A comparison among tools for the diagnosability of discrete event systems

Maria Paola Cabasino, Alessandro Giua, Laura Marcias, Carla Seatzu [*]

August 27, 2013

Abstract

In this paper we compare three tools for the diagnosability analysis of discrete event systems. After introducing the notion of diagnosability, we recall for each tool the main theoretical results on which it is based. A benchmark that describes a parametric manufacturing system is defined and used for comparison. We report the numerical results obtained for different values of the parameters and draw some general conclusions on the advantages and possible improvements of these tools.

Published as:
M.P. Cabasino, A. Giua, L. Marcias, C. Seatzu, "A comparison among tools for the diagnosability of discrete event systems", *CASE12: 8th IEEE Conference on Automation Science and Engineering*, (Seoul, Korea), Aug 2012.

[*]M.P. Cabasino, A. Giua, L. Marcias and C. Seatzu are with the Department of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy. `E-mail: {cabasino,giua,seatzu}@diee.unica.it, laura.marcias83@gmail.com`.

1 Introduction

Failure detection and diagnosis in industrial systems is a subject that has received a lot of attention in the past few decades. Within this body of research two different problems are addressed: diagnosis and diagnosability. Solving a problem of diagnosis means that we associate with each observed string of events a diagnosis state, such as “normal” or “faulty” or “uncertain”. Solving a problem of diagnosability is equivalent to determining if the system is diagnosable, i.e., if once a fault has occurred the system can detect its occurrence in a finite number of steps.

In the framework of discrete event systems, several approaches for the diagnosability analysis have been proposed for both automata [19, 14, 13, 10] and Petri nets (PNs) [16, 4, 9, 11, 21, 6, 22, 2]. We also mention two recent approaches that addressed the problem of on-line diagnosis (but not diagnosability) of Petri nets [1, 8].

In a seminal paper [19] Sampath *et al.* presented the so called *Diagnoser Approach* and proposed necessary and sufficient conditions for the diagnosability of a system modeled as an automaton. The Discrete Event System Group of the University of Michigan developed the *UMDES* tool [20]. UMDES is a library of C routines written for the study of discrete event systems modeled by finite-state automata. In particular, in UMDES there is a function that implements the theoretical results in [19] and allows one to analyze the diagnosability of a given automaton.

In [4] Cabasino *et al.* presented an original approach for diagnosability analysis of *bounded* PNs based on the notion of basis markings previously introduced in [5, 3]. The main feature of this approach it that of not requiring the exhaustive enumeration of the set of reachable states of the system: only the (possibly much) smaller subset of states corresponding to basis markings need to be enumerated. Based on this approach, a MATLAB tool for the diagnosability analysis of bounded PNs has been developed at the University of Cagliari, called *PN_DIAG* [15].

Unfortunately, in the case of unbounded PNs, i.e., nets with an infinite state space, the approach founded on basis markings cannot be used: in fact, in such a case the set of basis markings is also infinite. In [2] Cabasino *et al.* presented an approach for diagnosability analysis of labeled PNs that applies to unbounded nets (and thus, as a particular case, also to bounded nets). Necessary and sufficient conditions for diagnosability and diagnosability in k steps of unbounded systems were obtained. Furthermore, in this work it was also presented a test to analyze diagnosability and diagnosability in k steps based on the reachability/coverability graph of a particular net, called *Verifier Net*, built from the PN model of the system to be diagnosed. Based on this approach a MATLAB tool called *PN_DIAG_UNBOUNDED* [18] has been developed.

The salient features of the three tools mentioned above for the diagnosability analysis of discrete event systems, are shown in Table 1 where the meaning of the last but one column will be explained in the following. In this paper these tools are compared, to clarify their respective advantages/disadvantages.

Note that while the last two tools, namely PN_DIAG and PN_DIAG_UNBOUNDED, are based on PNs, the tool UMDES is based on automata. Thus, to compare them on the same test

Tool	Model	Basis Markings	Bounded/Unbounded	Diagnoser	Software
UMDES	Automata	Not applicable	Bounded	$Diag(G)$	C executable
PN_DIAG	Petri nets	Yes	Bounded	BRD	MATLAB
PN_DIAG_UNB.	Petri nets	No	Bounded + Unbounded	None	MATLAB

Table 1: A summary of the main features of the considered tools

cases we made use of a software platform developed within the FP7 project DISC (Distributed Supervisory Control of Complex Plants) [7]. Using the software platform, we constructed for a given PN system its reachability graph, converted it into a file format suitable for UMDES, and run the relevant UMDES functions.

Comparisons have been carried out on a benchmark that describes a parametric manufacturing system. A similar benchmark was also used in [12] to compare two approaches for the diagnosis of discrete event systems. Here the benchmark has been modified to address the diagnosability analysis.

The paper is structured as follows. Section 2 provides some background on labeled PNs. Section 2 summarizes the main notions behind the diagnosability approach for automata on which UMDES is based. The philosophy behind PN_DIAG is summarized in Section 4. The philosophy behind PN_DIAG_UNBOUNDED is recalled in Section 5. The considered benchmark is presented in Section 6. The results of numerical simulations are illustrated in Section 7. Conclusions are finally drawn in Section 8.

