A survey on efficient diagnosability tests for automata and bounded Petri nets

A survey on efficient diagnosability tests for automata and bounded Petri nets

Mona Noori Hosseini, Bengt Lennartson
Chalmers University of Technology,
SE-412 96, Gothenburg, Sweden
{noori,bengt.lennartson}@chalmers.se

Maria Paola Cabasino, Carla Seatzu
University of Cagliari,
Piazza DArmi, 09123 Cagliari, Italy
{cabasino, seatzu}@diee.unica.it

Abstract

This paper presents a survey and evaluation of the efficiency of polynomial diagnosability algorithms for systems modeled by Petri nets and automata. A modified verification algorithm that reduces the state space by exploiting symmetry and abstracting unobservable transitions is also proposed. We show the importance of minimal explanations on the performance of diagnosability verifiers. Different verifiers are compared in terms of state space and elapsed time. It is shown that the minimal explanation notion involved in the modified basis reachability graph, a graph presented by Cabasino et al. [3] for diagnosability analysis of Petri nets, has great impact also on automata-based diagnosability methods. The evaluation often shows improved computation times of a factor 1000 or more when the concept of minimal explanation is included in the computation.

1. Introduction

The ability to deduce about previously occurred failures in a system within a limited number of observations is called diagnosability [10]. In the past decades there have been many works in this field of research for both Petri nets (PNs) and automata models.

In [10], where the diagnosability notion, and necessary and sufficient conditions for diagnosability are introduced, a diagnoser is constructed for diagnosability verification which has exponential complexity. Later many polynomial automata-based diagnosability approaches have been proposed. A polynomial diagnosability test that abstracts away all unobservable transitions is proposed in [6]. In [11], a different synchronization rule is defined for unobservable and failure transitions, which has less worst case complexity with respect to (wrt) [6]. In the two mentioned approaches it is assumed that there is no cycle of unobservable transitions in the system and the generated language is live. The assumptions on liveness and non-existence of unobservable cycles are removed in [9]. To construct the verifier, the model is then split into coaccessible states from failure states and non-failure parts, so that only the traces that lead to violation of diagnosability are searched. The approach in [9] is claimed to have the lowest complexity compared to all existing methods found.

In some recent works, e.g., [1, 3, 4], the smallest sequence of unobservable transitions that must have been fired to explain an observation is considered. It is called minimal explanation and has a great impact on the complexity reduction of diagnosability analysis. Its main advantage is that it does not require the exhaustive enumeration of the state space. In [7], this notion is explained and combined with the diagnosability test in [11], where the constructed verifier includes both observable and failure transitions. Since based on [7] there is a close connection between minimal explanation notion and automata, in this paper we evaluate the automata-based diagnosability approaches mentioned above both for PNs and automata.

This paper presents a survey and evaluation of the efficiency of polynomial algorithms for diagnosability, showing the importance of minimal explanation and basis marking notions. Basis markings are determined through minimal explanations firing and are a subset of the reachability set. The contribution of this work is a systematic evaluation on how complexity increases based on the number of sequences and tokens in PNs. Furthermore, we show that the theoretical complexity analysis, which is based on the worst case scenario, does not necessarily give the correct picture from a practical point of view and does not always lead to selecting an appropriate diagnosability algorithm. We also show the importance of considering modular automata as PNs, implying that the minimal explanation notion can be used for both modeling formalisms. We evaluate different polynomial diagnosability verifiers for both the reachability graph (RG) and the modified basis reachability graph (MBRG) [3] of a PN. A modified version of a recent diagnosability method, which has a state space with a smaller cardinality, is also introduced. The conclusion is that significantly larger systems can be analyzed when the most efficient automata-based diagnosability algorithms are combined with minimal explanations. For systems where the RG can be used, the improved computation time using the MBRG is often a factor 1000 or more.

The rest of the paper is organized as follows. Section II presents preliminary concepts on PNs, while Section III

gives the definition of diagnosability. In Section IV, the notion of the minimal explanations is presented. Section V summarizes different polynomial verifiers that can be used for analysis of diagnosability. In Section VI an evaluation and comparison of the different verifiers is made with and without considering the minimal explanations. Finally, conclusions are drawn in Section VII.

