Fault detection of Discrete-Event Systems based on an identified timed model

Ryan P C de Souza, Marcos V Moreira, Jean-Jacques Lesage

Fault detection of Discrete-Event Systems based on an identified timed model

Ryan P. C. de Souza[a], Marcos V. Moreira[a], Jean-Jacques Lesage[b]

[a] *Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica, 21949-900, Rio de Janeiro, R.J., Brazil (e-mail: {ryanpitanga,moreira.mv}@poli.ufrj.br)*

[b] *LURPA, ENS Paris-Saclay, Univ. Paris-Sud, Université Paris-Saclay, 94235 Cachan, France (e-mail: jean-jacques.lesage@ens-paris-saclay.fr)*

Abstract

In this paper, a method for fault detection of Discrete-Event Systems (DES) based on a timed model called Timed Automaton with Outputs and Conditional Transitions (TAOCT), obtained by identification, is presented. The TAOCT is an extension of a recent untimed model proposed in the literature, called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT). Differently from the DAOCT, where only the logical behavior of the DES is considered, the TAOCT takes into account information about the time that the events are observed, and, for this reason, it can be used for the detection of faults that cannot be detected by using untimed models, such as faults that lead the fault detector to deadlocks. The TAOCT represents the fault-free system behavior, and a fault is detected when the observed behavior is different from the one predicted by the model, considering both logical and timing information. A practical example is used to illustrate the results of the paper.

Keywords: Fault diagnosis, System identification, Discrete-event systems, Finite automata.

1. Introduction

In the past years, interest in the domain of fault diagnosis of Discrete-Event Systems (DES) has increased. Since the introduction of the concept of fault diagnosis and diagnosability analysis of DES in [27], several methods have been proposed in the literature for fault diagnosis of untimed DES [8, 19, 36, 28, 5, 24, 14], and of timed DES [7, 22, 34, 35]. These methods provide a theoretical framework for the study of fault diagnosis of DES.

In general, obtaining an analytical white-box model of a real-world system is very laborious and time consuming, since, depending on its size and complexity, it is very difficult to take into account all possible behaviors of the system in the model. This problem is amplified when the post-fault behavior is considered, since there may exist unpredictable consequences to a fault occurrence. In addition, the modeling process requires engineers that are familiar with discrete-event modeling techniques. All these problems restrict the application of methods based on the complete system model to small systems, where white-box models of the fault-free and faulty behaviors can be obtained.

In order to overcome the modeling difficulties that arise due to the aforementioned problems, some solutions based on system identification were presented using Petri net models [18, 4, 9, 12, 10, 3, 11, 1, 2]. In the Petri net identification methods proposed in the literature with the objective of fault detection, it is supposed that the system structure or dynamics are completely or partially known, which makes this formalism suitable for modeling these systems. However, when no previous knowledge of the system is given, then automata become a suitable formalism for identification due to its more basic structure. In addition, generating an automatic fault detection method for systems modeled by automata is simpler than for systems modeled by Petri nets. Since, in this work, we assume that no previous knowledge about the system is available, then we have focused on the identification of automaton models.

Fault detection techniques based on an identified automaton model of the system have been proposed in the literature [16, 25, 20, 21]. In these works, the two main ideas are: (i) to automate the process of obtaining the fault-free automaton model of the system by identification; and (ii) when a fault has been detected through a discrepancy between the system behavior and the model, to use a technique based on residuals for fault localization [26, 21].

A model for the identification of closed-loop industrial DES, called Non-Deterministic Autonomous Automaton with Outputs (NDAAO), is presented in [16]. The NDAAO is identified from observed fault-free paths of the system, composed of sequences of vectors whose entries are the values of the binary input and output signals of the controller, as shown in Figure 1. These sequences form the input data of the identification procedure. With a view to obtaining a compact model, loops are introduced in the identified NDAAO, leading to the generation of sequences that have not been observed during the identification process. These sequences form the exceeding language gener-

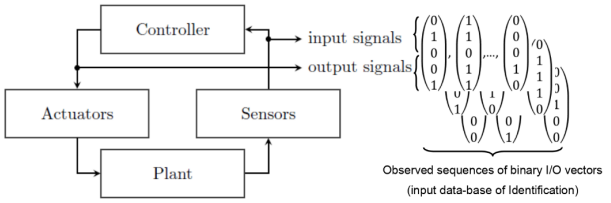

Figure 1: Identification scheme showing the signals exchanged between plant and controller, as well as the input data for the identification procedure.

ated by the identified model, and can be associated with non-detectable faults.

In [20], a different automaton model, called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT), is proposed. Conditions for the transposition of the transitions associated with the observed fault-free paths used in the identification procedure are added to the model by using a path estimation function. The use of the path estimation function reduces the exceeding language generated by the model in comparison with the NDAAO, and, consequently, it reduces the number of non-detectable faults. In [21], an algorithm for fault diagnosis using the DAOCT model proposed in [20] is presented.

It is important to remark that some faults cannot be detected by using untimed models. For instance, faults that prevent the system from generating events, leading the fault detector to a deadlock, cannot be detected from the logical behavior of the system. In addition, in some cases, timing information of event occurrences can be used to distinguish behaviors of the system, improving the capability of detecting faults. In this paper, a timed model for DES identification, called Timed Automaton with Outputs and Conditional Transitions (TAOCT), is proposed. The proposed model is an extension of the DAOCT in which timing information is added to the transitions in the form of guards for the transposition of the transitions, improving the capabality of the model to detect faults. Since the TAOCT is based on the DAOCT, it inherits the same advantages of the underlying DAOCT model in terms of exceeding language reduction considering only the logical behavior of the DES. A fault detection scheme based on the TAOCT model is also presented in this paper.

This paper is an extension of [32], where preliminary results were presented. In the present paper, the procedure for obtaining the TAOCT model is further developed by the proposition of an improved method for the identification of timing intervals associated with the transitions of the model. All results that have not been proved in [32] due to lack of space are proved in this paper, and new examples are presented to illustrate the results. In addition, differently from [32], an algorithm for performing fault detection using the TAOCT model is proposed, and conditions for the detection of a fault are formally introduced. The practical example is discussed in this paper in more detail than in [32].

1.1. Related works

Recent works on the identification of DES using the formalism of untimed Petri nets have been proposed in the literature. An identification method suitable for fault diagnosis is presented in [3], but only the faulty behavior is obtained by identification, and a model of the fault-free system behavior is required. In [9], even though the white-box model of the fault-free behavior of the system is not required, partial knowledge about the model is assumed, *e.g.*, it is necessary to provide an upper bound on the number of places in the net. In [10], the problem of identifying the unobservable behavior of a DES is addressed, while the fault-free system structure and dynamics are assumed to be previously known.

In the context of timed Petri net identified models, some interesting results have been reported in the literature. In [1], it is assumed that a time Petri net model for the nominal behavior of the system is available, and models for the faulty behaviors are identified using the discrepancies between the observed and nominal behaviors of the system. In [2], a time Petri net that models the complete system, and that can be used for fault detection, is identified. Both in [1] and [2], the adopted strategy for computing the identified model is based on the solution of a mixed integer linear programming problem.

Petri nets are normally used when some knowledge of the structure or dynamics of the system is available. However, for black-box identification of DES, automaton models can be more directly obtained due to its more basic structure. A timed automaton obtained by identification is used in [33] in a fault diagnosis scheme, where the measured data on which the identification model is built are based on the quantization of continuous signals, and not on discrete binary signals.

A timed automaton model identified from the observation of discrete binary signals, called the Timed Autonomous Automaton with Outputs (TAAO), is presented in [29]. It is an extension of the NDAAO to consider timing information in the model by adding timing constraints in the form of guards, defined as single time intervals, to its transitions.

1.2. Contributions of this paper

The main contributions of this paper are the proposal of the TAOCT model and its use in a fault detection scheme. The advantages of using the TAOCT are listed as follows:

- (*i*) the TAOCT is an extension of the DAOCT model, which has been shown in [21] to be more efficient for fault detection than the NDAAO model, which is the basis for the TAAO model presented in [29];
- (*ii*) the timing information added to the TAOCT model improves the capability of detecting faults of the fault detector system in comparison with the one obtained using the underlying DAOCT [21].

(*iii*) each guard associated with a transition of the TAOCT can be a disjoint union of several timing intervals, increasing the accuracy of the timing information added to the TAOCT in comparison with the models presented in [33], [29], [1], and [2];

This paper is structured as follows. Firstly, in Section 2, preliminary concepts are presented. In Section 3, the TAOCT is defined. The identification procedure for obtaining the TAOCT model is explained in Section 4. In Section 5, the fault detection scheme based on the TAOCT model is proposed. A practical example is presented and discussed in Section 6. Finally, conclusions are drawn in Section 7.

2. Preliminaries

Let G be a Timed Automaton with Timing Intervals (TATI) defined as follows [6].

Definition 1. *A timed automaton with timing intervals is a six-tuple:*

$$G = (X, \Sigma, f, c_g, guard, x_0),$$

where X is the set of states, Σ is the finite set of events, $f : X \times \Sigma \to X$ is the transition function, c_g is the global clock with value $c_g(t) \in \mathbb{R}^+$, $t \in \mathbb{R}^+$, $guard : X \times \Sigma \to \mathcal{A}$ is the guard function, where $\mathcal{A}$ is the set of admissible timing intervals for the global clock c_g, and $x_0 \in X$ is the initial state of the system. □

The domain of the transition function f can be extended to $X \times \Sigma^\star$, where $\Sigma^\star$ denotes the Kleene-closure of Σ, as follows: $f(x, \varepsilon) = x$, and $f(x, s\sigma) = f(f(x, s), \sigma)$, for $s \in \Sigma^\star$, and $\sigma \in \Sigma$, where ε denotes the empty sequence [6]. The feasible event function $\Gamma : X \to 2^\Sigma$ is defined as $\Gamma(x) = \{\sigma \in \Sigma : f(x, \sigma)!\}$, where symbol '!' denotes 'is defined'. It is important to remark that $guard(x, \sigma)$ is defined if, and only if, $f(x, \sigma)$ is defined.

