis.

In the Petri net (PN) framework the first main distinction is between Time PNs [**?**] and Timed PNs [**?**]. In Time PNs enabled transitions may fire within given time intervals that may either be associated with places or transitions (P-Time PNs and T-Time PNs, respectively). In Timed PNs enabled transitions fire as soon as given time delays have elapsed. As in the previous case, delays may either be associated with places or transitions (P-Timed PNs and T-Timed PNs, respectively).

In this paper we deal with the problem of *state* estimation and fault diagnosis of labeled T-Time PNs and we call them labeled Time PNs (TPNs) for short.

In a TPN a state consists of a marking and a set of constraints, one for each transition enabled at the corresponding marking, that specify when transitions may actually fire. We assume that only the firing of certain transitions can be observed, e.g., because a sensor, that produces an observable output when they fire, is associated with them. We call such transitions *observable*. The other transitions are called *unobservable* (or *silent*) because their firing cannot be observed. Moreover, to keep the problem statement more general, we assume that the same sensor can be associated with more than one transition, i.e., more than one transition may share

the same label. We call such transitions *indistinguishable*. We first present an algorithm for the construction of a graph called *Modified State Class Graph* (MSCG) that collects all the information on the evolution of the TPN system; then, we present a procedure based on the exploration of the MSCG and on the solution of some linear programming problems, that allows one to determine which states are consistent with a given observation and a given time instant. Finally, we show that the MSCG can also be effectively used to perform online fault diagnosis.

The MSCG takes inspiration from the State Class Graph (SCG) firstly presented in [**?**]. Two are the main differences between our graph and the one in [**?**]: 1) the introduction of labels associated with transitions, and 2) the introduction of timing variables and constraints associated with edges. Thanks to these information it is possible to obtain an estimation of the state at a certain time instant τ, given a timed observation, i.e., a sequence of labels occurring at given time instants. The key idea behind the MSCG definition is that of expressing the firing domain in a class as a function of the firing intervals of transitions entering in the class, associating a variable with each arc entering in a class. Due to the modifications introduced in the graph, the number of nodes in the MSCG is always greater than or equal to the number of nodes in the SCG, but the finiteness property for bounded TPNs is preserved.

A preliminary version of this paper was presented in [**?**]. Specifically, in [**?**] we solved the problem of *marking* estimation of a TPN. Here, we extend such a procedure to labeled TPNs. Then, we study the problem of *state* estimation, thus considering for each timed observation not only the estimation of the set of markings consistent with the observation, but also a series of time intervals, one for each transition enabled at any marking in the set. Such time intervals specify when the enabled transitions may fire and, as clarified in the sequel, they are characterized in linear algebraic terms. In particular, they depend on a certain number of parameters related to the possible time occurrences of unobservable transitions interleaved with the observed transitions. Moreover, we formalize the procedure for the state estimation giving an algorithm where the MSCG is used. Finally, we present a procedure to perform online fault diagnosis of a TPN using the MSCG.

Diagnosis can be performed both online and offline. Both approaches have their advantages and disadvantages. Online diagnosis, that uses an interpreted diagnoser, allows one a saving in memory with respect to offline diagnosis, that uses a compiled diagnoser. On the other hand, online diagnosis requires to solve online linear programming problems. In this paper, the proposed method aims to keep the advantages of both approaches, providing a trade-off between memory requirements and online computations: MSCG is off-line computed and linear programming problems are solved online to select the set of possible sequences consistent with the timed observations.

The presentation is organized as follows. In Subsection I-B we give an overview of relevant literature on state estimation and diagnosis of TPNs, and contrast our contribution with respect to such works. In Subsection II-A we present necessary background on Time Petri nets. In Subsection II-B we provide a series of definitions related to the labeling function. In Section III we formalize the problem statement and clarify the assumptions on which our approach is based. Section IV presents the algorithm for the computation of the MSCG. In Section V we propose a procedure for the computation of the state estimation of a labeled TPN given a sequence of labels observed at given time instants, and a final time τ. In Section VI we analyze the complexity of the proposed procedure. In Section VII an algorithm that allows to perform diagnosis of a labeled TPN using the MSCG is introduced. In Section VIII conclusions are drawn and the future lines of our research in this topic are outlined.

B. Literature review

The problem of state estimation and diagnosis of a dynamic system is a fundamental issue in system theory. Just like for time driven systems, also in the case of Discrete Event Systems (DES) state estimation and diagnosis under partial observation has been discussed in the literature.

As an example, in [**?**] Giua and Seatzu using place/transition nets solved the problem of marking estimation under the assumption that the net structure is known, the initial marking is not known, and all transition firings can be observed. Later on, Cabasino *et al.* considered different forms of nondeterminism, namely unobservable transitions and indistinguishable transitions, but assume that the initial marking is known [**?**], [**?**]. In particular, in [**?**], [**?**] the authors not only address the problem of marking estimation, but also deal with online fault diagnosis. Other important contributions in this area have been proposed in [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**].

Both the problems of marking estimation and diagnosis have been extensively studied using logic DES. On the contrary, there are relatively few works dealing with these topics in the *timed* DES framework using automata

[?], [?], [?], [?], [?] as well as PNs [?], [?], [?], [?], [?], [?]. As a consequence, several related problems are still open.

The first approach to compute the state-space of a TPN was proposed by Berthomieu and Diaz in [?]. They presented a graph, called State Class Graph, whose nodes are sets of states of the TPN and whose edges are labeled with transitions. They proved that if $Lang$ is the language accepted by the SCG and $Lang'$ is the untimed language accepted by the TPN, then $Lang = Lang'$. Then, since the SCG does not accept the timed language of the TPN, the SCG can only be used conveniently for checking untimed reachability properties. In order to check real-time properties a set of places and transitions representing a non-intrusive observer need to be added to the TPN and then compute the SCG of the system "TPN + observer". However the observer may be as big as the net, if the property to be verified involves markings, and so the number of classes may be squared. Furthermore, each property requires a specific observer, and thus a new computation of the SCG [?].

In [?] Ghazel *et al.* presented a state estimation method based on the SCG for unlabeled TPNs without cycles of unobservable transitions. Differently from the approach here presented, in [?] all sequences firable from each class of the state estimator and their firing time intervals are explicitly associated with each node of the estimator. This makes larger the memory needed to store the estimator, even if this speeds up the estimation process. In our approach the solution of some linear programming problems is required to determine which states are consistent with the timed observation, thus reducing the memory requirements. Moreover, the approach here presented also applies to labeled nets and does not require (despite of [?]) absence of cycles of unobservable transitions (it is only required that cycles of unobservable transitions do not fire in time intervals of null length).

In [?] Bonhomme approached the problem of marking estimation for P-Time PN models from another perspective. A marking observer is assumed to be available for the untimed underlying PN. Then, a linear programming problem is formulated to check the set of firing sequences feasible on the untimed underlying PN and consistent with the current timed observation. The state explosion problem of SCG is avoided but differently from this paper, the full state of the system, marking and firing time interval associated with transitions (tokens for P-Time PN models), cannot be estimated.

In [?] Wang *et al.* proposed a fault diagnosis approach for unlabeled TPNs based on a graph called Fault Diagnosis Graph (FDG). The FDG, that is obtained from the SCG by only keeping the necessary information for the computation of the fault states and removing the unnecessary states, is updated online after each observation. Several are the differences between our approach and the one presented in [?]. In [?] it is required that the unobservable portion of the net is acyclic (since an unobservable subnet is computed each time an observable transition fires), while in our approach it is only required that cycles of unobservable transitions do not fire in time intervals of null length. In [?] no constraints are imposed on the time intervals associated with transitions, while we require that the boundaries of such intervals are rational numbers. In this paper we deal with labeled TPNs, while in [?] unlabeled TPNs are considered. Finally, our approach is based on the online inspection of the MSCG that can be computed offline, thus reducing the online complexity of the estimation procedure.

In [?] Lime and Roux presented an approach that builds the state class graph as a timed automaton (TA), called State Class Timed Automaton, thus keeping the temporal information of the TPN. Obtaining a TA instead of a graph allows one to use all the model checking tools available for TA. At this aim, translation techniques from TPNs to TA have been proposed in [?] and [?], thus some results, such as fault diagnosis using TA, could be re-used to study fault diagnosis on TPNs. However, we believe that using methods that are specifically designed for TPNs is very important and leads to significant advantages. Firstly, a direct method avoids any interpretation difficulty coming from using different models. Secondly, as well as we did for untimed PNs ([?], [?], [?], [?]), we want to build a research line that goes from state-estimation/fault diagnosis of labeled time Petri nets to diagnosability (see [?]), up to methods that make diagnosable a non-diagnosable timed system and/or optimally select sensors for ensuring diagnosability (this is one of the next steps of our research line - see the contribution [?] for untimed systems).

