Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 835312, 21 pages
http://dx.doi.org/10.1155/2014/835312

Research Article

Identification of Stochastic Timed Discrete Event Systems with st-IPN

Doyra Mariela Muñoz,[1] Antonio Correcher,[2] Emilio García,[2] and Francisco Morant[2]

[1] *Grupo de Automática Industrial, Universidad del Cauca, Popayán, Colombia*
[2] *Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera Street, s/n, 46022 Valencia, Spain*

Correspondence should be addressed to Doyra Mariela Muñoz; mamunoz@unicauca.edu.co

Received 18 October 2013; Revised 14 April 2014; Accepted 28 April 2014; Published 9 July 2014

Academic Editor: Yingwei Zhang

This paper presents a method for the identification of stochastic timed discrete event systems, based on the analysis of the behavior of the input and output signals, arranged in a timeline. To achieve this goal stochastic timed interpreted Petri nets are defined. These nets link timed discrete event systems modelling with stochastic time modelling. The procedure starts with the observation of the input/output signals; these signals are converted into events, so that the sequence of events is the observed language. This language arrives to an identifier that builds a stochastic timed interpreted Petri net which generates the same language. The identified model is a deterministic generator of the observed language. The identification method also includes an algorithm that determines when the identification process is over.

1. Introduction

Industrial systems usually have a sequential evolution, so they behave as discrete event systems (DES). A DES is a discrete-state, event-driven system; that is, its state evolution depends entirely on the occurrence of asynchronously discrete events over time [1]. A DES describes the system behavior by means of the occurrence of events from an initial state. In an industrial system two kinds of events can be observed. Input (I) events are related to control commands or external interactions with the system and output (O) events are related to sensor measures. In this proposal, the external signals not related with control commands define the operating modes, which affects the controller and allows modelling the system under different strategies; in [2] the authors use a similar concept to differentiate production patterns in batch processes, called multimode; they propose to separate the original space of a mode into two different parts (the common and the specific) and a monitoring process is carried out in each block; the multiblock monitoring method proposed is used for fault diagnosis purposes in multimode multivariate continuous processes; some of its proposals could be applied in stochastic DES regarding times identification, so it will probably reduce the sizes of time matrices and the complexity of the identification method.

Regarding DES, one problem which is being studied recently is system identification. This problem can be defined as follows: given a set of observed ordered timed I/O signals, determine a model such that, given a set of ordered I signals, the model approximates a set of the observed O signals. When a model of the system is not available, identification can be effectively used to obtain a model that can be then used to formally prove if the system meets the requirements and to improve its dependability [3].

Therefore, the first step in an identification process is to define the characteristics and the format of the model to fit. DES modelling starting form I/O signals has been addressed by many authors using different approaches.

Petri nets (PNs) have been recognized as a suitable model to describe DES [4], particularly when a system is asynchronous [5–8]. Hiraishi [9] presented an algorithm for the construction of a free labeled Petri net model from the knowledge of a finite set of its firing sequences. In [10], the authors define a class of continuous-time discrete event dynamic systems (DEDS) with two types of discrete-valued I/O signals:

conditions signals and event signals called condition/event systems (C/E systems) and they define models of C/E systems too that are based on an extension of PNs (C/E PNs). C/E systems provide an intuitive modelling framework amenable to block diagram representation. The way of thinking and modelling a system is like a set of modules with a particular dynamic behavior and their interconnection via signals. This way of modeling is intuitive, and the modules can be pretailored and used over and over again. Each module is equipped with I/O signals which are of two types: condition I/O carrying state information and event I/O carrying state transition information. This way of system extension with I/O signals clearly reflects the duality of PNs, namely, the clear distinction between states and states transitions with their own graphical representation, semantics, and formal properties.

Rausch and Hanisch [11] proposed formalism for composition of I/O PNs to new systems which they called net condition/event systems (NCEs). I/O PNs are coupled by means of condition signals and event signals. The composition is performed in the same way as known from the composition of continuous systems in state space representation. This model can simulate the systems behavior by firing maximal steps and can also compute the complete state graph for the system, similar to the state graph for timed place/transition nets under the maximum firing strategy.

The general idea of NCEs is modelling a system as a set of modules with a particular dynamic behavior and their interconnection via signals [12]. Condition and event inputs can be connected with some transitions inside the module by condition and event arcs. Module places can be connected to the condition outputs by condition arcs, and transitions can be connected to the event outputs by event arcs. This concept provides a basis for a compositional approach to build larger models from smaller components [12].

Once the model has been defined, the following step is defined as the DES identification method. In the literature there exist a lot proposals related to DES identification methods, these can be compared by their complexity in terms of the system characteristics, identification process, and the model or algorithm applied [13], as well as by the identification model objective. From the theory of DES some identification methods have been developed using different techniques such as finite state machines (FSM), Automata, and PN; currently, the most popular models are PNs and Automata [7, 14].

The simplest methods are based on data from I/O, where concurrent states are not identified and its execution is off-line [13]. If the identification process is incremental, then the algorithm must be executed on-line, but its complexity is higher, because it can be polynomial or exponential, depending on the approach used for modeling. Furthermore, if the identification goal is the analysis of the model structure, the model must be developed to verify PN properties like being alive, conservative, achievable, among others. One of the most well-studied PNs problems is estimating the state of a given PN, based on its event sequence observation; in [15] the problem of finding the least-cost transition firing sequence(s) for a given interpreted PN (IPN) based on the observation of labels sequence is addressed. The PN possesses both observable transitions (which are associated with a possibly nonunique label) and unobservable transitions (whose firings do not generate any label observations). This paper assumes that each transition in the given IPN is associated with a nonnegative cost that captures its likelihood (e.g., in terms of the workload or the power amount needed to execute a certain transition). Given the observation of a labels sequence, the task is to find the transition firing sequence which (i) is consistent with both the observed label sequence and the PN structure and (ii) has the least total cost.

Recent literature presents results and trends to solve the identification problem. These trends can be grouped according to [13], in three groups: IPN identification by I/O signals (I/O IPN identification), nondeterministic finite automaton (NDFA), and application of integer linear programming (IPL) for IPN identification (ILP + IPN).

I/O IPN identification is related to the generation of DES, based on the observation of output signals [16]; an on-line algorithm computes an IPN model Q_i, describing the behavior of an unknown system Q. Every time a cyclic behavior is detected, the previous computed model is updated. This approach considers that each input to the system is reflected in the output. It means that, even if the system is not completely instrumented, the information provided by the sensors is enough to detect any change of state [17]. The algorithm receives a sequence of output signals obtained from observations during the system operation. These output signals must be binary vectors representing the current state of every one of the sensors measuring the output behavior of the system. In [17] the authors present an evolution of the method which is able to deal with concurrent systems by means of detecting concurrent transitions. The identified models are useful for structural diagnosis.

Regarding NDFA method, its approach is to build a model that approximates the original system language. The observed language of the closed loop DES is to be translated into a finite state machine by an appropriate identification algorithm. In [18, 19] a nondeterministic autonomous automaton with output (NDAAO) is chosen as an appropriate model to reproduce the observed language of closed loop DES. This approach identifies a closed loop DES based on the interaction of the controller with deterministic behavior and the physical process with nondeterministic behavior, combining internal and external events. It presents a proposal for division of the system into smaller subsystems. Also [20] addresses the problem of identification and diagnosis in DES based on timed Automata; the algorithm identifies and detects failures in the actuators in low complexity systems. An evolution of the method of [20] is proposed by Jarvis in [21]. This paper deals with the identification of timed event systems with timed Automata models. Some disadvantages of this trend are that it does not model the concurrence of states. Moreover, in complex systems there is a combinatorial explosion of states.

ILP + IPN uses an ILP approach and it computes an IPN. The model is built by using observed events and the available output vector ([22–26]). The identification problem is to determine a set of places P of cardinality m, a set of transitions T of cardinality n, and the function of labeling

and the PN defined as an ILP problem. Silent transitions are those that identify faults. A contribution of these methods is given in [3] that deals with the identification problem for deterministic systems as of timed λ − free PN, starting from the timed observed sequences. This method generates a net language, whereas the timer information is used to identify language strings that are not accepted by the net; therefore, the algorithm does not need to know the language of the full net. The solution is based on ILP and knowledge of the timed net structure. It allows, together with the observation of events, determining whether a transition time has expired. The algorithm is carried out through two subroutines, one to observe the language on-line and another for the off-line ILP. Furthermore, [27] contributes with a technique that reduces the number of sets of inequalities and the number of inequalities in each set to identify the IPN as an ILP problem. The other contribution in this area is given in [28], presenting a method that uses ILP with IPN, to prevent exhaustive generation of reachable states; it is considered nondeterminism in the model, since different observable transitions share the same label; it has a timing structure of events leading to the imposition of new restrictions and provides the most accurate diagnosis. It is a very efficient method in small models but to find an optimal solution in complex models has very high computational requirements.

These trends can be compared regarding to criteria such as kind of system to identify, resulting model and computational cost, we can say that: The I/O IPN identification cannot be used with timed systems; it can be applied to complex systems; the obtained models are less complex than those obtained using the Automata based methods and computational cost is normal. The NDFA trend applies to timed systems, and it works best on simple systems; it generates nondeterministic models and the computational cost is normal; furthermore, there is a combinatorial explosion of states. And regarding IPL + IPN, some proposals are for timed systems and some present nondeterminism so these methods cannot be applied today to identify real complex DES because of the high computational complexity of the algorithm.

Timed event systems identification with output symbols has been also treated in [21] using a technique from computational learning theory in the context of timed Automata and in [3] with free labeled nets; in this paper it is assumed that a delay is associated with each transition and the timer variables are used to determine if a potentially enabled transition; on the basis of the observed firings, a subset of the net language is build, while the timing information is exploited to determine a subset of counterexamples, that is, a set of strings that do not belong to the net language.

Considering the advantages and disadvantages of the existing identification methods, this paper proposes an identification method of timed discrete event complex systems, which identifies a deterministic language generator. The language generator is defined as a stochastic timed IPN (st-IPN). The proposal to model a system as st-IPN is inspired in some concepts of the modelling formalism by NCES, specifically on conditions and I/O events with the purpose of labeling the transitions and places. We have added properties to generate a DES modelling approach with the purpose of performing diagnostics on complex systems.

This paper is organized as follows. Section 2 describes the background on PN; Section 3 defines a st-IPN; Section 4 presents the identification method; Section 5 shows an application case; finally, the concluding remarks and discussion are showed in Section 6.

2. Background on PN

Petri nets (PNs) are widely used for modeling DES [29]. PN provides compact models and captures main DES characteristics as concurrence, asynchronism, causal relationships, mutual exclusions, and so forth. This paper will model DES with PN, so some basics of PN are presented.

Definition 1 (Petri net (PN)). A Petri net structure N is a bipartite digraph represented by the five-tuple $N = (P, \mathrm{TR}, \mathrm{Pre}, \mathrm{Post}, M_0)$, where P is a set of places with cardinality n and TR is a set of transitions with cardinality m, and $\mathrm{Pre} : P \times \mathrm{TR} \to \mathbb{N}$, $\mathrm{Post} : \mathrm{TR} \times P \to \mathbb{N}$ are the *pre-* and *postincidence matrices*, respectively, which specify the arcs connecting places and transitions. Matrix $I = \mathrm{Post} - \mathrm{Pre}$ is the $m \times n$ incidence matrix of the net. The marking function $M : P \to \mathbb{N}$ represents the number of tokens residing inside each place, and M_0 is the initial marking [26, 30].

For *pre-* and *post-sets*, the dot notation is used. $\bullet \mathrm{tr} = \{p \in P : \mathrm{Pre}(p, \mathrm{tr}) > 0\}$ [26].

Definition 2 (Reachability set of a PN). Let N be a PN. Transition tr_j is enabled at marking M_k if $\forall p_i \in \bullet \mathrm{tr}_j, M(p_i) \geq I(p_i, \mathrm{tr}_j)$. The enabled transition tr_j can be fired reaching a new marking M_{k+1} that can be computed by $M_{k+1} = M_k + I \cdot \overrightarrow{\mathrm{tr}_j}$, where $\overrightarrow{\mathrm{tr}_j}$ is a |TR| vector of zeroes, except in entry j, that is equal to 1. An enabled sequence $\sigma = \mathrm{tr}_1 \mathrm{tr}_2 \cdots \mathrm{tr}_k$ is denoted as $M[\sigma > M_k$. The reachability set of a PN is the set of all possible reachable markings from M_0, firing only enabled transitions; this set is denoted by $R(N, M_0)$ [31].

The system inputs will be the control commands and the outputs will be the sensor readings. At each time instant each input will have a particular value and so the outputs. In order to organize the signals, this paper encodes the different I/O vectors using its binary representation; therefore, it is defined an input (output) symbol as the set of control commands (sensor readings) values at a time instant in binary representation.

Definition 3 (I/O symbol). Assuming binary values in input and output signals, u_s is a binary representation of s, s stands for the input symbol, and $s = 0 \cdots |2^m| - 1$, with $u_0 = [\vec{0}]$; …; $u_{|2^m|-1} = [\vec{1}]$, the same applies to y_j, y_j is a binary representation of j, j stands for the output symbol, and $j = 0 \cdots |2^n| - 1$, with $y_0 = [\vec{0}]$; …; $y_{|2^n|-1} = [\vec{1}]$.

For example, given a set of control commands $Cc = \{\mathrm{cc}_1, \mathrm{cc}_2\}$; if $\mathrm{cc}_1 = 1$ and $\mathrm{cc}_2 = 0$ then the input symbol is represented by u_2.

Note that is possible that the controller will not generate all the input symbols, since some control commands cannot be enabled simultaneously. In the same way, the system will not generate all the output symbols if there is no failure.

Definition 4 (interpreted Petri net). An IPN is a tuple $Q = (N, U, Y, \lambda, \varphi)$ (see [32]), where N is a PN. $U = \{u_0, u_1, \ldots, u_{|2^m|-1}\}$ is the input alphabet, u_s is an input symbol, and m is the number of inputs; $Y = \{y_0, y_1, \ldots, y_{|2^n|-1}\}$ is the output alphabet, y_j is an output symbol, and n is the number of outputs; $\lambda : \mathrm{TR} \rightarrow U$ is a labeling transition function that assigns an input symbol to each transition. $\varphi : R(N, M_0) \rightarrow Y$ is an output function that assigns an output symbol to each reachable marking.