A timed automaton with timing intervals is an automaton to which a global clock c_g and a guard function are added. Thus, timed sequences are generated by the TATI. Function $guard : X \times \Sigma \to \mathcal{A}$ specifies the timing conditions that need to be satisfied on the global clock for the transition to occur, and $\mathcal{A}$ are the admissible clock constraints defined as timing intervals. In the TATI, the global clock is reset to zero each time an event occurs. Let t' denote the time that a state $x \in X$ is reached in the timed automaton, and let τ denote the time that has elapsed after reaching state x. Then, transition (x, σ, x'), where $x' = f(x, \sigma)$, is executed in the TATI if event σ occurs at time $t' + \tau$, and $\tau \in guard(x, \sigma)$.

In this paper, a formalism that is similar to the TATI is used to add timing information to the proposed identification model.

A timed path of a TATI G is a sequence of states, events and time values that can be executed by G. A timed path can be written as $\pi = (x_1, \sigma_1, \tau_1, x_2, \sigma_2, \tau_2, ..., x_{l-1}, \sigma_{l-1}, \tau_{l-1}, x_l)$, where $x_j \in X$, $j = 1, ..., l$, are the states visited in the path, $\sigma_j \in \Sigma$ are the events that trigger the transition from state x_j to x_{j+1}, $j = 1, ..., l-1$, and $\tau_j \in \mathbb{R}^+$ are the time durations where the system remains in state x_j up to the occurrence of event σ_j, $j = 1, ..., l-1$. The length of a path π, denoted by $|\pi|$, is equal to the number of vertices in the path, *i.e.*, $|\pi| = l$.

Let $\Sigma_t := \Sigma \times \mathbb{R}^+$. Then, the set of all sequences formed of elements $\sigma_t \in \Sigma_t$, including the empty sequence ε, is denoted by $\Sigma_t^\star$. The timed sequence associated with timed path $\pi = (x_1, \sigma_1, \tau_1, x_2, \sigma_2, \tau_2, \ldots, x_{l-1}, \sigma_{l-1}, \tau_{l-1}, x_l)$ is the sequence $s_t = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \ldots (\sigma_{l-1}, \tau_{l-1}) \in \Sigma_t^\star$. It is important to remark that, in general, in the literature, timed sequences consider an absolute time that specifies when the event has occurred with respect to the beginning of the sequence execution, and not with respect to the sojourn time of each state of the path. However, the same information is contained in both types of timed sequences, since it is always possible to obtain a timed sequence with respect to the beginning of the process, from the timed sequence considering time durations s_t, and vice-versa.

Given a timed language $L_t \subseteq \Sigma_t^\star$, the prefix-closure of L_t is given by $\overline{L}_t = \{w_t \in \Sigma_t^\star : (\exists z_t \in \Sigma_t^\star)[w_t z_t \in L_t]\}$.

The set of non-negative integers is denoted by $\mathbb{N}$, and the set formed only with 0 and 1 is denoted by $\mathbb{N}_1 = \{0, 1\}$.

The length of a sequence s, timed or not, is denoted by $\|s\|$.

3. Timed Automaton with Outputs and Conditional Transitions

Consider a closed-loop DES whose scheme is shown in Figure 1, and suppose that the controller has m_i input signals, which correspond to sensor readings, and m_o output signals, which correspond to actuator commands. Let $u(t)$ be the I/O vector, whose elements are the current status of each one of the controller signals, defined as:

$$u(t) := [i_1(t)\ \ i_2(t)\ \ ...\ \ i_{m_i}(t)\ \ o_1(t)\ \ o_2(t)\ \ ...\ \ o_{m_o}(t)]^T,$$

where $i_a(t) \in \mathbb{N}_1$, $a \in \{1, ..., m_i\}$, are the values of the input signals at time $t \in \mathbb{R}^+$, and $o_a(t) \in \mathbb{N}_1$, $a \in \{1, ..., m_o\}$, are the values of the output signals at time $t \in \mathbb{R}^+$. Then, an event of the identified model can be defined as follows [20].

Definition 2. *An event of the identified model is any observed instantaneous change in one or more signals of the I/O vector u.* □

Note that in Definition 2, it is possible for an event to be defined as the change of more than one I/O signal, since multiple signals may change their values during the same scan cycle of the controller.

Suppose that at a given time instant t_j, $j \in \mathbb{N}$, the I/O vector becomes $u_j := u(t_j)$. Then, this vector remains unchanged until at least one of the controller signals

changes its value at time t_{j+1}, leading to I/O vector $u_{j+1} = u(t_{j+1})$. Let event σ_j denote the signal changes from u_j to u_{j+1}, and let the sojourn time τ_j be the time duration during which the I/O vector remains equal to u_j, *i.e.*, $\tau_j = t_{j+1} - t_j$. Then, when the observed I/O vector changes from u_j to u_{j+1}, a *timed transition* $(u_j, \sigma_j, \tau_j, u_{j+1})$ is observed. A finite sequence of observed timed transitions constitutes an observed timed path $p = (u_1, \sigma_1, \tau_1, u_2, \sigma_2, \tau_2, ..., u_{l-1}, \sigma_{l-1}, \tau_{l-1}, u_l)$. If the timing information is removed from a timed transition $(u_j, \sigma_j, \tau_j, u_{j+1})$, then an *untimed transition* (u_j, σ_j, u_{j+1}) is obtained. Thus, if all timing information is removed in the same way from timed path p, we obtain the untimed path $p_u = (u_1, \sigma_1, u_2, \sigma_2, ..., u_{l-1}, \sigma_{l-1}, u_l)$. As in [20], in this work, the observed paths used in the identification procedure are associated with the execution of complete system tasks. The following assumption is considered in this work.

A1. The system has a unique initial state, whose corresponding I/O vector is denoted by u_0, and all observed timed paths start with vector u_0.

The initial state may correspond, for example, to the beginning of a production cycle in the case of an industrial process. In this case, a path corresponds to the observation of the execution of a production cycle.

It is important to remark that, in order to obtain a model by identification of a DES, capable of reproducing all possible behaviors of the system in fault-free operation, it is necessary to observe the paths executed by the system for an arbitrarily long time. However, in a finite time, only part of the paths generated by the system can be observed. This fact has a direct impact on the accuracy of the identified model and, therefore, on fault detection, since an observed behavior that the model is not capable of executing is interpreted as being faulty. If it is a fault-free behavior, then the fault detection system generates a false alarm. Thus, in order to reduce the number of false alarms, as in [20], it is assumed in this paper that the system has been observed for a sufficiently long time such that all different untimed paths of length up to a given number $n_0 \in \mathbb{N}$ have been observed.

The set formed of all events that were observed during the identification procedure is denoted by Σ. The set of all I/O vectors that have been observed is denoted by Ω. Thus, $\Omega \subseteq \mathbb{N}_1^{m_i+m_o}$.

Assume that ν timed paths (not necessarily distinct) have been observed, where each path is denoted by

$$p_q = \left(u_{q,1}, \sigma_{q,1}, \tau_{q,1}, u_{q,2}, \sigma_{q,2}, \tau_{q,2}, ..., u_{q,l_q-1}, \sigma_{q,l_q-1}, \tau_{q,l_q-1}, u_{q,l_q}\right), \quad (1)$$

where $q \in \{1, ..., \nu\}$, $u_{q,j} \in \Omega$ for $j = 1, ..., l_q$, and $\sigma_{q,j} \in \Sigma$ and $\tau_{j,q} \in \mathbb{R}^+$ for $j = 1, ..., l_q - 1$. Then, the observed timed sequence s_t^q associated with each path p_q is given by $s_t^q = (\sigma_{q,1}, \tau_{q,1})(\sigma_{q,2}, \tau_{q,2})...(\sigma_{q,l_q-1}, \tau_{q,l_q-1})$. Thus, the observed timed language $L_{t,Obs}$ is defined as follows:

$$L_{t,Obs} := \bigcup_{q=1}^{\nu} \overline{\{s_t^q\}}. \quad (2)$$

Let P denote the set formed of all observed timed paths p_q, for $q = 1, \ldots, \nu$, and let P be partitioned as $P = P_1 \dot\cup P_2 \dot\cup ... \dot\cup P_r$, $r \leq \nu$, such that all paths that form P_i are logically equivalent to each other, *i.e.*, there is a unique untimed path $p_{u_i} = (u_{i,1}, \sigma_{i,1}, u_{i,2}, \sigma_{i,2}, ..., u_{i,l_i'-1}, \sigma_{i,l_i'-1}, u_{i,l_i'})$, associated with each set P_i, for $i = 1, \ldots, r$. As in [20], the following assumption is considered.

A2. None of the untimed paths p_{u_i} has an associated sequence of events that is a prefix of the sequence of events of another untimed path p_{u_z}, where $i \neq z$, for $i, z \in \{1, 2, \ldots, r\}$. □

It is important to remark that Assumption **A2** holds true in several cases, since each observed path is associated with a system task, and, in a large number of applications, the execution of a sequence of events associated with a system task is not a prefix of another sequence of events associated with other possible task executed by the system.