Finally, several abstractions of TPN space exist in the literature [?], [?]. Among the abstractions preserving linear properties like reachability and firing sequences Relaxed SCG (RSCG) and Contracted SCG (CSCG) are available [?] in addition to SCG. However, RSCG and CSCG cannot be used for state estimation since the firing time constraints of the transitions are not preserved. Other abstractions, that use clocks to characterize state classes instead of firing domains, are considered in [?]. A clock is associated with each transition to measure the time elapsed, since it became enabled most recently. An element of a state class domain gives the clock value for each transition, enabled at the class marking, but these

abstractions such as the Strong SCG (SSCG) yield a larger graph that do not preserve the finiteness property for all bounded TPNs. Clocks become essential if the atomicity property of the state classes within a state class graph must be preserved. The atomicity property states that if a state in a state class SC1 has successors in some successor state class SC2, then all the other states of SC1 have successors in SC2. This property is very important in a distributed settings since by making consistency between the local traces in the local state classes graphs of different models one can discard the local states that have no continuations in the successor state classes in each component [**?**]. Since, this paper focuses on state estimation in a centralized settings, then interval representation can be used avoiding to introduce clocks.

II. Time Petri nets with labels

A. Background on Time Petri nets

A TPN is a *Place/Transition net* (P/T net) [**?**], [**?**] where timing is associated with transition firing. In particular, time intervals are associated with transitions: a transition may fire at a given time instant if and only if it has remained logically enabled for an amount of time within its own time interval. More details are provided in the following.

As in *Place/Transition nets* (P/T nets) the net structure is defined as a quadruple $N = (P, T, Pre, Post)$ where: P is a set of m places; T is a set of n transitions; and $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the pre– and $post$– incidence functions that specify the arcs. The incidence matrix C of the net is equal to $C = Post - Pre$.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place a nonnegative integer number of tokens, represented by black dots. We denote $M(p)$ the marking of place p.

A transition t is said to be *logically* enabled at M iff $M \geq Pre(\cdot, t)$. The firing of a transition t at a marking M yields to marking $M' = M + C(\cdot, t)$. We denote $\mathcal{A}(M)$ the set of transitions logically enabled at M, i.e., $\mathcal{A}(M) = \{t \in T \mid M \geq Pre(\cdot, t)\}$.

A *Time PN* is defined as a couple $N_d = (N, Q)$ where $N = (P, T, Pre, Post)$ defines the net structure and $Q : T \to \mathbb{Q} \times (\mathbb{Q} \cup \{\infty\})$ defines the set of *static* closed intervals associated with transitions. In particular, given a transition $t_i \in T$, the function Q associates two rational numbers with t_i (the second one may also be ∞), namely $Q(t_i) = (l_i, u_i)$, where

$$l_i \geq 0, \quad u_i \geq l_i, \quad l_i \neq \infty.$$

Transition t_i may fire iff it remains logically enabled for $l_i \leq \theta_i \leq u_i$, i.e., for a time interval included in $[l_i, u_i]$. Moreover, here it is assumed that an enabled transition must fire within its firing interval unless it is disabled (*strong time semantic*). Logical enabling condition must hold consecutively (enabling memory policy) or may not hold consecutively (total memory policy), depending on the considered enabling policy [**?**].

A TPN N_d with a marking M_0 at the initial time instant $\tau_0 = 0$ is called a *marked* TPN, or a TPN *system*, and is denoted $\langle N_d, M_0 \rangle$.

A TPN evolution is defined by a *time-transition sequence* (TTS), namely a sequence of pairs (transition, time instant), that specify the sequence of transitions that have fired, and the corresponding time instants. As an example, the firing of $\sigma = (t_{i_1}, \tau_1)(t_{i_2}, \tau_2) \dots (t_{i_k}, \tau_k) \in (T \times \mathbb{R}_0^+)^*$ at the initial marking M_0 means that transition with index i_1 has fired at time τ_1, transition with index i_2 has fired at time τ_2, and so on. We denote this in a compact form as $M_0[t_{i_1}(\tau_1)\rangle M_1[t_{i_2}(\tau_2)\rangle M_2 \cdots [t_{i_k}(\tau_k)\rangle M_k$, where obviously it is $\tau_1 \leq \tau_2 \leq \cdots \tau_k$. More concisely, we also write $M_0[\sigma\rangle M_k$ or simply $M_0[\sigma\rangle$ if we do not want to specify the marking reached after the firing of σ, but only want to denote that σ may fire at M_0. In the following we denote $t_l(\sigma)$ the instant of time at which the last transition in σ fires. Thus according to the above notation, it is $t_l(\sigma) = \tau_k$.

Note that while in untimed Petri nets the state coincides with the marking of the net, this is no more true in a timed net. In particular, in a TPN the *state* is an $(e_k + 1)$-tuple (M_k, Θ_k), where $M_k \in \mathbb{N}^{|P|}$ is a reachable marking and Θ_k is a set of e_k inequalities $\bar{l}_{k_i} \leq \theta_{k_i} \leq \bar{u}_{k_i}$, $i = 1, \dots, e_k$, and e_k is the number of transitions enabled at M_k.

The generic inequality $\bar{l}_{k_i} \leq \theta_{k_i} \leq \bar{u}_{k_i}$ means that transition t_{k_i} may fire at M_k only after $\bar{l}_{k_i}$ time units have elapsed and must fire before $\bar{u}_{k_i}$ time units have elapsed, unless another enabled transition has fired meanwhile, disabling t_{k_i}. It holds $\bar{l}_{k_i} \geq l_{k_i}$ and $\bar{u}_{k_i} \leq u_{k_i}$, where $(l_{k_i}, u_{k_i}) = Q(t_{k_i})$. Obviously, if t_{k_i} has just been enabled at marking M_k, it is $\bar{l}_{k_i} = l_{k_i}$ and $\bar{u}_{k_i} = u_{k_i}$. Otherwise, it is $\bar{l}_{k_i} > l_{k_i}$ and/or $\bar{u}_{k_i} < u_{k_i}$. In the following sections we show how the bounds $\bar{l}_{k_i}$ and $\bar{u}_{k_i}$ can be characterized in linear algebraic terms as a function of a series of parameters related to the possible time occurrences of silent transitions interleaved with the observed transitions.

A marking M is *reachable* in $\langle N_d, M_0 \rangle$ if there exists a TTS σ such that $M_0 \ [\sigma\rangle \ M$. The set of all markings reachable from M_0 defines the *time reachability set* of $\langle N_d, M_0 \rangle$ and is denoted by $R_t(N_d, M_0)$. Note that $R_t(N_d, M_0)$ is always a subset (usually a strict subset)

of the reachability set [**?**] of the underlying untimed PN.

A TPN system $\langle N_d, M_0\rangle$ is *bounded* if there exists a positive constant k such that, for all $M \in R_t(N_d, M_0)$, $M(p) \leq k$. By virtue of the above consideration, it obviously may happen that a TPN system is bounded even if the underlying untimed PN is unbounded.

Finally, the *enabling degree* of a transition t logically enabled at a marking M is the highest integer number k such that $M \geq k\ Pre(\cdot, t)$. In simple words, in a purely logic model, the enabling degree of a transition t at a marking M denotes how many times t may fire at M at most.

In the rest of the paper we assume that the considered TPNs follow a *single server semantics* and an *enabling memory policy*. More details on this can be found in [**?**], [**?**]. In simple words, when using a single server semantic each transition represents an operation that can be executed by a single operation unit (a single server). Therefore, regardless of the current enabling degree, a transition may only fire once at a time. The enabling memory policy implies that a transition has no memory of previous enabling, i.e., if it remains enabled for a certain time and some other transition fires disabling it, when it is enabled again it does not keep into account the time intervals in which it has already been enabled. Assume as an example that a transition models the execution of a given operation that requires a time interval defined by its lower and upper bound. If such an operation is interrupted, it should start from the beginning, i.e., the time required to perform the given operation should be consecutive.

B. Labeling function and timed labels sequences

A *labeling function* $\mathcal{L} : T \to L \cup \{\varepsilon\}$ assigns to each transition $t \in T$ either a symbol from a given alphabet L or the empty string ε. We call labeled Petri net system the triple $\langle N, M_0, \mathcal{L}\rangle$.

We denote as T_u the set of transitions whose label is ε, i.e., $T_u = \{t \in T \mid \mathcal{L}(t) = \varepsilon\}$. Transitions in T_u are called *unobservable* or *silent*.

We denote as T_o the set of transitions labeled with a symbol in L. Transitions in T_o are called *observable* because, when they fire, their label can be observed. In this paper we assume that the same label $\gamma \in L$ can be associated with more than one transition. In particular, two transitions $t_1, t_2 \in T_o$ are called *indistinguishable* if they share the same label, i.e., $\mathcal{L}(t_1) = \mathcal{L}(t_2) = \gamma \in L$.

We extend the labeling function to define the *projection operator* $\mathcal{L} : T^* \to L^*$ recursively as follows:

(i) if $t_j \in T_o$ then $\mathcal{L}(t_j) = \gamma$ for some $\gamma \in L$;
(ii) if $t_j \in T_u$ then $\mathcal{L}(t_j) = \varepsilon$;
(iii) if $\sigma \in T^* \wedge t_j \in T$ then $\mathcal{L}(\sigma t_j) = \mathcal{L}(\sigma)\mathcal{L}(t_j)$;
Moreover, $\mathcal{L}(\lambda) = \varepsilon$ where λ is the empty sequence.