From the definition given in [33], an event is a pair (u_s, y_j). A new event is generated when there is a change in u_s, in y_j, or in both.

The system alphabet Ω relates I/O symbols; that is, $\Omega = U \cdot Y$; therefore $\omega^e = u_s y_j \in U \cdot Y$ if $u_s \in U$ and $y_j \in Y$, where $e = 1 \cdots |2^m| \times |2^n|$.

If $U \cdot Y$ is the system alphabet and U (Y) is the input (output) alphabet, then $P_U : (U \cdot Y)^* \rightarrow U^*$ (resp., $P_Y : (U \cdot Y)^* \rightarrow Y^*$), where $P_U(u_s y_j) := u_s$ if $u_s \in U$, $P_U(u_{s'} y_j) := \varepsilon$ if $u_{s'} \notin U$. ($P_Y(u_s y_j) := y_j$ if $y_j \in Y$, $P_Y(u_s y_{j'}) := \varepsilon$ if $y_{j'} \notin Y$).

Definition 5 (timed event). A timed event ω^i is an I/O symbol at time τ_i,

$$\omega^i = \left(u_s y_j\right) \tau_i, \tag{1}$$

where $u_s y_j$ is an I/O symbol at time τ_i. The time elapsed from τ_i to τ_{i-1} is t^i (see [34]) and $t^i = |\tau_i| - |\tau_{i-1}|$.

Definition 6 (firing language (see [35])). A firing sequence in an IPN (Q) is a transition sequence $\sigma = \mathrm{tr}_1 \mathrm{tr}_2 \cdots \mathrm{tr}_k \cdots$ such that $M_0 \xrightarrow{\mathrm{tr}_1} M_1 \xrightarrow{\mathrm{tr}_2} \cdots M_w \xrightarrow{\mathrm{tr}_k} \cdots$. Given a firing sequence σ, a marking $M_k \in R(N, M_0)$ is reached from M_0 if there exists a σ_i such that $M_0 \lceil \sigma_i > M_k$. The set of all firing sequences is called the firing language $\mathscr{L}(F) = \{\sigma | \sigma = \mathrm{tr}_1 \mathrm{tr}_2 \cdots \mathrm{tr}_k \cdots \wedge M_0 \xrightarrow{\mathrm{tr}_1} M_1 \xrightarrow{\mathrm{tr}_2} \cdots M_w \xrightarrow{\mathrm{tr}_k} \cdots | M_0, M_1, M_w \in R(N, M_0)\}$. Each transition tr_j is fired at time τ_j ($\tau_1 \leq \tau_2 \leq \cdots \leq \tau_k$); therefore σ is an arranged sequence in a timeline, where each transition happens at time τ_i.

$\mathscr{L}(F)$ generates a timed event sequence (based on Definition 5), called system language $\mathscr{L}(Q) = \{\lambda(\mathrm{tr}_1)\varphi(M_1), \lambda(\mathrm{tr}_2)\varphi(M_1), \ldots, \lambda(\mathrm{tr}_k)\varphi(M_k)\}$ whose events are arranged at the same timeline as $\mathscr{L}(F)$.

We define the projection $P_U : \mathscr{L}(Q) \rightarrow \mathscr{L}_{\mathrm{in}}(Q)$ (resp., $P_Y : \mathscr{L}(Q) \rightarrow \mathscr{L}_{\mathrm{out}}(Q)$), where $P_U(\lambda(\mathrm{tr}_k)\varphi(M_k)) = \lambda(\mathrm{tr}_k)$ if $\lambda(\mathrm{tr}_k) \in U$ and $P_U(\lambda(\mathrm{tr}_{k'})\varphi(M_k)) = \varepsilon$ if $\lambda(\mathrm{tr}_{k'}) \notin U$. ($P_Y(\lambda(\mathrm{tr}_k)\varphi(M_k)) = \varphi(M_k)$ if $\varphi(M_k) \in Y$ and $P_Y(\lambda(\mathrm{tr}_k)\varphi(M_{k'})) = \varepsilon$ if $\varphi(M_{k'}) \notin Y$). Therefore consider the following.

(i) The input language is the generated language by the input symbols, is:

$$\mathscr{L}_{\mathrm{in}}(Q) = \left\{P_U(s)\, s \in \mathscr{L}(Q)\right\} \tag{2}$$

Note that there could be several transitions with the same label if λ is not isomorphic.

(ii) The The output language is the generated language by the output symbols, is:

$$\mathscr{L}_{\mathrm{out}}(Q) = \left\{P_Y(s)\, s \in \mathscr{L}(Q)\right\} \tag{3}$$

Note that there could be several places with the same label if $\varphi(M_w)$ is not isomorphic.

For identification purposes, it is convenient to use isomorphic φ functions. Isomorphic mappings will lead to equivalences between output symbols and net markings. Therefore, it will be immediate to infer the net state from a particular combination of output symbols.

Note that the evolution on an IPN could be nondeterministic if φ is nonisomorphic. For example, a nondeterministic evolution will occur if there exists at least one place with at least two transitions with the same label that lead to different places. Nevertheless, the language generated by many systems (see [18]) is represented by a nondeterministic NDA when the system state is described by means of its outputs. For some applications, such as diagnosis, nondeterminism is an undesirable characteristic. Next section will present an extension of IPN that will avoid the nondeterminism in this case.

3. Definition of Stochastic Timed Interpreted Petri Net (st-IPN)

This section presents the definition of st-IPN. st-IPN definition includes the concept of "differential of output signals" (dy), which represents the historical behavior of the output signals. This concept will improve the understanding of the process states. For example, in a fluid tank it is possible to differentiate between the filling or emptying tasks. This knowledge takes a great importance for identification purposes.

Definition 7 (differential of output symbol, dy). Given an output symbol $y_j \in Y$ and $y_{j-1} \in Y$; dy_j is defined as

$$dy_j\left(y_j \times y_{j-1}\right) \longrightarrow \{-1, 0, 1\}, \tag{4}$$

where $dy_j(a, b) = c$, $a \in Y$ is an output symbol at time τ_i, $b \in Y$ is an output symbol at time τ_{i-1}, and $c \in \{-1, 0, 1\}$; for an output symbol y_j; $c = a - b$, so dy_j possible values are $0 \times 0 \rightarrow 0$; $0 \times 1 \rightarrow 1$; $1 \times 0 \rightarrow -1$; $1 \times 1 \rightarrow 0$.

st-IPN combines an IPN and a timed PN. IPN is defined in Definition 4 and the timing concepts are based on [3] where the authors define a timed PN with a delay associated with each transition, represented by $\delta(\mathrm{tr})$. This delay represents the time that must elapse from the enabling of the

transition until it fires. Timer variables are used to determine if a potentially enabled transition tr is *timed-out*, that is, if its firing time is elapsed and tr has not yet fired. If $\delta(\mathrm{tr}) \neq 0$, then tr is said to be *timed*, while if $\delta(\mathrm{tr}) = 0$, then tr is said to be *immediate*.

Definition 8 (timer variables (based on [3])). Given a transition sequence $\sigma = \mathrm{tr}_1 \mathrm{tr}_2 \cdots \mathrm{tr}_k \cdots$, a timer variable t_{tr_j} is associated with each timed transition tr_j, where t_{tr_j} is enabled if $M_j >$ pre; t_{tr_j} is disabled; that is, $t_{\mathrm{tr}_j} = 0$, before the firing of σ.

Moreover, it is important to introduce the concept of *operation mode* which is related to external events signals that affects the controller, such as SCADA commands, operator requirements, among others. The operation modes can be grouped in a set $\mathrm{OM} = \{o_1, o_2, \ldots\}$.

A stochastic system is affected by a number of aspects such as the variability in the material flow, the fluctuation of power supply, and the deterioration of machinery and devices. These variations change the enabling and the firing times. It is assumed that each transition time follows a stochastic distribution that can be different for each operation mode.

Considering that this variability is not always normally distributed, st-IPN includes a density function of transition firing time for each transition and for each operation mode.

Definition 9 (stochastic timed interpreted Petri net (st-IPN)). A st-IPN is a structure represented by

$$\mathrm{st}Q = (Q, \Omega, \delta), \tag{5}$$

where $Q = (N, U, Y, \lambda, \varphi)$ is an IPN where N, U, Y have the same meaning as in Definition 4, but input function λ is defined as $\lambda : \mathrm{TR} \rightarrow U \cdot \delta$ (a labeling function that assigns an input symbol and a timer density function to each transition) and φ is defined as $\varphi : R(N, M_0) \rightarrow y_j/dy_j$; φ is isomorphic over y_j/dy_j.

$\Omega = U \cdot Y := \{\omega = (u_s y_j)\}$ is the system alphabet.

$\delta := \mathrm{TR} \times \mathrm{OM} \rightarrow f(t_{\mathrm{TR}\times\mathrm{OM}})$ is a density function of transition firing time for each operation mode $\mathrm{OM} = \{o_1, o_2, \ldots\}$.

A transition $\mathrm{tr}_j \in \mathrm{TR}$ with $\lambda(\mathrm{tr}_j) = u_s \cdot t_{\mathrm{tr}_j}$ is enabled at operation mode o_i if and only if $M_j >$ pre and if $\mathrm{Prob}(t_{\mathrm{tr}_j, o_i}) \geq (1 - \alpha)$, where $\mathrm{Prob}(t_{\mathrm{tr}_j}) = \int f(t_{\mathrm{tr}_j \times \mathrm{O}_i})$ and $1 - \alpha$ is the confidence level. Therefore, a transition is enabled if each input place meets the marking requirements and the time elapsed has reached a certain confidence interval.

When tr_j is fired, a new marking is reached, such that $\varphi(M_k) = y_j/dy_j$; thereby, each place of the st-IPN represents not only the current system state, but also includes information about directionality.

(i) Generated language: Stochastic timed event sequence generated from the evolution of a st-IPN, so $\mathscr{L}(\mathrm{st}Q) = \{s \in (U \cdot Y)^* : \mathrm{Prob}(s \mid \varphi(M_0)) \geq (1 - \alpha)\}$, where $s = \omega^0, \omega^1, \ldots, \omega^k$ at times $\tau_0 \leq \tau_1 \leq \cdots \leq \tau_k$. A symbol of the language generated is a concatenation of an input symbol and an output symbol ($\omega^i = u_s y_j$) at time τ_i.

3.1. Properties. Given an I/O event sequence s, where $s = (u_k y_k), (u_x, y_x), (u_1, y_1), \ldots, (u_v, y_v), (u_x y_x), (u_1 y_3)$, with $y_k \neq y_v$, if φ is an isomorphic output function between the set of markings and the set of output symbols, then the net that generates the sequence is nondeterministic because input symbols u_1 can be followed by two different output symbols, y_3 and y_x. One of the consequences of a net nondeterministic is that it generates event sequences that are not part of the system language. The generation of event strings not included in the system language must be avoided if the generator is going to be used as a diagnosis model. The Proposition 10 proof that a st-IPN from s is deterministic.

Proposition 10. *Let stQ be a st-IPN defined as in Definition 9. If φ is isomorphic, then stQ is deterministic.*

Proof. Let s be the sequence described in the previous paragraph. Let us split the symbol y_x into two y'_x and y''_x; $s = (u_k y_k), (u_x y'_x), (u_1 y_1), \ldots, (u_v y_v), (u_x y''_x), (u_1 y_3)$; the differential of output symbols in y'_x is $dy'_x = y'_x - y_k$ and in y''_x is $dy''_x = y''_x - y_v$; as $y_k \neq y_v$ then $dy'_x \neq dy''_x$; therefore, the output function is different in the two states, thus eliminating the nondeterminism and increasing the number of states. So, φ is isomorphic between the set of markings and the set of output symbols y_j/dy_j. □

The inclusion of dy in the model allows the computing of a maximum number of states.

Definition 11 (maximum number of states, $\mathrm{Q}_{\max}$). The system state is determined by the output symbols and their historical behavior; therefore, the maximum number of identifiable states is related to these output signals; thus

$$\mathrm{Q}_{\max} = 2^n \times 3^n, \tag{6}$$

where n is the number of output symbols.

For example in a system with only one binary sensor, $\mathrm{Q}_{\max} = |2|^1 \times |3|^1 = 6$; in a system with two binary sensors, $\mathrm{Q}_{\max} = |2|^2 \times |3|^2 = 36$.

Proposition 12. *Let stQ be a st-IPN and $\mathscr{L}(stQ)$ the language generated by stQ and let $s = \omega^i \cdots \omega^k$ at times $\tau_i, \ldots, \tau_k$, ($\omega^i = u_s y_j$) be an event sequence. If $s \in \mathscr{L}(stQ)$ and $\varphi(M_i) = \varphi(M_k)$ then $s^* \in \mathscr{L}(stQ)$.*

Proof. Because a st-IPN is deterministic (see Proposition 10), if $\varphi(M_i) = \varphi(M_k)$ then $M_i = M_k$ so s can be generated infinitely. □

Proposition 13. *Let stQ_1, stQ_2 be st-IPNs, with incidence matrices I_1 and I_2 and initial markings M_0^1 and M_0^2, and let $\mathscr{L}(stQ_1)$, $\mathscr{L}(stQ_2)$ be the language generated by stQ_1 and*

stQ_2, respectively, with $\varphi_1(M) = \varphi_2(M)$ and $M_0^1 = M_0^2$. $\mathscr{L}(stQ_1) = \mathscr{L}(stQ_2)$ if $I_1 = I_2$.

Proof. The system evolves the following PN state equation:

$$M_{k+1} = M_k + I \cdot \sigma_k,$$
$$y_k/dy_k = \varphi(M_k). \tag{7}$$

Let us suppose that $\mathscr{L}(\mathrm{stQ}_1) = \mathscr{L}(\mathrm{stQ}_2) = \mathscr{L}(\mathrm{stQ})$ but $I_1 \neq I_2$.