The *time-interval paths* can be defined as follows:

$$p_i' = \left(u_{i,1}, \sigma_{i,1}, I_{i,1}, ..., u_{i,l_i'-1}, \sigma_{i,l_i'-1}, I_{i,l_i'-1}, u_{i,l_i'}\right), \quad (3)$$

for every $i \in \{1, \ldots, r\}$. The logical behavior of p_i' is given by the untimed path p_{u_i} associated with P_i. The timing behavior associated with path p_i' is modeled by sets $I_{i,j}$, $j = 1, \ldots, l_i' - 1$, that represent the possible time values for the occurrence of events of each one of the transitions of p_{u_i}, obtained from the observed timed paths of P_i. In this paper, it is assumed that each untimed path, representing a possible logical behavior of the system, has been observed a number of times sufficient to capture the timing information regarding event occurrences. The process of building the sets $I_{i,j}$ from the timed paths in P_i is explained in Section 4. It is important to remark that the TAOCT model is computed from the time-interval paths p_i', $i = 1, \ldots, r$, identified from the timed paths $p_q \in P$ generated by the system. The set of time-interval paths is denoted in this paper as $P' := \{p_1', p_2', \ldots, p_r'\}$.

The timed model proposed in this paper for the identification of DES is presented in the sequel. The model is based on Definition 1 of Timed Automaton with Timing Intervals, and on the DAOCT model proposed in [20].

Definition 3. *The Timed Automaton with Outputs and Conditional Transitions is a ten-tuple:*

$$TAOCT = (X, \Sigma, f, c_g, \Omega, \lambda, R, g, x_0, X_f),$$

where

- X *is the finite set of states;*
- Σ *is the finite set of events;*

- $f : X \times \Sigma^\star \to X$ *is the transition function;*
- c_g *is the global clock, with value* $c_g(t) \in \mathbb{R}^+$, $t \in \mathbb{R}^+$*;*
- $\Omega \subseteq \mathbb{N}_1^{m_i+m_o}$ *is the set of outputs;*
- $\lambda : X \to \Omega$ *is the state output function;*
- $R = \{1, 2, ..., r\}$ *is the set of time-interval path indexes;*
- $g : X \times \Sigma \times R \to \mathcal{C}$ *is the guard function;*
- $x_0 \in X$ *is the initial state;*
- $X_f \subseteq X$ *is the set of final states.* □

The set of admissible constraints $\mathcal{C}$ is formed of all sets $I \subset \mathbb{R}^+$. Note that set I can be formed of the union of disjoint time intervals instead of a single time interval. As in the TATI, presented in Definition 1, in the TAOCT a unique global clock c_g is used, and it is reset every time a transition occurs. Function $g(x, \sigma, i)$ specifies a subset of $\mathbb{R}^+$ to which the clock value $c_g(t)$ must belong so that transition (x, σ, x'), where $x' = f(x, \sigma)$, associated with path p'_i, can occur. The output function λ is the same presented in the DAOCT model, and associates an I/O vector with each state $x \in X$ of the model.

Differently from the DAOCT, where a path estimation function is defined in the model, in the TAOCT the path estimation function $\theta : X \times \Sigma \times \mathbb{R}^+ \to 2^R$ can be defined using the guard function g as follows:

$$\theta(x, \sigma, \tau) = \{i \in R : \tau \in g(x, \sigma, i)\}. \tag{4}$$

Function $\theta(x, \sigma, \tau)$ provides the path estimate when the current state is x, and event σ occurs at clock value $c_g(t) = \tau$. If $f(x, \sigma)$ is not defined for a given path p'_i, then $g(x, \sigma, i)$ is, by convention, the empty set, and $i \notin \theta(x, \sigma, \tau)$ for any value of τ. Moreover, if $f(x, \sigma)$ is defined, but the observed time τ does not belong to $g(x, \sigma, i)$, then $i \notin \theta(x, \sigma, \tau)$.

Function $\psi : \Sigma_t^\star \to \Sigma^\star$ removes the timing information from a timed sequence in $\Sigma_t^\star$, obtaining its equivalent untimed sequence, *i.e.*, for any $s_t = (\sigma_1, \tau_1) \dots (\sigma_l, \tau_l) \in \Sigma_t^\star$, then $\psi(s_t) = \sigma_1 \dots \sigma_l \in \Sigma^\star$. By convention, $\psi(\varepsilon) := \varepsilon$.

The extended path estimation function $\theta_s : X \times \Sigma_t^\star \to 2^R$ is defined, recursively, as follows: $\theta_s(x, \varepsilon) = R$, and for any sequence $s_t(\sigma, \tau) \in \Sigma_t^\star$, where $s_t \in \Sigma_t^\star$ and $(\sigma, \tau) \in \Sigma_t$,

$$\theta_s(x, s_t(\sigma, \tau)) = \begin{cases} \theta_s(x, s_t) \cap \theta(x', \sigma, \tau), \text{ where} \\ \quad x' = f(x, \psi(s_t)), \text{ if } f(x, \psi(s_t)\sigma)! \\ \emptyset, \text{ otherwise.} \end{cases}$$

The timed language generated by the TAOCT model is given by:

$$L_{t,Iden} := \{s_t \in \Sigma_t^\star : \theta_s(x_0, s_t) \neq \emptyset\}. \tag{5}$$

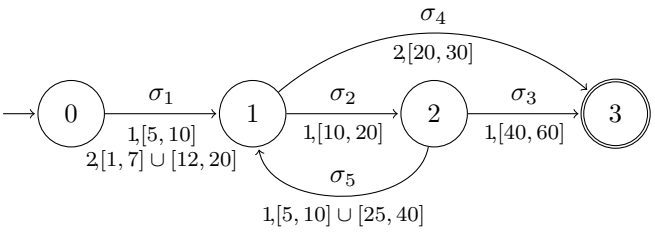

Figure 2: TAOCT model of Example 1.

Remark 1. *Note that the TAOCT is an extension of the DAOCT proposed in [20] by adding timing information to the transitions. Thus, since the DAOCT has been proven to be suitable for fault detection in [20], considering only the logical behavior of the system, and since the addition of timing information reduces even more the number of sequences that can be generated by the TAOCT, improving the capability of detecting faults, then the TAOCT model is also suitable for fault detection.* □

In the sequel, an example of a TAOCT model is presented, and it is discussed how timed event sequences can be generated by the model.

Example 1. *Consider the TAOCT shown in Figure 2. Each circle represents a state* $x \in X$*, where* $X = \{0, 1, 2, 3\}$*, and the directed arcs represent the transitions. A transition from state* x *to* x' *is labeled with an event* $\sigma \in \Sigma$*, where* $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$*. In addition, a label of the form* i, I *attached to a transition from* x *to* x' *indicates that there is a guard* $g(x, \sigma, i) = I$ *defined for that transition. In this example, the set of time-interval path indexes is given by* $R = \{1, 2\}$*, which means that the TAOCT model has been computed from two observed time-interval paths* p'_i*,* $i = 1, 2$*. The initial state is given by* $x_0 = 0$*, and the set of final states is* $X_f = \{3\}$*, represented in Figure 2 with a double circle. Note that a disjoint union of time intervals can be defined as the guard of a transition associated with a time-interval path. For instance, the transition* $(0, \sigma_1, 1)$ *of Figure 2 has guard* $g(0, \sigma_1, 2) = [1, 7] \cup [12, 20]$ *associated with time-interval path* p'_2*.*

Suppose that we want to verify if the timed sequence $s_t = (\sigma_1, 6)(\sigma_2, 14)(\sigma_3, 56)$ *can be executed by the model. It can be verified that* $\theta_s(0, (\sigma_1, 6)) = \{1, 2\}$*, since* $6 \in g(0, \sigma_1, 1) = [5, 10]$*, and* $6 \in g(0, \sigma_1, 2) = [1, 7] \cup [12, 20]$*. In addition,* $\theta_s(0, (\sigma_1, 6)(\sigma_2, 14)) = \theta_s(0, (\sigma_1, 6)) \cap \theta(1, \sigma_2, 14) = \{1\}$*, since* $14 \in g(1, \sigma_2, 1) = [10, 20]$*, and* $\theta_s(0, (\sigma_1, 6)(\sigma_2, 14)(\sigma_3, 56)) = \theta_s(0, (\sigma_1, 6)(\sigma_2, 14)) \cap \theta(2, \sigma_3, 56) = \{1\}$*, since* $56 \in g(2, \sigma_3, 1) = [40, 60]$*. Thus, according to Equation (5),* $s_t \in L_{t,Iden}$*.*

Consider now sequence $s'_t = (\sigma_1, 15)(\sigma_4, 14)$*. Then,* $\theta_s(0, (\sigma_1, 15)(\sigma_4, 14)) = \theta_s(0, (\sigma_1, 15)) \cap \theta(1, \sigma_4, 14)$*. Since,* $\theta_s(0, (\sigma_1, 15)) = \{2\}$*, and* $\theta(1, \sigma_4, 14) = \emptyset$*, then* $\theta_s(0, s'_t) = \emptyset$*. Thus, according to Equation (5),* $s'_t \notin L_{t,Iden}$*.* □

In the following section, the procedure for identifying the TAOCT model that represents the fault-free behavior of a closed-loop DES is presented.

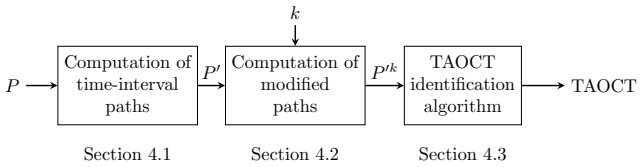

Figure 3: Scheme of the identification procedure.

4. Identification procedure

In this section, the steps that are necessary to compute the TAOCT model are presented. In Figure 3, the complete scheme of the identification procedure with three steps is depicted, where the subsection associated with each one of the steps is shown. It can be seen from Figure 3 that the input data for identification is set $P = \{p_1, p_2, \ldots, p_\nu\}$, containing all observed fault-free timed paths, and the output is the TAOCT that models the fault-free behavior of the system.

In the TAOCT, differently from other timed models presented in the literature, the timing behavior associated with each transition can be represented as a set of disjoint intervals. In [33], a single time interval is always assigned to each transition, where the lower (resp. higher) endpoint is simply the lowest (resp. highest) timing value observed during identification for that transition. In [31], the guards are determined using a method called Skewness Adaption, which always results in single intervals. In [2], a single timing interval is assigned to each transition of a time Petri net. To the best of the authors' knowledge, the method proposed in this work is the first one in the literature on DES identification that allows the definition of time guards that are not restricted to single intervals, which increases the accuracy of the guards, improving the efficiency of the fault detector system.

