Int. J. Appl. Math. Comput. Sci., 2018, Vol. 28, No. 2, 269–281
DOI: 10.2478/amcs-2018-0019

ON–THE–FLY DIAGNOSABILITY ANALYSIS OF BOUNDED AND UNBOUNDED LABELED PETRI NETS USING VERIFIER NETS

BEN LI [a,*], MANEL KHLIF-BOUASSIDA [a], ARMAND TOGUYÉNI [a]

[a]CRIStAL, UMR 9189
Centrale Lille, 59650 Villeneuve d'Ascq, France
e-mail: {ben.li,manel.khlif-bouassida,armand.toguyeni}@centralelille.fr

This paper considers the problem of diagnosability analysis of discrete event systems modeled by labeled Petri nets (LPNs). We assume that the LPN can be *bounded* or *unbounded* with no deadlock after firing any fault transition. Our approach is novel and presents the on-the-fly diagnosability analysis using verifier nets. For a given LPN model, the verifier net and its reachability graph (for a *bounded* LPN) or coverability graph (for an *unbounded* LPN) are built on-the-fly and in parallel for diagnosability analysis. As soon as a diagnosability decision is established, the construction is stopped. This approach achieves a compromise between computation limitations due to efficiency and combinatorial explosion and it is useful to implement an engineering approach to the diagnosability analysis of complex systems.

Keywords: fault diagnosis, discrete event systems, labeled Petri nets, on-the-fly diagnosability analysis, verifier nets.

1. Introduction

This paper addresses the fault diagnosis problem of discrete event systems (DESs). Two main issues of fault diagnosis of DES models include (i) on-line diagnosis of the system (Sampath *et al*., 1995; Lefebvre and Delherm, 2007; Basile *et al*., 2009; Cabral *et al*., 2015) and (ii) off-line diagnosability analysis of the system (Sampath *et al.*, 1995; Basile *et al.*, 2012; Cabasino *et al.*, 2012; 2014; Liu *et al.*, 2014). The on-line diagnosis is to deduce the occurrence of faults (represented by unobservable events) and their types by using observable events, while the system is running. The diagnosability represents the ability to detect a fault in a finite delay after its occurrence, based on observations. The diagnosability is checked "off-line" (at the design stage of the system) and must be ensured before implementing the system. The on-line diagnosis is executable if and only if all the faults are "diagnosable".

Sampath *et al*. (1995) provided a formal definition of diagnosability in the framework of automata and formal languages. The diagnoser is introduced to perform both diagnosability analysis and on-line diagnosis. For a given automaton, an intermediary model (i.e., generator) is built, which reduces all the unobservable transitions by using the ϵ-reduction technique. Afterwards, the diagnoser is built according to the idea of state estimation. The system diagnosability is related to the existence of an indeterminate cycle in the diagnoser. The verification of the existence of an indeterminate cycle contains two steps: (i) if there exists an F-uncertain cycle in the diagnoser and each of its macro-states (states of the diagnoser) contains at least one normal state and one faulty state of the system; (ii) if there exist one normal cycle and one faulty cycle in the intermediary model, which correspond to the F-uncertain cycle in diagnoser. The system is diagnosable if there is no indeterminate cycle. The diagnoser approach can be applied for the diagnosability analysis of Petri nets (PNs) by working on the reachability graph (RG).

One of the problems of the diagnoser approach by Sampath *et al*. (1995) is the combinatorial explosion problem. The construction of an intermediary model involves memory costs. The diagnoser is built by enumerating all the possible states. In particular, for a PN, the RG is built at first by enumerating all the markings. In order to overcome this problem, some research efforts have been made. In the work of Cabasino *et al*. (2014), the modified basis reachability graph (MBRG) and basis reachability diagnoser (BRD) were developed. The authors provided a compact manner for the construction of the state space for diagnosability analysis of bounded

*Corresponding author

labeled Petri nets (LPNs). Liu *et al*. (2014) proposed an on-the-fly diagnosability analysis technique to analyze the diagnosability of bounded LPNs. Only a part of the FM-graph (fault marking graph, which plays the role of the RG) and the FM-set tree (fault marking set tree, which plays the role of the diagnoser) are built on the fly and in parallel with stop conditions, instead of building the whole state space and the whole diagnoser *a priori*. The approach was improved by using minimal explanations and T-invariants by Li *et al.* (2015b; 2015a). In the work of Boussif *et al*. (2015), a new variant of the diagnoser was proposed. The variant diagnoser was built without building any intermediate model. Moreover, Boussif *et al*. (2015) proved that if there exists an F-uncertain cycle in the diagnoser, the normal cycle always exists. Therefore, it is only necessary to verify the existence of a faulty cycle and the authors verify it by using the model of the given automaton. However, it is worth noticing that the complexity of these approaches is the same as that of the diagnoser approach by Sampath *et al*. (1995).

Another problem of the diagnoser approach is its computational complexity. In the worst case, the complexity of building a diagnoser is exponential with respect to the state space of the initial model. Some approaches have been proposed to reduce the computational complexity. In the works of Jiang *et al*. (2001) as well as Yoo and Lafortune (2002), a twin-plant and a verifier was created. The idea is to build a parallel composition of the given automaton and a copy of itself. The verification of diagnosability is simplified compared with the diagnoser approach. It is only necessary to verify, in the verifier or the twin-plant, the existence of an F-confused cycle, which is composed of the nodes that contains both normal and faulty states. The complexity of these approaches is polynomial. Moreira *et al*. (2011) improved the verifier approach by building a parallel composition of the normal and faulty parts of a given automaton modeling the complete behavior of the system. The size of the verifier is reduced. In terms of the PN-based approach, in the work of Cabasino *et al*. (2012), a structure called the verifier net (VN) was developed to check the diagnosability for both bounded and unbounded PNs. The K-diagnosablity of unbounded PNs is analyzed as well. A necessary and sufficient condition for diagnosability was proposed, which is based on searching for the existence of a repetitive sequence in the reachability graph (RG) (resp. the coverability graph (CG)) of *bounded* PNs (resp. *unbounded* PNs). The complexity of analyzing a *bounded* LPN is polynomial, but that of analyzing an *unbounded* LPN is still an open issue. However, the problem of these polynomial-time approaches is the combinatorial explosion. Since these approaches build the parallel composition of two models having the same size of the given model, the model constructed for diagnosability analysis (verifier, twin-plant, VN and its RG/CG) is too large to analyze a complex system.

In this paper, we propose a VN-based approach, because of the advantages of the VN approach (Cabasino *et al*., 2012): (i) the capacity to analyze both bounded and unbounded LPN; (ii) the simplicity of verifying the necessary and sufficient condition for diagnosability; (iii) the polynomial complexity of the diagnosability analysis of bounded LPNs. We improve the VN approach to reduce the combinatorial explosion problem in certain cases. Moreover, we aim at proposing a new approach for practical and industrial use.

In practice, there is no guarantee that the diagnosability property is initially fulfilled, because the system is instrumented from the perspective of the normal control. Therefore, this shows the needs for the development of engineering methods that allow iterating diagnosability analysis, followed by the modification of the system model, until it becomes diagnosable.

We suppose that the LPN system is initially nondiagnosable. Hence, we need to develop an engineering approach as mentioned above. Most of the approaches to diagnosability analysis proposed in literature are based on an *a-priori* construction of the state space. However, if the model is changed in order to make it diagnosable, we must rebuild the state space, which is very large for a complex system. To overcome this problem, we choose to use a standard technique in model checking called "on-the-fly analysis." The state space is constructed on the fly. As soon as a stop condition is found, the construction and the analysis are stopped immediately. This technique increases slightly the computational complexity, but it reduces the memory cost.

In this study, the idea of on-the-fly analysis is applied. A new approach is developed in order to reduce the combinatorial explosion and to save the memory cost, while analyzing the diagnosability of bounded and unbounded LPNs. This approach is based on the depth-first search. The VN and its RG/CG are built on-the-fly and in parallel with stop conditions. The computational complexity of this approach is analyzed. This approach achieves a compromise between computational efficiency and limitations of combinatorial explosion.

This paper is an extended version of our earlier work (Li *et al*., 2016). It is structured as follows. In Section 2, the idea of the on-the-fly diagnosability analysis is explained. In Section 3, some preliminary notions used in this paper are given. Section 4 recalls the VN approach. The on-the-fly diagnosability analysis using VNs is proposed in Section 5. In Section 6, a comparative analysis of our approach and the VN approach is given. Finally, conclusions and some perspectives of this work are given in Section 7.

2. On-the-fly diagnosability analysis

In practice, the initial design of an LPN model is based on the requirement of supervisory control without considering the diagnosability property. Once the LPN model is built, the diagnosability is checked by using the proposed diagnosability analysis approaches. If the system is not diagnosable, the DES model needs to be modified and its diagnosability is checked again. The above process is iterated until the LPN becomes diagnosable.

The process of most diagnosability analysis approaches in literature is shown in Fig. 1. For a given LPN, the diagnosability analysis is transferred to the study of its reachability graph (RG) or some kinds of modified RG (e.g., MBRG (Cabasino *et al*., 2014)). Therefore, the whole RG is necessarily built. Then, a diagnoser is built based on the RG for the diagnosability analysis. The idea of the VN approach by Cabasino *et al*. (2012) is similar, and the reachability graph (for a *bounded* LPN) or the coverability graph (for an *unbounded* LPN) of the VN is the model that is analyzed to make the diagnosability decision.