2. Preliminaries

A Place/Transition net (P/T net) is a structure $N = (P, T, Pre, Post)$, where P is a set of n_P places; T is a set of n_T transitions; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the pre and post incidence functions that specify the arcs; the incidence matrix is $C = Post - Pre$.

The marking vector $M : P \to \mathbb{N}$ assigns to each place of a P/T net a nonnegative integer number of tokens. The marking of place p is denoted $M(p)$. A P/T system or net system $\langle N, M_0\rangle$ is a net N with an initial marking M_0. A transition t is enabled at M if $M \geq Pre(\cdot, t)$ and may fire yielding the marking $M' = M + C(\cdot, t)$. We write $M\,[\sigma\rangle$ to denote that the sequence of transitions $\sigma = t_{j_1} \cdots t_{j_k}$ is enabled at M, and $M\,[\sigma\rangle\ M'$ represents that the firing of σ yields M'.

Given a sequence $\sigma \in T^*$, $\pi : T^* \to \mathbb{N}^{n_T}$ is called the function that associates with σ a vector $y \in \mathbb{N}^{n_T}$, named the firing vector of σ. Particularly, $y = \pi(\sigma)$, is such that $y(t) = h$ if the transition t is contained h times in σ.

A marking M is reachable in $\langle N, M_0\rangle$ if there exists a firing sequence σ such that $M_0\,[\sigma\rangle\ M$. The set of all markings reachable from M_0 defines the reachability set of $\langle N, M_0\rangle$ and is denoted $R\,(N, M_0)$.

A net system $\langle N, M_0\rangle$ is bounded if there exists a positive constant c such that, $\forall M \in R(N, M_0)$, $M(p) \leq c$. A labeling function $\mathcal{L} : T \to L \cup \{\varepsilon\}$ assigns to each transition $t \in T$ either a symbol from a given alphabet L or the empty string ε . We call labeled PN system the triple $\langle N, M_0, \mathcal{L}\rangle$.

The set of transitions is partitioned into the set of observable transitions T_o and set of unobservable transitions T_u, $T = T_o \dot{\cup} T_u$. The set of fault transitions T_f is a subset of T_u, $T_f \subseteq T_u$. If there are r different fault classes, T_f can be partitioned into r different subsets T_f^i, where $i = 1, \cdots, r$.

The PN depicted in Fig. 1, has k similar sequences from t_1 to t_5, (including e.g., t_{11}, t_{12} and t_{13}), which for number of tokens $n = 1$ can also be considered as the synchronization of k local automata G_i, one for each sequence, and the complete system is $G_1 \| \cdots \| G_k$. In Fig. 1 only transitions t_1 and t_5 are observable. In the following the RG of a bounded PN is defined as an automaton, which is used in the rest of paper.

Definition 1 The RG automaton is a tuple $G_{RG} = \left\langle X_{RG}, E_{RG}, \to_{RG}, x_0^{RG}\right\rangle$ where, $X_{RG} \in R(N, M_0)$ and $x_0^{RG} = M_0$. $E_{RG} = T$ is the set of transitions, and $\to_{RG} \subseteq X_{RG} \times E_{RG} \times X_{RG}$ denotes the transition relation where $M\,[t\rangle\ \acute{M}$ means that $(M, t, \acute{M}) \in \to_{RG}$. □

Consider the transition set T. Generally, some transitions $T_a \subseteq T$ in a PN can be abstracted. In Section IV this will be done by the concept of minimal explanations. The remaining transitions in the abstracted model are then denoted T_r, meaning that $T = T_r \dot{\cup} T_a$.

The RG includes all transitions in T, both the observable T_o and the unobservable T_u. Hence, for the RG, $T_r = T_o \cup T_u$ and $T_a = \emptyset$. The alphabet of the RG is the set of transitions of the corresponding PN. Considering the RG automaton for PNs, the diagnosability definition of automata is presented in the following section.