Furthermore, to avoid introducing too many different notations, we also extend the labeling function to TTSs $\sigma \in (T \times \mathbb{R}_0^+)^*$. In particular, we denote as $\mathcal{L}(\sigma) \in (L \times \mathbb{R}_0^+)^*$ the observable projection of σ that only contains those pairs (label-time instant) relative to observable transitions.

Finally, we denote as $\delta_o = (\gamma_{e_1}, \tau_1)(\gamma_{e_2}, \tau_2)\ldots(\gamma_{e_h}, \tau_h) \in (L \times \mathbb{R}_0^+)^*$ a *time-label sequence* (TLS), namely a sequence of pairs (label-time instant) that specify the sequence of labels that have been observed, and the corresponding time instants. In general, given a TLS δ_o there exist more than one TTS σ such that $\mathcal{L}(\sigma) = \delta_o$. In analogy to TTSs, we denote as $t_l(\delta_o)$ the instant of time at which the last label in δ_o is observed.

III. Problem statement

In this paper we deal with two closely connected problems, namely the estimation of the state of a labeled TPN and the fault diagnosis of a timed discrete event system modeled by a labeled TPN.

As in [**?**] we make the following assumptions whose importance will be clarified in the rest of the paper.

(A1) The considered TPN is *bounded*.
(A2) For all $t \in T$, the bounds l and u associated with t are rational numbers.
(A3) There do not exist cycles of unobservable transitions that can fire in a time interval of null length.

A. State estimation problem

Given a TLS $\delta_o \in (L \times \mathbb{R}_0^+)^*$, and a time instant $\tau \geq t_l(\delta_o)$, we want to find the set of states in which the system may be at time τ given the observation δ_o. We assume that there is no further observation since the last event in δ_o.

Now, the following three main definitions can be given.

***Definition** 1:* Given an observable TLS $\delta_o \in (L \times \mathbb{R}_0^+)^*$ and a time instant $\tau \geq t_l(\delta_o)$, the *set of sequences consistent with the observation δ_o and the time instant* τ is

$$\Sigma(\delta_o, \tau) = \{\sigma \in (T \times \mathbb{R}_0^+)^* \mid M_0[\sigma\rangle M, \ \mathcal{L}(\sigma) = \delta_o, \ t_l(\sigma) = \bar{\tau}, \text{ with } \bar{\tau} \leq \tau, \ \nexists t \in T_u : M[t\rangle \text{ and } r_t(M_0, \sigma) \leq \tau - \bar{\tau}\} \quad (1)$$

where $r_t(M_0, \sigma)$ denotes the *residual time*[1] of t after the firing of σ at M_0, i.e., the amount of time within which t surely fires at M. ■

Note that the last constraint in (1) ensures that it is possible to remain in marking M during the time interval $\tau - \bar{\tau}$. This is obviously not possible if there exists at least one unobservable transition whose residual time, when M is reached, is smaller than or equal to $\tau - \bar{\tau}$. Indeed if such a transition exists, then for sure it fires in a time instant smaller than or equal to τ.

***Definition** 2:* Given a TLS $\delta_o \in (L \times \mathbb{R}_0^+)^*$ and a time instant $\tau \geq t_l(\delta_o)$, the *set of markings consistent with the observation δ_o and the time instant τ* is

$$\mathcal{C}(\delta_o, \tau) = \{M \in \mathbb{N}^m \mid \quad M_0[\sigma\rangle M, \quad \sigma \in \Sigma(\delta_o, \tau)\}. \tag{2}$$

■

***Definition** 3:* Given a TLS $\delta_o \in (L \times \mathbb{R}_0^+)^*$ and a time instant $\tau \geq t_l(\delta_o)$, the *set of states consistent with the observation δ_o and the time instant τ* is

$$\mathcal{S}(\delta_o, \tau) = \{(M_k, \Theta_k) \mid \begin{array}{l} M_k \in \mathcal{C}(\delta_o, \tau) \text{ and} \\ \Theta_k = \{(\bar{l}_{k_i} \leq \theta_{k_i} \leq \bar{u}_{k_i})\} \ : \\ t_{k_i} \in \mathcal{A}(M_k)\}. \end{array} \tag{3}$$

■

Solving a problem of state estimation of a labeled TPN consists in providing a systematic approach to characterize the set $\mathcal{S}(\delta_o, \tau)$ for any observation δ_o and any time instant τ.

B. Fault diagnosis

As in most of the literature in the fault diagnosis framework [**?**], [**?**], [**?**], the set of unobservable transitions is partitioned in two subsets, namely $T_u = T_f \cup T_{reg}$ where T_f includes all fault transitions (modeling anomalous or faulty behavior), while T_{reg} includes all transitions pertaining to unobservable but "regular" events. The set T_f is further partitioned into r different subsets T_f^i, where $i = 1, \ldots, r$, that model the different fault classes.

Obviously, in the fault diagnosis framework rather than being interested in the set of markings in which the system can be, we are interested in the set of transitions that may have fired.

Given a TTS $\sigma = (t_{i_1}, \tau_1)(t_{i_2}, \tau_2) \ldots (t_{i_h}, \tau_h) \in (T \times \mathbb{R}_0^+)^*$, we denote as $log(\sigma) = t_{i_1} t_{i_2} \ldots t_{i_h}$ the "logic" sequence of transitions associated with σ, neglecting the time instants at which they have fired.

[1]The residual time of a transition t is clearly not a function of the current marking but is a function of the particular evolution that led to it, i.e., according to our notation, it is a function of M_0 and σ.

Based on the definitions given in the previous subsection, we generalize the notion of diagnoser we provided in [**?**] in the case of untimed labeled PNs.

***Definition** 4:* A *diagnoser* is a function

$$\Gamma : [L \times \mathbb{R}_0^+]^* \times \mathbb{R}_0^+ \times \{T_f^1, T_f^2, \ldots, T_f^r\} \to \{N, U, F\}$$

that associates with each observation $\delta_o \in (L \times \mathbb{R}_0^+)^*$, each time instant $\tau \in \mathbb{R}_0^+$, and each fault class T_f^i, $i = 1, \ldots, r$, a *diagnosis state*.

- $\Gamma(\delta_o, \tau, T_f^i) = N$ if $\forall \sigma \in \Sigma(\delta_0, \tau)$ and $\forall t_f \in T_f^i$, it is $t_f \notin log(\sigma)$. In such a case the ith fault cannot have occurred, because none of the TTSs consistent with the observation and the time τ, contains fault transitions of class i.
- $\Gamma(\delta_o, \tau, T_f^i) = U$ if:
  (i) $\exists \sigma \in \Sigma(\delta_0, \tau)$ and $t_f \in T_f^i$ such that $t_f \in log(\sigma)$,
  (ii) $\exists \sigma' \in \Sigma(\delta_0, \tau)$ such that $\forall t_f \in T_f^i$, it is $t_f \notin log(\sigma')$.
  In such a case a fault transition of class i may have occurred or not, i.e., it is uncertain, and we have no criteria to draw a conclusion in this respect.
- $\Gamma(\delta_o, \tau, T_f^i) = F$ if $\forall \sigma \in \Sigma(\delta_0, \tau)$ $\exists t_f \in T_f^i$ such that $t_f \in log(\sigma)$.
  In such a case the ith fault must have occurred, because all firable sequences consistent with the observation and the time instant τ contain at least one fault transition of class i. ■

Solving a problem of fault diagnosis in a labeled TPN framework consists in providing a systematic approach to compute the diagnosis states.

IV. Modified State Class Graph

In this section we introduce a graph called *Modified State Class Graph* (MSCG). This graph takes inspiration from the State Class Graph presented in [**?**] but, as explained in the Introduction, it has significant novelties with respect to it. In particular, it contains a series of information that allow to estimate the set $\mathcal{S}(\delta_o, \tau)$, as explained in the next section.

Let us consider a labeled TPN system $\langle N_d, M_0 \rangle$, where $N_d = (N, Q)$, $N = (P, T, Pre, Post)$, $Q : T \to \mathbb{Q} \times (\mathbb{Q} \cup \{\infty\})$. Let $\mathcal{L} : T \to L \cup \{\varepsilon\}$ be its labeling function defined over an alphabet L.

The *Modified State Class Graph* is a directed graph whose nodes are called *classes*. With each class is associated a *state* of the net, namely a reachable marking $M \in R_t(N_d, M_0)$ and a set of inequalities Θ that define the timing constraints relative to all transitions enabled at M, as explained in Subsection III-A.

Edges are labeled as $(t, \gamma, \Delta \in [l^*, u^*])$, where $t \in T$ is the transition whose firing leads from the marking in the target node to the marking in the source node; $\gamma = \mathcal{L}(t)$ is the label associated with t, and $\Delta \in [l^*, u^*]$ is a constraint on the time to go from the target node to the source node of the edge.

We say that two classes are *equivalent* if they share the same marking and the same set of inequalities.

The construction of the MSCG first requires the definition of the *Modified State Class Tree* (MSCT) according to Algorithm 1.

In simple words, Algorithm 1 constructs the MSCT starting from the initial class (Step 1) to which it corresponds the initial marking M_0 at time $\tau = 0$ and a set of inequalities, denoted Θ_0, associated with transitions enabled at M_0. Such inequalities simply define the static intervals of such transitions.