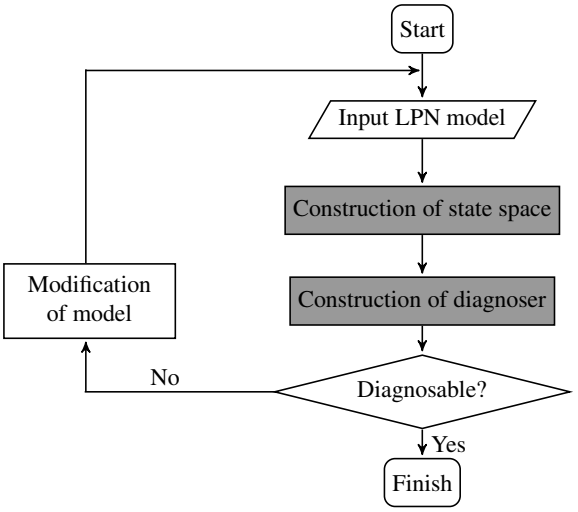

Fig. 1. Process of most approaches to diagnosability analysis.

If the LPN model is nondiagnosable, the model is modified and then its VN and its RG/CG of the modified model are built once again. This "serial" process is not favorable for industrial use, because the state space of the monolithic model is rebuilt each time when the system is modified. Hence, for a large-scale LPN, these approaches based on this process are not efficient for iterating the diagnosability analysis and there exists a combinatorial explosion problem.

To overcome the combinatorial explosion problem while iterating the diagnosability analysis, the process of the proposed technique is different, which is shown in Fig. 2. The on-the-fly diagnosability analysis does not build entirely the state space *a priori*, nor the model for diagnosis: they are built on the fly and in parallel with stop conditions, which stop building some of their branches. Along with their on-the-fly construction, the diagnosability and is analyzed. When the condition of undiagnosability is satisfied, the result that the LPN model is not diagnosable is immediately given. This technique shows that a part of the state space could suffice for diagnosability analysis, particularly when the system is nondiagnosable, in order to reduce the memory cost and solve the combinatorial explosion problem. In this paper, the VN approach is improved by applying the on-the-fly diagnosability analysis. The VN and its RG/CG are built on-the-fly and in parallel. The construction is stopped, when the condition of undiagnosability is satisfied. This approach is favorable for iterating the diagnosability analysis for industrial use.

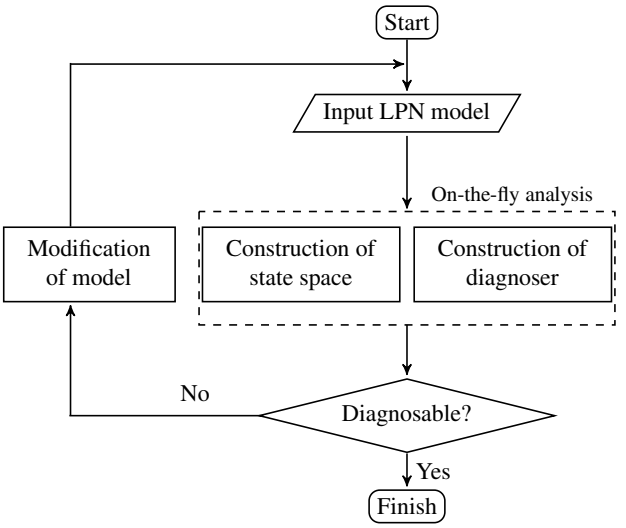

Fig. 2. Process of on-the-fly diagnosability analysis

3. Preliminaries

3.1. Labeled Petri nets. Petri nets are a mathematical and graphical modeling notation for DESs, which are presented by the quadruple $N = (P, T, Pre, Post)$, where P is a finite set of places; T is a finite set of transitions; Pre and $Post$ are the pre- and post-incidence matrices. A marking is a vector $M \in \mathbb{N}^{|P|}$, which assigns a nonnegative integer to each place. The markings of a Petri net represent different states and dynamic behaviors of the system. $M(p)$ denotes the number of tokens in place p and M_o is the initial marking of N. (N, M_0) is a marked PN with initial marking M_0. The incidence matrix is $C = Post - Pre$.

A transition t is enabled at M iff $M \geq Pre(\cdot, t)$,

denoted by $M[t>$. $M[\sigma>$ denotes that the sequence of transitions $\sigma \in T^*$ is enabled at M, where T^* is the Kleene closure of the set T. Here λ denotes an empty sequence of transitions, i.e., $\forall \sigma \in T^*, \sigma\lambda = \sigma$.

For a given sequence $\sigma \in T^*$, $\pi : T^* \to \mathbb{N}^{|T|}$ is the function that assigns a vector $\overrightarrow{y} \in \mathbb{N}^{|T|}$ to σ. Furthermore, $\overrightarrow{y} = \pi(\sigma)$ is called the firing vector of σ, and $\overrightarrow{y}(t) = k$ means that transition t is contained k times in σ. The reached marking M' is computed by $M' = M + C \cdot \overrightarrow{y}$. A marking M is reachable in (N, M_0) iff a sequence σ exists such that $M_0[\sigma > M$. The set of all the reachable markings from M_0 is denoted by $R(N, M_0)$ and called the reachability set of (N, M_0).

A PN (N, M_0) is said to be *bounded* if there exists a positive number m such that $\forall M \in R(N, M_0),\ M(p) \leq m$. The reachability space of a bounded PN is finite and it is represented by a graph called the reachability graph (RG). If the number of tokens in one or more places can be arbitrarily large, the PN is unbounded and the coverability graph (CG) is used to represent the infinite state space.

Definition 1. Given a PN (N, M_0), a transition t is

- *dead*: if there does not exist a reachable marking $M \in R(N, M_0)$ that enables t;
- *semilive*: if there exists at least one reachable marking $M \in R(N, M_0)$ that enables t;
- *live*: if for each reachable marking $M \in R(N, M_0)$, t is semilive in (N, M);

A PN (N, M_0) is *live* if each transition $t \in T$ is live. In other words, a PN is live if, from any marking in $R(N, M_0)$, it is possible to fire any transition by progressing through some further firing sequences. A *deadlock* occurs at marking M if no transition can be enabled at M.

Definition 2. A sequence $\sigma \in T^*$ is called *repetitive* if there exists a marking $M_1 \in R(N, M_0)$ such that $M_1[\sigma > M_2[\sigma > \cdots$, i.e., if it can fire infinite times starting from M_1. A repetitive sequence is called

(i) *stationary*: if $M_{i+1} = M_i$ for all $i = 1, 2, \ldots$;

(ii) *increasing*: if $M_{i+1} \gneq M_i$ for all $i = 1, 2, \ldots$. If there exists an increasing repetitive sequence, the PN system is unbounded.

A labeled Petri net (LPN), an extension of the PN, is the quadruple $N_L = (N, M_o, \Sigma, \mathcal{L})$, where (N, M_o) is a marked PN. Σ is a finite set of events, $\Sigma = \Sigma_o \uplus \{\varepsilon\}$ ("$\uplus$" is used to emphasize that the two sets are disjoint). Σ_o is the set of observable events that are associated to observable transitions and the label of all unobservable transitions is ε. $\mathcal{L}\colon T \to \Sigma$ is the transition labeling function which assigns a label to every transition. The same label could be shared by different transitions. $\mathcal{L}$ can be extended to $\mathcal{L}\colon T^* \to \Sigma^*$.

Definition 3. Given a Petri net $N = (P, T, Pre, Post)$. $T' \subseteq T$ is a subset of the transitions in T. The T'-induced subnet of N is defined as the new Petri net $N' = (P, T', Pre', Post')$, where $Pre', Post'$ are the restrictions of $Pre, Post$. In this case, we write $N' \prec_{T'} N$. The net N' can be considered as being obtained by removing all transitions in $T \setminus T'$ and all related arcs.

3.2. Diagnosability. In event-based diagnosis of DESs using LPN models, the set of transitions is partitioned into two disjoint sets, $T = T_o \uplus T_u$, where T_o is the set of observable transitions, and T_u is the set of unobservable transitions. The label of an observable transition can be observed when it fires. The fault transitions are unobservable. The set of unobservable transitions is partitioned into two disjoint sets, $T_u = T_f \uplus T_{reg}$, where T_f includes all fault transitions, while $T_{reg} = T_u \backslash T_f$ is the set of regular unobservable transitions. The set T_f can be further partitioned into k different subsets T_f^i, where $i = 1, \ldots, k$, which represents different classes of faulty transitions. $|T|$ is the size of T which indicates the number of transitions in T.

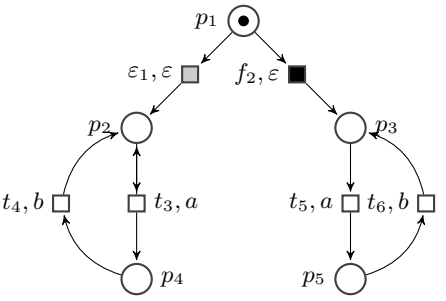

Fig. 3. Example of the LPN.