3. Diagnosability of Discrete Event Systems

Definition 2 (Event Observation Projection) The event observation projection [5], is a mapping from the original event set E to a smaller observable event set $E_o \subseteq E$, i.e., $P : E \to E_o \cup \{\varepsilon\}$ that can be extended to E^*, so we have $s \in E^*$, $\sigma \in E$: $P\,(s\sigma) = P\,(s)\,P\,(\sigma)$, with $P\,(\varepsilon) = \varepsilon$ and $P\,(\sigma) = \varepsilon$ for all $\sigma \in E_u$. Here, E_u denotes the unobservable event set, and E^* the set of all event traces generated from E. □

Definition 3 (Failure Assignment Function) Failure assignment function is a mapping from the original event set E to either 0 or $\mathcal{F}$. $\psi : E \to \mathcal{F} \cup \{0\}$, where $\mathcal{F} = \{T_f^i, i = 1, \ldots, r\}$. It means that if $\sigma \in E$ is not a failure event it is projected to 0. Otherwise it is projected to the T_f^i type failure set where it belongs to [6]. □

Assume G is live, without any cycle of unobservable events. Consider a trace $\sigma \in \mathcal{L}(G)$ ending with a T_f^i type failure, and a sufficiently long trace m obtained by extending σ. The system G is then *diagnosable* if every trace w whose observation is equivalent to m also contains a failure class T_f^i. Formally, diagnosability is defined as follows.

Definition 4 (Diagnosability) With respect to the event observation projection introduced in Definition 2 and the failure assignment function $\psi : E \to \mathcal{F} \cup \{0\}$, a system G is diagnosable if

$$(\forall T_f^i \in \mathcal{F})(\exists n_i \in \mathbb{N})(\forall s \in \mathcal{L},\, \psi(s_f) = T_f^i)$$
$$(\forall m = st \in \mathcal{L}, \|t\| \geq n_i) \Rightarrow$$
$$(\forall w \in \mathcal{L},\, P(w) = P(m))(\exists l \in pr(\{w\}),\, \psi(l_f) = T_f^i)$$

σ_f and l_f are the last events in traces σ and l, respectively, and $pr(\{w\})$ is the set of all prefixes of w [6]. □

For the sake of simplicity, here one fault type is considered. Necessary and sufficient conditions for diagnosability of PNs are introduced under the following assumptions. The net system $\langle N, M_0\rangle$ is bounded and has no deadlock, i.e., at least one transition fires at each reachable marking. The unobservable subnet is acyclic, i.e, $\forall \sigma \in T^*$, σ is not an unobservable trace and $\forall M$ if $M\,[\sigma\rangle\ M \Rightarrow \sigma \notin T_u^*$. The assumptions are also often considered for automata models. Moreover minimal explanations are introduced based on these assumptions.

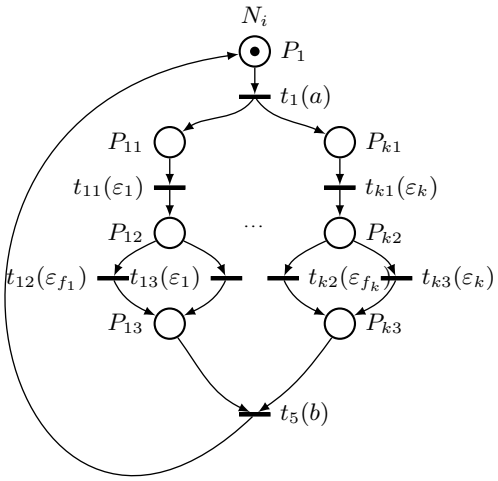

Figure 1 PN with k sequences, and generally n initial tokens in place P_1 (here $n = 1$).

4. Generation of Minimal Explanation

There are many diagnosability algorithms for automata, while for PNs it is still an open problem to find a computationally efficient analysis and there are few approaches. A straightforward diagnosability test for PNs is to construct the RG(PN), which for bounded PNs is an automaton, and then apply one of the automata-based diagnosability tests. This is a non-efficient approach, since constructing RG(PN) is exponentially complex wrt the number of tokens and places. PN is a more compact representation than an automaton for a system, so a small increase in the number of tokens or places often results in a huge increase of the number of states of the RG.