Let us consider an event sequence $s = \omega^0, \omega^1, \ldots, \omega^k \in \mathscr{L}(stQ)$ at times $\tau_0 \leq \tau_1 \leq \cdots \leq \tau_k$ that is generated by a firing sequence $\sigma = \mathrm{tr}_1 \mathrm{tr}_2 \cdots \mathrm{tr}_k$.

If $I_1 \neq I_2$ then $\exists k > 0 \mid M_0^1 \xrightarrow{\mathrm{tr}_1} M_1^1 \xrightarrow{\mathrm{tr}_2} \cdots \xrightarrow{\mathrm{tr}_k} M_k^1$; $M_0^2 \xrightarrow{\mathrm{tr}_1} M_1^2 \xrightarrow{\mathrm{tr}_2} \cdots \xrightarrow{\mathrm{tr}_k} M_k^2$, for each st-IPN, respectively, and $M_i^1 = M_i^2$ but, $M_k^1 \neq M_k^2$. As φ is isomorphic then $\varphi(M_k^1) \neq \varphi(M_k^2)$.

As $\mathscr{L}(\mathrm{stQ}_1) = \mathscr{L}(\mathrm{stQ}_2)$ then $\mathscr{L}_{\mathrm{out}}(\mathrm{stQ}_1) = \mathscr{L}_{\mathrm{out}}(\mathrm{stQ}_2)$.

If $\mathscr{L}_{\mathrm{out}}(\mathrm{stQ}_1) = \{\varphi(M_0), \varphi(M_1^1), \ldots, \varphi(M_k^1)\}$ and $\mathscr{L}_{\mathrm{out}}(\mathrm{stQ}_2) = \{\varphi(M_0), \varphi(M_1^2), \ldots, \varphi(M_k^2)\}$ by (2) and (3) respectively.

Then $\varphi(M_0^1) = \varphi(M_0^2) \cdots \varphi(M_k^1) = \varphi(M_k^2)$.

Therefore $M_k^1 = M_k^2$. Whereof $I_1 = I_2$, which contradicts the initial condition. □

It is concluded that the st-IPN that generates a language is unique.

Proposition 14. *Let $\varphi : R(N, M_0) \to Y/dY$ be an output function and let $s \in \mathscr{L}(stQ)$, $s = \omega^i \cdots \omega^k$ be a sequence that models a cycle ($\varphi(M_i) = \varphi(M_k)$); $\sum_i^k dy_j = 0$, for each, $j = 1, \ldots, n$ where n is the number of sensors.*

Proof. The system evolves the following st-IPN state. Equation (7) is as follows.

$s = \{\omega^i \cdots \omega^k\} = \{\lambda(\mathrm{tr}_i)\varphi(M_i) \cdots \lambda(\mathrm{tr}_k)\varphi(M_k)\}$ observed at times $\tau_i, \cdots \tau_k$; then $\mathscr{L}_{\mathrm{out}}(\mathrm{stQ}) = \{\varphi(M_i), \ldots, \varphi(M_k)\}$ by (3), that is, $\mathscr{L}_{\mathrm{out}}(\mathrm{stQ}) = \{y_i/dy_i, \ldots, y_k/dy_k\}$.

In compliance with the condition that in a cycle $\varphi(M_i) = \varphi(M_k)$, $(i \neq k)$, the behavior of each output signal can be as follows. (i) It does not change; then, the value of dy_j is always zero during the cycle; therefore $\sum_i^k dy_j = 0$; (ii) it changes; therefore the value of $\varphi(M_i)$ can be 1 or 0, if it starts in 1; the value along the cycle will be $1 \to 0 \to 1$; therefore, $dy_j : 0 \to -1 \to 1 : \sum_1^3 dy_j = 0$. If it starts in 0, the value along the cycle will be $0 \to 1 \to 0$; therefore, $dy_j : 0 \to 1 \to -1 : \sum_1^3 dy_j = 0$. That is to say, after $2n + 1$ steps (n: number of outputs whose value has changed), the output values must return at their initial values when reaching M_k. □

According to their definition st-IPNs are an extension of PNs which include input and output signals in places and transitions (IPNs) and stochastic firing times for each transition (st-IPNs). The link with I/O signals and the capacity of including stochastic time variations make st-IPNs a powerful tool to model real systems.

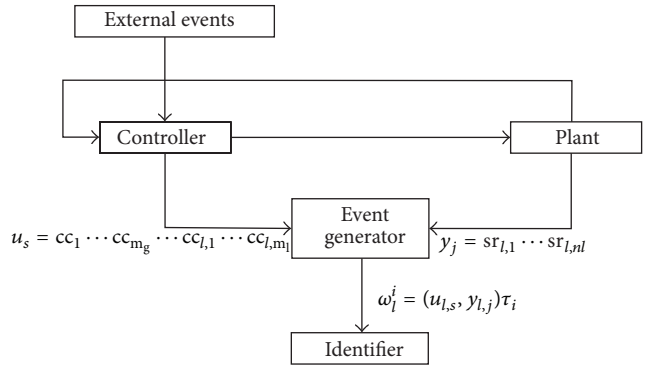


Figure 1: Event Generator.

4. DES Identification Method

4.1. System to Identify. The generation of internal and external events in a timeline defines the behavior of the system (see Figure 1). A set of external events defines an operation mode that affects the controller and an internal event is the concatenation of control commands and sensor readings in at time instant.

The system consists of the closed loop connection of a dynamic system (plant) and a control law (controller). Once the operation mode is defined (by the external events), the controller executes a control strategy by generating a sequence of control commands. This sequence input gets into the plant and activates the actuators. The system reacts and generates a sensor readings sequence (Figure 1). Therefore the behavior of the system can be described in terms of event sequences.

During a system evolution like a production cycle, it is possible to get information of the system state (by analyzing these sequences), with the language theory ([1]). Each internal event is a letter of the alphabet and a letter sequence is a word that models the system performance. Therefore, the system can be described as a regular language generator.

4.1.1. Definition of a Subsystem. To decrement the complexity of the model, the system is split into subsystems. A subsystem is a part of a system which has a particular behavior. The division into subsystems is carried out according to physical or functional criteria, because the system model to identify is unknown and is not possible to define the subsystems in other ways.

For each subsystem the inputs will be the control commands and the outputs will be the sensor readings. Control commands can be classified as global or local. A control command is global when it is applied to more than one subsystem and it is local when it is applied only to one subsystem. Therefore the control commands set is $\mathrm{Cc} = \mathrm{Cc}_g \cup \mathrm{Cc}_l$, where $\mathrm{Cc}_g = \{\mathrm{cc}_1, \ldots, \mathrm{cc}_{m_g}\}$ and m_g is the number of global control commands; $\mathrm{Cc}_l = \cup \mathrm{cc}_{l,m_l}$; $l = 1 \cdots c$, c is the index of the subsystem and m_l is the number of local control commands in subsystem l.

Each sensor reading is assigned to a subsystem, so the set of sensor readings will be $\mathrm{Sr} = \cup \mathrm{sr}_{l,n_l}$, where n_l is the number of sensors in subsystem l and $l = 1 \cdots c$.

4.1.2. Restrictions over the System. The system has the following restrictions.

(i) The system is closed loop controlled and its working is cyclic. Moreover, each subsystem can only be in one state at a time.

(ii) The exchanged signals between the system and the controller are discrete with only two possible values (binary coding).

(iii) Its behavior is defined by the generation of internal and external events in a timeline.

(iv) An external event affects the controller; an internal event is the concatenation of control commands and sensor readings at time instant. An external event sequence is a timed sequence which generates a system operation mode.

(v) An internal event can only be generated by a subsystem at a time.

4.1.3. System Operation. Given an operation mode, the controller executes a control strategy then a timed internal event sequence is generated; that is, at each time the controller generates a set of control commands $u_s = \{\mathrm{cc}_1, \ldots, \mathrm{cc}_{m_g}, \mathrm{cc}_{l,1}, \ldots, \mathrm{cc}_{l,m_l}\}$. This set is the input set to system which changes its state and generates an output set $\{y_j = \mathrm{sr}_{l,1}, \ldots, \mathrm{sr}_{l,n_l}\}$ containing sensor readings. $u_s \in U$; $U = \{u_0, u_1, \ldots, u_{|2^{(m_g+m)}|-1}\}$ and $y_j \in Y$; $Y = \{y_0, y_1, \ldots, y_{|2^n|-1}\}$.

(a) Event Generator. It is a procedure that concatenates the I/O symbols at time τ_i when any signal changes (control command or sensor reading) from τ_{i-1} to τ_i and it generates an event as $\omega^i = (u_s y_j)\tau_i$.

Applying this concept to system defined in definition of Section 4.1, an event in a subsystem l will be noted with the subscript l, so an event in a subsystem l is noted as ω_l^i. Consider $\omega_l^i = (u_{l,s} y_{l,j})\tau_i; u_{l,s} \in U_l; U_l = \{u_{l,0}, u_{l,1}, \ldots, u_{l,|2^{(m_g+m_l)}|-1}\}$ and $y_{l,j} \in Y_l; Y_l = \{y_{l,0}, y_{l,1}, \ldots, y_{l,|2^{n_l}|-1}\}$ (see Figure 1).

Note that when there are changes in control commands or sensor readings shared by over one subsystem, the event is generated in all the subsystems involved.

4.2. Identification Process. This section presents the identification method of the DES presented in Section 4.1.

Problem 15. Given an event sequence of size e, $\omega^0 \omega^1 \cdots \omega^e$, generated from generator defined in Section 4.1.3(a) in a system defined as in Section 4.1, find the st-IPN that generates the same sequence.

The first step in the identification process will be the division of the system into subsystems.

Each subsystem will be identified as a st-IPN. For this purpose the identification process requires monitoring the signals exchanged between the system and the controller (I/O signals). These signals enter the event generator, which concatenates the signals and generates an internal event.

The event arrives to the identifier, which continuously builds the st-IPN which is able to generate the same event, in the same sequence and at the same time.

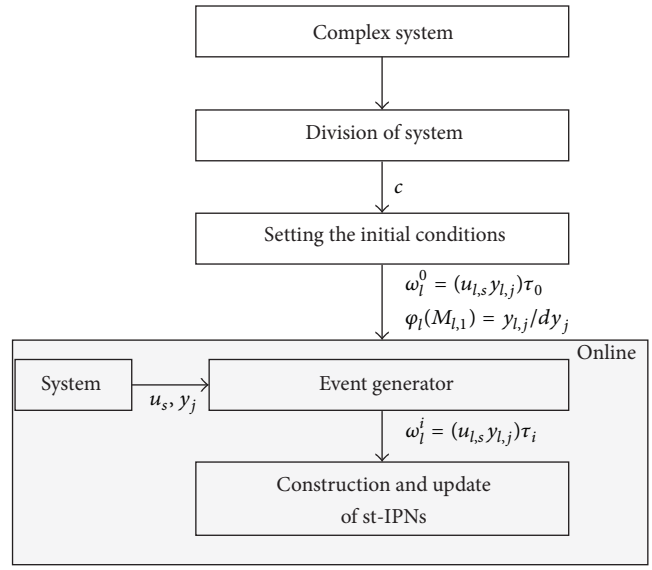


Figure 2: Identification process.

4.3. Identification Algorithm. The identification algorithm is shown in Figure 2. First, the system to identify has to be split into subsystems. Then, the initial conditions must be set, which consist of the initial event and the initial state. Let us assume that the system is split into c subsystems, so each subsystem will be noted with the index l ($l = 1 \cdots c$).

The starting event for each subsystem is $\omega_l^0 = (u_{l,s} y_{l,j})\tau_0$, where $u_{l,s}$, $y_{l,j}$ stand for the starting values of control commands and sensor readings, respectively, $l = 1 \cdots c$ at time τ_0. Therefore each subsystem starts with a st-IPN with one place. The inputs and output functions will be $\varphi_l(M_{l,1}) = y_{l,j}/\vec{0}$, and $\lambda_l(\mathrm{tr}_{l,0}) = u_{l,s} \cdot t_{l,0}^0$ and $t_{l,0}^0 = 0$ is the initial value of the timer; that is, the subsystem l starts without transitions.

Following the establishment of an operation mode o_i ($o_i \in \mathrm{OM}$), the event detection procedure generates events when any change happens in the I/O signals. It reads the input symbol and waits for the sensor setting. The next step is checking if there has been a change in one of the symbols of the I/O pair, in order to generate an event in that case.

The st-IPN identification algorithm starts with a new event (see Figure 3); then it compares the event with the previous one in order to classify it as a change in an input, in an output, or in both.