Example 1. Consider the example of the LPN in Fig. 1, where $T_o = \{t_3, t_4, t_5, t_6\}$, $T_{reg} = \{\varepsilon_1\}$ and $T_f = \{f_2\}$. a and b are observable events such that $\mathcal{L}(t_3) = \mathcal{L}(t_5) = a$ and $\mathcal{L}(t_4) = \mathcal{L}(t_6) = b$. The label of all unobservable transitions is ε. This LPN is unbounded. At the initial marking $[1\ 0\ 0\ 0\ 0]^\tau$, if the transition ε_1 is fired, then the transition t_3 can be fired infinite times. Therefore, the number of tokens in place p_4 can be arbitrarily large. ♦

The definition of the diagnosability of a system modeled by an LPN is given as follows:

Definition 4. (*Diagnosability*) Given a live LPN $N_L = (N, M_o, \Sigma, \mathcal{L})$, N_L is diagnosable with respect to a fault class T_f^i if there are not two sequences σ_1 and σ_2 which satisfy the following conditions: (i) $\forall t_f \in T_f^i$, $t_f \notin \sigma_1$; (ii) $\exists t_f \in T_f^i$ such that $t_f \in \sigma_2$ and σ_2 can be arbitrarily long after the occurrence of t_f; (iii) $\mathcal{L}(\sigma_1) = \mathcal{L}(\sigma_2)$.

In other terms, in a diagnosable LPN, two sequences of transitions with the same observation cannot be found, such that one contains a fault transition and can be arbitrarily long after its occurrence; the other does not contain a fault transition.

The following assumptions are given for all the approaches mentioned in this paper: (i) the LPN does not deadlock after firing any fault transition; (ii) no cycle of unobservable transitions exists; (iii) the same label may be associated with different transitions; (iv) the structure of LPN and the initial marking M_0 are well known.

4. Verifier net

In the work of Cabasino *et al*. (2012), the concept of a verifier net (VN) is developed for diagnosability analysis of bounded and unbounded PNs. In this section, the VN approach is recalled. Without loss of generality, in the current paper, the diagnosis issue is discussed for a single class of faults. For simplicity, the superscript i in T_f^i will be omitted. The VN approach cannot analyze the diagnosability of an LPN system with several fault classes at the same time. For each fault class, one VN and its RG/CG need to be built.

Given an LPN system, $N_L = (N, M_o, \Sigma, \mathcal{L})$, where $N = (P, T, Pre, Post)$. Let $N' = (P', T', Pre', Post')$ be the T'-induced subnet, where $T' = T \backslash T_f = T_o \cup T_{reg}$. P and P' are used to distinguish among places of N and N', and they are disjoint even if they represent the same places. Analogously, T and T' are used to distinguish among transitions of N and N', and they are disjoint even if they represent the same transitions. The LPN system associated with N' is called $T'-induced\ sub-LPN$ and denoted by $N'_L = (N', M'_o, \Sigma', \mathcal{L}')$, where $M'_o = M_o$, $\Sigma' = \Sigma$ and the labeling function $\mathcal{L}'$ is defined by $\mathcal{L}$ but restricted to T'.

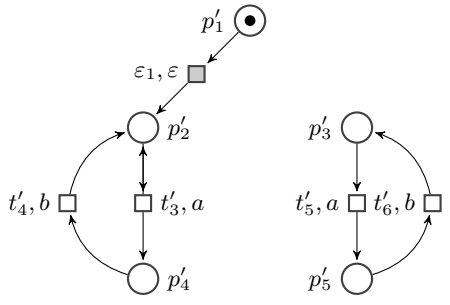

Fig. 4. T'-induced sub-LPN of the LPN in Fig. 3.

Example 2. For the LPN system N_L in Fig. 3, its T'-induced sub-LPN N'_L is depicted in Fig. 4. N'_L is obtained by removing the fault transition f_2, ε and the corresponding arcs. ♦

The VN is an LPN system constructed by composing N'_L with N_L with the synchronization on the observable transition labels. The VN is denoted by $\widetilde{N_L} = (\widetilde{N}, \widetilde{M}_o, \widetilde{\Sigma}_o, \widetilde{\mathcal{L}})$. $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{Pre}, \widetilde{Post})$, where $\widetilde{P} = P' \cup P$, $\widetilde{T} = \widetilde{T}_o \cup (T'_{reg} \times \{\lambda\}) \cup (\{\lambda\} \times T_{reg}) \cup (\{\lambda\} \times T_f)$ and $\widetilde{T}_o = \{(t', t) \mid t' \in T'_o, t \in T_o, \mathcal{L}'(t') = \mathcal{L}(t)\}$. Furthermore,

$$\widetilde{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix},$$

$\widetilde{\Sigma}_o = \{(\Sigma'_o \times \Sigma_o) \cup \{\varepsilon\}\}$ and $\widetilde{\mathcal{L}} : \widetilde{T} \to \widetilde{\Sigma}_o$. The incidence matrix of VN is $\widetilde{C} = \widetilde{Post} - \widetilde{Pre}$.

Example 3. For the LPN system N_L displayed in Fig. 3, the VN is depicted in Fig. 5. $\widetilde{T}_o = \{(t'_3, t_3), (t'_3, t_5), (t'_5, t_3), (t'_5, t_5), (t'_4, t_4), (t'_6, t_4), (t'_4, t_6), (t'_6, t_6)\}$, $T'_{reg} \times \{\lambda\} = \{(\varepsilon'_1, \lambda)\}$, $\{\lambda\} \times T_{reg} = \{(\lambda, \varepsilon_1)\}$ and $\{\lambda\} \times T_f = \{(\lambda, f_2)\}$. ♦

To analyze the diagnosability of an LPN, the VN is constructed. Afterwards, its RG (for a bounded LPN) or CG (for an unbounded LPN) is built. A sufficient and necessary condition for diagnosability is proposed. Let $F(VN)$ denote the set of faulty nodes in the RG/CG of the VN. A node belongs to $F(VN)$, if it can be reached by firing a sequence that contains a fault transition.

Theorem 1. (Cabasino *et al.*, 2012) *An LPN system $N_L = (N, M_o, \Sigma, \mathcal{L})$ is diagnosable iff there does not exist any cycle associated with a firable repetitive sequence in the VN that is reachable starting from any node in the set $F(VN)$.*

For a bounded LPN, the cycle in the RG of the VN corresponds to a firable repetitive sequence. For an unbounded PN, if a cycle in the CG is reachable starting from a node in the set $F(VN)$, it is necessary to check if it is a real cycle, i.e., if the cycle is associated with a repetitive sequence, i.e., if $\widetilde{C} \cdot \overrightarrow{y} \geq \overrightarrow{0}$, where $\overrightarrow{y}$ is the firing vector that contains all the transitions of the cycle.

Example 4. For the LPN system N_L in Fig. 3, the VN is depicted in Fig. 5. Since N_L is unbounded, its CG is built in Fig. 6. The markings in Fig. 6 are shown in Table 1. There exists a cycle after the occurrence of the fault:

$$\widetilde{M}_9 \xrightarrow{(t'_3, t_5), a} \widetilde{M}_{10} \xrightarrow{(t'_4, t_6), b} \widetilde{M}_9 \cdots$$

One can check that the sequence of transitions $((t'_3, t_5)(t'_4, t_6))^*$ corresponds to a repetitive sequence. According to Theorem 1, the LPN is not diagnosable. ♦

5. On-the-fly diagnosability analysis using verifier nets

In this section, a new approach is developed for the on-the-fly diagnosability analysis using VNs. The algorithm, which is based on the depth-first search, is

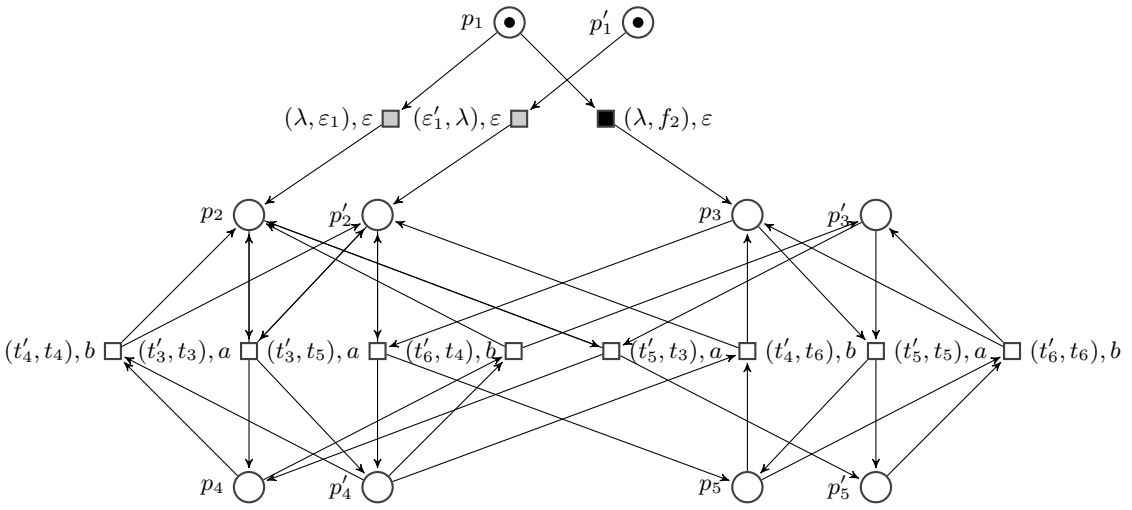

Fig. 5. VN of the LPN in Fig. 3.