To solve this problem, a basic idea is to abstract away unobservable transitions, which do not convey any information about the system and its failures. In this way, a PN still reproduces all words generated by the labeled observable transitions. Based on this idea, a non-deterministic observer considering all observable and failure transitions (OF) with a smaller state space than the RG is constructed in [7], where $t \in T_r = T_o \dot{\cup} T_f$. In the OF, t is defined at state M and reaches M' iff in the PN there is an accessible trace $\tau = \sigma t$ that connects M to M', where σ indicates the finite number of non-failure unobservable transitions.

In most cases, the OF is a smaller automaton compared to RG. However, as shown in Table 1 for Fig. 1, where $k = 2$, the size of the OF for the number of tokens $n = 2$, even gets more transitions compared to RG. In Table 1, n_s is the number of states, and n_t is the number of transitions. Although for $n = 1$ the OF size is smaller than the RG, we still need further reduction of the PN. This is obtained based on the *minimal explanation* notion as in [2], and *strictly minimal explanation* as in [8]. These notions represent the reachability space in a compact form and hence, do not require the exhaustive enumeration of the state space. Based on [8], for PNs with acyclic unobservable sub-nets, both definitions on minimal explanation and strictly minimal explanation are equivalent.

Table 1 Number of states and transitions of the RG, OF and MBRG of the PN in Fig. 1 for $k = 2$.

	RG	OF	ROF/MBRG
	$n = 1$		
n_s	10	7	5
n_t	20	17	9
	$n = 2$		
n_s	46	46	14
n_t	146	546	36

The advantage of using minimal explanation is also presented in [4], where the basis reachability graph (BRG) is introduced, where $T_r = T_o$ and $T_a = T_u$. For diagnosability test the information provided by the BRG is however not enough, due to the fact that the BRG does not include enough information on all failure transitions [2]. Hence, a modified version of the BRG, called MBRG, is introduced where the failure events are not abstracted, and therefore $T_r = T_o \dot{\cup} T_f$ and $T_a = T_u \backslash T_f$. The size of the MBRG of Fig. 1 for $n = 1, 2$, is shown in Table 1. Moreover in [2], a diagnoser called basis reachability diagnoser (BRD) is defined using the MBRG.

In [7], a reduced version of OF, called ROF, is also constructed based on a backward search for the computation of minimal explanations. Both OF and ROF are non-deterministic automata, while all polynomial diagnosability algorithms considered in this paper assume that the original system is deterministic. Since the MBRG in [3] can be considered as a deterministic automaton, our presentation of minimal explanations for PNs is based on the MBRG formulation. We consider the definition of minimal explanation as in [3], where a tabular search algorithm to compute the set of minimal explanations is implemented. An explanation is then defined as minimal if its firing vector is component-wise minimal.

4.1. Minimal Explanation Notion

A transition $t \in T_r = T_o \cup T_f$ fires at marking M yielding $\acute{M}$ iff in the PN $\langle N, M \rangle$ the relation $M\,[\sigma\rangle\,\acute{M}$ is verified, where $\tau = \sigma t$ exists, σ is a minimal explanation of t, and $\sigma \in T_a^*$ ($T_a = T_u \backslash T_f$).

Example 1 Consider the PN in Fig. 1 for $k = 1$. t_{12} is a failure transition, which is considered as an observable transition in the MBRG generation. The procedure for finding minimal explanations and basis markings is as follows. Starting from M_0 there is only one transition $t_1 \in T_r$. Hence, t_1 connects the basis marking $M_0 = [1, 0, 0, 0]$ to $M_1 = [0, 1, 0, 0]$ with no e-vector. From M_1 there are two possible transitions belonging to T_r, t_{12} and t_5. For t_{12} the minimal explanation is t_{11}, $e_1 = [1, 0]$ and the next basis marking is $M_2 = [0, 0, 0, 1]$. Transition t_5 is in the outgoing traces of both basis markings M_1 and M_2, and the minimal explanations are $t_{11}t_{13}$, i.e., $e_2 = [1, 1]$ and ε, respectively. The MBRG automaton is depicted in Fig. 2. □

The formal definition of minimal explanation for a marking M and a transition $t \in T_r$ is as follows.