2 Background on labeled Petri nets

In this section we briefly recall the formalism used in the paper. For more details on PNs we refer to [17].

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

A *marking* is a vector $M : P \rightarrow \mathbb{N}$ that assigns to each place of a P/T net a nonnegative integer number of tokens, represented by black dots. We denote $M(p)$ the marking of place 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 iff $M \geq Pre(\cdot, t)$ 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 we write $M\ [\sigma\rangle\ M'$ to denote that the firing of σ yields M'. We also write $t \in \sigma$ to denote that a transition t is contained in σ. The set of all sequences that are enabled at the initial marking M_0 is denoted $L(N, M_0)$, i.e., $L(N, M_0) = \{\sigma \in T^* \mid M_0[\sigma\rangle\}$.

A marking M is *reachable* in $\langle N, M_0 \rangle$ iff 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 PN having no directed circuits is called *acyclic.* A net system $\langle N, M_0\rangle$ is *bounded* if there exists a positive constant k such that, for all $M \in R(N, M_0)$, it is $M(p) \leq k$.

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

We denote as T_o the set of transitions whose label is a symbol in L and transitions in T_o are called *observable.* The same label $l \in L$ can be associated with more than one transition. Two transitions $t_1, t_2 \in T_o$ are called *undistinguishable* if they share the same label, i.e., $\mathcal{L}(t_1) = \mathcal{L}(t_2) = l \in L$. The set of transitions sharing a label l are denoted T_l.

3 Diagnoser Approach for Automata

In this section we briefly recall the seminal approach to the diagnosis of automata developed by Sampath *et al.* in [19] known in the literature as *Diagnoser Approach.* The considered model is an automaton $G = (X, E, \delta, x_0)$, where X is the set of states, E is the set of events, $\delta : X \times E \to X$ is the transition function and x_0 is the initial state. The set of events E is partitioned into observable events E_o and unobservable events E_u. The set of unobservable events is further partitioned into two disjoint subsets: the subset of unobservable and regular events E_{reg} and the set of unobservable and faulty events E_f. Finally the set of fault events is partitioned into r fault classes $E_f = E_f^1 \cup E_f^2 \cup \cdots \cup E_f^r$. The assumptions under which this approach is valid are:

(A1) The language $L(G)$ generated by G is live. This means that there not exists a state $x \in X$ from which no event is possible.

(A2) There does not exist in G any cycle of unobservable events.

Assumption (A1) is made for the sake of simplicity. On the contrary, assumption (A2) is necessary and ensures that G does not generate sequences of unobservable events whose length can be infinite.

The diagnosability analysis is based on the search of *indeterminate cycles* in the diagnoser. The diagnoser $Diag(G)$ is an automaton built starting from the system model G whose set of events is E_o and whose language is the projection of the language of G onto E_o. Each state of $Diag(G)$ contains one or more states of G each one associated with a row vector of labels l_f having r columns. The ith entry of l_f is equal to N if reaching that state no fault event belonging to the ith fault class has occurred, while it is equal to Y otherwise. The initial state of $Diag(G)$, called $x_{0,diag}$, contains x_0 that has l_f equal to a vector of N by definition and all those states that are reachable from x_0 firing one or more unobservable events. Starting from $x_{0,diag}$ we compute the set of events that are enabled by at least one state in $Diag(G)$. For each one of such events we consider all states $x \in X$ that can be reached from a state in $x_{0,diag}$. This set of states is then extended to all those states that can be reached from one of such states firing one or more unobservable events. The vector of labels will be computed in the way described above, and obviously, for the so called *propagation rule*, if one entry of one state x is equal to Y all states reached starting from x will have as well that entry equal to Y. The procedure is repeated until all states have been explored.

We now recall the notion of indeterminate cycles and for the sake of simplicity consider one single fault event f. In $Diag(G)$ we can distinguish three different kinds of states:

- *negative state*, if in the node of $Diag(G)$ for all pairs (x, l), $l = N$, i.e., reaching this node we are sure that fault f has not occurred yet;
- *positive state*, if in the node of $Diag(G)$ for all pairs (x, l), $l = Y$, i.e., reaching this node we are sure that fault f has occurred;
- *uncertain state*, if in the node of $Diag(G)$ there exists at least one pair (x, l) such that $l = N$ and at least one pair (x', l') such that $l' = Y$, i.e., we cannot say nothing about the occurrence of the fault f.

A cycle in the diagnoser is said *uncertain* if it is composed only by uncertain states. An *indeterminate cycle* in $Diag(G)$ is a cycle composed exclusively of uncertain states for which there exist:

- a corresponding cycle of observable events in G involving only states that carry Y in their labels in the cycle in $Diag(G)$ and
- a corresponding cycle of observable events in G involving only states that carry N in their labels in the cycle in $Diag(G)$.