Table 1. Markings in Fig. 6.

j	$\widehat{M}_j$
0	$[1\,0\,0\,0\,0 \mid 1\,0\,0\,0\,0]^\tau$
1	$[1\,0\,0\,0\,0 \mid 0\,1\,0\,0\,0]^\tau$
2	$[0\,1\,0\,0\,0 \mid 1\,0\,0\,0\,0]^\tau$
3	$[1\,0\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
4	$[0\,1\,0\,0\,0 \mid 0\,1\,0\,0\,0]^\tau$
5	$[0\,1\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
6	$[0\,1\,0\,\omega\,0 \mid 0\,1\,0\,\omega\,0]^\tau$
7	$[0\,1\,0\,1\,0 \mid 0\,0\,1\,0\,0]^\tau$
8	$[0\,\omega\,0\,\omega\,0 \mid 0\,\omega\,0\,\omega\,0]^\tau$
9	$[0\,\omega\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
10	$[0\,\omega\,0\,1\,0 \mid 0\,0\,0\,0\,1]^\tau$

given for the on-the-fly construction and analysis of the VN and its RG/CG.

For a given LPN model, the LPN $N'_L = (N', M'_o, \Sigma', \mathcal{L}')$ associated with the T'-induced subnet of the considered LPN system $N_L = (N, M_o, \Sigma, \mathcal{L})$ is built, as presented in Section 4.

In order to distinguish from the symbols of VN in Section 4, the partial VN is denoted by $\widehat{N_L} = (\widehat{N}, \widehat{M}_o, \widehat{\Sigma}_o, \widehat{\mathcal{L}})$, where $\widehat{N} = (\widehat{P}, \widehat{T}, \widehat{Pre}, \widehat{Post})$. The set of places is $\widehat{P} = P \cup P'$ and the initial marking is

$$\widehat{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix}.$$

The transitions of the VN $\widehat{T}$ are built on the fly.

Proposition 1. *At a marking*

$$\widehat{M} = \begin{bmatrix} M' \\ M \end{bmatrix}$$

of the VN, a transition $\widehat{t}$ is enabled iff one of the following conditions is satisfied:

(i) $\widehat{t} = (\lambda, t_f)$*, the transition $t_f \in T_f$ is enabled in N at the marking M;*

(ii) $\widehat{t} = (\lambda, t_{reg})$*, the transition $t_{reg} \in T_{reg}$ is enabled in N at the marking M;*

(iii) $\widehat{t} = (t'_{reg}, \lambda)$*, the transition $t'_{reg} \in T'_{reg}$ is enabled in N' at the marking M';*

(iv) $\widehat{t} = (t'_o, t_o)$*, the transition $t'_o \in T'_o$ is enabled in N' at the marking M' and the transition $t_o \in T_o$ is enabled in N at the marking M. Meanwhile, $\mathcal{L}'(t'_o) = \mathcal{L}(t_o)$.*

Proof.
(If) Given a transition $\widehat{t} = (t', t)$ (one of t' and t can be λ). According to the construction of transitions in the VN (Cabasino *et al.*, 2012),

$$\widehat{Pre}(\cdot, \widehat{t}) = \begin{bmatrix} Pre'(\cdot, t') \\ Pre(\cdot, t) \end{bmatrix}.$$

For the condition (i), $\widehat{t} = (\lambda, t_f)$, $t_f \in T_f$ and t_f is enabled in N at the marking M i.e., $M \geq Pre(\cdot, t_f)$. Therefore,

$$\begin{bmatrix} M' \\ M \end{bmatrix} \geq \begin{bmatrix} Pre'(\cdot, \lambda) \\ Pre(\cdot, t_f) \end{bmatrix}$$

(where $Pre'(\cdot, \lambda) = \overrightarrow{0}$), i.e., $\widehat{M} \geq \widehat{Pre}(\cdot, \widehat{t})$. The transition $\widehat{t} = (\lambda, t_f)$ is enabled at $\widehat{M}$. The conditions (ii) and (iii) can be proved in the same way.

For the condition (iv), $\widehat{t} = (t'_o, t_o)$, the transition $t'_o \in T'_o$ is enabled in N' at the marking M' and the transition $t_o \in T_o$ is enabled in N at the marking M, i.e., $M' \geq Pre'(\cdot, t'_0)$ and $M \geq Pre(\cdot, t_0)$. Therefore,

$$\begin{bmatrix} M' \\ M \end{bmatrix} \geq \begin{bmatrix} Pre'(\cdot, t'_o) \\ Pre(\cdot, t_o) \end{bmatrix},$$

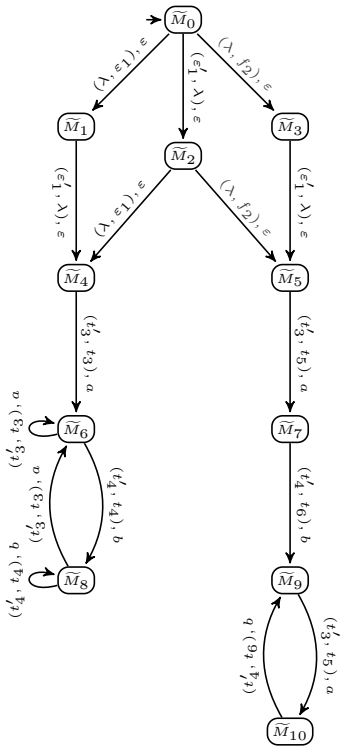

Fig. 6. CG of the VN in Fig. 5.

i.e., $\widehat{M} \geq \widehat{Pre}(\cdot, \widehat{t})$. The transition $\widehat{t} = (t'_o, t_o)$ is enabled at $\widehat{M}$.

(Only if) Assuming that, in VN, a transition $\widehat{t} = (t', t)$ (one of t' and t can be λ) is enabled. Hence, $\widehat{M} \geq \widehat{Pre}(\cdot, \widehat{t})$ i.e.,

$$\begin{bmatrix} M' \\ M \end{bmatrix} \geq \begin{bmatrix} Pre'(\cdot, t') \\ Pre(\cdot, t) \end{bmatrix}.$$

It can be deduced that $M' \geq Pre'(\cdot, t')$ and $M \geq Pre(\cdot, t)$. Therefore, the transition $t' \in T'$ is enabled in N' at the marking M' and the transition $t \in T$ is enabled in N at the marking M. ■

Algorithm 1 is given to calculate the enabled transition at a marking $\widehat{M}$ according to Proposition 1. This algorithm is called in Algorithm 3 for the on-the-fly construction of the VN and its RG/CG.

Example 5. For the LPN system N_L in Fig. 3, its T'-induced sub-LPN N'_L is built in Fig. 4. The initial marking of N_L is $M_0 = [1\ 0\ 0\ 0\ 0]^\tau$ and the initial marking of N'_L is $M'_0 = [1\ 0\ 0\ 0\ 0]^\tau$. At M_0, ε_1 and f_2 are enabled. At M'_0, only the transition ε'_1 is enabled. According to Proposition 1, at the initial marking of the VN

$$\widehat{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix},$$

Algorithm 1. Algorithm for $EnabledT$ function: find the enabled transitions at a marking $\widehat{M}$.

Input: N, N' and $\widehat{M} = \begin{bmatrix} M' \\ M \end{bmatrix}$;
Output: $(\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o)$;
Function $EnabledT(N, N', \widehat{M})$
$\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o \leftarrow \emptyset$;
$T'_{con,o}, T_{con,o} \leftarrow \emptyset$; {Local variables}
for all $t' \in T'$ **do**
  if t' is enabled at M' **then**
    if $t' \in T'_{reg}$ **then**
      $\widehat{T}'_{reg} \leftarrow \widehat{T}'_{reg} \cup \{(t', \lambda)\}$;
    else
      $T'_{con,o} \leftarrow T'_{con,o} \cup \{t'\}$;
    end if
  end if
end for
for all $t \in T$ **do**
  if t is enabled at M **then**
    if $t \in T_{reg}$ **then**
      $\widehat{T}_{reg} \leftarrow \widehat{T}_{reg} \cup \{(\lambda, t)\}$;
    else if $t \in T_f$ **then**
      $\widehat{T}_f \leftarrow \widehat{T}_f \cup \{(\lambda, t)\}$;
    else
      $T_{con,o} \leftarrow T_{con,o} \cup \{t\}$;
    end if
  end if
end for
for all $t'_o \in T'_{con,o}$ and $t_o \in T_{con,o}$ s.t. $\mathcal{L}(t'_o) = \mathcal{L}'(t_o)$ **do**
  $\widehat{T}_o \leftarrow \widehat{T}_o \uplus \{(t'_o, t_o)\}$;
end for
return $(\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o)$;

(λ, f_2) (condition (i)), (λ, ε_1) (condition (ii)) and $(\varepsilon'_1, \lambda)$ (condition (iii)) are enabled. Therefore, the output of the function $EnabledT(N, N', \widehat{M}_0)$ is $(\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o)$, where $\widehat{T}_f = \{(\lambda, f_2)\}$, $\widehat{T}_{reg} = \{(\lambda, \varepsilon_1)\}, \widehat{T}'_{reg} = \{(\varepsilon'_1, \lambda)\}$ and $\widehat{T}_o = \emptyset$. ♦

It is worth noticing that if the entire VN is built (in Fig. 5), the transitions, which are never enabled, are also built such as (t'_5, t_3), (t'_5, t_5), (t'_6, t_4) and (t'_6, t_6). However, since these transitions are never enabled, they will never be generated by Algorithm 1. Therefore, these transitions will not be built in the on-the-fly construction of the VN and it can be proposed that $\widehat{T} \subseteq \widetilde{T}$.