The identification algorithm (Algorithm 1) follows the flow diagram presented in Figure 3. It identifies a change in a subsystem state when there is a difference between $y_{l,j}/dy_{l,j}$ and the current output symbol in subsystem l. In this case, the system executes Algorithm 2, (Case 1). This algorithm creates

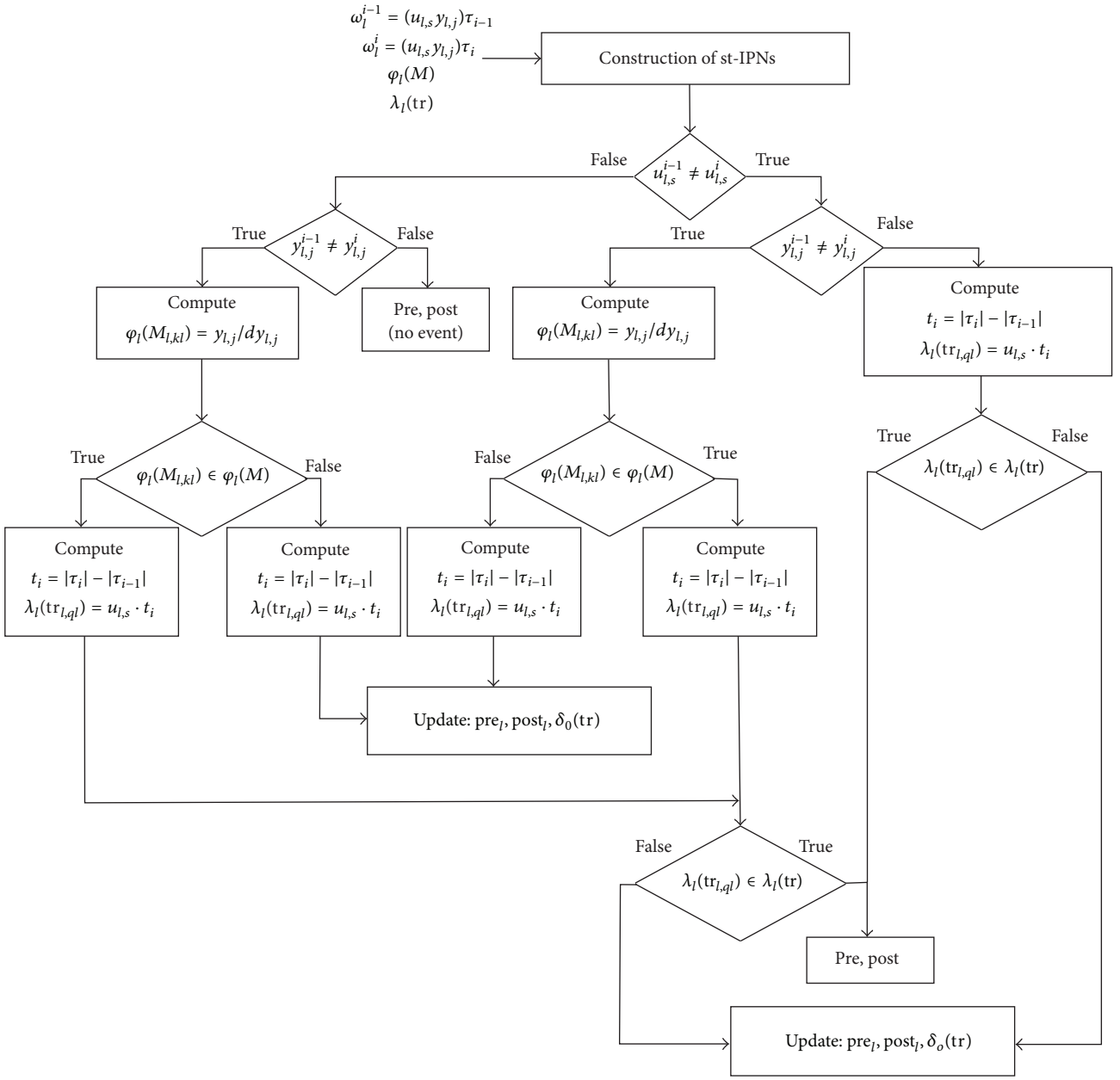


Figure 3: Algorithm 1 flowchart.

new transitions and places when necessary and it also updates the current st-IPN marking.

Otherwise, a change only in the input symbol will involve a transition, so Algorithm 2, (case 2) includes this new transition (if it does not exist) in the st-IPN and updates the current state of the st-IPN.

4.4. Including Time Information. This section explains how to exploit the time to solve the identification problem.

Time that must elapse from the enabling of the transition until it fires is accumulated in time data arrays $\delta_{o_i}(\mathrm{tr}_{l,q_l}) = \{t_{l,q_l}^1 \cdots t_{l,q_l}^k\}$, where $o_i \in \mathrm{OM}$, q_l, l stand for the indexes of the transition and subsystem, respectively, and k is the number of samples.

The goal to monitoring the process data is finding a statistical model that describes statistical behavior of each transition; therefore, the system must be observed for long time to have enough data for the statistical analysis, until the process converges. The convergence criterion is based on finding a reliable estimate.

The estimation error of population mean is the estimator we use to study the convergence of the method. It has denoted the maximal accepted difference between the population mean and the sample mean as d_{op}.

Sample size determination is the act of choosing the number of times that the process to identify is observed. The sample size is increased until the process has been identified totally and has converged.

Inputs: $\omega_l^i = \left(u_{l,s} y_{l,j}\right) \tau_i$; $\omega_l^{i-1} = \left(u_{l,s} y_{l,j}\right) \tau_{i-1}$; $\varphi_l(M)$ $\lambda_l(\mathrm{tr})$; $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right)$; k_l; q_l.
Output: Pre_l; Post_l; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$; $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right)$.
Initial Conditions: $\varphi_l(M) = \{\varphi_l\left(M_{l,1}\right)\}$; $\lambda_l(\mathrm{tr}) = \lambda_l\left(\mathrm{tr}_{l,0}\right)$; $\mathrm{Pre}_l = [1]$; $\mathrm{Post}_l = [0]$; $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right) = 0$; $k_l = 1$; $q_l = 0$
read $u_{l,s}$, $y_{l,j}$
(1) If $u_{l,s}^i(:) \neq u_{l,s}^{i-1}(:)$ and $y_{l,j}^i(:) \neq y_{l,j}^{i-1}(:)$
execute Algorithm 2 CASE = 1
(2) else if $u_{l,s}^i(:) = u_{l,s}^{i-1}(:)$ and $y_{l,j}^i(:) \neq y_{l,j}^{i-1}(:)$
execute Algorithm 2 CASE = 1
(3) else if $u_{l,s}^i(:) \neq u_{l,s}^{i-1}(:)$ and $y_{l,j}^i(:) = y_{l,j}^{i-1}(:)$
execute Algorithm 2 CASE = 2
end if

Algorithm 1: st-IPN Identification.

Inputs: $u_{l,s}$; $y_{l,j}$; k_l; q_l; $\varphi_l\left(M_{l,k_l}\right)$; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$
Outputs: Pre_l; Post_l; $\varphi_l\left(M_{l,k_l}\right)$; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$
Case 1.
read (l)
$k_l' = k_l + 1$;
$q_l' = q_l + 1$;
$u_{l,s} = \{\mathrm{cc}_1, \ldots, \mathrm{cc}_{m_g}, \mathrm{cc}_{l,1}, \ldots, \mathrm{cc}_{l,m_l}\}$;
$y_{l,j} = \{\mathrm{sr}_{l,1}, \ldots, \mathrm{sr}_{l,n_l}\}$;
(1) $\varphi_l\left(M_{l,k_l'}\right) = y_{l,j}^i / \left(y_{l,j}^i - y_{l,j}^{i-1}\right) = y_{l,j}^i / dy_{l,j}$
(2) $t_{l,q_l}^i = |\tau_i| - |\tau_{i-1}|$
(3) $\lambda_l\left(\mathrm{tr}_{l,q_l'}\right) = u_{l,s} \cdot t_{l,q_l}^i$
(4) Update Pre_l; Post_l; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$; $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right)$.
Case 2.
read (l)
$k_l' = k_l$;
$q_l' = q_l + 1$;
$u_{l,s} = \{\mathrm{cc}_1, \ldots, \mathrm{cc}_{m_g}, \mathrm{cc}_{l,1}, \ldots, \mathrm{cc}_{l,m_l}\}$;
(1) $t_{l,q_l}^i = |\tau_i| - |\tau_{i-1}|$
(2) $\lambda_l\left(\mathrm{tr}_{l,q_l'}\right) = u_{l,s} \cdot t_{l,q_l}^i$
(3) Update Pre_l; Post_l; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$; $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right)$
The updating of Pre_l; Post_l; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$; $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right)$ can be seen in Figure 4.

Algorithm 2: Subroutine Case.

The number of samples needed for fitting a distribution (n) can be computed as [36–38] $n = (\mathrm{st}_{\alpha/2})^2 \times S^2/d^2$. Moreover, the significance level (α) can be defined as $\mathrm{Prob}(|\overline{X} - \mu| \geq d \times \mu) = \alpha$,$d$ being the difference between the population mean and the sample mean and S^2 is the sample quasi-variance. α is the risk of having an error larger than d_{op}. Generally α is set in 0.05 [39] and $\mathrm{st}_{\alpha/2}$ is the factor determining the length of a confidence interval of population mean.

From to equation of n, d is

$$d = \left(\mathrm{st}_{\alpha/2}\right) \times \frac{S}{\sqrt{n}} \tag{8}$$

Algorithm 3 is executed each time a time array gets a new value, so its size has increased. It computes d and compares it with the defined upper limit (d_{op}). That is, where, if $n \leq 30$, $\mathrm{st}_{\alpha/2} = t_{n-1}$ is computed as a Student's t distribution and, if $n > 30$, $\mathrm{st}_{\alpha/2} = z_{\alpha/2}$ is computed as a normal distribution. Once d_{op} is reached, Algorithm 3 is no longer applied to this transition and operation mode, because its distribution can be identified with the available data. Moreover, no more data is included in the time vector.

Algorithm 3 first determines if there is enough data to fit a distribution, once d_{op} is reached, that is, the size sample is sufficient, then it is necessary to find the probability density function (PDF) that fits the data. The methods of

Input: $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right) = \left\{t^1_{l,q_l}, \ldots, t^k_{l,q_l}\right\}$ with $k \geq 5$, α, d_{op},
Output: n_{l,q_l}
for $l := 1$ to c
for $q_l := 1$ to $|q_l|$
(1) $n_{l,q_l} = \left|\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right)\right|$;
(2) compute $\overline{X}$, S;
(3) if $n_{l,q_l} \leq 30$ compute st $= t_{n-1}$
(i) else compute st $= z_{\alpha/2}$
end if
(4) $d = \left(\mathrm{st}_{\alpha/2}\right) \times S/\sqrt{n_{l,q_l}}$;
(i) If $d \leq d_{op}$
execute Algorithm 4
(ii) end if
end for
end for

Algorithm 3: Stochastic Identification.

Input: $\delta_{o_i}\left(\mathrm{tr}_{l,q_l}\right) = \left\{t^1_{l,q_l}, \ldots, t^k_{l,q_l}\right\}$, $\left(n_{l,q_l}\right)$, α_{ct} for contrast test, CD
Output: $F_0(X)$
search p-value$_{\alpha_{ct}}$ in table of Kolmogorov-Smirnov.
for $j = 1, \ldots, |\mathrm{CD}|$
(i) Design goodness-of-fit test
$H_0 : F_0(X) \sim F_0^j(x_i)$;
$H_1 : F_0(X) \nsim F_0^j(x_i)$.
(ii) Estimate parameters of $F_0^j(X)$.
(iii) Apply Kolmogorov-Smirnov test
(a) Compute p-value $= \sup_{1 \leq i \leq n} \left|\widehat{F}_n(x_i) - F_0^j(x_i)\right|$.
(iv) If p-value $\leq p$-value$_{\alpha_{ct}}$,
then $F_0 = F_0^j(X)$; break
else $j = j + 1$.
(v) end if
end for

Algorithm 4: Data fitting.

estimating the PDF fall into two main categories. One is parametric method that assumes the form of the PDF of the population which is known a priori and the collected process sample is used to determine the parameters of the assumed PDF and another category is nonparametric methods. In this procedure we analyze a set CD of specific continuous distributions known; CD $= \{F_0^j(x_i)\}$. Among the most commonly used continuous distributions it includes the normal distribution $N(\mu, \sigma^2)$, the exponential distribution $\mathrm{Exp}(\lambda)$, and the uniform distribution $U(a, b) \cdots$.

Algorithm 4 computes the distribution type and its parameters. Algorithm 4 uses the goodness-of-fit test (see [40]) to define the distribution type with a Kolmogorov-Smirnov test. First it computes the distribution parameters using the maximum likelihood method; second, it computes the p-value statistic as

$$p\text{-value} = \sup_{1 \leq i \leq n} \left|\widehat{F}_n(x_i) - F_0(x_i)\right|, \tag{9}$$

where x_i is the ith observed value in the sample (whose values have previously been ordered from the lowest to the highest); $\widehat{F}_n(x_i)$ is an estimator of the probability of observing values less than or equal to x_i and $F_0(x_i)$ is the probability of observing values less than or equal to x_i from a specified distribution [41]; that is, $F_0(x_i) \in \mathrm{CD}$.

4.4.1. Modelling Error. The proposed identification algorithm is a tool for identifying stochastic DES which estimates the probability distributions of transitions times. Because full times population cannot be observed, there will be a modelling error. As the sample mean is the estimator of a population mean, then the modelling error is related to the difference between value estimated and the true value of the mean. The standard error of the mean is the way to measure the modelling error. The standard error of the mean is the standard deviation of sample means over all possible samples (of a given size) drawn from the population. To measure it,

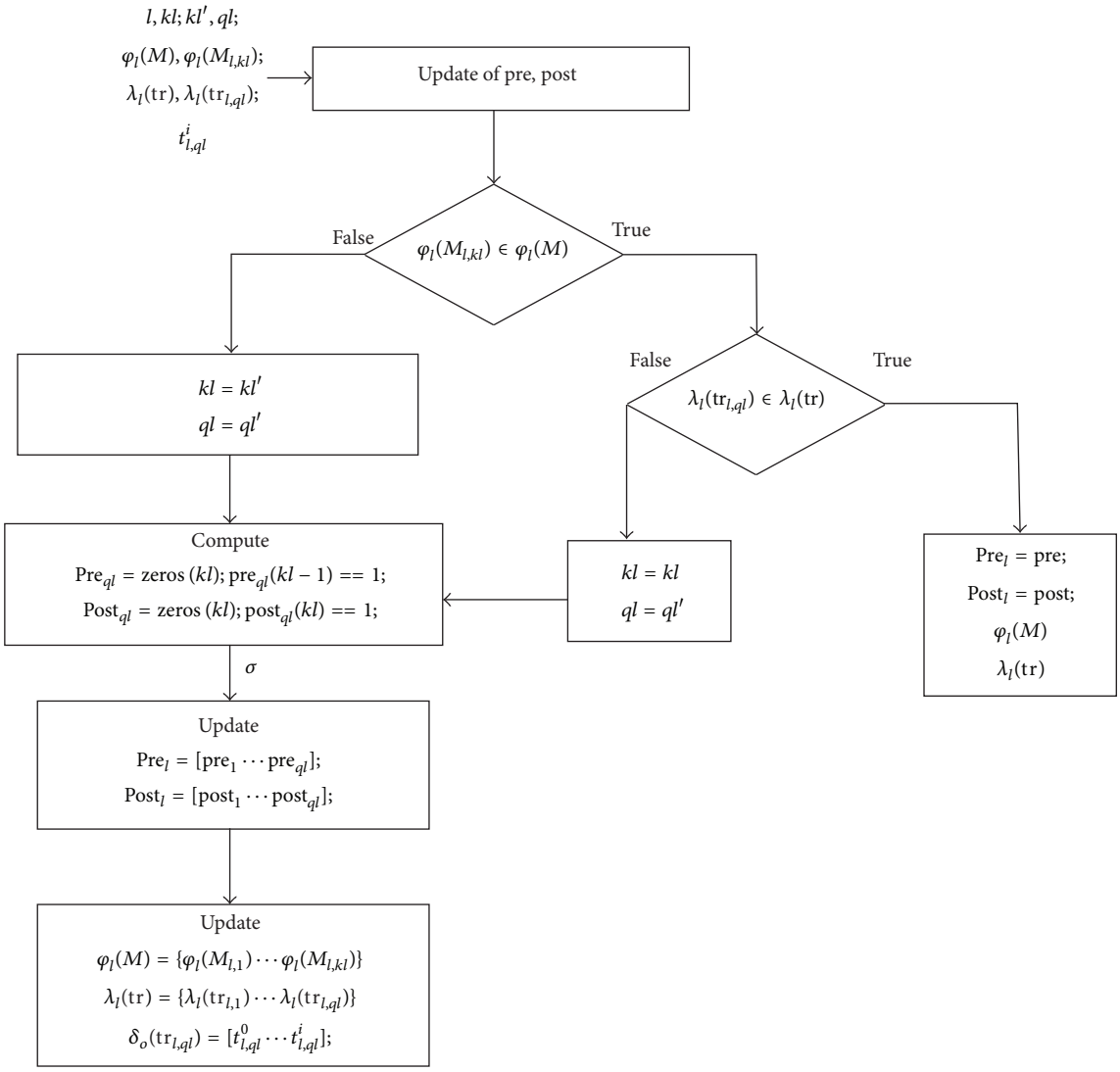


FIGURE 4: Update of Pre_l; Post_l; $\varphi_l(M)$; $\lambda_l(\mathrm{tr})$; $\delta_{o_i}(\mathrm{tr}_{l,ql})$.

the root-mean-square error (RMSE) is used, where RMSE = $S/\sqrt{n}$.