4.1. Computation of the time-interval paths

In order to compute the time-interval paths p'_i, $i = 1, \ldots, r$, it is necessary to obtain the sets $I_{i,j}$ corresponding to the transitions of p'_i, associated with the time values that the transitions of paths $p_q \in P_i$ have been observed.

Let $T_{i,j} := \{\tau_{q,j} \in \mathbb{R}^+ : (q \in \{1, \ldots, \nu\}) \wedge (p_q \in P_i)\}$ be the set formed of the time values $\tau_{q,j}$ that are observed in the j-th transition of the timed paths $p_q \in P_i$. It is possible that the values in $T_{i,j}$ are arranged on the real axis in such a way that some values in a subset of $T_{i,j}$ are closer to each other in comparison with the other values in $T_{i,j}$. Each such subset forms a *cluster* of time values, and each cluster corresponds to a different observed timing behavior for that transition. Thus, different timing behaviors can be identified by applying a clustering method to the values in $T_{i,j}$.

Several clustering algorithms have been proposed in the literature [13, 17, 15, 23]. In this paper, the Nearest Neighbors Method [17] is used for obtaining the clusters of $T_{i,j}$. In this method, a value is assigned to a cluster if the minimum distance between this value and the other members of the cluster is smaller than or equal to a user-specified threshold ζ. If the distance between two values is greater than ζ, then they are assigned to different clusters.

Note that an uncertainty $\delta \in \mathbb{R}^+$ must be considered for defining the value of ζ to take into account possible errors in measuring the time values of the event occurrences, caused, for example, by the scan cycle of the controller. Due to this uncertainty, the threshold ζ must be chosen so that $\zeta > 2\delta$.

After applying the Nearest Neighbors Method to $T_{i,j}$, set $S = \{S_1 \ldots, S_K\}$ is obtained, where $S_h \subseteq T_{i,j}$, for $h = 1, \ldots, K$, such that $S_1 \dot\cup S_2 \dot\cup \ldots \dot\cup S_K = T_{i,j}$. Each set $S_h \in S$ corresponds to a different cluster in $T_{i,j}$, representing a distinct timing behavior associated with the j-th transition of time-interval path p'_i. Set $I_{i,j}$ is then given by the following expression:

$$I_{i,j} = \bigcup_{h=1}^{K} [\max\{0, \min S_h - \delta\}, \max S_h + \delta].$$

Note that set $I_{i,j}$ is a disjoint union of intervals, where each interval corresponds to a cluster S_h. Each interval is extended by δ on both ends in order to reduce the number of false alarms that may be raised due to disturbances in time measurements. Since only non-negative values are allowed in $I_{i,j}$, if the minimum value of a cluster is less than δ, then the lower endpoint of the interval corresponding to this cluster is set to zero.

A *time-interval transition* is defined as the 4-tuple $(u_{i,j}, \sigma_{i,j}, I_{i,j}, u_{i,j+1})$ and represents the transition between $u_{i,j}$ and $u_{i,j+1}$ in a time-interval path. Differently from a timed transition, the third element of the time-interval transition is a set of values instead of a single value, which allows to distinguish both types of transition.

For the choice of the threshold ζ, the following considerations must be taken into account. Let ζ be chosen to be a large value such that two distinct timing behaviors of time-interval transition $(u_{i,j}, \sigma_{i,j}, I_{i,j}, u_{i,j+1})$ of path p'_i are merged into a single cluster, *i.e.*, $I_{i,j}$ is formed of a single time interval. In this case, if a fault in the system causes the transition to occur between the two distinct timing behaviors, then the fault is not detected, since this time value belongs to $I_{i,j}$, which means that it is included in the fault-free model. On the other hand, if time-interval transition $(u_{i,j}, \sigma_{i,j}, I_{i,j}, u_{i,j+1})$ of path p'_i has a unique timing behavior, but ζ is chosen to be a small value and the number of observations of the transition is small, then it is possible that several clusters are formed instead of a single cluster. As a consequence, $I_{i,j}$ is formed of several time intervals, and a false alarm is raised every time transition $(u_{i,j}, \sigma_{i,j}, u_{i,j+1})$ is observed between the time intervals of $I_{i,j}$. Thus, for the correct computation of the clusters associated with $I_{i,j}$, ζ must be a small value greater than 2δ, and the system must be observed for a sufficiently long time such that there are enough observations of each transition of p'_i to adequately represent its distinct timing be-

haviors. It is worth noting that in the ideal case where infinite observations of the same transition are recorded, the distance that separates two successive values in the same cluster would drop to zero. Thus, it is reasonable to choose a small value for ζ, provided that a large amount of data recorded during identification is available. Nevertheless, if false alarms are raised during system operation, the model can still be improved by increasing the value of ζ for the computation of each $I_{i,j}$.

It is important to remark that the clustering procedure must be performed for the determination of each set $I_{i,j}$ from set $T_{i,j}$, for every time-interval p'_i, $i \in \{1, \ldots, r\}$.

4.2. Parametric identification approach

In [20], as in [16], the parametric identification algorithm allows to obtain a model satisfying an important property called k-completeness which guarantees that, once the value of a free parameter k has been chosen, all and only all sequences of length 1 up to k that have been observed are generated by the identified DAOCT. By increasing the value of the free parameter k, the exceeding language generated by the identified DAOCT model reduces, but the size of the model grows. Thus, there is a trade-off to be found between complexity and accuracy of the identified model.

Since the timed model proposed in this paper is based on the DAOCT model presented in [20], the same strategy proposed in [20] to obtain a parameterized model for identification can be used. In order to do so, the same steps for the computation of the DAOCT model are considered here, where the first step is the computation of modified paths p'^k_i, according to the free parameter k, as follows:

$$p'^k_i = \left(y_{i,1}, \sigma_{i,1}, I_{i,1}, ..., y_{i,l'_i-1}, \sigma_{i,l'_i-1}, I_{i,l'_i-1}, y_{i,l'_i}\right), \quad (6)$$

where

$$y_{i,j} = \begin{cases} (u_{i,j-k+1}, ..., u_{i,j}), \text{ if } k \leq j \leq l'_i \\ (u_{i,1}, ..., u_{i,j}), \text{ if } j < k \end{cases}.$$

Note that the vertices $y_{i,j}$ of path p'^k_i are formed of sequences of I/O vectors of length up to k, capable of memorizing the last $k-1$ events occurred in the model.

Define set Ω^k as the set formed of all $y_{i,j}$, $j = 1, \ldots, l'_i$, $i \in \{1, \ldots, r\}$. Let $\hat{y}_{i,j}$ denote the last element of $y_{i,j}$. Then, $\hat{y}_{i,j} := u_{i,j}$, for $j = 1, \ldots, l'_i$, $i = 1, \ldots, r$. The set of modified paths is denoted as $P'^k := \{p'^k_1, p'^k_2, \ldots, p'^k_r\}$.

4.3. Computation of the TAOCT model

Before introducing the procedure to construct the TAOCT from the observed data, define function $\tilde{\lambda} : X \to \Omega^k$, which provides a bijective correspondence between a state $x \in X$ of the model and a symbol $y \in \Omega^k$. This function will be used in the procedure for constructing the TAOCT model from the set of fault-free modified paths p'^k_i, $i \in \{1, \ldots, r\}$, described in Algorithm 1.

Since the TAOCT model is based on the DAOCT, Algorithm 1 is based on the algorithm for computing the DAOCT model presented in [20]. At the beginning of Algorithm 1, the initial state x_0 is created, and $\tilde{\lambda}(x_0)$ and $\lambda(x_0)$ receive $y_{1,1}$ and $\hat{y}_{1,1}$, respectively, such that the output of the initial state is equal to the first I/O vector of the observed paths $p_q \in P$, used for the computation of the modified time-interval paths p'^k_i, for $i = 1, \ldots, r$. In line 3, the set of states X, the set of outputs Ω, and the set of final states X_f are initialized. In addition, the set of time-interval path indexes R is specified. In line 4, the set of events Σ is defined as the set containing all observed events, considering every transition in each observed fault-free time-interval path p'_i, for $i = 1, \ldots, r$. In line 5, guard function g is initialized for state x_0 by assigning the empty set for every observed event in Σ and every path index in R. In the inner for-loop, from lines 7 to 21, every vertex $y_{i,j}$ of the modified time-interval path p'^k_i is visited. Each time a vertex is visited, the corresponding state $x \in X$ is found. Then, it is checked if there is a state x' that has already been created and that corresponds to the next vertex $y_{i,j+1}$. If such a state does not exist, then a new state is created in the if-block that starts in line 9, and set X is updated accordingly. In lines 12 to 14, vertex $y_{i,j+1}$ is associated to the newly created state x', and its output symbol, given by the last element of $y_{i,j+1}$, is added to Ω. Similarly to what is done in line 5 for the initial state x_0, in line 15 the empty set is assigned to the guard function for state x' and every pair (σ, i) in $\Sigma \times R$, ensuring that the guard function is completely defined on its domain. In line 18, the transition from the current state x to the next state x' through event $\sigma_{i,j}$ is defined in the transition function f. Differently from g, function f may be partially defined on its domain. In line 19, the guard function $g(x, \sigma_{i,j}, i)$ is updated by aggregating set $I_{i,j}$ to $g(x, \sigma_{i,j}, i)$. Finally, in line 20, it is tested if the next vertex $y_{i,j+1}$ is the last one of path p'^k_i. If this is the case, then next state x' is added to the set of final states X_f.

Language $L_{t,Iden}$, generated by the TAOCT model obtained using Algorithm 1, and whose expression is given by Equation (5), simulates the timed sequences of the paths used in the identification procedure, as stated in the sequel.