Algorithm 2 is developed to build on-the-fly the VN and it is called in Algorithm 3. Algorithm 3 which is called in Algorithm 4, is developed to build on-the-fly the RG/CG of the VN and gives the diagnosability verdict.

Initially, the VN contains only all the places but no transitions. The RG/CG contains one node with the initial marking of the VN. From the initial marking, Algorithm 1 is used to calculate all the enabled transitions. Since the approach is based on a depth-first search, selecting one

enabled transition and build it by using Algorithm 2. After firing the chosen transition, the new marking is calculated and built by Algorithm 3. The above process is iterated until the undiagosability condition is fulfilled or the whole RG/CG is built. Proposition 2 is provided to prove the correctness of our approach.

Proposition 2. *For an LPN system, the $MAIN$ function in Algorithm 4 terminates and its diagnosability verdict is correct.*

Proof. The on-the-fly diagnosability using a VN is presented by Algorithm 4. Algorithm 3 is called in Algorithm 4 and Algorithm 2 is called in Algorithm 3.

First, we will prove that the algorithm terminates for both bounded and unbounded LPNs. The investigation of a branch of the RG/CG is stopped, when one of the following conditions is satisfied:

1. There exists a deadlock at the new node (Line 9 of Algorithm 3);
2. A new node in RG/CG is equal to a previous one (Lines 31–42 of Algorithm 3).

In an unbounded case, the unbounded places are treated in Lines 15–20 of Algorithm 3.

In the on-the-fly construction of the RG/CG, for any branch, these two conditions will be met sooner of later. Therefore, the algorithm terminates well.

Second, when the above stop condition is satisfied, the following three cases can occur: (i) a deadlock is found; (ii) the cycle is reachable starting from a node in the set $F(VN)$; (iii) the cycle is not reachable starting from any node in the set $F(VN)$.

In Case (i), if there exists a deadlock, the investigation of this branch is stopped. However, the diagnosability of the LPN system cannot be determined. The construction needs to be continued. It restarts from its previous node, and the other branches are investigated (Lines 9 and 10 of Algorithm 3). In Case (ii), if the cycle corresponds to a firable repetitive sequence, the result that the LPN system is nondiagnosable can be obtained immediately according to Theorem 1 (Lines 39 and 40 of Algorithm 3). Otherwise, it needs to investigate other branches. In Case (iii), the construction need to be continued.

If the LPN system is diagnosable or a cycle is reachable starting from a node in the set $F(VN)$ in the end of the construction, all the *semilive* and *live* transitions (Algorithm 2) of the VN are built and the whole RG/CG is built.

Since all the possible cases are considered, it can be guaranteed that Algorithm 4 terminates well and its diagnosability verdict is correct. ■

Proposition 2 guarantees the correctness of this approach. When a cycle corresponding to a firable

Algorithm 2. Algorithm for $BuildVNTransition$ function.

1: Input: $\widehat{N_L} = (\widehat{N}, \widehat{M}_o, \widehat{\Sigma}_o, \widehat{\mathcal{L}})$, where $\widehat{N} = (\widehat{P}, \widehat{T}, \widehat{Pre}, \widehat{Post})$; a transition $\widehat{t} \in \widehat{T}_{con}$, where $\widehat{T}_{con} = \widehat{T}_{con,f} \cup \widehat{T}_{con,reg} \cup \widehat{T}'_{con,reg} \cup \widehat{T}_{con,o}$, where $(\widehat{T}_{con,f}, \widehat{T}_{con,reg}, \widehat{T}'_{con,reg}, \widehat{T}_{con,o})$ is the output of $EnabledT$ function;
2: Output: $\widehat{N'_L} = (\widehat{N}', \widehat{M}_o, \widehat{\Sigma}_o, \widehat{\mathcal{L}}')$, where $\widehat{N}' = (\widehat{P}', \widehat{T}', \widehat{Pre}', \widehat{Post})'$;
3: **Function** $BuildVNTransition(\widehat{N_L}, \widehat{t})$
4: **if** $\widehat{t} \in \widehat{T}$ **then**
5: **return** $\widehat{N_L}$;
6: **else**
7: **if** $(\widehat{t} = (\lambda, t_f) \in \widehat{T}_{con,f})$ **then**
8: **for all** $p \in P'$ **do**
9: $\widehat{Pre}'(p, \widehat{t}) = \widehat{Post}'(p, \widehat{t}) = 0$;
10: **end for**
11: **for all** $p \in P$ **do**
12: $\widehat{Pre}'(p, \widehat{t}) = Pre(p, t_f), \widehat{Post}'(p, \widehat{t}) = Post(p, t_f)$;
13: **end for**
14: $\widehat{\mathcal{L}}(\widehat{t}) = \varepsilon$;
15: **else if** $(\widehat{t} = (\lambda, t_{reg}) \in \widehat{T}_{con,reg})$ **then**
16: **for all** $p \in P'$ **do**
17: $\widehat{Pre}'(p, \widehat{t}) = \widehat{Post}'(p, \widehat{t}) = 0$;
18: **end for**
19: **for all** $p \in P$ **do**
20: $\widehat{Pre}'(p, \widehat{t}) = Pre(p, t_{reg}), \widehat{Post}'(p, \widehat{t}) = Post(p, t_{reg})$;
21: **end for**
22: $\widehat{\mathcal{L}}(\widehat{t}) = \varepsilon$;
23: **else if** $(\widehat{t} = (t'_{reg}, \lambda) \in \widehat{T}'_{con,reg})$ **then**
24: **for all** $p \in P'$ **do**
25: $\widehat{Pre}'(p, \widehat{t}) = Pre'(p, t'_{reg})$, $\widehat{Post}'(p, \widehat{t}) = Post'(p, t'_{reg})$;
26: **end for**
27: **for all** $p \in P$ **do**
28: $\widehat{Pre}'(p, \widehat{t}) = \widehat{Post}'(p, \widehat{t}) = 0$;
29: **end for**
30: $\widehat{\mathcal{L}}(\widehat{t}) = \varepsilon$;
31: **else**
32: **for all** $p \in P'$ **do**
33: $\widehat{Pre}'(p, \widehat{t}) = Pre'(p, t'_o), \widehat{Post}'(p, \widehat{t}) = Post'(p, t'_o)$;
34: **end for**
35: **for all** $p \in P$ **do**
36: $\widehat{Pre}'(p, \widehat{t}) = Pre(p, t_o), \widehat{Post}'(p, \widehat{t}) = Post(p, t_o)$;
37: **end for**
38: $\widehat{\mathcal{L}}(\widehat{t}) = (l, l)$; $\{\mathcal{L}(t_o) = \mathcal{L}'(t'_o) = l\}$
39: **end if**
40: $\widehat{T}' \leftarrow \widehat{T} \cup \widehat{t}$;
41: Update $\widehat{Pre}'$ by adding the column $\widehat{Pre}'(\cdot, \widehat{t})$ to $\widehat{Pre}$;
42: Update $\widehat{Post}'$ by adding the column $\widehat{Post}'(\cdot, \widehat{t})$ to $\widehat{Post}$;
43: **return** $\widehat{N_L}'$;
44: **end if**

Algorithm 3. Algorithm for $DIAG$ function.