4.5. Global Language Reconstruction from to Language of the Subsystems. Our objective is to identify the language of global system. It is found from the language of its subsystem.

A global system event is obtained by Timed Synchronized Splice so.

Definition 16 (operation of timed synchronized splice). Le Let Q be a system composed by c subsystems. Let ω^i_l with $l = 1 \cdots c$ be c events at time τ_i, where $\omega^i_l = (u_{l,s} y_{l,j})\tau_i$. A global system event, at time τ_i, can be obtained as:

$$\omega^i = \omega^i_1 \oplus \cdots \oplus \omega^i_c \tag{10}$$

where $\omega^i_1 \oplus \cdots \oplus \omega^i_c = (u_{1,s} y_{1,j}) \oplus \cdots \oplus (u_{c,s} y_{c,j}) = (u_{1,s} \oplus \cdots \oplus u_{c,s})(y_{1,j} \oplus \cdots \oplus y_{c,j})$ and $(u_{1,s} \oplus \cdots \oplus u_{c,s}) = u_{s_g}$ where u_{s_g} is a binary representation of s_g, s_g stands for all input symbols and $(y_{1,j} \oplus \cdots \oplus y_{c,j}) = y_{j_g}$ where y_{j_g} is a binary representation of j_g, j_g stands for all output symbols.

Then, the global system language can be built from the sequence of synchronized events of its subsystem. Therefore, the global generated language at times $\tau_0 \leq \ldots \leq \tau_k$ is $\mathscr{L}(stQ) = \omega^0 \cdots \omega^k$.

Proposition 17. *Given a global system Q composed by c subsystems, the global system language $\mathscr{L}(stQ)$ built by the temporal synchronization of the languages of its subsystems is equal to system language, $\mathscr{L}(stQ) = \mathscr{L}(Q)$.*

Proof. Let Q be a complex system with c subsystems. Let $\mathscr{L}(Q) = \omega^0 \omega^1 \cdots \omega^e$ be an event sequence generated from generator defined in Section 4.1.3(a) and let $\mathscr{L}(stQ)$ be the global system language built from a sequence of synchronized events of its subsystems.

Let $s = \omega^0\omega^1\cdots\omega^e$ be an event sequence such that $s \in \mathscr{L}(Q)$ and let $s_1 = \omega^{*0}\cdots\omega^{*e}$ be the sequence of synchronized events of the $\mathscr{L}(stQ_l)$ with $l = 1\cdots c$.

Assuming $\omega^e \neq \omega^{*e}$.

$P_U(s) = \{u_s^0\cdots u_s^e\}$, $u_s^e = \{u_{1,s},\ldots,u_{c,s}\}$ and $P_Y(s) = \{y_j^0\cdots y_j^e\}$, as $\omega_l^i = (u_{l,s}y_{l,j})$ for $l = 1\cdots c$. $\omega_l^e = (u_{l,s}, y_{l,j})$; $\omega_l^e = (u_{1,s}, y_{1,j})\ldots(u_{c,s}, y_{c,j})$. Based on Definition 16 $\omega^{*e} = \omega_1^e \oplus\cdots\oplus\omega_c^e$ then $\omega^e = \omega^{*e}$, which contradicts the initial condition. □

4.5.1. Synchronous Product of st-IPNs. To find the global system st-IPN we have proposed the operation of synchronous product of st-IPNs defined as follows.

Definition 18 (synchronous product of st-IPNs). Given a set st$Q$ of c st-IPNs, st$Q = \{\text{st}Q_1,\ldots,\text{st}Q_c\}$ defined as (5), synchronous product of st$Q$ is the st-IPN defined as:

$$\begin{aligned}\text{st}Qs &= \|_{l=1}^{c}\text{st}Q_l,\\ \text{st}Qs &= (Qs, \Omega s, \delta s),\end{aligned} \tag{11}$$

where $Qs = (Ns, Us, Ys, \lambda s, \varphi s)$. $Ns = (Ps, \text{TR}s, \text{Pre}, \text{Pos}, M_0)$ is an ordinary PN, with $Ps = \{P1\times P2\times\cdots\times Pc\}$ set of places, $\text{TR}s = \{2^{|P_s|-1}\}$ set of system transitions, $|P_s|$ is the length set of places, c number of subsystems, $\text{Pre} : Ps\times\text{TR}s \to Z^+$, $\text{Pos} : Ps\times\text{TR}s \to Z^+$), and $Ms_0 = M1_0\times M2_0\times\cdots\times Mc_0$ is the initial marking. $Us = \{U_1\cup U_2\cup\cdots Uc\}$ is the set of input to the system. $Ys = \{Y_1\cup Y_2\cup\cdots Yc\}$ is the set of output to the system. $\lambda s : \text{TR}s \to Us\cdot\delta s$ (labeling function that assigns an input symbol and a timer density function to each transition) and φs is defined as $\varphi s : R(Ns, Ms_0) \to y_{l,j}/dy_{l,j}$; φs is isomorphic over $y_{l,j}/dy_{l,j}$.

$\Omega s = Us\cdot Ys := \{\omega s = (u_{l,s}y_{l,j})\}$ is the system alphabet.

$\delta s := \text{TR}s\times\text{OM} \to f(t_{\text{TR}s\times\text{OM}})$ is a density function of transition firing time for each operation modes $\text{OM} = \{o_1, o_2,\ldots\}$.

In order to verify that the structure of a st-IPN solves Problem 15, we define the conditions for a PN which is considered a deterministic generator and the conditions for a system to be considered deterministically identifiable.

Definition 19 (α-Generator). Let $U\cdot Y$ be an alphabet and let $\mathscr{L}(Q)$ be a language generated by alphabet; a PN is an α-Generator for $\mathscr{L}(Q)$ if given a set of inputs $u_s \in P_U : (U\cdot Y)^* \to U^*$, PN generates a set of outputs y_j such that the strings $u_s y_j \in \mathscr{L}(Q_l)\forall u_s \in U$ with a probability greeter than $1-\alpha$.

4.5.2. Assumptions of a System so that It Is a Deterministically Identifiable (DI) System. Definition 19 enables us to set the assumptions of a system for it can be a DI system.

(i) Let U and Y be a set of I/O observed signals of system; if given a set of inputs $u_s \in U$ associated with a time t, $P_{y_j} : (Y) \to y_j$ is always the same.

(ii) Moreover, given two identical inputs u_s, but with different length of time and two different projections of outputs, the system can be considered DI, since length of time defines the mark to reach.

Proposition 20. *Let $\mathscr{L}(Q) = \{\omega^0\omega^1\cdots\omega^k\}$ be a system observed language, at times $\tau_0 \le \tau_1 \le\cdots\le \tau_k$; if st-IPN is a PN obtained with proposed identification algorithm then st-IPN is a α-Generator of $\mathscr{L}(Q)$.*

Proof. A st-IPN is a α-Generator of $\mathscr{L}(Q)$ if $\forall s \in \mathscr{L}(Q)$, $s \in \mathscr{L}(P)$ and if $\forall s \in \mathscr{L}(P)$, $s \in \mathscr{L}(Q)$.

Let P be a st-IPN obtained by proposed identification algorithm for $\mathscr{L}(Q)$.

Assuming that ss_1 is a string such that $s \in \mathscr{L}(Q)\wedge\mathscr{L}(P)$ but $s_1 \in \mathscr{L}(Q)\wedge s_1 \notin \mathscr{L}(P)$, where $s = \omega^0,\ldots,\omega^k$, then the postlanguage of s is given by $\mathscr{L}(Q)/s = \{s_1 \in (U\cdot Y)^* \mid ss_1 \in \mathscr{L}(Q)\}$. Then given the event s_1 with $s_1 = \omega^{k+1} = (u_s y_j)^{k+1}$ at time τ_{k+1}, applying the Algorithm 1 a new event generates a transition with $\lambda(\text{tr}_{k+1}) = u_s^{k+1}\cdot t_{\text{tr}_{k+1}}$ and the marking reached by tr_{k+1}: $M_k \xrightarrow{\text{tr}_{k+1}} M_{k+1}$ will be $\varphi(M_{k+1}) = y_j^{k+1}/y_j^{k+1} - y_j^k$ and as $u_s^{k+1}\cdot y_j^{k+1}$ are signals vectors observed at time τ_{k+1}, (based on description of event generator, Section 4.1.3(a)), then $\omega^{k+1} = (u_s y_j)^{k+1} \in \mathscr{L}(P)$ then assumption is rejected.

Now, we assume that ss_2 is a string such that $s \in \mathscr{L}(Q)\wedge\mathscr{L}(P)$ but $s_2 \notin \mathscr{L}(Q)\wedge s_2 \in \mathscr{L}(P)$, where $s = \omega^0,\ldots,\omega^k$. Then the postlanguage of s is given by $\mathscr{L}(P)/s = \{s_2 \in (U\cdot Y)^* \mid ss_2 \in \mathscr{L}(P)\}$. Then given the event s_2 with $s_2 = \omega^{k+1} = (u_s y_j)^{k+1}$ at time τ_{k+1}, the st-IPN evolves a state from to a transition with $\lambda(\text{tr}_{k+1}) = u_s^{k+1}\cdot t_{\text{tr}_{k+1}}$ and the marking reached by tr_{k+1}: $M_k \xrightarrow{\text{tr}_{k+1}} M_{k+1}$ will be $\varphi(M_{k+1}) = y_j^{k+1}/y_j^{k+1} - y_j^k$, where tr_{k+1} has not been observed and reaches states that have not been observed, but according to Algorithm 1 the construction of st-IPN is made based on reached state by the observed language, then $\omega^{k+1} = (u_s y_j)^{k+1} \notin \mathscr{L}(P)$ and then assumption is rejected and therefore Proposition 20 is proved. □

Moreover, $\mathscr{L}(Q)/s = \{s_1 \in (U\cdot Y)^* \mid ss_1 \in \mathscr{L}(Q)\}$ reaches the mark $\varphi(M_{k+1})$ if $\text{Prob}(\varphi(M_{k+1}), \omega^{k+1}/\varphi(M_k)) \geqq (1-\alpha) \mid \omega^{k+1} \in \mathscr{L}(Q)$ and if $\varphi(M_{k+1}) = \varphi(M_i)$ for some $i = 1\cdots k$, then ss_1 is a cycle (Proposition 12).

Definition 21. A $\mathscr{L}(Q)$, where $\mathscr{L}(Q) = \{s \in (U\cdot Y)^* : \text{Prob}(s \mid \varphi(M_0)) \geqq (1-\alpha)\}$ is DI language if $\forall s \in \mathscr{L}(Q), \wedge\forall s_1, s_2 \in \mathscr{L}(Q)/s$, where $s = \omega^0,\ldots,\omega^k$, such as $P_{u_s} : (ss_1) = P_{u_s} : (ss_2)$ then $\text{Prob}(P_{y_j} : (ss_1) = P_{y_j} : (ss_2)) \geq 1-\alpha$ if they started from the same state $\varphi(M_k)$.

In other words, a $\mathscr{L}(Q)$ is DI language, if from a state two equal inputs are given, the output symbol is always the same.

Theorem 22. *A language $\mathscr{L}(Q)$ is DI language if and only if $\exists$ an α-Generator for $\mathscr{L}(Q)$.*

Proof. Necessary condition: if $\mathscr{L}(Q)$ is a DI language, then $\exists$ an α-Generator.

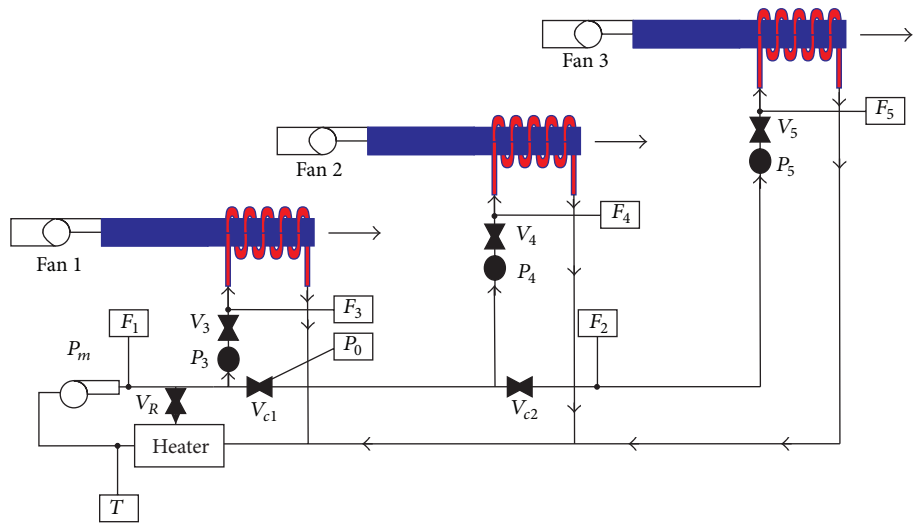


Figure 5: AHS system.