Theorem 1. *$L_{t,Obs} \subseteq L_{t,Iden}$.*

Proof. Consider a timed event sequence $w_t \in L_{t,Obs}$, which, according to Equation (2), is the prefix of a timed sequence associated with some observed timed path p_q, whose expression is given by Equation (1). Since $p_q \in P_i$, for some $i \in \{1, \ldots, r\}$, then its corresponding modified time-interval path p'^k_i, given by Equation (6), has the same untimed sequence as p_q, and p_q is such that $\tau_{q,j} \in I_{i,j}$, for $j = 1, ..., \|w_t\|$. Since $\tilde{\lambda}(x_0) = \hat{y}_{i,1}$ (as a consequence of Assumption **A1**), Algorithm 1 ensures that: (i) there exist states x_j, such that $f(x_j, \sigma_{i,j}) = x_{j+1}$, starting at the initial state x_0 (line 18); and (ii) $\tau_{q,j} \in I_{i,j} \subseteq g(x_j, \sigma_{i,j}, i)$, $j = 1, ..., \|w_t\|$, according to line 19. Hence, according to Equation (4), $i \in \theta(x_j, \sigma_{i,j}, \tau_{q,j})$, $j = 1, ..., \|w_t\|$, which implies that $i \in \theta_s(x_0, w_t)$, and, consequently, $\theta_s(x_0, w_t) \neq$

Algorithm 1: TAOCT identification

Input: Modified time-interval paths $p_i'^k$, $i = 1, ..., r$
Output: TAOCT $= (X, \Sigma, f, c_g, \Omega, \lambda, R, g, x_0, X_f)$

Create initial state x_0
$\lambda(x_0) \leftarrow \hat{y}_{1,1}$ and $\tilde{\lambda}(x_0) \leftarrow y_{1,1}$
$X \leftarrow \{x_0\}$, $\Omega \leftarrow \{\hat{y}_{1,1}\}$, $R \leftarrow \{1, ..., r\}$, and $X_f \leftarrow \emptyset$
$\Sigma \leftarrow \bigcup_{i \in R} \{\sigma_{i,j} : j = 1, \ldots, l_i' - 1\}$
$g(x_0, \sigma, i) \leftarrow \emptyset$, $\forall (\sigma, i) \in \Sigma \times R$
for $i = 1$ **to** r **do**
  for $j = 1$ **to** $l_i' - 1$ **do**
    Find state $x \in X$ such that $\tilde{\lambda}(x) = y_{i,j}$
    if $\tilde{\lambda}(x') \neq y_{i,j+1}$ $\forall x' \in X$ **then**
      Create state x'
      $X \leftarrow X \cup \{x'\}$
      $\tilde{\lambda}(x') \leftarrow y_{i,j+1}$
      $\lambda(x') \leftarrow \hat{y}_{i,j+1}$
      $\Omega \leftarrow \Omega \cup \{\lambda(x')\}$
      $g(x', \sigma, i) \leftarrow \emptyset$, $\forall (\sigma, i) \in \Sigma \times R$
    else
      Find $x' \in X$ such that $\tilde{\lambda}(x') = y_{i,j+1}$
    $f(x, \sigma_{i,j}) \leftarrow x'$
    $g(x, \sigma_{i,j}, i) \leftarrow g(x, \sigma_{i,j}, i) \cup I_{i,j}$
    if $j = l_i' - 1$ **then**
      $X_f \leftarrow X_f \cup \{x'\}$

$\emptyset$. Thus, from Equation (5), $w_t \in L_{t,Iden}$. Therefore, $L_{t,Obs} \subseteq L_{t,Iden}$. □

Let L_{Obs} and L_{Iden} denote the languages formed of the untimed sequences of events obtained from all timed sequences of $L_{t,Obs}$ and $L_{t,Iden}$, respectively, *i.e.*, $L_{Obs} := \{\psi(s_t) \in \Sigma^\star : s_t \in L_{t,Obs}\}$ and $L_{Iden} := \{\psi(s_t) \in \Sigma^\star : s_t \in L_{t,Iden}\}$. Then, the following result can be stated.

Corollary 1. $L_{Obs} \subseteq L_{Iden}$.

Proof. The proof is straightforward from Theorem 1. □

5. Fault detection scheme

The fault detector proposed in this paper is inspired by the one presented in [21] based on the DAOCT model. Thus, as long as a sequence of timed events s_t, whose corresponding untimed behavior $s_u = \psi(s_t)$ belongs to L_{Obs}, is executed by the system, the fault detector observes the events, and plays the model following the behavior of s_t. If sequence s_t is compatible with one of the time-interval paths p_i', then, after sequence s_t has been observed, the model is reinitialized and a new sequence can be played by the fault detector. The sequence of events that is played by the fault detector without reinitializing the model is called a *model run*. Thus, the fault detector must evaluate if the current model run can be executed by the TAOCT model. If the model is unable to execute the observed sequence of events, or if the occurrence of an expected event does not occur within a feasible time interval, then a fault is detected.

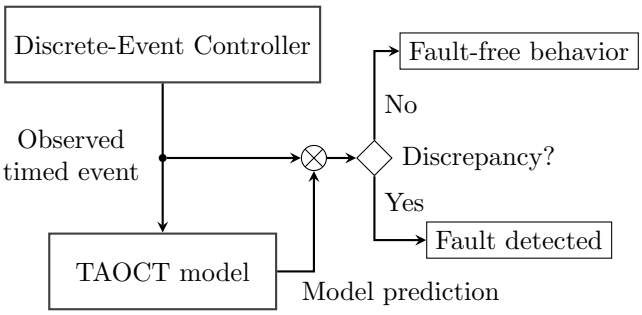

Figure 4: Fault detection scheme based on the TAOCT model.

In Figure 4, the scheme for fault detection based on the TAOCT model is presented. The idea behind this scheme is that, during system operation, the current I/O vector of controller signals is read in real-time by the fault detector. Then, if the observed behavior of the system is different from what is predicted by the TAOCT model, the fault that caused the unexpected behavior is detected. After detection, the fault can be localized by using residuals [26, 30, 21]. Fault localization is not addressed in this paper.

In [21], the minimum number of event observations, denoted as μ_{u_i}, such that each fault-free path $p_{u_i} \in P_u$ can be distinguished from the other fault-free paths in P_u, where P_u is the set formed of all fault-free paths used in the DAOCT identification procedure, is used in the fault detection scheme. Similarly, in this paper, the minimum number of event observations of p_i' such that p_i' can be distinguished from any other path p_z', $i \neq z$, $i, z \in R$, denoted as μ_i, is used in the fault detection scheme. However, due to the fact that the timing behavior of the TAOCT is also considered, the value of μ_i may be different from the value of μ_{u_i}, obtained for the underlying DAOCT model, for each $i \in R$. Consider, for example, that $R = \{1, 2\}$, and that both paths p_1' and p_2' start with the same event, *i.e.*, $\sigma_{1,1} = \sigma_{2,1}$. Since p_1' and p_2' have the same first event, then they are logically equivalent considering only the first event occurrence, which means that certainly both μ_{u_1} and μ_{u_2} are greater than one by considering only the logical behavior. However, if $I_{1,1} \backslash I_{2,1} \neq \emptyset$, then it is possible to distinguish path p_1' from p_2' upon the occurrence of the first event. In order to do so, the first event must be observed with a time value belonging to $I_{1,1} \backslash I_{2,1}$. In this case, $\mu_1 = 1$. Note that, if $I_{2,1} \subset I_{1,1}$, then it is impossible to distinguish p_2' from p_1' upon the first event occurrence. Thus, $\mu_2 > 1$. The value of μ_i, for $i = 1, \ldots, r$, can be formally defined as follows:

$$\mu_i = \min\{j \in \{1, \ldots, l_i' - 1\} : (\nexists z \in R \setminus \{i\}) \\ [(\sigma_{i,v} = \sigma_{z,v}) \wedge (I_{i,v} \backslash I_{z,v} = \emptyset), v = 1, \ldots, j]\}. \quad (7)$$

It is important to remark that it is always possible to compute μ_i, $i = 1, \ldots, r$, since, according to Assumption

A2, none of the untimed paths p_{u_i}, associated with p'_i, has an associated sequence of events that is a prefix of the sequence of events of another untimed path p_{u_z}, associated with p'_z, where $i \neq z$, for $i, z \in R$.

In the definition of a viable event given in [21], four conditions are presented in order to verify if an observed event indicates the occurrence of a fault. In the sequel, a *viable timed event* is defined by adapting the definition of viable event to the context of the TAOCT model.

Definition 4. *Let $s_t \in \Sigma_t^\star$ be a model run such that $x = f(x_0, \psi(s_t))$. Then, $(\sigma, \tau) \in \Sigma_t$ is said to be a viable timed event in state $x \in X$ of the TAOCT model if it satisfies the following conditions:*

C1. *$\sigma \in \Gamma(x)$;*

C2. *$\theta_s(x_0, s_t(\sigma, \tau)) \neq \emptyset$;*

C3. *If $|\theta_s(x_0, s_t)| > 1$ and $\theta_s(x_0, s_t(\sigma, \tau)) = \{i\}$, then $\|s_t(\sigma, \tau)\| \geq \mu_i$;*

C4. *If $\|s_t(\sigma, \tau)\| = l'_i - 1$, for $i \in \theta_s(x_0, s_t(\sigma, \tau))$, then $\tilde{\lambda}(x') = y_{i,l'_i}$, where $x' = f(x, \sigma)$, or there exists $j \in \theta_s(x_0, s_t(\sigma, \tau))$ such that $\|s_t(\sigma, \tau)\| < l'_j - 1$.* □

If Conditions **C1** and **C2** are verified, then $s_t(\sigma, \tau) \in L_{t,Iden}$. If Condition **C3** is violated, then path p'_i has been wrongly identified before the minimum number of timed event occurrences μ_i, given by Equation (7). Finally, if Condition **C4** is not true, then the length of the observed trace $s_t(\sigma, \tau)$ is equal to the maximum length among all sequences of the estimated paths in $\theta_s(x_0, s_t(\sigma, \tau))$, without reaching the final vertex of any of these paths, which implies that a fault has occurred.