Input: $(N_L, N'_L, \widehat{N_L}, \widehat{M}, n)$;
Output: $(\widehat{N'_L}, \widehat{M'}, n')$;
Function $DIAG(N_L, N'_L, \widehat{N_L}, \widehat{M}, n)$
$\widehat{T}_{con,f}, \widehat{T}_{con,reg}, \widehat{T}'_{con,reg}, \widehat{T}_{con,o} \leftarrow \emptyset$;
$\widehat{N''_L}, \widehat{M''} \leftarrow \emptyset$, $n'' \leftarrow 1$; {n'' is a local Boolean variable}
{$\widehat{\mathcal{N}}, \widehat{\mathcal{A}}$ and $F(VN)$ are global, cf.in Algorithm 4.}
$(\widehat{T}_{con,f}, \widehat{T}_{con,reg}, \widehat{T}'_{con,reg}, \widehat{T}_{con,o}) \leftarrow EnabledT(N_L, N'_L, \widehat{M})$;
$\widehat{T}_{con} \leftarrow \widehat{T}_{con,f} \cup \widehat{T}_{con,reg} \cup \widehat{T}'_{con,reg} \cup \widehat{T}_{con,o}$;
$n' \leftarrow 1$; {Local Boolean variable}
if $\widehat{T}_{con} = \emptyset$ **then**
  return $(\widehat{N_L}, \widehat{M}, n')$;
else if $\widehat{T}_{con} \neq \emptyset$ **then**
  for all $\widehat{t} \in \widehat{T}_{con}$ **do**
    $\widehat{N_L}' \leftarrow BuildVNTransition(\widehat{N_L}, \widehat{t})$; {for $BuildVNTransition$ function, see Algorithm 2}
    $\widehat{M'} \leftarrow \widehat{M} + \widehat{C}'(\cdot, \widehat{t})$; {$\widehat{C}'$ is the incidence matrix of $\widehat{N_L}'$}
    Let $\widehat{M}^\alpha$ be the first node met on the backward path from $\widehat{M'}$ to the initial marking $\widehat{M}_0$ s.t. $\widehat{M}^\alpha < \widehat{M'}$; {As it was presented in Karp and Miller (1969)}
    if $\widehat{M}^\alpha$ exists **then**
      for all $p \in \widehat{P}$ s.t. $\widehat{M}^\alpha(p) < \widehat{M'}(p)$ **do**
        $\widehat{M'}(p) = \omega$;
      end for{Lines 39–42 are only for an unbounded LPN.}
    end if
    if $(\forall \widehat{M}^* \in \widehat{\mathcal{N}}, \widehat{M}^* \neq \widehat{M'})$ ($\widehat{M}^*$ is the node already built in $\widehat{\mathcal{N}}$) **then**
      if $(\widehat{M} \in F(VN)) \vee (\widehat{t} \in \widehat{T}_{con,f})$ **then**
        $F(VN) \leftarrow F(VN) \cup \widehat{M'}$;
      end if
      $\widehat{\mathcal{N}} \leftarrow \widehat{\mathcal{N}} \cup \widehat{M'}$;
      $\widehat{\mathcal{A}} \leftarrow \widehat{\mathcal{A}} \cup (\widehat{M}, \widehat{\mathcal{L}}(\widehat{t}), \widehat{M'})$;
      $(\widehat{N''_L}, \widehat{M''}, n'') \leftarrow DIAG(N_L, N'_L, \widehat{N'_L}, \widehat{M'}, n')$;
      if $n'' = 0$ **then**
        return $(\widehat{N''_L}, \widehat{M''}, n'')$;
      end if
    else if $(\exists \widehat{M}^* \in \widehat{\mathcal{N}}, \widehat{M}^* = \widehat{M'})$ **then**
      $\widehat{\mathcal{A}} \leftarrow \widehat{\mathcal{A}} \cup (\widehat{M}, \widehat{\mathcal{L}}(\widehat{t}), \widehat{M}^*)$;
      if $((\widehat{M} \in F(VN)) \vee (\widehat{t} \in \widehat{T}_{con,f})) \wedge (\widehat{M}^* \notin F(VN))$ **then**
        $F(VN) \leftarrow F(VN) \cup \widehat{M}^*$
        for all $\widehat{M}^o$ can be reached from $\widehat{M}^*$ **do**
          $F(VN) \leftarrow F(VN) \cup \widehat{M}^o$;
        end for
      end if
      if There exists a cycle associated with a firable repetitive sequence from a node in $F(VN)$ **then**
        return $(\widehat{N'_L}, \widehat{M'}, n' \leftarrow 0)$; {$path\_exists$ in the library $digraph$ (Rushton (2012)).}
      end if
    end if
  end for
  return $(\widehat{N'_L}, \widehat{M'}, n')$; {$n' = 1 \Rightarrow N_L$ is diagnosable.}
end if

Algorithm 4. Algorithm for $MAIN$ function.

Input: N_L;
Output: Diagnosability analysis of the N_L;
Function $MAIN(N_L)$
$\widehat{\mathcal{N}} \leftarrow \emptyset$; {$\widehat{\mathcal{N}}$ is the set of RG/CG nodes;}
$\widehat{\mathcal{A}} \leftarrow \emptyset$; {$\widehat{\mathcal{A}}$ is the set of RG/CG arcs;}
$F(VN) \leftarrow \emptyset$; {$F(VN)$ is the set of fault nodes;}
{$\widehat{\mathcal{N}}, \widehat{\mathcal{A}}$ and $F(VN)$ are global variables;}
$\widehat{N'_L}, \widehat{M'} \leftarrow \emptyset$, $n' \leftarrow 1$; {n' is a local boolean variable}
$n \leftarrow 1$; {n is the tag to indicate the diagnosability of the system;}
build $N'_L =< N', M'_0, \Sigma', \mathcal{L}' >$;
initialize $\widehat{N_L}$, where $\widehat{P} = P' \cup P$, $\widehat{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix}$;
$\widehat{\mathcal{N}} \leftarrow \widehat{\mathcal{N}} \cup \widehat{M}_0$;
$(\widehat{N'_L}, \widehat{M'}, n') \leftarrow DIAG(N_L, N'_L, \widehat{N_L}, \widehat{M}_0, n)$;
if $n' = 1$ **then**
  assert (The LPN system N_L is diagnosable);
else if $n' = 0$ **then**
  assert (The LPN system N_L is non-diagnosable);
end if

repetitive sequence is found and is reachable starting from any node in the set $F(VN)$, the LPN is determined immediately as nondiagnosable.

It is worth noticing that our approach does not define priorities in the investigation of branches. In the work of Li *et al*. (2015a), a kind of heuristic is proposed for the on-the-fly diagnosability analysis by using the T-invariant to define the priorities in the investigation of the branches. However, this heuristic is not available for the approach in this paper. Since we build on-the-fly the VN and its RG/CG, we cannot get the T-invariants of the VN from the beginning. This problem will not be addressed in this paper and we define the priorities as follows.

In the paper, the transitions in $\widehat{T}_f$ have higher priorities, because, according to Theorem 1, the diagnosability analysis is based on finding the cycle after firing a fault transition. For the transitions in $\widehat{T}_{reg}$, $\widehat{T}'_{reg}$ and $\widehat{T}_o$, the priorities cannot be defined reasonably. To analyze the example in this paper, the priorities between the branches to be investigated are heuristically defined as follows:

$$\widehat{T}_f > \widehat{T}_{reg} > \widehat{T}'_{reg} > \widehat{T}_o$$

The priorities of the transitions in the same set are defined by the numerical order of the transitions. For example, for the transition in $\widehat{T}_o$, $(t'_3, t_3) > (t'_3, t_5) > (t'_5, t_3) > (t'_5, t_5)$.

Example 6. Let us consider again the LPN $N_L = (N, M_o, \Sigma, \mathcal{L})$ in Fig. 3. In this paper, we study an unbounded LPN system, because a bounded LPN is a special case of an unbounded LPN. The T'-induced sub-LPN $N'_L = (N', M'_o, \Sigma', \mathcal{L}')$ is built in Fig. 4. The places of VN are built as $\widehat{P} = P \cup P'$ and the initial

marking is

$$\widehat{M}_0 = \begin{bmatrix} M_0' \\ M_0 \end{bmatrix}.$$

At the initial marking $\widehat{M}_0 = [1\ 0\ 0\ 0\ 0 \mid 1\ 0\ 0\ 0\ 0]^\tau$, $(\lambda, \varepsilon_1), \varepsilon$, $(\varepsilon_1', \lambda), \varepsilon$ as well as $(\lambda, f_2), \varepsilon$ are enabled (Algorithm 1). The transition $(\lambda, f_2), \varepsilon$ is built in the VN (Lines 7–14, Algorithm 2) and is fired because of its higher priority. The next node $\widehat{M}_1 = [1\ 0\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$ is generated and built in the CG (Lines 25–26, Algorithm 3). The set of fault nodes $F(VN)$ is updated (Line 36, Algorithm 3).

At $\widehat{M}_1$ (Line 27, Algorithm 3), only $(\varepsilon_1', \lambda), \varepsilon$ is enabled. This transition is built and by firing this transition, the next node in CG is $\widehat{M}_2 = [0\ 1\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$. The set of fault nodes $F(VN)$ is updated.

At $\widehat{M}_2$, only $(t_3', t_5), a$ is enabled according to Condition (iv) in Proposition 1. This transition is built and the next node in CG is $\widehat{M}_3 = [0\ 1\ 0\ 1\ 0 \mid 0\ 0\ 0\ 0\ 1]^\tau$. The set of fault nodes $F(VN)$ is updated.

At $\widehat{M}_3$, only $(t_4', t_6), b$ is enabled. This transition is built. By firing this transition, the computed marking is $\widehat{M}_4' = [0\ 2\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$. Since on the backward path, the marking $\widehat{M}_2$ can be found such that $\widehat{M}_4' \gtrsim \widehat{M}_2$. Therefore, the number of the token in place p_2' is set as ω (Lines 15–19, Algorithm 3). The next node in the CG is $\widehat{M}_4 = [0\ \omega\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$ and it is put into $F(VN)$.

At $\widehat{M}_4$, only $(t_3', t_5), a$ is enabled and $\widehat{M}_5 = [0\ \omega\ 0\ 1\ 0 \mid 0\ 0\ 0\ 0\ 1]^\tau$ is built in CG. The node is put into $F(VN)$.

At $\widehat{M}_5$, only $(t_4', t_6), b$ is enabled. The computed node is $[0\ \omega\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$, which is equal to $\widehat{M}_4$ (Lines 31–40, Algorithm 3). Therefore, a cycle from a node in $F(VN)$ is found. Since $\widehat{C}' \cdot \overleftarrow{y} = \overleftarrow{0}$ (where $\widehat{C}'$ is the updated incidence matrix; $\overleftarrow{y}$ is the firing vector that contains (t_3', t_5) and (t_4', t_6), i.e., $\overleftarrow{y}((t_3', t_5)) = \overleftarrow{y}((t_4', t_6)) = 1$ and for all other transitions $\widehat{t} \in \widehat{T}$, $\overleftarrow{y}(\widehat{t}) = 0$), the cycle is associated with a stationary repetitive sequence. Therefore, the construction of VN and its CG is stopped and the immediate result is that the unbounded LPN system is nondiagnosable. ♦

Table 2. Markings in Fig. 8.

j	$\widehat{M}_j$
0	$[1\ 0\ 0\ 0\ 0 \mid 1\ 0\ 0\ 0\ 0]^\tau$
1	$[1\ 0\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$
2	$[0\ 1\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$
3	$[0\ 1\ 0\ 1\ 0 \mid 0\ 0\ 0\ 0\ 1]^\tau$
4	$[0\ \omega\ 0\ 0\ 0 \mid 0\ 0\ 1\ 0\ 0]^\tau$
5	$[0\ \omega\ 0\ 1\ 0 \mid 0\ 0\ 0\ 0\ 1]^\tau$

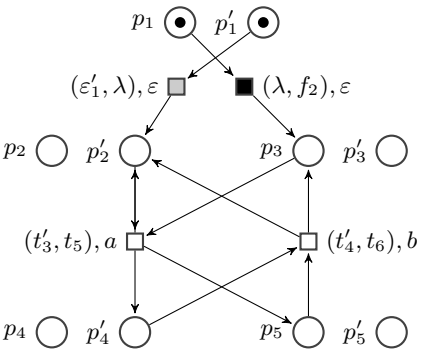

Fig. 7. On-the-fly construction of the VN of the LPN in Fig. 3.