Tags are used to guarantee an exhaustive exploration of the graph with no repetition. At Step 2 an unexplored node, i.e., a node tagged "new", is arbitrarily selected. Let C_k be the corresponding class with marking M_k. At marking M_k a certain number of transitions are logically enabled, i.e., they belong to $\mathcal{A}(M_k)$. In general, only a subset of them may actually fire, in particular those whose lower bound at that class is less than or equal to the minimum of the upper bounds at the same class of the transitions logically enabled at M_k (Step 4). Indeed if such a constraint is violated for a given transition t_i, it means that such a transition, even if logically enabled, requires a time interval to fire that is too long with respect to the other transitions enabled at the same marking. So, in practice it cannot occur. Note that this is correct since we are considering an enabling memory policy, so transitions have no memory of previous enabling conditions. Finally, in Step 4 we write $max\{0, l_i^k\}$ to impose that the lower bound is always a nonnegative number.

Now, for all transitions that may actually fire, we compute the marking resulting from their firing. In Step 5 we denote M_q the marking resulting from the firing of the generic transition t_i at marking M_k. Note that such a marking is not related to the enabling degree since we are considering single server semantics, so all transitions at any marking, may only fire once at a time. To completely define the class we need to compute the set of inequalities Θ_q, namely the timing constraints relative to all transitions that are enabled at M_q. Obviously such constraints are different, depending on the fact that a transition has just become enabled, or it was already enabled at the previous class. In the first case the timing constraints simply coincide with the timing constraints associated with the transition. In the

Algorithm 1: Construction of the Modified State Class Tree

input : A labeled TPN system.
output: Its Modified State Class Tree.

1 **Initialize:** The root node C_0 is labeled with the initial marking M_0 and a set of inequalities Θ_0 defined as follows: $\forall t_i \in \mathcal{A}(M_0)$, let $l_i^0 \leq \theta_i \leq u_i^0$ where $l_i^0 = l_i$ and $u_i^0 = u_i$. Tag the root node "new". **while** *a node tagged "new" exists* **do**

2 Select a node C_k tagged "new".

3 **for** *all* $t_i \in \mathcal{A}(M_k)$ **do**

4 **if** $max\{0, l_i^k\} \leq min_{j:\ t_j \in \mathcal{A}(M_k)}\{u_j^k\}$, *where* l_i^k (u_j^k) *is the lower (upper) bound associated with* t_i (t_j) *at class* C_k, **then**

5 Let $M_q = M_k + C(\cdot, t_i)$ be the marking reached from M_k firing t_i.
for *all transitions* $t_r \in \mathcal{A}(M_q)$ **do**
if $t_r \in \mathcal{A}(M_k)$, *i.e., if* t_r *was already enabled at class* C_k, *and* $M_k - Pre(:, t_i) \geq Pre(:, t_r)$, **then**
let
$$l_r^q = l_r^k - \Delta_i, \quad u_r^q = u_r^k - \Delta_i$$
else
let
$$l_r^q = l_r, \quad u_r^q = u_r.$$

6 Add a new node C_q labeled with marking M_q and a set of inequalities Θ_q defined as follows: $\forall t_r \in \mathcal{A}(M_q)$, let $max\{0, l_r^q\} \leq \theta_r \leq u_r^q$.

7 Add an edge from C_k to C_q labeled "$t_i, \mathcal{L}(t_i),\ \Delta_i \in [max\{0, l_i^k\}, min_{j:\ t_j \in \mathcal{A}(M_k)}\{u_j^k\}]$".

8 **if** *there already exists a node equivalent to* C_q *in the tree* **then**
tag node C_q "duplicate",
else
tag it "new".

9 Untag node C_k.

second case, we need to reduce the upper and lower bounds at the previous class by an amount Δ_i, equal by definition to the time elapsed when going from the previous class to the new one. Such bounds should obviously be nonnegative.

Finally, at Step 8 we associate labels with edges. The label associated with an edge obviously contains information on the transition whose firing leads from the marking in the target node to the marking in the source node. Moreover, the edge also contains information on the time that must elapse when moving from such two markings. The condition is obviously the same as in the "if" condition in Step 4, thus explaining the label in Step 8.

The MSCG can be immediately obtained from the MSCT simply merging duplicate nodes corresponding to equivalent classes.

Under Assumptions A1 and A2 the number of classes of the MSCG, as well as the number of classes of the SCG [**?**], is finite, as proved in the following proposition.

***Proposition** 5:* Let $\langle N_d, M_0 \rangle$ be a TPN. Under Assumptions A1 and A2 the MSCG built according to Algorithm 1 is finite.

Proof: Assumption A1 guarantees that the number of reachable markings is finite. This is clearly not enough to guarantee the boundedness of the graph since the same marking may be associated with several classes. However, since a reachable marking always enables the same set of transitions, and since the timing constraints associated with the enabled transitions are in a finite number (due to Assumption A2 and to their structure $0 \leq l_r^q \leq \theta_r \leq u_r^q$), the combinations reachable markings–timing constraints are in a finite number. Thus the number of classes corresponding to the same marking is finite as well, and consequently the number of classes in the MSCG is also finite. □

The following example clarifies the notions introduced until now. Note that we voluntarily chose the example simple for the sake of clarity.

***Example** 6:* Let us consider the labeled TPN system whose net structure and labeling function are shown in Fig. 1. It is $T_o = \{t_1, t_5\}$, $T_u = \{t_2, t_3, t_4\}$, $\mathcal{L}(t_1) = a$, and $\mathcal{L}(t_5) = b$. Moreover, it is $Q(t_1) = (0, 1)$, $Q(t_2) = (0, 2)$, $Q(t_3) = (1, 3)$, $Q(t_4) = (1, 5)$, and $Q(t_5) = (2, 3)$. Finally, the initial marking is $M_0 = [1\ 0\ 0\ 0\ 0]^T$.

The corresponding MSCG is shown in Fig. 2. Each class is labeled with a reachable marking $M \in R_t(N_d, M_0)$ and a set of inequalities Θ that characterize the timing intervals of the transitions enabled at that class. As an example, the initial marking M_0 and the set of timing constraints $\Theta_0 = \{0 \leq \theta_1 \leq 1\}$ are associated with the initial class C_0. In fact, M_0 only enables t_1 and it is $Q(t_1) = (0, 1)$.

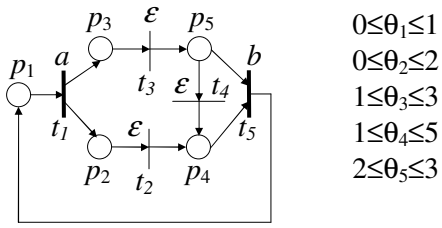

Fig. 1. The TPN considered in Example 6.

Class C_1 is labeled with marking $M_1 = [0\ 1\ 1\ 0\ 0]^T$ resulting from the firing of t_1 at M_0. Moreover, at marking M_1 only two transitions, namely t_2 and t_3, are enabled. Therefore this class is characterized by the timing constraints $\Theta_1 = \{0 \leq \theta_2 \leq 2,\ 1 \leq \theta_3 \leq 3\}$ that result from the fact that t_2 and t_3 are not enabled at M_0, and it is $Q(t_2) = (0, 2)$, $Q(t_3) = (1, 3)$.

Class C_3 is labeled with marking $M_3 = [0\ 1\ 0\ 0\ 1]^T$ resulting from the firing of t_3 at M_1, and with the set of timing constraints $\Theta_3 = \{0 \leq 0 - \Delta_3 \leq \theta_2 \leq 2 - \Delta_3,\ 1 \leq \theta_4 \leq 5\}$ with $\Delta_3 \in [1, 2]$. Indeed, t_2 is also enabled at the class C_1 from which C_3 is reached, while t_4 is not enabled at C_1, and it is $Q(t_4) = (1, 5)$.

Moreover, according to Algorithm 1, each edge of the MSCG is labeled as "$t_i, \mathcal{L}(t_i), \Delta_i \in [max\{0, l_i^k\}, min_{j:\ t_j \in \mathcal{A}(M_k)}\{u_j^k\}]$". As an example, the edge directed from C_1 to C_3 is labeled "$t_3, \varepsilon, \Delta_3 \in [1, 2]$", since the minimum of the upper bounds of the set of transitions logically enabled at C_1 is 2. In fact, either t_3 will fire before 2 time instants elapse after C_1 is reached, or it will never fire from C_1 (since the upper bound of t_2 is 2).