Algorithm 2 describes the procedure for performing fault detection using the identified TAOCT model. The basic idea is to detect the occurrence of a fault if the observed timed event is not viable, according to Definition 4, or a deadlock is reached. In Algorithm 2, sets $V := \{(i, y_{i,l'_i}) : i \in R\}$ and $N := \{(i, \mu_i) : i \in R\}$ are used for fault detection. Verification of Conditions **C1**-**C4** is carried out for the TAOCT in a similar way as for the underlying DAOCT presented in [21]. The main difference between Algorithm 2 and the algorithm proposed in [21], is the use of the path estimation function θ_s that takes into account the information about the time that an event is observed, τ, for determining which time-interval paths are possibly being executed by the system. Since θ_s uses both the logical and the timing behaviors of the system, then its estimation is improved with respect to the estimation provided by the path estimation function proposed in [21] that uses only the logical behavior.

There are three faulty scenarios for which a fault can be detected by using the TAOCT instead of the underlying DAOCT model, namely: (*i*) faults that lead the system to a deadlock; (*ii*) faults that cause the occurrence of a feasible event σ at a time instant τ that does not belong to any set defined by the guard conditions $g(x, \sigma, i)$, for all $i \in R$; and (*iii*) faults that cause the occurrence of a feasible event σ at a time instant τ that satisfies a guard condition, but leads the path estimation function to $\theta_s(x_0, s_t) = \emptyset$. In order to detect the first type of fault, the global clock c_g is reset in line 5 every time a transition is transposed in the TAOCT. Then, c_g is used to detect a deadlock in lines 7 to 12, by verifying if an event is observed in a time smaller than or equal to the maximum time τ_{max} allowed for the occurrence of an event in the current state x_c. If an event is not observed within τ_{max} time units, then the fault is detected. The second and third types of faults can be detected by verifying if the observed timed sequence can be generated by the model, since, in both cases, although $f(x_0, \psi(s_t))$ is defined, $\theta_s(x_0, s_t) = \emptyset$, *i.e.*, the observed timed sequence is not in $L_{t,Iden}$.

In line 14, it is checked if Condition **C1** is violated, in which case the fault is detected in line 15. In lines 17 to 20, Condition **C2** is verified and, if it does not hold, the fault is detected. In lines 21 to 25, Condition **C3** is tested and the fault is detected if it is violated. In lines 26 to 29 of Algorithm 2, the TAOCT is reinitialized every time a cyclical fault-free path, with behavior compatible with a time-interval path p'_i, $i \in \{1, \ldots, r\}$, is executed by the system. In lines 30 to 34, it is verified if the final vertex of a non-cyclical observed path is reached, in which case no event should be observed anymore. In this case, in line 32 the path estimate is made equal to the empty set, in line 33 the value of τ_{max} is set to infinity, and the algorithm returns to line 5. Then, if an event is observed, the fault is detected in lines 14 and 15, since Condition **C1** is violated. In line 35, Condition **C4** is tested and, if it is violated, the fault is detected in line 36. If the end of the path that is being executed by the system has not been reached yet, then the path estimation function and the state of the model are updated in lines 38 to 42, and the algorithm returns to line 4.

The concept of reinitializability is introduced in [21] to present a condition for ensuring that the model can always be reinitialized after the completion of some fault-free path. This condition is restated for the TAOCT model as follows.

Definition 5. *Let s_t denote an observed sequence of timed events compatible with path p'_i, i.e., $i \in \theta_s(x_0, s_t)$ and $\|s_t\| = l'_i - 1$, for some $i \in R$. Then, the TAOCT is said to be reinitializable if there does not exist $s'_t \in \overline{\{s_t\}}$ of length $\|s'_t\| = l'_z - 1$, where $z \in \theta_s(x_0, s'_t)$ and $l'_z < l'_i$, such that $x' = f(x_0, \psi(s'_t))$, and $\tilde{\lambda}(x') = y_{z,l'_z}$.* □

If the condition for reinitializability presented in Definition 5 is not satisfied, then Algorithm 2 cannot be used for fault detection, since it is not guaranteed that the model will be reinitialized after playing a fault-free path associated with p'_i. Sufficient conditions and a method for verifying the reinitializability of a DAOCT model are presented in [21]. These conditions and verification method remain valid for the TAOCT, since they are still valid for the underlying DAOCT model.

Algorithm 2: Fault detection algorithm

Input: Identified TAOCT model, $\tilde{\lambda}$, V, N
Output: Fault detection

1 Define the current state of the model $x_c \leftarrow x_0$
2 Define the current path estimate $\theta_{s,c} \leftarrow R$
3 Define the counter of event observations $\eta \leftarrow 0$
4 $\tau_{max} \leftarrow \sup \bigcup_{i \in R} \bigcup_{\sigma' \in \Gamma(x_c)} g(x_c, \sigma', i)$
5 Reset the global clock c_g
6 **while** $c_g \leq \tau_{max}$ **do**
7 Check the occurrence of an event σ
8 **if** σ *is detected* **then**
9 $\tau \leftarrow c_g(t)$
10 Go to line 13
11 Fault detected
12 Stop the algorithm
13 $\eta \leftarrow \eta + 1$
14 **if** $\sigma \notin \Gamma(x_c)$ **then**
15 Fault detected
16 Stop the algorithm
17 $\theta_{s,n} \leftarrow \theta_{s,c} \cap \theta(x_c, \sigma, \tau)$
18 **if** $\theta_{s,n} = \emptyset$ **then**
19 Fault detected
20 Stop the algorithm
21 **if** $|\theta_{s,n}| = 1 \wedge |\theta_{s,c}| > 1$ **then**
22 Find the pair $(i, \mu_i) \in N$ such that $\theta_{s,n} = \{i\}$
23 **if** $\mu_i > \eta$ **then**
24 Fault detected
25 Stop the algorithm
26 Define state $x_n \leftarrow f(x_c, \sigma)$
27 Define set $\Lambda \leftarrow \{l'_i : i \in \theta_{s,n}\}$
28 **if there exists** $l'_i \in \Lambda$ **such that** $\eta = l'_i - 1$, $\tilde{\lambda}(x_n) = y_{i,l'_i}$, **and** $u_{i,l'_i} = u_{i,1}$ **then**
29 Go to line 1
30 **if there exists** $l'_i \in \Lambda$ **such that** $\eta = l'_i - 1$, $\tilde{\lambda}(x_n) = y_{i,l'_i}$, **and** $u_{i,l'_i} \neq u_{i,1}$ **then**
31 $x_c \leftarrow x_n$
32 $\theta_{s,c} \leftarrow \emptyset$
33 $\tau_{max} \leftarrow \infty$
34 Go to line 5
35 **if** $\max_{l'_j \in \Lambda} l'_j = \eta + 1$ **then**
36 Fault detected
37 Stop the algorithm
38 $\theta'_{s,n} \leftarrow \{i \in \theta_{s,n} : l'_i = \eta + 1\}$
39 $\theta_{s,n} \leftarrow \theta_{s,n} \setminus \theta'_{s,n}$
40 $x_c \leftarrow x_n$
41 $\theta_{s,c} \leftarrow \theta_{s,n}$
42 Go to line 4

In the following example, the three situations in which a fault can be detected using the TAOCT, and not using the underlying DAOCT model, are illustrated.

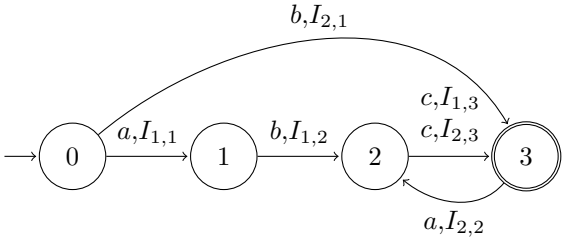

Figure 5: TAOCT model of Example 2.

Example 2. *Consider the following time-interval paths* p'_1 *and* p'_2*:*

$$p'_1 = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, a, I_{1,1}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, b, I_{1,2}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, c, I_{1,3}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right),$$

where $I_{1,1} = [28.5, 90.6]$, $I_{1,2} = [0, 46.3] \cup [153.8, 242.7] \cup [519.4, 694.8]$ *and* $I_{1,3} = [161.3, 452.6] \cup [844.8, 1125.3]$, *and*

$$p'_2 = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, b, I_{2,1}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, a, I_{2,2}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, c, I_{2,3}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

with $I_{2,1} = [200, 215]$, $I_{2,2} = [90, 110]$ *and* $I_{2,3} = [590, 620]$.

The TAOCT obtained by applying Algorithm 1, for $k = 1$, *is shown in Figure 5. The set of states of the TAOCT is* $X = \{0, 1, 2, 3\}$. *Label* σ,I *on each transition from a state* $x \in X$, *means that a guard* $g(x, \sigma, i) = I$ *is defined. The final states* X_f *are represented by double circles as in [20].*

Suppose that during system operation timed sequence $s_t = (a, 60)(b, 200)(c, 600) \in \Sigma_t^\star$ *is observed. Thus, since event* a *is feasible in state* 0, *and* a *occurs at time instant* $60 \in I_{1,1}$, *then* $(a, 60) \in L_{t,Iden}$, *and the path estimation function is* $\theta_s(0, (a, 60)) = \{1\}$. *After the occurrence of event* b, *no fault is detected since* b *is feasible in state* 1, $200 \in I_{1,2}$ *and* $\theta_s(0, (a, 60)(b, 200)) = \theta_s(0, (a, 60)) \cap \theta(1, b, 200) = \{1\} \cap \{1\} = \{1\}$. *However, after the occurrence of event* c *after* 600 *seconds in state* 2, *the fault is detected, since, although* c *is feasible in state* 2, *time instant* $600 \notin I_{1,3}$ *and* $\theta(2, c, 600) = \{2\}$. *Thus,* $\theta_s(0, (a, 60)(b, 200)(c, 600)) = \theta_s(0, (a, 60)(b, 200)) \cap \theta(2, c, 600) = \{1\} \cap \{2\} = \emptyset$. *It is important to remark that the fault would not be detected in this example if the underlying DAOCT model were used, since transition* $(2, c, 3)$ *belongs to the untimed paths associated with* p'_1 *and* p'_2.