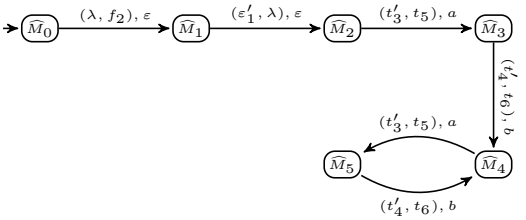

Fig. 8. On-the-fly construction of a CG.

6. Comparative analysis

6.1. Memory cost analysis. A comparison of the VN approach by Cabasino *et al*. (2012) and our approach is given for the diagnosability analysis of the LPN system in Fig. 3. For the diagnosability analysis of this LPN system, our approach generates fewer transitions in the on-the-fly construction of the VN and fewer nodes in the CG. The result that the system is non-diagnosable is immediately given, when a cycle associated with a firable repetitive sequence is found which is reachable starting from a node in the set $F(VN)$.

Table 3. Comparison of the VN approach in and the on-the-fly diagnosability analysis using the VN.

Approach	$\lvert\widetilde{P}\rvert/\lvert\widehat{P}\rvert$	$\lvert\widetilde{T}\rvert/\lvert\widehat{T}\rvert$	Number of nodes in CG
VN approach (Cabasino *et al.*, 2012)	10	11	11
On-the-fly diagnosability analysis using VN	10	4	6

If the removal of the fault transitions makes some transitions blocked in a T-induced sub-LPN, fewer transitions will be built in our approach compared with the VN approach by Cabasino *et al*. (2012), because in Section 5 we argued that $\widehat{T} \subseteq \widetilde{T}$. Even for a diagnosable system, the whole RG/CG needs to be built, but only the

semi live and *live* transitions are built in the VN.

The on-the-fly diagnosability analysis using the VN avoids building, *a priori*, the whole VN and its RG/CG. The VN structure and its RG/CG are built on-the-fly and in parallel with stop conditions. When a cycle corresponding to a firable repetitive sequence is found which is reachable by firing a fault, there is no need to continue the construction and the LPN system is determined as non diagnosable. Our approach is favorable for industrial use, because it allows iterating the diagnosability analysis without building each time the whole VN and its RG/CG.

6.2. Computational complexity analysis. In this subsection, the computational complexity of our approach is analyzed for both bounded and unbounded cases.

6.2.1. Analysis for bounded LPNs. In this section, the computational complexity of the VN approach is discussed at first, because the computational complexity is not analyzed in detail by Cabasino *et al*. (2012). After that, the computational complexity of our approach is analyzed.

Let us denote by $n_1 = |P|$ the number of the places of the initial LPN model N_L, $n_2 = |T|$ the number of the transitions of N_L and $n_3 = |\mathcal{N}_{RG}|$ the number of the nodes in the RG of N_L.

Proposition 3. *The computational complexity of the construction of the VN of the LPN model N_L is $\mathcal{O}(n_1 n_2^2)$.*

Proof. First it is evident that the number of the places in the VN is $|\widetilde{P}| = |P'| + |P| = 2n_1$.

Second, the set of transitions of N_L is $T = T_o \uplus T_{reg} \uplus T_f$. As presented in Section 3.2, since T_o, T_{reg} and T_f are disjoint, $|T_o| + |T_{reg}| + |T_f| = |T| = n_2$. The set of the transitions of the VN is $\widetilde{T} = \widetilde{T}_o \cup (T'_{reg} \times \{\lambda\}) \cup (\{\lambda\} \times T_{reg}) \cup (\{\lambda\} \times T_f)$. It is intuitive that $|T'_{reg} \times \{\lambda\}| = |\{\lambda\} \times T_{reg}| = |T_{reg}|$ and $|\{\lambda\} \times T_f| = |T_f|$. The number of the observable transitions in VN is $|\widetilde{T}_o| \leq |T_o| \cdot |T'_o| = |T_o|^2$, and the equality is true in the case when all the observable transitions have the same label. Therefore, the number of transitions in the VN is

$$\begin{aligned} |\widetilde{T}| &= |\widetilde{T}_o| + |T'_{reg} \times \{\lambda\}| \\ &\quad + |\{\lambda\} \times T_{reg}| + |\{\lambda\} \times T_f| \\ &= |T_o|^2 + 2 \cdot |T_{reg}| + |T_f| \\ &\leq 2 \cdot |T|^2 = 2 \cdot n_2^2. \end{aligned}$$

■

Remark 1. The coefficient 2 is added to cover the case when T contains only one regular unobservable transition. Besides, the number of arcs built in the VN between the places and transitions is at most $2 \cdot |\widetilde{T}| \cdot |\widetilde{P}| \leqslant 2 \cdot (2 \cdot n_2^2) \cdot (2 \cdot n_1)$.

Above all, the computational complexity of the construction of the VN of N_L is $\mathcal{O}(n_1 n_2^2)$.

Proposition 4. *The computational complexity of the construction of the RG of the VN is $\mathcal{O}(n_3^2 n_2^2)$.*

Proof. Assuming that the $\mathcal{N}_{RG}^{VN}$ is the set of the nodes in RG of the VN and the $\mathcal{A}_{RG}^{VN}$ is the set of the arcs in the RG of the VN. Since a marking in the VN is composed of

$$\widetilde{M} = \begin{bmatrix} M' \\ M \end{bmatrix},$$

the number of the nodes in $\mathcal{N}_{RG}^{VN}$ is at most n_3^2. The numbers of arcs in the RG of the VN is $|\mathcal{A}_{RG}^{VN}| \leq |\mathcal{N}_{RG}^{VN}| \cdot |\widetilde{T}| \leq (n_3^2) \cdot (2n_2^2)$. Above all, the computational complexity of the construction of the RG of the VN is $\mathcal{O}(n_3^2 n_2^2)$. ■

Proposition 5. *The existence of a cycle that is reachable starting from a node in the set $F(VN)$ can be decided in $\mathcal{O}(n_3^2 n_2^2)$.*

Proof. Deciding if there exists a cycle that is reachable starting from a node in the set $F(VN)$ takes $\mathcal{O}(|F(VN)| + |\mathcal{A}_{F(VN)}|)$ as proposed by Cormen *et al*. (1990), where $\mathcal{A}_{F(VN)}$ is the set of arcs between the nodes in $F(VN)$. In the worst case (all nodes belong to $F(VN)$), it can be determined that $|F(VN)| \leq |\mathcal{N}_{RG}^{VN}| \leq n_3^2$ and $|\mathcal{A}_{F(VN)}| \leq |\mathcal{A}_{RG}^{VN}| \leq (n_3^2) \cdot (2 \cdot n_2^2)$ according to the proof of Proposition 3. Therefore, the existence of a cycle that is reachable starting from a node in the set $F(VN)$ can be decided in $\mathcal{O}(n_3^2 n_2^2)$. ■

From Theorem 1 and Proposition 3–5, the following result can be deduced:

Theorem 2. *The diagnosability of N_L with respect to a given fault class using the VN approach of (Cabasino et al., 2012) can be decided in $\mathcal{O}(n_1 n_2^2 + n_3^2 n_2^2)$.*

The result of the computational complexity analysis of the VN approach is the same with the analysis by Cabasino *et al*. (2012). The first term of the complexity $(n_1 n_2^2)$ is the complexity of the construction of the VN, which is the same the on presented by Cabasino *et al*. (2012). The second term $(n_3^2 n_2^2)$ represents the complexity of two steps: the construction of the RG and the verification of the existence of cycles. Cabasino *et al*. (2012) argued that the total complexity of these two steps is linear in the sum of the numbers of states and transitions of the RG of the VN, i.e., the complexity is $\mathcal{O}(|\mathcal{N}_{RG}^{VN}| + |\mathcal{A}_{RG}^{VN}|)$. The result is the same as our second term, because $|\mathcal{N}_{RG}^{VN}| + |\mathcal{A}_{RG}^{VN}| \leqslant n_3^2 + (n_3^2) \cdot (2 \cdot n_2^2)$ as presented in the proof of Proposition 4. Therefore, the complexity of the VN approach is $\mathcal{O}(n_1 n_2^2 + n_3^2 n_2^2)$. Our results define more precisely the complexity of the VN approach of Cabasino *et al*. (2012). It is based on the size

of the initial model which makes it comparable to other approaches, such as the diagnoser approach by Sampath *et al*. (1995).