Analogously the edge directed from C_3 to C_5 is labeled "$t_2, \varepsilon, \Delta_2 \in [max\{0, 0 - \Delta_3\}, min\{(2 - \Delta_3, 5\}]$". Note that instead of writing $max\{0, 0 - \Delta_3\}$ we can obviously simply write 0 since by definition it is $\Delta_3 \geq 0$. We preferred to leave it like that only to better explain the procedure for the MSCG construction. ■

***Remark** 7:* Consider a generic class C_k labeled with marking M_k. If M_k is not a deadlock marking, i.e, if there exists at least one transition that is logically enabled at M_k, then for sure there exists at least one transition $t_i \in \mathcal{A}(M_k)$ that satisfies the *if* condition in Step 4. This can be intuitively explained by simply observing that timing constraints on the transition firings impose a sort of priority among transitions, that may also end up in transition disabling, however they cannot induce deadlock. ■

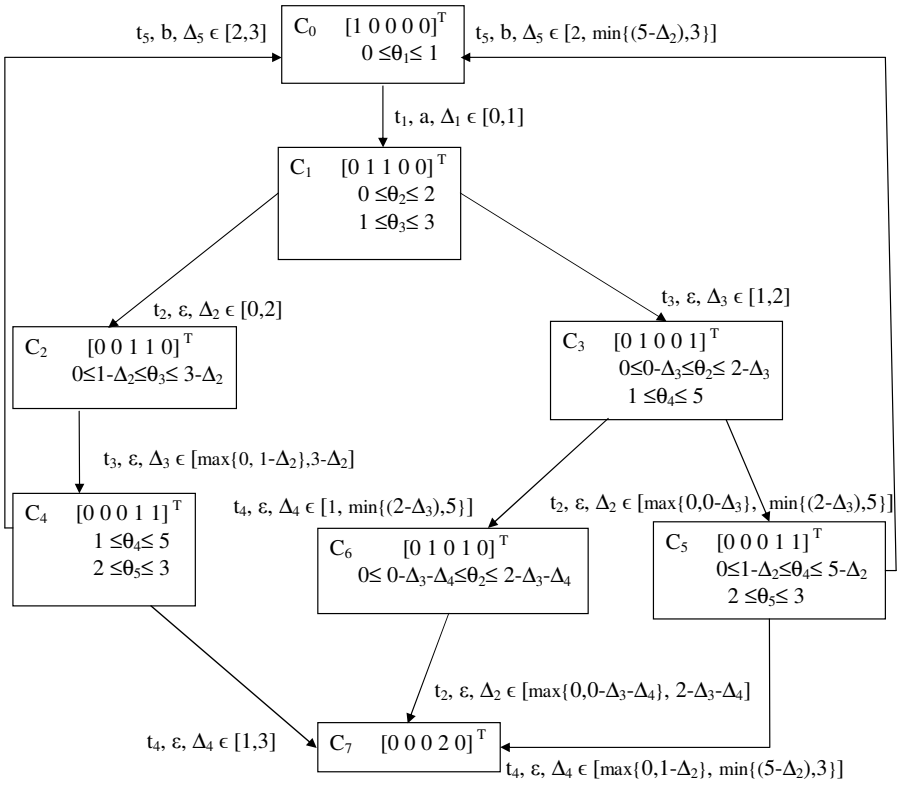

Fig. 2. The MSCG relative to the TPN in Fig. 1 assuming $M_0 = [1\ 0\ 0\ 0\ 0]^T$.

V. State estimation of a Time Petri net

We now present a procedure based on the MSCG to compute the set of states $\mathcal{S}(\delta_o, \tau)$ defined as in eq. (3), consistent with an observation $\delta_o = (\gamma_{i_1}, \tau_1)(\gamma_{i_2}, \tau_2)\cdots(\gamma_{i_h}, \tau_h) \in (L \times \mathbb{R}_0^+)^*$ and a time instant $\tau \geq 0$. With no ambiguity in the notation we extend the log operator to TLSs.

Such a procedure consists in the following three main steps.

- First, we compute the set of paths in the MSCG that are *logically* consistent with the observation δ_o, namely those paths to which it corresponds a sequence of labels equal to $log(\delta_o)$.
- Then, a set of linear constraints is associated with such paths to select those paths that are also *timing* consistent with the given δ_o and the given time τ, i.e., such that the observation of labels may occur at the same time instants as in δ_o and no further observation has occurred until time τ. A path is consistent if and only if the corresponding set is feasible.
- Each consistent path ends in a class that identifies a *consistent marking*. Moreover, the set of inequalities associated with transitions enabled at such a marking allows us to fully identify the corresponding *consistent states* as explained in the following Proposition 8.

To explain the above steps in detail we introduce some preliminary notation and prove a key result.

Given a class C_q we denote $out(C_q)$ the set of transitions associated with edges that exit from C_q. Therefore, if $\delta_o = (\gamma_{i_1}, \tau_1)(\gamma_{i_2}, \tau_2)\ldots(\gamma_{i_h}, \tau_h) \in (L \times \mathbb{R}_0^+)^*$ is a given TLS, then $log(\delta_o) = \gamma_{i_1}\gamma_{i_2}\ldots\gamma_{i_h}$ denotes the "logic" sequence of labels associated with δ_o, neglecting the time instants at which labels have been observed.

Now, let

$$\pi = \begin{array}{ll} C_{q_0} & \xrightarrow{\begin{array}{l} t_{i_1}, \mathcal{L}(t_{i_1}), \\ \Delta^{(1)} \in [max\{0, l_{i_1}^{q_0}\}, min_{j:\ t_j \in \mathcal{A}(M_{q_0})}\{u_j^{q_0}\}] \end{array}} \\ C_{q_1} & \xrightarrow{\begin{array}{l} t_{i_2}, \mathcal{L}(t_{i_2}), \\ \Delta^{(2)} \in [max\{0, l_{i_2}^{q_1}\}, min_{j:\ t_j \in \mathcal{A}(M_{q_1})}\{u_j^{q_1}\}] \end{array}} \\ \quad \ldots \quad \ldots & \\ C_{q_{k-1}} & \xrightarrow{\begin{array}{l} t_{i_k}, \mathcal{L}(t_{i_{k-1}}), \\ \Delta^{(k)} \in [max\{0, l_{i_k}^{q_{k-1}}\}, \\ \quad min_{j:\ t_j \in \mathcal{A}(M_{q_{k-1}})}\{u_j^{q_{k-1}}\}] \end{array}} \\ C_{q_k} & \end{array} \tag{4}$$

be a generic path in the MSCG[2].

The following results holds.

***Proposition** 8:* Consider a path π defined as in equation (4), starting from the root node of the MSCG and terminating in a class C_{q_k}, that is logically consistent with a given TLS $\delta_o = (\gamma_{i_1}, \tau_1)(\gamma_{i_2}, \tau_2)\ldots(\gamma_{i_h}, \tau_h) \in (L \times \mathbb{R}_0^+)^*$, where it is $k \geq h$. Consider a time instant $\tau \geq \tau_h$. Associate with π the following set of constraints:

$$\begin{cases} \displaystyle\sum_{l=1}^{k} \Delta^{(l)} \leq \tau, & (a1) \\ \tau - \displaystyle\sum_{l=1}^{k} \Delta^{(l)} < \min\limits_{r: t_r \in out(C_{q_k})} \{u_r^{q_k}\}, & (a2) \\ \Delta^{(l)} \geq \max\{0, l_{i_l}^{q_{l-1}}\}, \quad l = 1, \ldots, k, & (b1) \\ \Delta^{(l)} \leq \min\limits_{j: t_j \in \mathcal{A}(M_{q_{l-1}})} \{u_j^{q_{l-1}}\}, \quad l = 1, \ldots, k, & (b2) \\ \displaystyle\sum_{l=1}^{q} \Delta^{(l)} = \text{given number}, \quad \forall t_{i_q} \in log(\delta_o), & (c) \end{cases} \tag{5}$$

where the meaning of "given number" in Constraints (c) is explained in detail in the following proof. If the above set of constraints is feasible then the corresponding path π is also *timing consistent* with δ_o. Moreover, if (M_k, Θ_k) is the state identified by class C_{q_k} where

$$\Theta_k = \{l_j^{q_k} \leq \theta_j \leq u_j^{q_k}, \quad t_j \in \mathcal{A}(M_k)\}, \tag{6}$$

then it is $(M_k, \Theta_k') \in \mathcal{S}(\delta_o, \tau)$ where:

$$\Theta_k' = \{\bar{l}_j \leq \theta_j \leq \bar{u}_j, \quad t_j \in \mathcal{A}(M_k)\ \}, \tag{7}$$

and

$$\begin{array}{l} \bar{l}_j = \max\{0, l_j^{q_k} - (\tau - \sum_{l=1}^{k} \Delta^{(l)})\}, \\ \bar{u}_j = \max\{0, u_j^{q_k} - (\tau - \sum_{l=1}^{k} \Delta^{(l)})\} \end{array} \tag{8}$$

are subject to constraints (5).

Proof: The consistency of the path with respect to time follows from the physical meaning of the constraints, that can be explained as follows.

— Constraints (a) impose limitations on the total length of the observation. In particular, (a1) implies that, if the observation really corresponds to the considered path, then the class at the end of the path, namely C_{q_k}, is reached at a time that is smaller than or equal to the given time instant τ. Now, since $\Delta^{(l)}$ denotes the time interval to go from $C_{q_{l-1}}$ to C_{q_l}, the sum in the left hand side of (a1) is equal to the time spent to reach class C_{q_k}. Constraint (a2) imposes that class C_{q_k} is the last visited class. This is verified if the difference between τ and the time spent to reach class C_{q_k} is less than the minimum of the upper bounds of the output transitions of class C_{q_k}. Indeed, once the node C_{q_k} is reached a transition enabled at M_{q_k} will fire at most after a time interval equal to the minimum of the upper bounds of the transitions enabled at C_{q_k}.