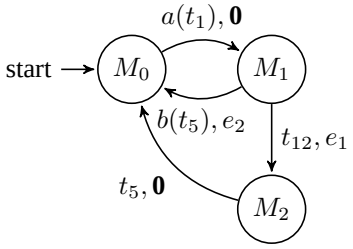

Figure 2 The MBRG automaton of Fig. 1 for $k = 1$.

Definition 5 [3] Given a marking M and a transition $t \in T_r$, we define

$$\Sigma(M,t) = \{\sigma \in T_a^* | M[\sigma\rangle \acute{M}, \acute{M} \geq Pre(\cdot,t)\}$$

as the set of explanations of t at M, and $Y(M,t) = \pi(\Sigma(M,t))$ the corresponding set of e-vectors (or explanation vectors), i.e., firing vectors associated with the explanations. □

Thus, $\Sigma(M,t)$ is the set of unobservable sequences whose firing at M enables t. Among the above sequences we want to select those whose firing vectors are minimal.

Definition 6 [3] Given a marking M and a transition $t \in T_r$, we define

$$\Sigma_{\min}(M,t) = \{\sigma \in \Sigma(M,t) \,|\, \nexists \sigma_1 \in \Sigma(M,t) : \pi(\sigma_1) \lneqq \pi(\sigma)\}$$

as the set of minimal explanations of t at M, and $Y_{\min}(M,t) = \pi(\Sigma_{\min}(M,t))$ the corresponding set of minimal e-vectors. □

4.2. Modified Basis Reachability Graph Automaton

In this section the MBRG automaton is defined based on minimal explanations and basis markings. The MBRG is here marginally modified compared to the MBRG in [3], where the authors did not consider it as an automaton. In the sequel, the basis marking and the MBRG automaton are defined.

Definition 7 The initial marking of a PN is also the initial basis marking, $X_{BM}^0 = M_0$. Starting from X_{BM}^0, the rest of basis markings are generated based on the following iteration until $X_{BM}^{i+1} = X_{BM}^i$.

$$X_{BM}^{i+1} = X_{BM}^i \cup \{\acute{M} \,|\, M \in X_{BM}^i \wedge t \in T_r \wedge \sigma \in \Sigma_{\min}(M,t) \wedge M\,[\sigma t\rangle\, \acute{M}\}$$

The set of basis markings for MBRG is $X_{MBRG} = X_{BM}^i$. □

Definition 8 The MBRG is an automaton $G_{MBRG} = \langle X_{MBRG}, E_{MBRG}, \rightarrow_{MBRG}, x_0^{MBRG}\rangle$, where $X_{MBRG} \subseteq X_{RG}$, $E_{MBRG}{=}T_r{\times}Y_{\min}(M,t)$, and $\rightarrow_{MBRG}{\subseteq}X_{MBRG}{\times}E_{MBRG}{\times}X_{MBRG}$ denotes the transition relation where $M\,[\sigma t\rangle\, \acute{M}$, and $\sigma \in \Sigma_{\min}(M,t)$ means that $(M,(t,e),\acute{M}) \in \rightarrow_{MBRG}$. □

Proposition 1 *[3]* G_{MBRG} is deterministic. □

Proposition 2 *[7]* $\mathcal{L}(G_{RG})$ is diagnosable wrt $\mathcal{F}$ iff $\mathcal{L}(G_{MBRG})$ is diagnosable wrt $\mathcal{F}$. □

Since G_{MBRG} is deterministic according to Proposition 1, any automata-based verifier can be used to verify diagnosability of a PN by investigating G_{MBRG} instead of G_{RG}, according to Proposition 2.

5. Different Verifier Automata

Here, three known verifiers are presented. A modified version of the first verifier, [6], is also introduced. To get a more fair comparison between the best known verifier, [9], and the concept used in [6], our modified version reduces the complexity of [6] in the same spirit as in [9].