A deterministic α-Generator for $\mathscr{L}(Q)$ is a st-IPN. $\mathscr{L}(Q)$ is a DI language, if α-Generator is a st-IPN (see Proposition 20).

Sufficient condition: if $\exists$ an α-Generator then $\mathscr{L}(Q)$ is a DI language.

Assume that the α-Generator for $\mathscr{L}(Q)$ is a st-IPN. Let $\mathscr{L}(P)$ be the generated language by α-Generator. Let $s = \omega^0, \ldots, \omega^k$ be a string such that $\mathscr{L}(P) = \{s \in (U \cdot Y)^* : \text{Prob}(s \mid \varphi(M_0)) \geqq (1 - \alpha)\}$. Then, given the events s_1 and s_2 such that $s_1, s_2 \in \mathscr{L}(P)/s$ with $s_1, s_2 = \omega^{k+1}$ with $\text{Prob}(\varphi(M_{k+1}), \omega^{k+1}/\varphi(M_k)) \geqq (1 - \alpha)$ if $P_{u_s} : (ss_1) = P_{u_s} : (ss_2)$, in the event ω^{k+1} then $P_{y_j} : (ss_1) = P_{y_j} : (ss_2)$, based on Definition 21 $\mathscr{L}(P)$ is a DI language and according to Proposition 13 $\mathscr{L}(P) = \mathscr{L}(Q)$; therefore $\mathscr{L}(Q)$ is a DI language. □

If a st-IPN is a deterministic α-Generator for $\mathscr{L}(Q)$, then st-IPN generates the same observed language; therefore Problem 15 is solved.

5. Application Example

Let us illustrate the approach with an example. The proposed example is the centralized air heating system (AHS) in Figure 5. The system includes three heating subsystems. Each heating subsystem has a fan creating an air flow that is heated with hot water. The water flow is controlled by pump-valve systems. Moreover, there is a central heater providing hot water to each heating subsystem and two valves (v_{c1} and v_{c2}) controlling the water flow through the whole system. The system can be split into five subsystems (1, 2, 3, 4, and 5). Subsystems 3, 4, and 5 are the local heaters, subsystem 2 is the distribution subsystem (v_{c1} and v_{c2}), and subsystem 1 is the main heating subsystem (heater, main pump (p_m), and reflux valve (v_r)). The heater works in three modes (0, 1, 2), each state is defined by the number of resistances it has activated, so, modes 0, 1, and 2 will represent no activation of resistance, activation of h_1, and activation of both h_1 and h_2.

The system is equipped with a set of sensors detailed in Table 1. Each subsystem i includes a flow sensor (F_i) that measures the presence or absence of flow in the subsystem. Nevertheless, flow level is affected when other subsystems are activated. So, a software sensor (NF_i) is designed in order to measure the deviation over a normal operation flow taking into account the activation of other subsystems.

The system also includes binary temperature sensors, pressure sensors, and a position sensor for valve Vc_1.

$u_{l,s}y_{l,j}$ is an I/O symbol based on Definition 3, where $u_{l,s}$ is a binary representation of s, s stands for the input symbol, $y_{l,j}$ is a binary representation of j, j stands for the output symbol, and l stands the subsystem index. For example, $(u_{1,30}y_{1,9})$ is an I/O symbol in subsystem 1, whose values are [11110, 1001], that is, $[v_r p_g v_g h_1 \overline{h}_2, T_o \overline{T}_1 \overline{T}_2 F_1]$.

5.1. System Operation. The system globally starts with the external event "*Son*"; the heating subsystems are locally started with events "*Ca*3," "*Ca*4," and "*Ca*5". These events are external events that change the controller strategy (See in Figure 6). Each combination of external events generates a system operation mode. When event "*Son*" is generated, the controller opens the reflux valve (v_r) and starts Pm (p_m), the heater starts heating (temperature set point is Tp_1) and the water flows through the path Heater-Pm-VR-Heater. When any heating subsystem is activated ("*Ca*3," "*Ca*4," or "*Ca*5"), subsystem 1 closes reflux valve ($\overline{v}_r$) and it changes the temperature set point to Tp_2 ($Tp_2 > Tp_1$). If "*Ca*4" and (or) "*Ca*5" are started, then, the corresponding valves 1 and 2 of subsystem 2 are opened (v_{c1}, v_{c2}). Moreover local valves and pumps start their work when necessary (v_3, v_4, v_5 and p_3, p_4, p_5). Each heating subsystem is stopped with events "$\overline{Ca3}$," "$\overline{Ca4}$," and "$\overline{Ca5}$" ($\overline{Ca} = \overline{Ca3} \wedge \overline{Ca4} \wedge \overline{Ca5}$). Then, the controller performs the closing actions; it closes the valves ($\overline{v}_{c1}, \overline{v}_{c2}, \overline{v}_3, \overline{v}_4, \overline{v}_5$) and stops the pumps ($\overline{p}_3, \overline{p}_4, \overline{p}_5$). When

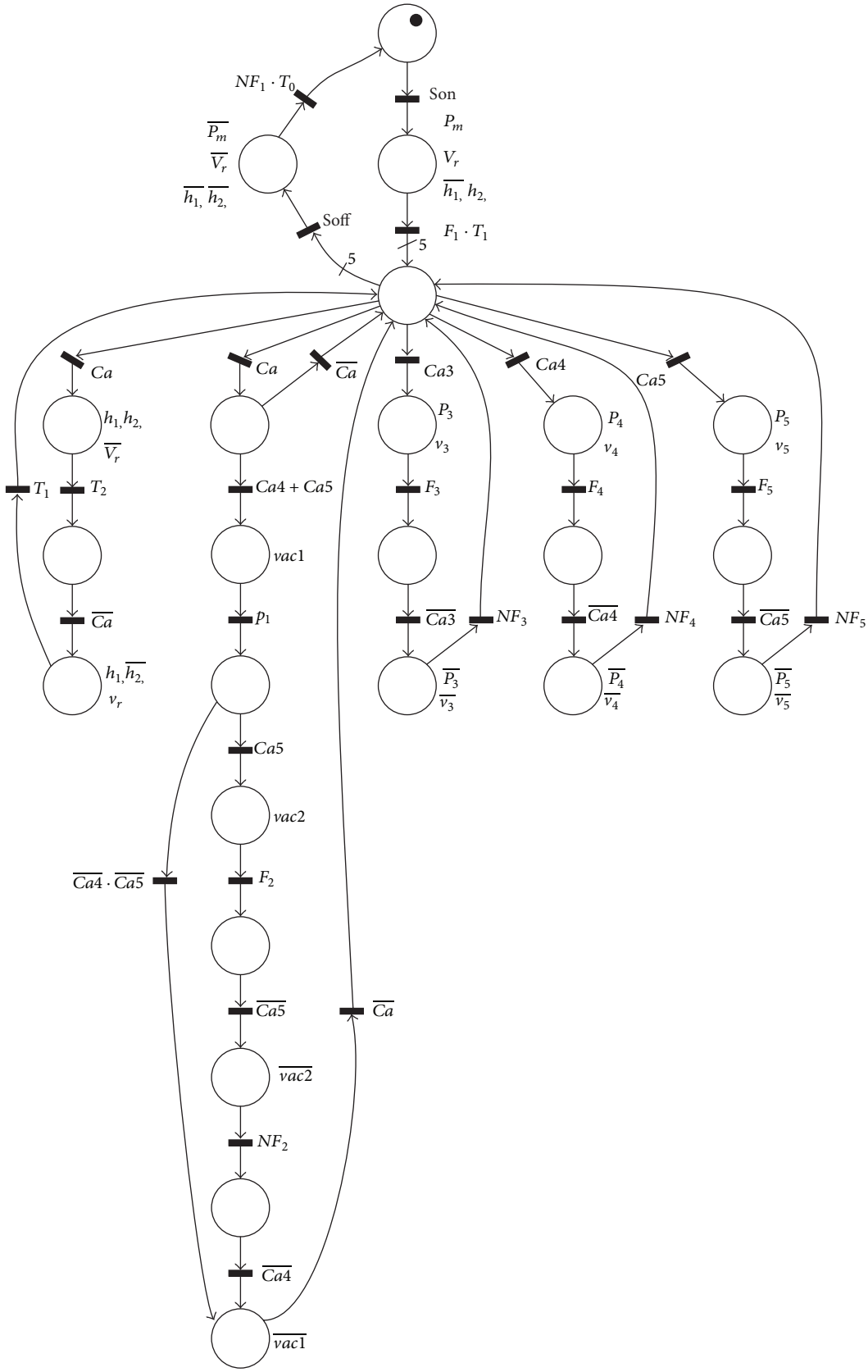
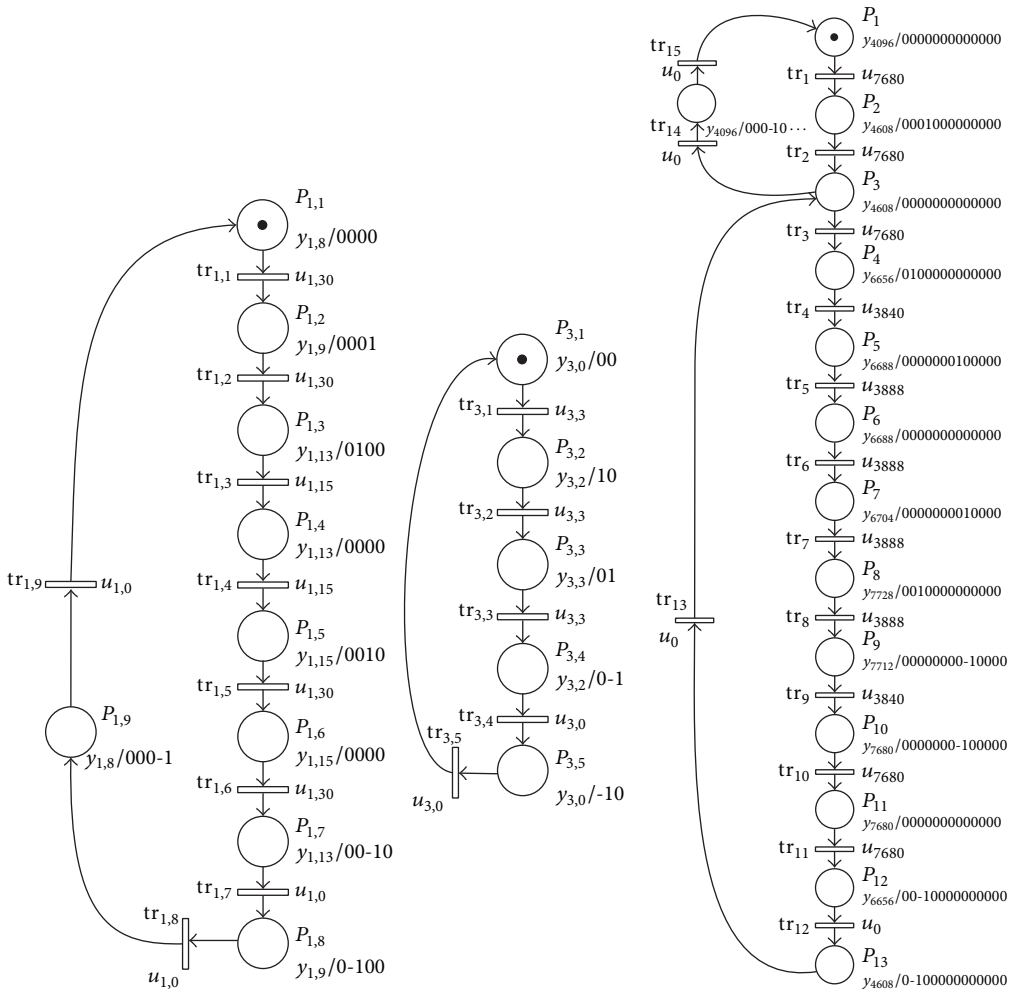


Figure 6: AHS system controller.



(a) Subsystems 1 and 3 (b) Global

Figure 7: st-IPNs system scenario 1.

every subsystem is completely stopped, the system can be globally stopped with event "*Soff*." Then, the controller closes the reflux valve ($\overline{v}_r$), and stops the heater and Pm ($\overline{p}_m$).

In order to demonstrate the advantages of the proposed identification method, the system will be identified over two scenarios.

The first test presented is the stand alone operation of subsystem 3 and the second test presented simulates all the possible operation modes sequentially. The identification method is additive, so after the identification of a language associated with a particular mode, the identified nets learn new languages by adding them to the previous knowledge.

Because the system cannot work if *Son* is not activated, there are 9 possible operations modes: OM $= \{o_0, \ldots, o_8\}$; where $o_0 =$ Soff, $\overline{Ca3}, \overline{Ca4}, \overline{Ca5}$, $o_1 =$ Son, $\overline{Ca3}, \overline{Ca4}, \overline{Ca5}$, and $o_2 =$ Son, $Ca3, \overline{Ca4}, \overline{Ca5}$; so on. Moreover, each test includes the starting and stopping operations. Therefore testing of operation mode o_x includes the sequence (o_0 - o_1 - o_x - o_1 - o_0).