— Constraints (b) simply imposes the limitations on the time intervals $\Delta^{(l)}$ as given in the MSCG.

— Constraints (c) imply that the observable transitions actually occur in accordance to the observation. Note that we do not specify the value of the "given number" to avoid heavy notation that would be required to completely formalize this. We clarify this via a simple example. Let

$$\delta_o = (a, 1)(b, 5)(c, 8)$$

and let

$$\begin{array}{r} \sigma = (\varepsilon_4, \Delta^{(1)})(t_1, \Delta^{(2)})(\varepsilon_5, \Delta^{(3)})(t_2, \Delta^{(4)}) \\ (\varepsilon_6, \Delta^{(5)})(t_3, \Delta^{(6)}) \end{array}$$

be the sequence corresponding to the considered path, where $\mathcal{L}(t_1) = a$, $\mathcal{L}(t_2) = b$, and $\mathcal{L}(t_3) = c$. Constraints

[2]A remark should be done concerning the above notation and the notation used in the previous section. Indeed, if we look at Fig. 2 the time interval associated with the generic transition t_k is denoted as Δ_k in all the edges in which it appears. Obviously, when a path is followed in the graph, the same edge may be visited multiple times. Whenever this occurs multiple variables should be associated with it, since the time intervals spent in the input class is usually not the same whenever it is visited. Analogously, paths visiting different edges characterized by the same transition label require different variables to denote the time interval spent in the input class.

(c) take the following form:

$$\begin{aligned}&\Delta^{(1)}+\Delta^{(2)}=1,\\&\Delta^{(1)}+\Delta^{(2)}+\Delta^{(3)}+\Delta^{(4)}=5,\\&\Delta^{(1)}+\Delta^{(2)}+\Delta^{(3)}+\Delta^{(4)}+\Delta^{(5)}+\Delta^{(6)}=8.\end{aligned}$$

Thus, in such a case the "given number" is 1 for constraint (c1), 5 for constraint (c2), and 8 for constraint (c3).

Now, if the path is both logically and timing consistent, then the marking M_k associated with the last class in the path obviously belongs to $\mathcal{C}(\delta_o,\tau)$, i.e., it is consistent with the observation δ.

Finally, by definition, the set of constraints associated with transitions enabled at a given class, tell us when a transition may fire as soon as the class is reached. Therefore, Θ_k provides the set of timing constraints that transitions enabled at M_k should satisfy as soon as class C_{q_k} is reached. Now, at time τ, namely when $\tau-\sum_{l=1}^{k}\Delta^{(l)}$ time instants have elapsed after reaching M_k, the residual times of all the enabled transitions are reduced by an amount equal to $\tau-\sum_{l=1}^{k}\Delta^{(l)}$, thus explaining the new lower and upper bounds in equation (8). □

Constraints (5) are clearly nonlinear. However, they can be easily linearized leading to the following result.

***Corollary** 9:* Consider a path π defined as in equation (4), starting from the root node of the MSCG and terminating in class C_{q_k}, that is logically consistent with a given TLS $\delta_o=(\gamma_{i_1},\tau_1)(\gamma_{i_2},\tau_2)\dots(\gamma_{i_h},\tau_h)\in(L\times\mathbb{R}_0^+)^*$. Consider a time instant $\tau\geq\tau_h$. Associate with π the following set of constraints:

$$\begin{cases}\displaystyle\sum_{l=1}^{k}\Delta^{(l)}\leq\tau, & (a1')\\ \displaystyle\tau-\sum_{l=1}^{k}\Delta^{(l)}<u_r^{q_k}, \quad \forall t_r\in out(C_{q_k}), & (a2')\\ \Delta^{(l)}\geq l_{i_l}^{q_{l-1}}, \quad l=1,\dots,k, & (b1')\\ \Delta^{(l)}\geq 0, \quad l=1,\dots,k, & (b2')\\ \Delta^{(l)}\leq u_j^{q_{l-1}}, \quad l=1,\dots,k, \quad \forall j: t_j\in\mathcal{A}(M_{q_{l-1}}), & (b3')\\ \displaystyle\sum_{l=1}^{q}\Delta^{(l)}=\text{ given number}, \quad \forall t_{i_q}\in log(\delta_o). & (c')\end{cases} \tag{9}$$

If the above set of constraints is feasible then the corresponding path π is also *timing consistent* with δ_o. Moreover, if (M_k,Θ_k) is the state identified by class C_{q_k} where

$$\Theta_k=\{l_j^{q_k}\leq\theta_j\leq u_j^{q_k}, \quad t_j\in\mathcal{A}(M_k)\}, \tag{10}$$

then it is $(M_k,\Theta_k')\in\mathcal{S}(\delta_o,\tau)$ where:

$$\Theta_k'=\{\bar{l}_j\leq\theta_j\leq\bar{u}_j, \quad t_j\in\mathcal{A}(M_k)\ \} \tag{11}$$

and

$$\begin{aligned}\bar{l}_j&=\max\{0,l_j^{q_k}-(\tau-\textstyle\sum_{l=1}^{k}\Delta^{(l)})\},\\ \bar{u}_j&=\max\{0,u_j^{q_k}-(\tau-\textstyle\sum_{l=1}^{k}\Delta^{(l)})\}\end{aligned} \tag{12}$$

are subject to constraints (9). ■

A. State estimation algorithm

In this subsection we provide an algorithm of state estimation based on the above results.

Note that given a path π in the MSCG, we denote as $obs(\pi)$ the sequence of labels associated with it. Therefore, for the path π in equation (4) it is $obs(\pi)=\mathcal{L}(t_{i_1}t_{i_2}\dots t_{i_k})$.

Algorithm 2 computes the set of states that are consistent with a timed observation and it can be explained as follows. If no label has been observed, then it is $\delta_o=\varepsilon$. We compute all paths that correspond to a null observation. Note that a path may also have one node and no edge, therefore such a set surely includes the path only containing the root node. However, there may also be other paths consistent with the null observation at time 0. Steps 2, 3 and 4 look for them. At Step 3 the feasibility of the set of constraints (9) is checked for any path that is logically consistent with the observation ε to establish if it is also consistent with respect to the time.

Note that the paths we need to evaluate at Step 3 are always in a finite number thanks to Assumption A3 that guarantees that we have no cycle of unobservable transitions that can fire in a null interval of time.

A similar reasoning can be applied to explain the remaining steps.

***Remark** 10:* The proposed approach, despite of similar approaches in the literature [**?**], does not require that the unobservable subnet is acyclic. It simply requires that cycles of unobservable transitions cannot fire in a time interval of null length (see Assumption A3). Indeed, as already mentioned in the explanation of Algorithm 2, under such an assumption, for any finite value of τ we have to check the feasibility of a finite number of constraints sets. ■

***Example** 11:* Let us consider again the TPN system and its MSCG shown respectively in Figures 1 and 2, and introduced in Example 6.

Let $\delta_o=(a,1)(b,5)(a,5)$ and $\tau=5.5$. We want to compute the set of consistent states $\mathcal{S}(\delta_o,\tau)$. The paths in the MSCG that start from the initial node and whose "logic" label has an observable projection equal to $log(\delta_o)$ are: $t_1t_2t_3t_5t_1$, $t_1t_3t_2t_5t_1$, $t_1t_2t_3t_5t_1t_2$,

Algorithm 2: State Estimation using the MSCG

input : A labeled TPN system, its MSCG, a TLS $\delta_o = (\gamma_{i_1}, \tau_1)\dots(\gamma_{i_h}, \tau_h)$, and a time instant $\tau \geq \tau_h$.

output: Set of consistent states $\mathcal{S}(\delta_o, \tau)$.

Initialize: Let $\delta_o = \varepsilon$ and $\mathcal{S}(\varepsilon, 0) = \emptyset$.

Determine all paths π that start from the initial node and such that $obs(\pi) = \varepsilon$. Let $\tilde{\Pi}$ be the set containing all such paths.

Select those paths $\pi \in \tilde{\Pi}$ such that constraints (9) are feasible for $\tau = 0$. Let Π be the set containing all such paths.

for *all* $\pi \in \Pi$, **do**

  let $\mathcal{S}(\varepsilon, 0) = \mathcal{S}(\varepsilon, 0) \cup \{(M, \Theta)\}$, where (M, Θ) is the state associated with the last node of π.

if *we are interested in* $\mathcal{S}(\varepsilon, \tau)$ *for a given* $\tau > 0$, **then**

  let $\mathcal{S}(\varepsilon, \tau) = \emptyset$.

  Select those paths $\pi \in \tilde{\Pi}$ such that constraints (9) are feasible for the given $\tau > 0$. Let Π_τ be the set containing all such paths.

  for *all* $\pi \in \Pi_\tau$, **do**

    let $\mathcal{S}(\varepsilon, \tau) = \mathcal{S}(\varepsilon, \tau) \cup \{(M, \Theta')\}$, where M is the marking associated with the last node of π and Θ' is defined as a function of the constraints Θ in the last node according to equations (11)–(12) and constraints (9).

Wait until a new observation $(e, \bar{\tau})$ occurs.