5.1. V1 Verifier

In [6] a verifier is introduced that abstracts away all unobservable and failure transitions by first constructing an observable automaton, G_o, whose definition is presented in Algorithm 1 in Subsection 5.4. Then $G_d = G_o \| G_o$. This verifier (V1) is checked for the existence of possible indeterminate cycles by identifying the uncertain states in G_d. All other states and their associated transitions are then deleted. If the remaining graph contains at least one cycle the system is not diagnosable. This verifier was the first proposed polynomial algorithm with the worst case complexity $\mathcal{O}(n_s^4 n_T)$.

5.2. V2 Verifier

In [11] a different synchronization rule for unobservable transitions compared to V1 is defined. The computational complexity is $\mathcal{O}(n_s^2 n_T)$. Thus, in worst case it has lower computational complexity than the V1 algorithm. In practice however, when there are a significant number of unobservable events in the system, V1 is often more efficient, due to the abstraction of unobservable transitions.

5.3. V3 Verifier

The algorithm in [9] starts with the construction of the non-failure automaton, G_N, the co-accessible states from faulty states, G_F, and parallel synchronization $G_N \| G_F$. The algorithm efficiency is due to the fact that in the synchronization step, only the traces that lead to the violation of diagnosability are searched. In the algorithm, the assumptions of liveness and non-existence of unobservable cycles are removed and its complexity is $\mathcal{O}(n_s^2(n_T - n_{T_f}))$, where n_{T_f} represents the number of failure transitions. It is claimed that this verifier has lower complexity than all other methods found in the literature. Note that, since verifier V3 has a more efficient formulation than verifier V2, in comparisons we only consider V3.

Table 2 Number of states and transitions of the verifier for the PN in Fig. 1.

		RG		V1				V3				V4			
n	k	n_s	n_t	n_s	n_t	t_s	t_{cd}	n_s	n_t	t_s	t_{cd}	n_s	n_t	t_s	t_{cd}
1	1	5	5	8	13	0.11	0.005	20	34	0.19	0.01	4	5	0.08	0.005
1	2	10	20	8	13	0.11	0.005	116	344	0.60	0.02	4	5	0.08	0.005
1	3	28	83	8	13	0.11	0.004	884	4086	4.33	0.09	4	5	0.08	0.006
2	1	10	20	76	889	0.21	0.008	92	248	0.47	0.02	38	335	0.14	0.008
2	2	46	146	652	66265	80.53	0.09	1772	8270	9.45	0.21	326	22907	5.66	0.07
2	3	244	1109	*	*	*	*	47972	336762	3130	246.20	2918	1716797	44193	52.62
3	1	20	50	328	11604	1.88	0.02	292	974	1.33	0.03	164	4308	0.51	0.03
3	2	146	578	*	*	*	*	14052	81306	224	10.54	5348	3279134	158066.74	164.78
3	3	1244	6941	*	*	*	*	*	*	*	*	*	*	*	*

5.4. V4 Verifier

We present V4 as an alternative strategy, which intuitively can give better performance than V3 due to smaller state space, particularly when there are a significant number of unobservable transitions. Increasing the number of unobservable transitions results in an obvious increase in the size of V3, while it does not change the number of states in V4. The proposed V4 verifier is based on V1, but exploits the symmetry of the states in $G_d = G_o \| G_o$ similar to V3. In V4 two parts of the G_o are generated: G_o^N, the non-failure part, and G_o^F, the co-accessible part of G_o from failure states. The resulting verifier $G_o^N \| G_o^F$ has less states, but typically more transitions than V3.

Algorithm 1: For a given system $G = (X, E, \rightarrow, x_0)$, the following algorithm constructs the V4 verifier:

1. Augment each state of G with failure labels (N, F), based on the failure assignment function.

2. Obtain a non-deterministic automaton $G_o = (X_o, E_o, \rightarrow_o, x_0^o)$, where $X_o = \{(x, f) \,|x \in X_1 \cup \{x_0\}, f \subseteq \mathcal{F}\}$ and $X_1 = \{x \in X | \exists\, (x', \sigma, x) \in \rightarrow \text{ with } P(\sigma) \neq \varepsilon\}$.