5.2. Identification of Scenario 1. The number of subsystems is 5 and the initial conditions are as follows.

(i) Definition of initial state at τ_0: $\omega^0 = \{\omega_1^0 \cdot \omega_2^0 \cdot \omega_3^0 \cdot \omega_4^0 \cdot \omega_5^0\}$:

$\omega_1^0 = [\overline{v}_r \overline{p}_g \overline{v}_g \overline{h}_1 \overline{h}_2, T_o \overline{T}_1 \overline{T}_2 \overline{F}_1] = [u_{1,0} y_{1,8}]$;

$\omega_2^0 = [\overline{v}_{c1} \overline{v}_{c2}, \overline{P}_o \overline{F}_2 \overline{NF}_2] = [u_{2,0} y_{2,0}]$;

$\omega_3^0 = [\overline{p}_3 \overline{v}_3, \overline{F}_3 \overline{NF}_3] = [u_{3,0} y_{3,0}]$;

$\omega_4^0 = [\overline{p}_4 \overline{v}_4, \overline{F}_4 \overline{NF}_4] = [u_{4,0} y_{4,0}]$;

$\omega_5^0 = [\overline{p}_5 \overline{v}_5, \overline{F}_5 \overline{NF}_5] = [u_{5,0} y_{5,0}]$.

(ii) Initial Output Function:

$P_{1,1} : \varphi(M_{1,1}) = y_{1,8}/\vec{0}$;

$P_{2,1} : \varphi(M_{2,1}) = y_{2,0}/\vec{0}$;

$P_{3,1} : \varphi(M_{3,1}) = y_{3,0}/\vec{0}$;

$P_{4,1} : \varphi(M_{4,1}) = y_{4,0}/\vec{0}$;

$P_{5,1} : \varphi(M_{5,1}) = y_{5,0}/\vec{0}$.

Table 1: Sensor Readings.

Sensor	Subs	Readings	Meaning
Temperature T	1	T_0, T_1, T_2,	if $T < Tp_1$ $T = T_0$, if $Tp_1 < T < Tp_2$, $T = T_1$, if $T_2 > Tp_2$ $T = T_2$
Flow	1	F_1, NF_1,	Flow, normal flow.
Position	2	P_o	Valve Vc_1 opened or closed
Flow	3	F_2, NF_2	Flow, normal flow.
Flow	3	F_3, NF_3	Flow, normal flow.
Flow	4	F_4, NF_4	Flow, normal flow.
Flow	5	F_5, NF_5	Flow, normal flow.
Pressure	3	P_3	Pressure.
Pressure	4	P_4	Pressure.
Pressure	5	P_5	Pressure.

The input vector arranges the control commands based on system operation as described in Section 5.1. The output vector arranges the sensor readings as in Table 1.

Once the system starts with "*Son*"; the controller generates the control commands when the external inputs evolve. Then the system reacts and the event generator starts generating events. (Note that the system starts with operation mode o_0, then it changes to o_1, and, when "*Ca3*" is externally activated, it changes to o_2.)

With each new event, Algorithm 1 creates new places and transitions. After a complete run of the system (*Soff* is generated), Algorithm 1 creates the st-IPNs for subsystems 1 and 3 showed in Figure 7(a).

At this identification stage, places and transitions are defined and the identified st-IPNs model the expected behavior of the system.

Initially, the system works in operation mode 0 (o_0) as Son is not active. Subsystem 1 starts at place $P_{1,1}$. Activation of "*Son*" changes the operation mode to o_1, the controller opens the valves (reflux and main), turns on the main pump, and sets the heater in mode 1 ($\mathrm{tr}_{1,1}$). A state change ($P_{1,2}$) is generated, because water flows through the path Heater-Pm-VR-Heater ($F_1 = 1$) (as it is showed in Figure 5). After a time ($\mathrm{tr}_{1,2}$) temperature reaches Tp_1 (18°C); therefore, the system evolves to other state ($P_{1,3}$). The controller starts the heating actions; it closes the reflux valve and sets heater in mode 2 ($\mathrm{tr}_{1,3}$), the state change cannot be measured by absence of sensors that detect the operation of the reflux valve ($P_{1,4}$). At this time "*Ca3*" is activated, so the system operation mode changes to o_2; then the controller turns on pump 3 and opens valve 3 ($\mathrm{tr}_{3,1}$) and subsystem 3 changes its state to $P_{3,2}$ because water is flowing through path Heater-Pm-P3-V3-Heater. After a time ($\mathrm{tr}_{3,2}$), flow reaches its normal level (NF_3) and subsystem 3 changes the state to $P_{3,3}$. Subsystem 1 continues with the reflux valve closed ($\mathrm{tr}_{1,4}$). It waits until reaching Tp_2 (28°C) and, therefore, it evolves to $P_{1,5}$. Subsystem 3 remains in $P_{3,3}$ until "*Ca3*" is externally deactivated ($\overline{Ca3}$) and o_1 is reached, then normal flow falls ($\mathrm{tr}_{3,3}$) and it changes to ($P_{3,4}$). Now, the controller turns off pump 3 and closes valve 3 ($\mathrm{tr}_{3,4}$). As a consequence, flow in the subsystem decreases (change to $P_{3,5}$) until flow level is low F_1. This transition makes the subsystem return to its initial state ($P_{3,1}$). Now, the controller opens the reflux valve and sets the heater in mode 1 ($\mathrm{tr}_{1,5}$). Subsystem 1 holds the state ($P_{1,6}$) until temperature is below Tp_2, ($P_{1,7}$), after the controller turns off the main pump, it closes the main valve and sets the heater in mode 0 ($\mathrm{tr}_{1,7}$). Subsystem 1 evolves to $P_{1,8}$ and after time ($\mathrm{tr}_{1,8}$) to $P_{1,9}$. Slightly, temperature decreases until it reaches the initial state ($P_{1,1}$). At the end, Son is deactivated (*Soff*) and the system operation mode changes to o_0.

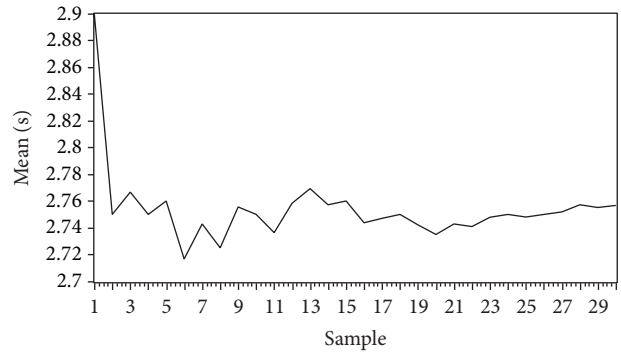


Figure 8: Behavior of transition $\mathrm{tr}_{3,2}$.

Nevertheless, the time array for each transition has only one value, so no probability distribution can be fit.

As stated in Section 4, at least 5 samples are needed to identify the distributions, so scenario 1 will be repeated until each distribution for each transition will be identified.

After 5 observations of a transition, Algorithm 3 is executed. This paper shows, as an example, the case with the greatest dispersion ($\mathrm{tr}_{3,2}$) as it is the worst case.

Inputs: $\delta_{o_2}(\mathrm{tr}_{3,2}) = \{t^1_{3,2} \cdots t^5_{3,2}\}$; $\alpha = 0.05$; $\alpha_{ct} = 0.05$; $d_{op} = 0.05$; CD $= \{F^j_0(x_i)\}$

$l = 3$, $q_l = 2$.

for $n = 5$;

$\delta_{o_2}(\mathrm{tr}_{3,2}) = \{2.9, 2.6, 2.8, 2.7, 2.8\}$

$\overline{X} = 2.76$, $S = 0.11$, $d = 0.1027$.

As $d > 0.05$, there is not enough data to fit the distribution.

After 20 observations, Algorithm 3 gives a positive result: $\overline{X} = 2.75$, $S = 0.09$; $d = 0.049$. As $d < 0.05$, Algorithm 4

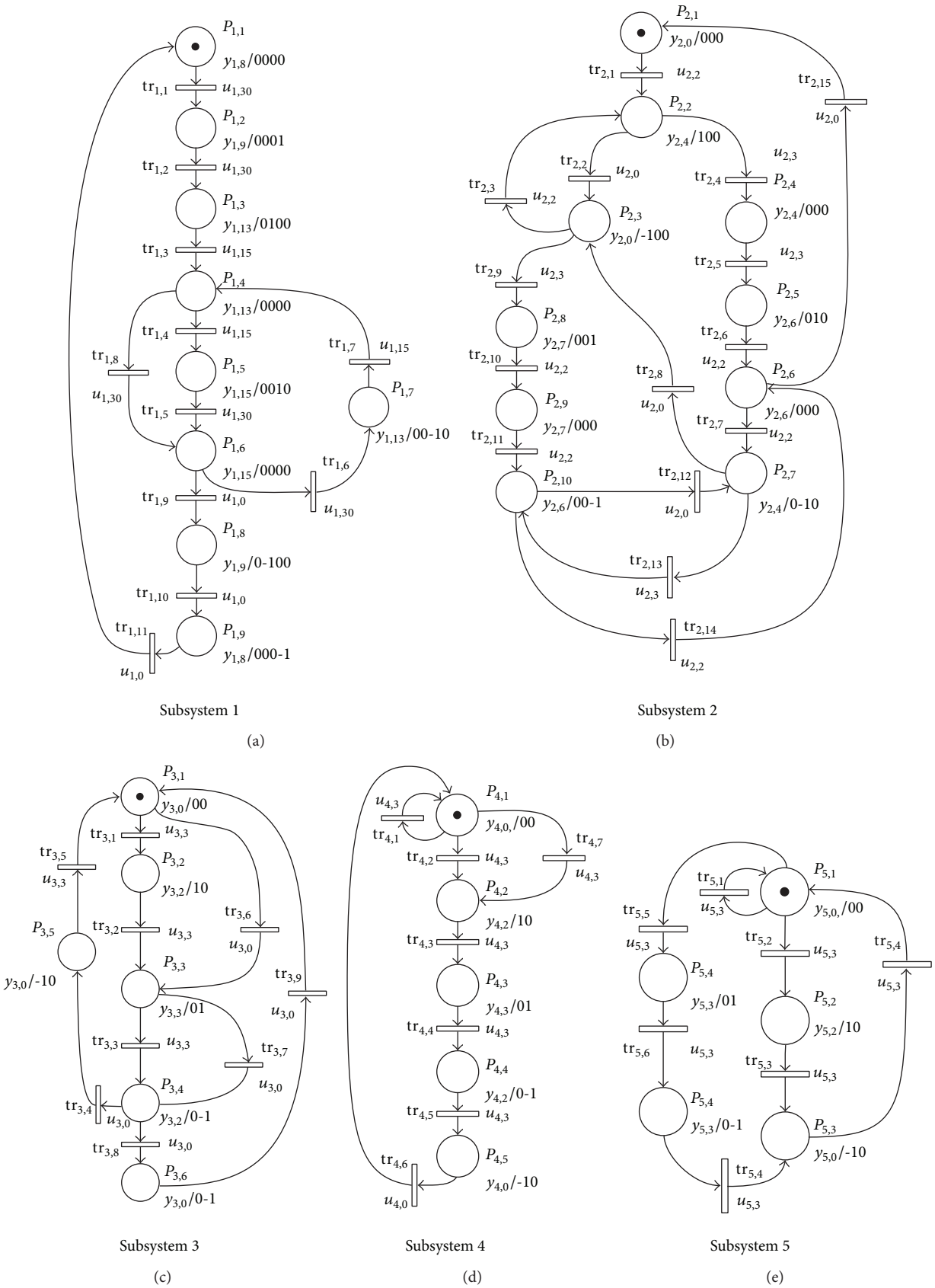


Figure 9: st-IPNs after the complete identification.

is executed and it will compute the distribution type and its parameters.

$\alpha_{ct} = 0.05$. $p\text{-value}_{\alpha_{ct}} = 0.29408$ in table of Kolmogorov-Smirnov, with $n = 20$.

Hypothesis test:

H_0: $F_0(X) \backsim N(\mu, \sigma^2)$;

H_1: $F_0(X) \nsim N(\mu, \sigma^2)$.

Parameters estimation $F_0 = N(\mu, \sigma^2) = N(2.75, 0.09)$ and *p-value* $= 0.1621$ based on test Kolmogorov-Smirnov.

Accept H_0 since $p\text{-value} \leq p\text{-value}_{\alpha_{ct}}$. Therefore the data is normally distributed.

To do a full identification of the process it is necessary to get 20 samples of each transition time.

Figure 8 shows the mean of the time array collected for $\text{tr}_{3,2}$ versus the number of acquired samples in operation mode 2. It can be seen that for 20 samples onwards the mean remains stable.

Thereby, the sequence of operation modes o_0 - o_1 - o_2 - o_1 - o_0 is fully identified. This identification includes the st-IPNs structures and all the density functions for each transition in the net.

5.3. Complete Identification of the System. The last test performed is the sequential generation of all the possible combinations of the external events, so the system must be able to generate a really complex language. Subsystem st-IPNs are showed in Figure 9.

Results show the advantages of identifying subsystems instead of the global system. Figure 7(b) shows the identification of the global system (without splitting in subsystems) in scenario 1. The complexity (number of places and transitions) of this st-IPN is similar to the complexity of the st-IPNs in the case of the subsystem identification. Nevertheless, as the complexity of the system increases, subsystem identification reduces complexity. Table 2 shows a comparison of the number of sates and transitions when identifying with or without splitting the system into subsystems.

It can be seen that a global system identification increases its complexity with the number of operation modes considered, as a subsystem identification does not.

This behavior is related to the number of possible events handled by the st-IPNs. The maximum number of global system states is $Q(g)_{\max} = 2^{n_t} \times 3^{n_t}$, ($n_t$ is the total number of output symbols) (6), whereas in identification by subsystem the maximum number of system states is $Q(\text{split})_{\max} = Q_{\max,1} + Q_{\max,2} + \cdots + Q_{\max,c}$; that is, $Q(\text{split})_{\max} = 2^{n_1} \times 3^{n_1} + 2^{n_2} \times 3^{n_2} + \cdots + 2^{n_c} \times 3^{n_c}$, where $n_1 + n_2 + \cdots + n_c = n_t$, it can be seen that $Q(g)_{\max} > Q(\text{split})$.

5.4. Language Generation. This section shows how the observed language is generated in both subsystem and global identification approach for scenario 1.

The observed language for each subsystem is equal to the one generated by the st-IPNs.

Table 2: Places and transitions.