Suppose now that sequence $s'_t = (a, 60)(b, 200)(c, 100) \in \Sigma_t^\star$ *is observed, i.e., event* c *is observed in state 2 after 100 seconds. In this case, although* c *is feasible in state* 2, *since* $100 \notin I_{1,3}$, *then a fault is detected using Algorithm 2. Note that this fault would not be detected if the underlying DAOCT were used.*

A fault that leads the fault detector to a deadlock would also be detected using Algorithm 2 if, for instance, after the occurrence of sequence $s''_t = (a, 60)(b, 200)$, *no event is*

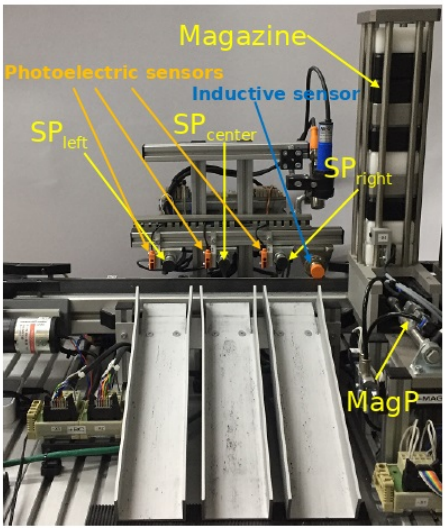

Figure 6: Sorting unit system of the practical example.

observed at state 2 *after the maximum time* $\tau_{max} = 1125.3$ *seconds has elapsed.* □

It is important to remark that all faults that can be detected using only the logical behavior of the system, modeled by the underlying DAOCT, can also be detected using the TAOCT model. Thus, there is an improvement in the fault detection capability using the timed model proposed in this paper.

6. Practical Example

6.1. Closed-loop system behavior

The identification method proposed in this paper is illustrated using the sorting unit system presented in Figure 6. Three different types of pieces are sorted in the system: white plastic pieces (WP), black plastic pieces (BP), and metallic pieces (M). Each type of piece is pushed to one of the three slides shown on the bottom of Figure 6, such that pieces of type WP are pushed to the right slide, pieces of type M are pushed to the slide in the middle, and pieces of type BP are pushed to the left slide.

On the right of Figure 6, there is a stack magazine where the pieces are stored in any order. In the sorting process, the pieces at the bottom of the stack magazine are placed onto the conveyor belt by a pneumatic pusher. Then, the conveyor belt is turned on, and the piece is moved in the direction of two sensors in order to determine its type. An inductive sensor detects metallic pieces (type M), and an optical sensor detects metallic (type M) and white plastic pieces (type WP). If a black plastic piece (type BP) is on the conveyor, then none of the sensors is capable of detecting it. The optical sensor is located close to the inductive sensor, such that metallic pieces are detected by both sensors almost at the same time.

It is also important to remark that there is a photoelectric sensor next to each sorting pusher on the conveyor.

Table 1: Table containing the description of each I/O signal.

d	**Description of the I/O signal**
1	Sensor indicating the extension of MagP
2	Sensor indicating the retraction of MagP
3	Sensor indicating the extension of SP_{left}
4	Sensor indicating the retraction of SP_{left}
5	Sensor indicating the extension of SP_{center}
6	Sensor indicating the retraction of SP_{center}
7	Sensor indicating the extension of SP_{right}
8	Sensor indicating the retraction of SP_{right}
9	Optical sensor
10	Inductive sensor
11	Photoelectric sensor of SP_{left}
12	Photoelectric sensor of SP_{center}
13	Photoelectric sensor of SP_{right}
14	Command to activate SP_{left}
15	Command to activate SP_{center}
16	Command to activate SP_{right}
17	Command to extend MagP
18	Command to retract MagP
19	Command to activate the conveyor belt

When a piece is detected by the photoelectric sensor next to the pusher that should remove it from the conveyor, the conveyor is stopped and the pusher is extended. Then, the pusher is retracted and a new piece can be placed on the conveyor by the pusher of the stack magazine.

The sorting pushers are denoted as SP_{left}, SP_{center} and SP_{right}, for the left, center and right sorting pusher of Figure 6, respectively. The magazine pusher is denoted as MagP. The sorting unit system has 13 sensors and 6 actuator signals. Thus, the controller has 19 input and output signals. Table 1 gives the position number d of each controller signal in the I/O vector, along with their description.

The initial state of all observed paths is defined as the I/O vector corresponding to the case where the conveyor belt is turned off, and all pushers are retracted.

6.2. TAOCT model computation

During the identification process, 2294 timed paths with lengths ranging from 6 to 14 I/O vectors were observed, which corresponds to two hours and fifty three minutes of observation of the controller signals. A total of 12 logically distinguishable paths were obtained. As there are only three types of pieces, only three different logically distinguishable sequences would be expected. However, it has been observed that, since the inductive and optical sensors are very close to each other, the order of sensor readings (rising and falling edges) may change for different sorting cycles of metallic pieces, increasing the number of paths.

In this practical example, it has been chosen the threshold $\zeta = 80$ms and uncertainty $\delta = 1$ms, which is equal to the measured scan cycle, to obtain the time-interval paths.

After obtaining the time-interval paths p'_i, the TAOCT model is computed following the steps of Algorithm 1. The identified TAOCT for $k = 1$ is depicted in Figure 7. In this case, the identified TAOCT has 26 states and 38 transitions. In Figure 7, the guards are not presented due to lack of space. The events of the TAOCT are rising and falling edges of the elements of the I/O controller vector, which are represented by $\uparrow d$ and $\downarrow d$, respectively, where d is the position of the signal in the I/O vector. It is important to remark that, according to Definition 2, an event can be formed of more than one rising or falling edge of controller signals.

The programming code for computing the TAOCT model was implemented using Python 3.7, on a computer Intel Core i7 with 2.4GHz and 8 GB RAM. The time required for computing the time-interval paths from the fault-free observed timed paths was 535 ms. The most time-consuming operations are those related to the formation of clusters, which are carried out for every transition of every observed path. The time required for the computation of the TAOCT model was only 2.4 ms. Thus, the overall time for the computation of the TAOCT model, including the computation of the time-interval paths p'_i, is approximately 538 ms.

6.3. Fault detection based on the TAOCT model

In the sequel, three faulty scenarios for which the fault can be detected thanks to the timing information added to the TAOCT model are presented.

In the first scenario, consider that, after a piece is placed on the conveyor belt by pusher MagP, the pusher stuck extended and cannot be retracted. The path associated with this behavior is $p = (x_0, \uparrow 17, x_1, \downarrow 2, x_2, \uparrow 1. \downarrow 17. \uparrow 18, x_3)$. In this case, the system deadlocks, since the conveyor belt is turned on only after the retraction of the pusher is detected ($\uparrow 2$), which never occurs. This fault cannot be detected by using the DAOCT model, but can be detected by using the TAOCT model, since the falling edge of the sensor that indicates that MagP is extended ($\downarrow 1$) must occur before the maximum time of the guard $g(x_3, \downarrow 1, 1) = [87, 161]$. Thus, when the elapsed time is greater than 161 milliseconds, the fault is detected.

To illustrate the second faulty scenario, consider a fault in the speed controller of the conveyor that makes it work faster than expected. Consider, for instance, that after a BP piece is placed on the conveyor belt, and the conveyor is turned on ($\uparrow 19$), which corresponds to state x_5 of the TAOCT, it reaches the photoelectric sensor next to the first sorting pusher ($\uparrow 13$) in a time smaller than the minimum time of the guard $g(x_5, \uparrow 13, 4) = [4058, 4146]$. Thus, the fault is detected. It is important to remark that, since $\uparrow 13$ is coherent with the logical behavior of the system, then the detection system based on the underlying DAOCT model would not be capable of detecting the fault.

The third faulty scenario can be illustrated by the following example. Consider all observed paths associated with BP or M pieces, and consider that the piece is in front of the photoelectric sensor next to the right sorting pusher, *i.e.*, the rising edge of the photoelectric sensor ($\uparrow 13$) has been observed, which corresponds to state x_8 of Figure 7. By analyzing the time elapsed between $\uparrow 13$ and $\downarrow 13$, two distinct sets of time values can be defined according to the type of piece, as shown in Figure 8. This occurs because metallic pieces are detected for a longer time than plastic pieces by the photoelectric sensor due to their brightness. Consider now a fault that causes both the optical and inductive sensors to fail at the same time. In this case, a piece of type M would lead to the same logical behavior as a BP piece, making such a fault non-detectable by the diagnosis system based on the underlying DAOCT model. However, as shown in Figure 8, it is possible to distinguish the types of pieces by using the guards associated with the timed paths. While a BP piece would take some time in the interval $g(x_8, \downarrow 13, 4) = [915, 937]$, a metallic piece would take some value in the interval $[1035, 1052]$ milliseconds, which corresponds to the union of all guards defined in state x_8, for event $\downarrow 13$, and the timed paths associated with metallic pieces. Thus, if a metallic piece is on the conveyor belt, and the fault occurs, the time elapsed between the rising and falling edges of the photoelectric sensor will be compatible with the guard condition associated with metallic pieces, and non-compatible with the guard condition of black plastic pieces. Since the logical behavior is not coherent with the timing behavior, the fault is detected.