3. Compute G_o^N that represents the non-faulty behavior of G_o. Construct the G_o^F by marking all states of G_o that have label F, denoted as $\bar{G}_o^F$, then start from $\bar{G}_o^F$ and construct $G_o^F = CoAcc(\bar{G}_o^F)$.

4. Compute the verifier automaton, $G_V = G_o^N \parallel G_o^F$.

Proposition 3 System G is diagnosable according to verifier V4 iff it is diagnosable according to verifier V1.

Proof: Since $G_d = G_o \| G_o$, G_d includes all different combinations of failure labels, i.e., (x_1, N, x_2, N), (x_1, N, x_2, F), (x_1, F, x_2, N), and (x_1, F, x_2, F). A system is not diagnosable if there exists at least one loop among the uncertain states in G_d. Assuming the existence of failure, G_o includes both failure labels. Thus, to avoid symmetry, we only perform strict composition of G_o with its G_o^N, which results in half of the number of uncertain states. To get all states that may lead to failure transitions, we calculate the co-accessible part of the states with label F, denoted as G_o^F. Hence, all uncertain states in G_d of type (x_1, N, x_2, F) are included in G_V. □

6. Comparisons

The non-diagnosable PN in Fig. 1 is evaluated for different number of initial markings n and sequences k. The cycle detection time t_{cd} is the elapsed time until the algorithm finds a cycle in the graph. t_s and t_e are the verifier construction and total elapsed times, respectively. All tests have been run on a PC Intel with a clock of 3.10 GHz, RAM 16 GB. Computation times are in seconds.

6.1 Comparing Verifiers Based on RG

Here, the verifiers are constructed based on the RG. The $*$ sign in Table 2 shows that $t_s > 15$ hours. V4 is more efficient than V1, and V3 is often better than V4, especially for large systems. However, increasing the number of unobservable transitions in each sequence, by the factor β and described in more details in Example 2, shows in Table 3 that V4 is more efficient than V3.

Example 2 Consider the PN in Fig. 1 with $n = 1$ and $k = 3$. Replace the transition t_{k3} with an observable one with a distinct label. The PN is diagnosable. Let β be the number of unobservable transitions from P_{i1} to P_{i2} (the same number for all $i = 1, \ldots, k$). Table 3 shows the results for V3 and V4. This example is a counterexample, where for large systems V4 has better results than V3. □

As already observed, V3 is often better for large systems. Although n_s is less in V4 than V3, n_t in V4 often increases dramatically for large systems. The reason is that abstracting away all unobservable transitions typically generates a number of non-deterministic transitions. In the synchronization in G_V this generates a huge number of transitions for large systems. However, in Example 2 the number of unobservable transitions in V3 dominates, which results in the opposite behavior where V4 is more efficient, avoiding all the unobservable transitions.

Table 3 Number of states and transitions of two verifiers for the PN in Fig. 1 for $n = 1$ and $k = 3$ and additional unobservable transitions β according to Example 2.

	V3			V4		
β	n_s	n_t	t_e	n_s	n_t	t_e
1	344	1158	1.65	129	554	0.30
2	2198	9090	13.01	471	3701	1.11
3	9262	42276	95.32	1263	15650	5.16
4	29792	144060	666.35	2793	50027	37.32
5	76508	399258	4328.43	5421	132410	251.55

Table 4 Number of states and transitions of the verifier for the PN with MBRG instead of RG in Fig. 1.