Theorem 3. *The complexity of the diagnosability analysis of N_L using Algorithm 4 with respect to a given fault class is $\mathcal{O}(n_1 n_2^2 n_3^2 + n_3^4 n_2^4)$.*

Proof. First, the complexity of Algorithm 1 is $\mathcal{O}(|\widehat{T}|) = \mathcal{O}(n_2^2)$ (Line 29 of Algorithm 1). Algorithm 1 is called in Algorithm 3 but not in any iteration. The Algorithm 3 contains one main iteration that is indicated by Lines 13–42 of Algorithm 3 (in the worst case, the number of iterations is $\mathcal{O}(|\widehat{T}|)$). The complexity of building a transition of the VN is $\mathcal{O}(|P|)$ (Algorithm 2). To verify the condition in Line 21 of Algorithm 3, the complexity is $\mathcal{O}(|\widehat{\mathcal{N}}_{RG}|)$. The verification of a cycle from a node in $F(VN)$ (Line 39 of Algorithm 3) has a complexity of $\mathcal{O}(n_3^2 n_2^2)$ (Proposition 6). The other steps in the main iteration can be neglected, because of their lower complexity. In the worst case, Algorithm 3 is called $|\widehat{\mathcal{N}}_{RG}|$ times (Line 27). Therefore, the entire complexity is $\mathcal{O}(|\widehat{\mathcal{N}}_{RG}| \cdot (|\widehat{T}| + |\widehat{T}| \cdot (|P| + |\widehat{\mathcal{N}}_{RG}| + n_3^2 n_2^2))) = \mathcal{O}(n_3^2 \cdot (n_2^2 + n_2^2 \cdot (n_1 + n_3^2 + n_3^2 n_2^2))) = \mathcal{O}(n_1 n_2^2 n_3^2 + n_3^4 n_2^4)$. ■

6.2.2. Analysis for unbounded LPN. Cabasino *et al*. (2012) argued that the complexity of the VN approach for unbounded LPNs cannot be given because the complexity of the construction of the CG is still an open issue. In the worst case, the on-the-fly diagnosability analysis using the VN needs to build the whole VN and its CG. Moreover, the on-the-fly diagnosability analysis using the VN builds the transitions in the VN and the nodes in its CG and analyze at the same time when a cycle in the CG is found. Assume that $|F(VN)|$ is the number of nodes in $F(VN)$, $\mathcal{A}_{F(VN)}$ the number of arc between these nodes and the complexity of the VN approach for unbounded LPN is $\mathcal{C}_{VN}$. Deciding if there is a cycle that is reachable starting from a node in the set $F(VN)$ takes $\mathcal{O}(|F(VN)| + |\mathcal{A}_{F(VN)}|)$. In the worst case (i.e., all the nodes belong to $F(VN)$), $\mathcal{O}(|F(VN)| + |\mathcal{A}_{F(VN)}|) \leq \mathcal{C}_{VN}$. Therefore, the complexity of the on-the-fly diagnosability analysis using VN is $(n_{cycle}+1) \cdot \mathcal{C}_{VN}$, where n_{cycle} is the number of cycles in CG, which do not correspond to repetitive sequences.

7. Conclusion

In this paper, a new approach to diagnosability analysis of DESs modeled by bounded or unbounded LPNs has been proposed. The VN approach by Cabasino *et al*. (2012) is improved by applying the on-the-fly diagnosability analysis. The VN and its RG/CG are built on the fly and in parallel with stop conditions. As soon as the undiagnosability condition is fulfilled, the construction and the analysis are stopped and the result that the system is nondiagnosable is immediately given. This approach allows iterating the diagnosability analysis without building each time the whole state space, until the system becomes diagnosable. The computational complexity of our approach is analyzed, which is slightly increased by using the on-the-fly analysis (Schwoon and Esparza, 2005). Our approach achieves a compromise between the computation efficiency and limitation resulting from combinatorial explosion.

In future research, some heuristics need to be defined in terms of priority between the branches to be investigated while building the transitions of the VN and the nodes of RG/CG, to make this approach more efficient.

Acknowledgment

This work was carried out thanks to the ELSAT2020 program co-financed by the European Union through the European Regional Development Fund, by the French State and the Hauts-de-France Region.

References

Basile, F., Chiacchio, P. and De Tommasi, G. (2012). On K-diagnosability of Petri nets via integer linear programming, *Automatica* **48**(9): 2047–2058.

Basile, F., Chiacchio, P. and Tommasi, G. (2009). An efficient approach for online diagnosis of discrete event systems, *IEEE Transactions on Automatic Control* **54**(4): 748–759.

Boussif, A., Ghazel, M. and Klai, K. (2015). Combining enumerative and symbolic techniques for diagnosis of discrete-event systems, *9th International Workshop on Evaluation of Computer and Communication Systems, Bucharest, Romania*, pp.1–11.

Cabasino, M., Giua, A., Lafortune, S. and Seatzu, C. (2012). A new approach for diagnosability analysis of Petri nets using verifier nets, *IEEE Transactions Automatic Control* **57**(12): 3104–3117.

Cabasino, M., Giua, A. and Seatzu, C. (2014). Diagnosis of discrete event systems using labeled Petri nets, *IEEE Transactions on Automation Science and Engineering* **11**(1): 144–153.

Cabral, F., Moreira, M., Diene, O. and Basilio, J. (2015). A Petri net diagnoser for discrete event systems modeled by finite state automata, *IEEE Transactions on Automatic Control* **60**(1): 59–71.

Cormen, T., Leiserson, C. and Rivest, R. (1990). *Introduction of Algorithms*, MIT Press, Cambridge, MA.

Jiang, S., Huang, Z., Chandra, V. and Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete event systems, *IEEE Transactions on Automatic Control* **46**(8): 1318–1321.

Karp, R. and Miller, R. (1969). Parallel program schemata: A mathematical model for parallel computation, *Journal of Computer and System Sciences* **3**(2): 147–195.

Lefebvre, D. and Delherm, C. (2007). Diagnosis of DES with Petri net models, *IEEE Transactions on Automation Science and Engineering* **4**(1): 114–118.

Li, B., Khlif-Bouassida, M. and Toguyéni, A. (2015a). On-the-fly Diagnosability analysis of labeled Petri nets using T-invariants, *5th International Workshop on Dependable Control of Discrete Systems, DCDS'2015, Cancun, Mexico*, pp. 64–70.

Li, B., Khlif-Bouassida, M. and Toguyéni, A. (2016). On-the-fly diagnosability analysis of LPN using verifier nets, *3rd International Conference on Control and Fault-Tolerant Systems, SYSTOL'16, Nice, France*, pp. 305–312.

Li, B., Liu, B. and Toguyéni, A. (2015b). On-the-fly diagnosability analysis of labeled Petri nets using minimal explanations, *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'2015, Paris, France*, pp. 326–331.

Liu, B., Ghazel, M. and Toguyéni, A. (2014). Toward an efficient approach for diagnosability analysis of DES modeled by labeled Petri nets, *13th European Control Conference, ECC'2014, Strasbourg, France*, pp. 1293–1298.

Moreira, M., Jesus, T. and Basilio, J. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems, *IEEE Transactions on Automatic Control* **56**(7): 1679–1684.

Rushton, A. (2012). *STLplus C++ Library Collection,* `stlplus.sourceforge.net`.

Sampath, M., Sengupta, R. and Lafortune, S. (1995). Diagnosability of discrete-event systems, *IEEE Transactions an Automatic Control* **40**(9): 1555–1575.

Schwoon, S. and Esparza, J. (2005). A note on on-the-fly verification algorithms, *11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Edinburgh, UK,* pp. 174–190.

Yoo, T. and Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems, *IEEE Transactions on Automatic Control* **47**(9): 1491–1495.

Ben Li was born in An Yang, China, in 1988. He received the BEng and MSc degrees in control science and technology from Beihang University, China, in 2014. He is currently pursuing the PhD degree in automation science at École Centrale de Lille, France. His current interests are fault diagnosis of discrete event systems, formal verification methods and the theory of Petri nets.

Manel Khlif-Bouassida was born in Ariana, Tunisia. She received her MSc(Res) degree in computer science in 2006 from Université de Versailles Saint Quentin en Yvelines, France. She obtained her PhD in computer engineering from Université de Technologie de Compiègne, France, in 2010. In 2010, she received a postdoctoral fellowship from the same university. She has been an associate professor at École Centrale de Lille since 2011. She is a researcher at the CRIStAL lab (Research Center in Computer Science, Signal and Automatic Control of Lille, France). Her research interests are in the field of modeling and verification of discrete event systems, including diagnosis and diagnosability analysis. One of the main applications of her research is transportation systems.

Armand Toguyéni was born in Dakar, Senegal, in 1964. He obtained the BEng degree in 1988 from Institut Industriel du Nord (French Grande Ecole) and the MSc degree in computer sciences in the same year. He obtained his PhD in automatic control for manufacturing and discrete events systems in 1992 and his accreditation to supervise research (HdR) in 2001. He is a full professor of industrial computer sciences at École Centrale de Lille, France. He performs his research at CRIStAL, a laboratory associated with CNRS. He is the leader of the MOSES team of CRIStAL. His research interests are the quality of service of discrete-event systems. He works in particular on modeling, fault-tolerant control and diagnosis of such systems. His main areas of application are railway systems, manufacturing systems and computer networks.

Received: 15 March 2017
Revised: 9 November 2017
Accepted: 13 January 2018