Let $\delta_o' = \delta_o$, $\delta_o = \delta_o'(e, \bar{\tau})$, $\Pi' = \Pi$, $\mathcal{S}(\delta_o, \bar{\tau}) = \emptyset$.

Select those paths in Π' that can be continued in an extended path π such that $obs(\pi) = log(\delta_o)$. Let $\tilde{\Pi}$ be the set containing all such extended paths.

Select those paths $\pi \in \tilde{\Pi}$ such that constraints (9) are feasible for $\tau = \bar{\tau}$. Let Π be the set containing all such paths.

for *all* $\pi \in \Pi$, **do**

  let $\mathcal{S}(\delta_o, \bar{\tau}) = \mathcal{S}(\delta_o, \bar{\tau}) \cup \{(M, \Theta)\}$, where (M, Θ) is the state associated with the last node of π.

if *we are interested in* $\mathcal{S}(\delta_o, \tau)$ *for a given* $\tau > \bar{\tau}$, **then**

  let $\mathcal{S}(\delta_o, \tau) = \emptyset$,

  Select those paths $\pi \in \tilde{\Pi}$ such that constraints (9) are feasible for the given $\tau > \bar{\tau}$. Let Π_τ be the set containing all such paths.

  for *all* $\pi \in \Pi_\tau$, **do**

    let $\mathcal{S}(\delta_o, \tau) = \mathcal{S}(\delta_o, \tau) \cup \{(M, \Theta')\}$, where M is the marking associated with

$t_1t_2t_3t_5t_1t_3$, $t_1t_3t_2t_5t_1t_2$, $t_1t_3t_2t_5t_1t_3$, $t_1t_2t_3t_5t_1t_2t_3$, $t_1t_2t_3t_5t_1t_3t_4$, $t_1t_2t_3t_5t_1t_3t_2$, $t_1t_3t_2t_5t_1t_2t_3$, $t_1t_3t_2t_5t_1t_3t_4$, $t_1t_3t_2t_5t_1t_3t_2$, $t_1t_2t_3t_5t_1t_2t_3t_4$, $t_1t_2t_3t_5t_1t_3t_4t_2$, $t_1t_2t_3t_5t_1t_3t_2t_4$, $t_1t_3t_2t_5t_1t_2t_3t_4$, $t_1t_3t_2t_5t_1t_3t_4t_2$, $t_1t_3t_2t_5t_1t_3t_2t_4$. [Note that for simplicity of notation we omitted here the classes and the edges along the path.]

Consider as an example the path $t_1t_2t_3t_5t_1$ that can be rewritten as $C_0 \xrightarrow{t_1, \Delta^{(1)} \in [0,1]} C_1 \xrightarrow{t_2, \Delta^{(2)} \in [0,2]} C_2 \xrightarrow{t_3, \Delta^{(3)} \in [max\{0, 1-\Delta^{(2)}\}, 3-\Delta^{(2)}]} C_4 \xrightarrow{t_5, \Delta^{(4)} \in [2,3]} C_0 \xrightarrow{t_1, \Delta^{(5)} \in [0,1]} C_1$. The set of constraints associated with this path is:

$$
\begin{cases}
\left.\begin{array}{l} \Delta^{(1)} + \Delta^{(2)} + \Delta^{(3)} + \Delta^{(4)} + \Delta^{(5)} \leq 5.5 \\ 5.5 - (\Delta^{(1)} + \Delta^{(2)} + \Delta^{(3)} + \Delta^{(4)} + \Delta^{(5)}) < 2 \end{array}\right\} (a) \\
\left.\begin{array}{l} \Delta^{(1)} \geq 0 \\ \Delta^{(1)} \leq 1 \\ \Delta^{(2)} \geq 0 \\ \Delta^{(2)} \leq 2 \\ \Delta^{(3)} \geq 0 \\ \Delta^{(3)} \geq 1 - \Delta_2 \\ \Delta^{(3)} \leq 3 - \Delta_2 \\ \Delta^{(4)} \geq 2 \\ \Delta^{(4)} \leq 3 \\ \Delta^{(5)} \geq 0 \\ \Delta^{(5)} \leq 1 \end{array}\right\} (b) \\
\left.\begin{array}{l} \Delta^{(1)} = 1 \\ \Delta^{(1)} + \Delta^{(2)} + \Delta^{(3)} + \Delta^{(4)} = 5 \\ \Delta^{(1)} + \Delta^{(2)} + \Delta^{(3)} + \Delta^{(4)} + \Delta^{(5)} = 5 \end{array}\right\} (c)
\end{cases}
$$

The existence of a solution has been tested using LINDO package[3], introducing a fictitious objective function (we used $\min \sum_l \Delta^{(l)}$) to obtain a LPP formulation. In particular, the solution we found is: $\Delta^{(1)} = 1, \Delta^{(2)} = 1, \Delta^{(3)} = 0, \Delta^{(4)} = 3$, and $\Delta^{(5)} = 0$. If we do the same for all "logic" consistent paths listed above we find that the set of constraints is feasible only for paths: $t_1t_2t_3t_5t_1$, $t_1t_3t_2t_5t_1$, $t_1t_2t_3t_5t_1t_2$, $t_1t_3t_2t_5t_1t_2$. Obviously, once we find that one path is not consistent, also all its continuations will not be consistent, so we discard them a priori. In fact, there is no possibility that a path, that is the continuation of an unfeasible path, is consistent.

Once all consistent paths are selected, we can compute the set of consistent states, i.e., the states that can be reached following such paths. In the case at hand it is $\mathcal{S}(\delta_o, 5.5) = \{(M_1, \Theta_1), (M_2, \Theta_2)\}$, where $\Theta_1 = \{(0 \leq$

[3]LINDO website: http://www.lindo.com/

$\theta_2 \leq 1.5), (0.5 \leq \theta_3 \leq 2.5)\}$ and $\Theta_2 = \{(0 \leq 0.5 - \Delta^{(2)} \leq \theta_3 \leq 2.5 - \Delta^{(2)}), \quad \Delta^{(2)} \in [0, 0.5]\}$. ■

Note that, the state estimation of a timed PN could also be done online, without using the MSCG, computing all information each time an event is observed. However, the construction of the MSCG is important because it allows to move offline the most burdensome part of the computations.

VI. Complexity of the procedure

In this section we analyze the complexity of each step of the proposed procedure.

1) *Construction of the MSCG.*
   The number of nodes of the MSCG increases exponentially with the system complexity (net structure, and number of tokens in the initial marking).
2) *Computation of the paths in the MSCG that are logically and timing consistent with the observation δ_o and the given time τ.*
   Our approach first requires the computation of all paths that are logically consistent with the observation δ_o. Now, in the case of cycles of unobservable transitions, there may also be paths of infinite length. However, as discussed above, thanks to Assumption (A3), given a finite value of τ, only finite paths may also be timing consistent with the observation. We denote l_{max} the length of the longest path in the MSCG that is logically consistent with the observation δ_o and the given value of τ. The complexity of finding all such paths is $\mathcal{O}(|T|^{l_{max}})$. Indeed we need to analyze all the output arcs from each node in the path. To establish if the above paths are timing consistent with all the labels in δ_o we associate with each path a set of linear constraints of the form (9), and evaluate if they are feasible. This can be done solving, for each path, a LPP with a number of constraints at most equal to $1+|T|+3l_{max}|T|+l_{max}$. In particular, one constraint is of type (a1'). The number of constraints of type (a2') is equal to the number of output transitions of the last node in the path, that is at most equal to $|T|$. Constraints of type (b1'), (b2') and (b3') are each in a number equal at most to $l_{max}|T|$. The number of constraints of type (c') is equal to h since one constraint is associated with each observable transition in the path. Therefore, in the worst case the number of constraints of type (c') is equal to the length of the path, that cannot be greater than l_{max}. Finally, the number of unknowns of each LPP is equal to k, i.e., to the length of the path that cannot be greater than l_{max}.

Note that our approach is based on the solution of some LPPs that as we know can be solved using polynomial time algorithms. Moreover there are many softwares, such as CPLEX, that allow to compute LPPs in a very efficient way. However, if the system evolution is very fast compared with the time required to solve the LPPs, it may be impossible to use our approach. Obviously, this is a limitation of the procedure but it is common to all approaches that require online computations.

VII. Fault diagnosis of a Time Petri net

In this section we address the problem of fault diagnosis introduced in Subsection III-B, namely we show how to compute the diagnosis state relative to an observation δ_o, a time instant τ, and a fault class $t_f \in T_f^i$.

The following corollary shows how the diagnosis states can be computed using the MSCG.

***Corollary** 12:* Consider an observation $\delta_o \in (L \times \mathbb{R}_0^+)^*$, a time instant $\tau \geq 0$, and a fault class T_f^i.