Scenario	Sub		Global	
	Places	Tr.	Places	Tr.
1	7	2	3	2
2	14	14	14	15
3	22	22	22	23
4	28	32	24	31
5	28	32	24	32
6	28	32	24	32
7	28	32	24	32
8	28	32	24	32
9	28	33	33	42
10	28	33	39	52
11	32	41	44	59
12	32	42	50	68
13	32	43	52	71
14	33	43	54	74
15	34	45	57	78
16	35	48	61	83

$\mathscr{L}_1 = (u_{1,0}y_{1,8})(u_{1,30}y_{1,8})(u_{1,30}y_{1,9})(u_{1,30}y_{1,13})(u_{1,15}y_{1,13})(u_{1,15}y_{1,15})(u_{1,30}y_{1,15})(u_{1,30}y_{1,13})(u_{1,0}y_{1,9})(u_{1,0}y_{1,8})$ at times $\tau_0, \tau_1, \tau_2, \tau_3, \tau_6, \tau_{10}, \tau_{11}, \tau_{12}, \tau_{13}, \tau_{14}$.

$\mathscr{L}_3 = (u_{3,0}y_{3,0})(u_{3,3}y_{3,2})(u_{3,3}y_{3,3})(u_{3,3}y_{3,2})(u_{3,0}y_{3,0})(u_{3,0}y_{3,0})$ at times $\tau_0, \tau_4, \tau_5, \tau_7, \tau_8, \tau_9$.

Figure 7(b) shows the result of the identification process in case that the system is not split into subsystems. The language observed also is equal to the generated.

$\mathscr{L}_g = (u_0y_{4096})(u_{7680}y_{4608})(u_{7680}y_{4608})(u_{7680}y_{6656})(u_{3840}y_{6688})(u_{3888}y_{6688})(u_{3888}y_{6704})(u_{3888}y_{7728})(u_{3888}y_{7712})(u_{3840}y_{7680})(u_{7680}y_{7680})(u_{7680}y_{6656})(u_0y_{4608})(u_0y_{4608})(u_0y_{4096})(u_0y_{4096})$ at times $\tau_0, \ldots, \tau_{15}$.

A global input symbol is

$$u_g = \left\{ \overbrace{v_r p_g v_g h_1 h_2}^{u_1} \overbrace{v_{c1} v_{c2}}^{u_2} \overbrace{p_3 v_3}^{u_3} \overbrace{p_4 v_4}^{u_4} \overbrace{p_5 v_5}^{u_5} \right\}. \tag{12}$$

And a global output symbol is

$$y_g = \left\{ \overbrace{T_0 T_1 T_2 F_1}^{y_1} \overbrace{P_o F_2 NF_2}^{y_2} \overbrace{F_3 NF_3}^{y_3} \overbrace{F_4 NF_4}^{y_4} \overbrace{F_5 NF_5}^{y_5} \right\}. \tag{13}$$

Note that, for example, at time τ_3, the input symbol in the global language is u_{7680} and at time τ_4 is u_{3840} (see Figure 7(b)). At these times, input symbols in local subsystems are $u_{1,30}$ and $u_{3,3}$. In this time the output symbol in the global language is y_{6656} and at time τ_4 is y_{6688} (see Figure 7(b)). Moreover, at these times output symbols in local subsystems are $y_{1,13}$ and $y_{3,2}$. Table 3 shows these symbols.

According to Definition 16, a global I/O symbol, ω^i, is obtained from splice of $\omega^i = \omega_1^i \oplus \omega_2^i \oplus \cdots \oplus \omega_5^i$ for $i = 3, 4$. Therefore the global and the subsystem approach represent the same language. Then, time synchronization splice of $\mathscr{L}_1$ and $\mathscr{L}_3$ gives $\mathscr{L}_g$.

Table 3: I/O SYMBOL IN τ_3 and τ_4.

	τ_3					τ_4						τ_3					τ_4				
u_g	11110	00	00	00	00	01111	00	11	00	00	y_g	1101	000	00	00	00	1101	000	10	00	00
u_1	11110					01111					y_1	1101					1101				
u_2		00					00				y_2		000					000			
u_3			00					11			y_3			00					10		
u_4				00					00		y_4				00					00	
u_5					00					00	y_5					00					00

Table 4: Parameters of $tr_{3,2}$.

Operation Mode	Distribution
o_0	NA*
o_1	NA
o_2	N (2.75, 0.094)
o_3	NA
o_4	NA
o_5	N (1.44, 0.0966)
o_6	NA
o_7	N (1.35, 0.085)
o_8	N (0.37, 0.0483)

* "Not applicable" means that the transition not is executed in this operation mode.

Therefore, the advantage of using subsystems is that the method identifies only active languages, which allows determining which subsystem is operating.

5.5. Timed Transitions Identification in Different Operation Modes. Transition time depends on the operation mode of the system although the device maintains its operation sequence.

Table 4 shows the parameters of $tr_{3,2}$ distribution for each operation mode. So, although subsystem 3 uses the same net for each operation mode (Figure 9) it uses a different distribution in each transition for each operation mode, so it can model complex behaviors.

Note that time changes can be notable with changes from 0.37 seconds to 2.75 seconds.

6. Conclusions

The proposed method identifies stochastic DES without previous model as a set of st-IPNs modelling the subsystems. The identification algorithm uses on-line I/O data to identify each subsystem, and it defines st-IPNs that generate the same language as the observed one. It has been proved that these st-IPNs are unique and deterministic, a desired feature for applications such as diagnosis. In addition the proposed method identifies systems with stochastic time. This is of great importance in industrial systems because processes are affected by a number of aspects such as the variability in the material flow, the fluctuation of power supply, the deterioration of machinery and devices, among others.

The identification methodology is incremental and is directly related to the objective of modeling, that is, modeling the devices that compose the system in a constructivist way. This approach allows identifying all observable behaviors of the devices.

Some of the main advantages of the methodology (with respect to the existing ones) are that it handles a lot of inputs and outputs with low computational cost (as they are split into subsystems) and it does not need a previous knowledge of the number of places neither the length of the language. Moreover, it generates reduced size models and it includes timed information. Besides, this proposal includes a process to determine whenever the system is fully identified which has not been proposed in other research.

Future research will deal with a forgetting algorithm in order to forget behaviors that will be not more in use.

Conflict of Interests

The authors declare that there is no conflict of interests regarding this paper.

Acknowledgment

This work was supported by a Grant from the Universidad del Cauca, reference 2.3-31.2/05 2011.

References

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, New York, NY, USA, 2008.

[2] Y. Zhang, J. An, and C. Ma, "Fault detection of non-Gaussian processes based on model migration," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1517–1526, 2013.

[3] F. Basile, P. Chiacchio, J. Coppola, and G. De Tommasi, "Identification of Petri nets using timing information," in *Proceedings of the 3rd International Workshop on Dependable Control of Discrete Systems (DCDS '11)*, pp. 154–161, June 2011.

[4] A. Ichikawa and K. Hiraishi, "Analysis and control of discrete event systems represented by Petri nets," in *Discrete Event Systems: Models and Applications*, P. Varaiya and A. Kurzhanski, Eds., vol. 103 of *Lecture Notes in Control and Information Sciences*, pp. 115–134, Springer, Berlin, Germany, 1988.

[5] M. P. Fanti, A. M. Mangini, and W. Ukovich, "Fault detection by labeled Petri nets in centralized and distributed approaches," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 392–404, 2013.

[6] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, 2010.

[7] H. Hu, M. Zhou, Z. Li, and Y. Tang, "An optimization approach to improved Petri net controller design for automated manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 772–782, 2013.

[8] H. Hu, M. Zhou, and Z. Li, "Supervisor optimization for deadlock resolution in automated manufacturing systems with Petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 4, pp. 794–804, 2011.

[9] K. Hiraishi, "Construction of a class of safe Petri nets by presenting firing sequences," in *Application and theory of Petri nets 1992*, K. Jensen, Ed., vol. 616 of *Lecture Notes in Computer Science*, pp. 244–262, Springer, Berlin, Germany, 1992.

[10] R. S. Sreenivas and B. H. Krogh, "Petri net based models for condition/event systems," in *Proceedings of the American Control Conference*, pp. 2899–2904, Boston, Mass, USA, June 1991.

[11] M. Rausch and H.-M. Hanisch, "Net condition/event systems with multiple condition outputs," in *Proceedings of the INRIA/IEEE Symposium on Emerging Technologies and Factory Automation (ETFA '95)*, vol. 1, pp. 592–600, October 1995.

[12] S. Patil, V. Vyatkin, and M. Sorouri, "Formal verification of Intelligent Mechatronic Systems with decentralized control logic," in *Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '12)*, pp. 1–7, Krakow, Poland, September 2012.

[13] A. P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, "A comparative analysis of recent identification approaches for discrete-event systems," *Mathematical Problems in Engineering*, vol. 2010, Article ID 453254, 21 pages, 2010.

[14] S. Shu and F. Lin, "I-detectability of discrete-event systems," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 1, pp. 187–196, 2013.

[15] L. Li and C. N. Hadjicostis, "Least-cost transition firing sequence estimation in labeled Petri nets with unobservable transitions," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 394–403, 2011.

[16] M. E. Meda-Campana and E. Lopez-Mellado, "Required event sequences for identification of discrete event systems," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 4, pp. 3778–3783, December 2003.

[17] M. Meda-Campana and E. Lopez-Mellado, "Identification of concurrent discrete event system using Petri nets," in *Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics*, pp. 11–15, Paris, France, July 2005.

[18] S. Klein, L. Litz, and J.-J. Lesage, "Fault detection of discrete event systems using an identification approach," in *Proceedings of the 16th Triennial World Congress of International Federation of Automatic Control (IFAC '05)*, pp. 92–97, July 2005.

[19] M. Roth, J.-J. Lesage, and L. Litz, "An FDI method for manufacturing systems based on an identified model," in *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM '09)*, vol. 13, pp. 1406–1411, June 2009.

[20] P. Supavatanakul, J. Lunze, V. Puig, and J. Quevedo, "Diagnosis of timed automata: theory and application to the DAMADICS actuator benchmark problem," *Control Engineering Practice*, vol. 14, no. 6, pp. 609–619, 2006.

[21] D. E. Jarvis, "An identification technique for timed event systems," in *Proceedings of the 10th International Workshop on Discrete Event Systems*, vol. 10, pp. 181–186, Berlin, Germany, 2010.

[22] M. Dotoli, M. P. Fanti, and A. M. Mangini, "An optimization approach for identification of Petri nets," in *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES '06)*, pp. 332–337, July 2006.

[23] M. Dotoli, M. P. Fanti, and A. M. Mangini, "On line identification of discrete event systems via Petri nets: an application to monitor specification," in *Proceedings of the 3rd IEEE International Conference on Automation Science and Engineering (CASE '07)*, pp. 893–898, September 2007.

[24] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Ukovich, "On-line identification of Petri nets with unobservable transitions," in *Proceedings of the 9th International Workshop on Discrete Event Systems, WODES '08)*, pp. 449–454, May 2008.

[25] M. P. Cabasino, A. Giua, and C. Seatzu, "Identification of deterministic Petri nets," in *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES '06)*, pp. 325–331, July 2006.

[26] M. Dotoli, M. P. Fanti, and A. M. Mangini, "Real time identification of discrete event systems using Petri nets," *Automatica*, vol. 44, no. 5, pp. 1209–1219, 2008.

[27] Y. Chen, Z. Li, M. Khalgui, and O. Mosbahi, "Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 374–393, 2011.

[28] F. M. Pia, M. A. Marcello, and U. Walter, "Fault detection by labeled Petri nets and time constraints," in *Proceedings of the 3rd International Workshop on Dependable Control of Discrete Systems (DCDS '11)*, pp. 168–173, June 2011.

[29] C. Girault and R. Valk, *Petri Nets for Systems Engineering—A Guide to Modeling, Verification, and Applications*, Springer, Berlin, Germany, 2003.

[30] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[31] A. Ramirez-Treviño, E. Ruiz-Beltran, J. Aramburo-Lizarraga, and E. Lopez-Mellado, "Structural diagnosability of des and design of reduced Petri net diagnosers," *IEEE Transactions on Systems, Man and Cybernetics A: Systems and Humans*, vol. 42, no. 2, pp. 416–429, 2012.

[32] K. Hernández and M. E. Meda-Campana, "Fault diagnosis using Petri nets: a case study," in *Proceedings of the 10th Latin American and Caribbean Conference for Engineering and Technology*, Panama City, Panama, July 2012.

[33] I. Rivera-Rangel, A. Ramírez-Treviño, L. I. Aguirre-Salas, and J. Ruiz-León, "Geometrical characterization of observability in Interpreted Petri nets," *Kybernetika*, vol. 41, no. 5, pp. 553–574, 2005.

[34] P. Supavatanakul and F. J. Lunze, *Identification of Timed Discreteevent Models for Diagnosis*, Ruhr-University Bochum, 2004.

[35] A. Ramirez-Trevino, E. Ruiz-Beltran, I. Rivera-Rangel, and E. López-Mellado, "Online fault diagnosis of discrete event systems. A Petri net-based approach," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 31–39, 2007.

[36] W. G. Cochran, *Sampling Techniques*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1977.

[37] J. E. Bartlett, J. W. Kotrlik, and C. C. Higgins, "Organizational research: determining appropriate sample size in survey research," *Information Technology, Learning and Performance*, vol. 19, no. 1, pp. 43–50, 2001.

[38] H. Toutenburg, "Fleiss, J. L.: statistical methods for rates and proportions. John Wiley & Sons, New York-London-Sydney-Toronto 1973. XIII, 233 S," *Biometrische Zeitschrift*, vol. 16, no. 8, p. 539, 1974.

[39] E. H. Livingston and L. Cassidy, "Statistical power and estimation of the number of required subjects for a study based on the *t*-test: a surgeon's primer," *Journal of Surgical Research*, vol. 126, no. 2, pp. 149–159, 2005.

[40] D. Ruppert, *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics, Springer, Berlin, Germany, 1st edition, 2010.

[41] W. Conover, *Practical Nonparametric Statistics*, Wiley Series in Probability and Statistics, John Wiley & Sons, New York, NY, USA, 3rd edition, 1999.