		RG		MBRG			V3			V4		
n	k	n_s	n_t	n_s	n_t	t_{MBRG}	n_s	n_t	t_e	n_s	n_t	t_e
1	1	4	5	3	4	0.09	5	7	0.16	4	5	0.08
1	2	10	20	5	9	0.09	7	13	0.18	4	5	0.08
1	3	28	83	9	21	0.13	11	26	0.19	4	5	0.08
1	4	82	326	17	49	0.36	19	55	0.21	4	5	0.08
1	5	244	1217	33	113	4.06	35	120	0.23	4	5	0.08
2	1	10	20	6	11	0.09	9	17	0.20	8	18	0.09
2	2	46	146	14	34	0.16	17	42	0.22	12	42	0.10
2	3	244	1109	36	110	2.85	39	120	0.29	20	120	0.11
2	4	1378	8264	98	362	170.74	101	374	0.48	36	390	0.14
2	5	8020	60023	276	1198	12826.11	279	1198	1.21	68	1368	0.22
3	1	20	50	10	21	0.10	14	30	0.21	13	40	0.10
3	2	146	578	30	83	0.69	34	95	0.28	27	170	0.13
3	3	1244	6941	100	345	101.42	104	360	0.55	67	964	0.23
4	1	35	100	15	34	0.11	20	46	0.22	19	73	0.11
4	2	371	1678	55	164	3.63	60	180	0.36	51	489	0.19
4	3	4619	29191	225	834	1908.13	230	854	1.08	169	4507	0.85
6	1	84	280	28	69	0.17	35	87	0.25	34	180	0.15
6	2	1596	8428	140	454	76.44	147	478	0.70	134	2382	0.56
10	1	286	1100	66	175	0.81	77	205	0.43	76	614	0.28
10	2	12298	75086	506	1770	6681.96	517	1810	3.01	496	20330	9.87

6.2 Comparing Verifiers Based on MBRG

The most efficient verifiers V3 and V4 are now constructed based on the MBRG. Using minimal explanation reduction, V3 and V4 are similar except for the failure abstraction case, which is not abstracted in the V3 generation. As shown in Table 4, except for the last row, t_e of V4 is less than V3. For small and medium sized systems, V4 in combination with minimal explanations has smaller state space and the cost of increasing n_t is not considerable. Hence, V4 has better results than V3. For large systems, although V4 has less n_s, V3 performs better due to the significant growth in n_t for V4.

A more important conclusion is that the introduction of the MBRG improves t_e dramatically, often with a factor 1000 or more. Many systems that cannot be solved based on RG are easily solved by first generating the MBRG. The bottleneck is not the diagnosability verifier but the calculation of the MBRG.

7. Conclusion

In this paper the efficiency of different polynomial diagnosability verifiers has been evaluated for PNs in terms of n_s, n_t and t_e. A modified verification algorithm was also proposed, which in the same way as other known verifiers, was constructed based on either the RG or MBRG. The verifiers were compared in different aspects, and the results showed that for small and medium sized systems verifier V4 was better, while for very large systems verifier V3 was more efficient. Moreover, it is emphasized that the verifier construction based on the reduced MBRG is applicable both for PNs and synchronized automata.

The main conclusion is however that significantly large systems can be analyzed when the most efficient automata-based diagnosability algorithms are combined with minimal explanations. For systems where the RG can be used, the improved t_e using MBRG is often 1000 times faster or even more.

References

[1] R. Boel and G. Jiroveanu. Distributed contextual diagnosis for very large systems. *IFAC WODES04: 7th Work. on Discrete Event Systems*, page 333, 2004.

[2] M. Cabasino. *Diagnosis and identification of discrete event systems using Petri nets*. PhD thesis, University of Cagliari, Italy, 2008.

[3] M. Cabasino, A. Giua, and C. Seatzu. Diagnosability of bounded Petri nets. *IEEE Conference on Decision and Control*, pages 1254–1260, 2009.

[4] M. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9):1491–1495, 2010.

[5] C. Cassandras and S. Lafortune. *Introduction to discrete event models*. Kluwer, Norwell, MA, 1999.

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

[7] G. Jiroveanu and R. Boel. The diagnosability of Petri net models using minimal explanations. *IEEE Transactions on Automatic Control*, 55(7):1663–1668, 2010.

[8] G. Jiroveanu, R. Boel, and B. Bordbar. On-line monitoring of large Petri nets models under partial observation. *Discrete Event Dyn Syst*, 18(3):323–354, 2008.

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

[10] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[11] T. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1531–1539, 2002.