- $\Gamma(\delta_o, \tau, T_f^i) = N$ if for all paths π in the MSCG starting from the root node such that $obs(\pi) = log(\delta_o)$ and such that the corresponding set of constraints (9) is feasible, it is $t_f \notin \pi$ for all $t_f \in T_f^i$.
- $\Gamma(\delta_o, \tau, T_f^i) = U$ if:
  (i) there exists a path π in the MSCG starting from the root node such that $obs(\pi) = log(\delta_o)$, the corresponding set of constraints (9) is feasible, and there exists at least a fault transition $t_f \in T_f^i$ such that $t_f \in \pi$ but
  (ii) there also exists another path π' in the MSCG starting from the root node, such that $obs(\pi') = log(\delta_o)$, the corresponding set of constraints (9) is feasible, and for all $t_f \in T_f^i$ it is $t_f \notin \pi'$.
- $\Gamma(\delta_o, \tau, T_f^i) = F$ if for all paths π in the MSCG starting from the root node, such that $obs(\pi) = log(\delta_o)$ and such that the corresponding set of constraints (9) is feasible, there exists a transition $t_f \in T_f^i$ such that $t_f \in \pi$.

Proof: It simply follows from Definition 4 of diagnosis states and the definition of MSCG. Indeed, if a path is such that $obs(\pi) = log(\delta_o)$ then the path is logically consistent with the observation. The feasibility of constraints (9) implies that it is also consistent with respect to time, i.e., the TLS associated with the path belongs to the set $\Sigma(\delta_o, \tau)$. □

Algorithm 3 summarizes how to perform online fault diagnosis using the MSCG and can be explained as follows. The state corresponding to the root node is surely consistent with the observation ε at $\tau = 0$. If

Algorithm 3: Diagnosis using the MSCG

input : A labeled TPN system, its MSCG, a TLS $\delta_o = (\gamma_{i_1}, \tau_1) \dots (\gamma_{i_h}, \tau_h)$, and a time instant $\tau \geq \tau_h$.

output: Diagnosis states $\Gamma(\delta_o, \tau, T_f^i)$, with $i = 1, \dots, r$.

1 **Initialize:** Let $\delta_o = \varepsilon$, $\Gamma(\varepsilon, 0, T_f^i) = N$, $i = 1, \dots, r$, and $\Pi = \emptyset$.

2 Determine all paths π that start from the initial node and such that $\pi_o = \varepsilon$. Let $\tilde{\Pi}$ be the set containing all such paths.

3 Select those paths $\pi \in \tilde{\Pi}$ such that constraints (9) are feasible for $\tau = 0$. Let Π be the set containing all such paths.

4 **for** *all* $i = 1, \dots, r$, **do**
  if $\exists \pi \in \Pi$ *and* $\exists t_f \in T_f^i$ *such that* $t_f \in \pi$, **then**
    $\Gamma(\varepsilon, 0, T_f^i) = U$.

5 Wait until a new observation $(e, \bar{\tau})$ occurs.

6 Let $\delta_o' = \delta_o$, $\delta_o = \delta_o'(e, \bar{\tau})$, $\Pi' = \Pi$.

7 Select those paths in Π' that can be continued in an extended path π such that $obs(\pi) = log(\delta_o)$. Let $\tilde{\Pi}$ be the set containing all such extended paths π.

8 Select those paths $\pi \in \tilde{\Pi}$ such that constraints (9) are feasible for $\tau = \bar{\tau}$. Let Π be the set containing all such paths π.

9 **for** *all* $i = 1, \dots, r$, **do**
  if $\forall\ \pi \in \Pi \wedge \forall t_f \in T_f^i$ *it is* $t_f \notin \pi$, **then**
    $\Gamma(\delta_o, \tau, T_f^i) = N$;
  else if $\exists\ \pi \in \Pi$ *and* $\pi' \in \Pi$ *s.t.: (i)* $\exists t_f \in T_f^i$ *such that* $t_f \in \pi$, *(ii)* $\forall t_f \in T_f^i$, $t_f \notin \pi'$, **then**
    $\Gamma(\delta_o, \tau, T_f^i) = U$;
  else if $\forall\ \pi \in \Pi\ \exists t_f \in T_f^i$ *such that* $t_f \in \pi$, **then**
    $\Gamma(\delta_o, \tau, T_f^i) = F$.

10 Go to Step 5.

it is the only consistent state then the diagnosis state is set equal to N. If there exists some other consistent state that is reachable at $\tau = 0$ via a sequence containing some fault transition in a given class, then the diagnosis state relative to such a class is set equal to U. Steps 2 and 3 check if other consistent states exist. Step 4 eventually updates the values of the diagnosis states initialized at N in Step 1.

Steps 7, 8 and 9 are relative to a generic observation. In particular, Steps 7 and 8 look for the paths that are consistent with the observation considered at Step 5. Step 9 computes the values of the diagnosis states accordingly using Corollary 12.

***Example** 13:* Let us consider again the TPN system and its MSCG shown respectively in Figures 1 and 2, and introduced in Example 6. Let us consider one single fault class $T_f = \{t_4\}$.

Let $\delta_o = (a, 1)(b, 5)(a, 5)$ and $\tau = 5.5$. We want to compute the diagnosis state for such time observation and for such time instant. The set of paths that are consistent with δ_o and τ have been already computed in Example 13 and are: $t_1t_2t_3t_5t_1$, $t_1t_3t_2t_5t_1$, $t_1t_2t_3t_5t_1t_2$, $t_1t_3t_2t_5t_1t_2$. Since none path contains the fault transition t_4, then it is $\Gamma(\delta_o, 5.5, T_f) = N$. ■

VIII. Conclusions and future work

In this paper we have first presented an algorithm for the construction of a graph, called *Modified State Class Graph* (MSCG), that collects in a compact form the main information on all the possible evolutions of a given labeled TPN system initialized at a given marking. Then, given a timed observation, i.e., a sequence of observed labels with the corresponding time instants of observation, and a given time instant, we have illustrated a procedure based on the exploration of the MSCG and on the solution of some integer linear programming problems that allows one to determine which states are consistent with the observation. Finally, we have given a procedure that uses the modified state class graph to associate with each time observation a diagnosis state.

As a future research we plan to investigate the problem of diagnosability of a labeled TPN. Some preliminary results on this topic have been presented in [**?**].

Francesco Basile Francesco Basile was born in Naples, Italy, in 1971. He received the Laurea degree in Electronic Engineering in 1995 and the Ph.D. degree in Electronic and Computer Engineering in 1999 from the University of Naples. In 1999 he was visiting researcher for six months at the Departamento de Ingenieria Informatica y Systemas of the University of Saragoza, Spain. He is currently assistant professor of Automatic Control at the Dipartimento di Ingegneria Elettronica ed Ingegneria Informatica of the University of Salerno, Italy. He has published more than 90 papers on international journals and conferences. He has been member of the editorial board of International Journal of Robotics and Automation. He is a member of the editorial board of IEEE Transactions on Control Systems Technology and of IEEE Control System Society Conference Editorial Board. His current research interest are: modelling and control of discrete event systems, automated manufacturing and robotic. He is IEEE Senior member since November 2011.

Maria Paola Cabasino Maria Paola Cabasino received the Laurea degree in electronic engineering and the Ph.D. degree in electronic and computer engineering, both from the University of Cagliari, Cagliari, Italy, in 2005 and 2009, respectively. She is a Post doctoral researcher of Automatic Control at the Department of Electrical and Electronic Engineering of the University of Cagliari. She has been a visiting researcher at the University of Illinois (Urbana-Champaign, IL, USA), University of Michigan (Ann Arbor, MI, USA), Universidad de Zaragoza (Spain) and Indiana University Purdue University Indianapolis (Indianapolis, IN, US). She was an instructor at Indiana University Purdue University Indianapolis during the Spring semester 2014. Her research interests are based on discrete event systems, automata, Petri nets, state estimation, diagnosis, identification, supervisory control. She has been the quality manager of the European project FP7-ICT2-3.7 DISC - Distributed Supervisory Control of Large Plants (2008-11). She has been member of the International Program Committee of the 2nd IFAC Conf. on the Analysis and Design of Hybrid Systems (ADHS'06) and of the 18th IEEE Int. Conf. on Emerging Technology and Factory Automation (ETFA2013). She has published 14 international journal papers, 7 book chapters and 35 international conference papers.

Carla Seatzu Carla Seatzu received the Laurea degree in electrical engineering and her Ph.D. degree in electronic engineering and computer science from the University of Cagliari, Italy, in 1996 and 2000, respectively. In 2002 she joined the Department of Electrical and Electronic Engineering of the University of Cagliari as an assistant professor of Automatic Control. She currently serves as an associate professor of Automatic Control in the same Department. Carla Seatzu's research interests include discrete-event systems, hybrid systems, Petri nets, manufacturing systems, networked control systems, and control of mechanical systems. She has published 190+ publications, including one textbook and 50+ international journals. Carla Seatzu served as General Co-Chair of the 18th IEEE Int. Conf. on Emerging Technology and Factory Automation (ETFA2013). She was Chair of the National Organizing Committee of the 2nd IFAC Conf. on the Analysis and Design of Hybrid Systems (ADHS'06) and member of the International Program Committee of over 50 international conferences. She is associate editor of IEEE Trans. on Automatic Control, IEEE Trans. on Automation, Science and Engineering, and Nonlinear Analysis: Hybrid Systems. She is associate editor of the Conference Editorial Board of the IEEE Control System Society and of the IEEE Robotics and Automation Society; she is chair of the IEEE IES Technical Committee on Factory Automation - Subcommittee on Industrial Automated Systems and Controls.