7. Conclusions

In this paper, a new timed model for the identification of DES with the aim of fault detection is proposed. In this model, called Timed Automaton with Outputs and Conditional Transitions (TAOCT), timing information regarding the occurrence of events are added as guards to the transitions. By doing so, a refinement of the path estimation can be carried out using the timing information, and faults that cannot be detected by using untimed models, can be detected by using the TAOCT. An algorithm describing the fault detection based on the TAOCT model has also been proposed in this paper, and a practical example has been used to illustrate the detection algorithm.

Declaration of competing interest

None declared.

Acknowledgments

This study was financed in part by CNPq, FAPERJ, and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Finance Code 001.

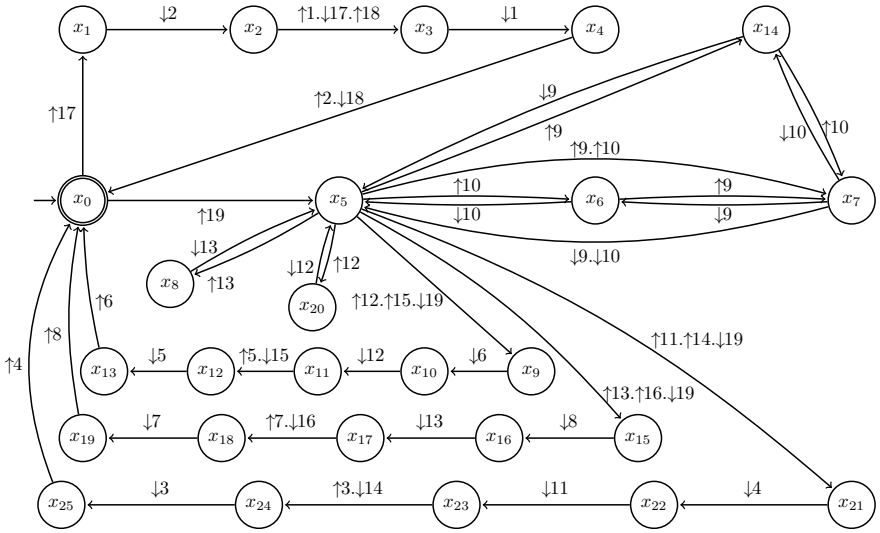

Figure 7: TAOCT model obtained for the practical example.

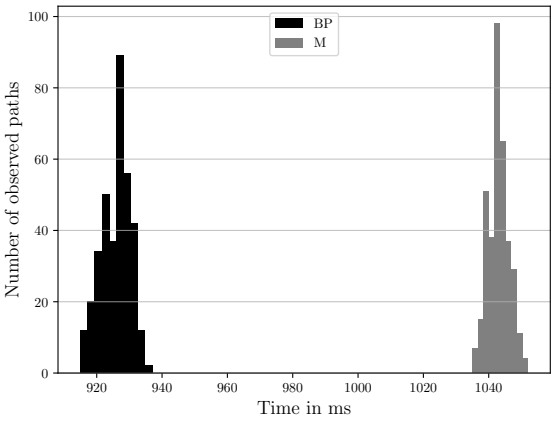

Figure 8: Histogram showing the observed times elapsed between the rising edge and falling edge of the photoelectric sensor next to the first sorting pusher (BP in black and M in grey) in the observed timed paths.

References

[1] Basile, F., Chiacchio, P., Coppola, J., 2016. A novel model repair approach of timed discrete-event systems with anomalies. IEEE Transactions on Automation Science and Engineering 13, 1541–1556.

[2] Basile, F., Chiacchio, P., Coppola, J., 2017. Identification of time petri net models. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 2586–2600.

[3] Cabasino, M.P., Giua, A., Hadjicostis, C.N., Seatzu, C., 2014. Fault model identification and synthesis in petri nets. Discrete Event Dynamic Systems 24, 275–307.

[4] Cabasino, M.P., Giua, A., Seatzu, C., 2007. Identification of petri nets from knowledge of their language. Discrete Event Dynamic Systems 17, 447–474.

[5] Cabral, F.G., Moreira, M.V., 2020. Synchronous diagnosis of discrete-event systems. IEEE Transactions on Automation Science and Engineering 17, 921–932.

[6] Cassandras, C.G., Lafortune, S., 2008. Introduction to Discrete Event Systems. Springer Publishing Company.

[7] Chen, Y.L., Provan, G., 1997. Modeling and diagnosis of timed discrete event systems - a factory automation example, in: Proceedings of the 1997 American Control Conference, IEEE. pp. 31–36.

[8] Debouk, R., Lafortune, S., Teneketzis, D., 2000. Coordinated decentralized protocols for failure diagnosis of discrete event systems. Discrete Event Dynamic Systems 10, 33–86.

[9] Dotoli, M., Fanti, M.P., Mangini, A.M., 2008. Real time identification of discrete event systems using petri nets. Automatica 44, 1209–1219.

[10] Dotoli, M., Fanti, M.P., Mangini, A.M., Ukovich, W., 2011. Identification of the unobservable behaviour of industrial automation systems by petri nets. Control Engineering Practice 19, 958–966.

[11] Estrada-Vargas, A.P., López-Mellado, E., Lesage, J., 2017. A black-box identification method for automated discrete-event systems. IEEE Transactions on Automation Science and Engineering 14, 1321–1336.

[12] Estrada-Vargas, A.P., López-Mellado, E., Lesage, J.J., 2010. A comparative analysis of recent identification approaches for discrete-event systems. Math. Probl. Eng. 2010, 1–21.

[13] Hartigan, J., 1975. Clustering algorithms. Wiley.

[14] Hu, Y., Ma, Z., Li, Z., 2020. Design of supervisors for active diagnosis in discrete event systems. IEEE Transactions on Automatic Control doi:10.1109/TAC.2020.2970011.

[15] Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: A review. ACM Comput. Surv. 31, 264–323.

[16] Klein, S., Litz, L., Lesage, J.J., 2005. Fault detection of discrete event systems using an identification approach. IFAC Proceedings Volumes 38, 92–97. 16th IFAC World Congress.

[17] Lu, S.Y., Fu, K.S., 1978. A sentence-to-sentence clustering procedure for pattern analysis. IEEE Transactions on Systems, Man, and Cybernetics 8, 381–389.

[18] Medhi, S.O.E., Leclercq, E., Lefebvre, D., 2006. Petri nets design and identification for the diagnosis of discrete event systems, in: IAR Annu. Meeting.

[19] Moreira, M.V., Jesus, T.C., Basilio, J.C., 2011. Polynomial time verification of decentralized diagnosability of discrete event systems. IEEE Transactions on Automatic Control 56, 1679–1684.

[20] Moreira, M.V., Lesage, J.J., 2019a. Discrete event system identification with the aim of fault detection. Discrete Event Dynamic Systems 29, 1–19.

[21] Moreira, M.V., Lesage, J.J., 2019b. Fault diagnosis based on identified discrete-event models. Control Engineering Practice 91, 104101.

[22] Pandalai, D.N., Holloway, L.E., 2000. Template languages for fault monitoring of timed discrete event processes. IEEE Transactions on Automatic Control 45, 868–882.
[23] Pham, D.T., Dimov, S.S., Nguyen, C.D., 2005. Selection of k in k-means clustering. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 219, 103–119.
[24] Ran, N., Giua, A., Seatzu, C., 2019. Enforcement of diagnosability in labeled petri nets via optimal sensor selection. IEEE Transactions on Automatic Control 64, 2997–3004.
[25] Roth, M., Lesage, J.J., Litz, L., 2009. An FDI method for manufacturing systems based on an identified model, in: 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2009), Moscow, Russia. pp. 1406–1411.
[26] Roth, M., Lesage, J.J., Litz, L., 2011. The concept of residuals for fault localization in discrete event systems. Control Engineering Practice 19, 978–988.
[27] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D., 1995. Diagnosability of discrete event systems. IEEE Transactions on Automatic Control 40, 1555 – 1575.
[28] Santoro, L.P.M., Moreira, M.V., Basilio, J.C., 2017. Computation of minimal diagnosis bases of discrete-event systems using verifiers. Automatica 77, 93–102.
[29] Schneider, S., 2015. Automatic Modeling and Fault Diagnosis of Timed Concurrent Discrete Event Systems. Ph.D. thesis. École Normale Supérieure de Cachan - ENS Cachan.
[30] Schneider, S., Litz, L., Danancher, M., 2011. Timed residuals for fault detection and isolation in discrete event systems, in: 3rd International Workshop on Dependable Control of Discrete Systems (DCDS2011), Saarbrücken, Germany. pp. 35–40.
[31] Schneider, S., Litz, L., Lesage, J.J., 2012. Determination of timed transitions in identified discrete-event models for fault detection, in: 51st IEEE Annual Conference on Decision and Control (CDC'12), Maui, HI, USA. pp. 5816–5821.
[32] de Souza, R.P.C., Moreira, M.V., Lesage, J.J., 2020. A timed model for discrete event system identification and fault detection, in: 21st IFAC World Congress, Berlin, Germany. pp. 826–831.
[33] Supavatanakul, P., Lunze, J., Puig, V., Quevedo, J., 2006. Diagnosis of timed automata: Theory and application to the damadics actuator benchmark problem. Control Engineering Practice 14, 609–619.
[34] Tripakis, S., 2002. Fault diagnosis for timed automata, in: International symposium on formal techniques in real-time and fault-tolerant systems, Springer, Oldenburg, Germany. pp. 205–221.
[35] Zad, S.H., Kwong, R.H., Wonham, W.M., 2005. Fault diagnosis in discrete-event systems: Incorporating timing information. IEEE Transactions on Automatic Control 50, 1010–1015.
[36] Zaytoon, J., Lafortune, S., 2013. Overview of fault diagnosis methods for discrete event systems. Annual Reviews in Control 37, 308–320.