The notion of indeterminate cycles is very important because their analysis gives us necessary and sufficient conditions for diagnosability. A language $L(G)$ is diagnosable if and only if its diagnoser $Diag(G)$ has no indeterminate cycles for all failure classes E_f^i, $i = 1, \dots, r$. Note that the presence of a cycle of uncertain states in a diagnoser does not necessarily imply inability to diagnose with certainty an occurrence of event f.

The UMDES library [20], that is a collection of C executable functions, also performs diagnosability analysis. In fact, given as input the automaton and the fault classes the function *dcycle* computes the diagnoser and the presence of indeterminate cycles in it, giving as output information on the diagnosability of the system.

4 The philosophy behind PN_DIAG

In [3, 5] we present a fault diagnosis approach that — using the notions of *basis marking* and *justification* — allows one to analyze diagnosability of a system modeled using PNs without enumerating the entire state space, but only a subset of it. We consider labeled PNs having undistinguishable transitions and assume that the set of unobservable transitions T_u is partitioned into regular transitions T_{reg} and faulty transitions T_f. The assumptions under which our approach holds are:

(B1) The unobservable subnet is acyclic.

(B2) The system does not enter a deadlock after the firing of any fault transition.

The first assumption is analogous to the classical hypothesis in the theory of automata where no cycle of unobservable events can appear and allows one to use the state equation and have necessary and sufficient conditions for the reachability. The second assumption is a weakened version of the usual "liveness" assumption in most works on diagnosability of discrete event systems; it avoids the technicalities that must be dealt with when the system may deadlock after a fault.

Given a word $w \in L^*$, let $\sigma_o \in T_o^*$ be a sequence of observable transitions such that $\mathcal{L}(\sigma_o) = w$. A *basis marking* M_b is a marking reached from M_0 with the firing of σ_o and of all unobservable transitions whose firing is *strictly* necessary to enable w. Such a sequence σ_u of unobservable transitions interleaved with σ_o whose firing enables σ_o and whose firing vector is *minimal* is called *justification.* Since in general σ_o is not unique and more than one σ_u may be associated with each σ_o, then the set of justifications of w is not necessarily a singleton.

In [4] we focus on bounded PN systems: we provide a necessary and sufficient condition for diagnosability and give a systematic method to analyze the diagnosability of a given PN system. Such a method requires the construction of two labeled and oriented graphs denoted respectively *Modified Basis Reachability Graph* (MBRG) and *Basis Reachability Diagnoser* (BRD). The MBRG nodes contain either a basis marking or a marking that can be reached firing a fault transition. A node of the BRD contains one or more basis markings and each one has associated a label that specifies if reaching that marking a fault transition has fired.

As for the Diagnoser Approach described in the previous section we can distinguish among negative states, positive states and uncertain states of the BRD. Basically, the diagnosability analysis first consists of determining if there are uncertain cycles in the BRD, namely cycles of uncertain states. If there are no uncertain cycles in the BRD the system is diagnosable. In the presence of uncertain cycles the following analysis should be performed for any of such a cycle. Let γ be an uncertain cycle in the BRD with observable projection $\rho \in L^*$ and let $p \in L^*$ be a path from the initial node to any node of the cycle. We need to verify if the cycle γ is *indeterminate* wrt a fault class T_f^i, namely if in the MBRG there exist two cycles γ_1 and γ_2 satisfying the following three conditions:

(i) their observable projection is equal to ρ;

(ii) there exist two paths p_1 and p_2 with observable projection p, that from the initial node in the MBRG enable γ_1 and γ_2;

(iii) both γ_2 and p_2 do not contain a fault in T_f^i, while either γ_1 or p_1 (or both) contain a fault in T_f^i.

The MATLAB tool that implements this approach is called PN_DIAG [15]. This tool explores all paths starting from the initial node of the BRD looking for uncertain cycles and does the same in the MBRG when looking for p_1, γ_1, p_2 and γ_2.

5 The philosophy behind PN_DIAG_UNBOUNDED

In [2] Cabasino *et al.* presented necessary and sufficient conditions for diagnosability of unbounded PNs. This is a significant result, because the diagnosability of systems with an infinite state space has never been studied before. As in the previous section, the considered model is a labeled PN where some transitions are undistinguishable. No assumption either on the structure of the net or on the labeling function is required. The only assumption that should hold is (B2) defined in Section 4. The test to analyze diagnosability is based on the analysis of the reachability/coverability graph of a new type of net, called *Verifier Net* (VN), built from the PN model of the system to be diagnosed. In particular, if the net is bounded, we just need to check if there exists a cycle in the reachability graph of the VN that is reachable starting from a node that can be reached firing a path that contains a fault transition. If this is the case the system is not diagnosable. In the case of unbounded nets, we need to verify if there exists a cycle in the coverability graph of the VN associated with a repetitive sequence in the VN that is reachable starting from a node that can be reached firing a path that contains a fault transition. If this is the case the system is not diagnosable.

Assume that n_N denotes the number of nodes of the reachability/coverability graph of the net to be analyzed, while n_V denotes the number of nodes of the reachability/coverability graph of the VN. It holds $n_N < n_V \leq n_N^2$, hence we need to study a system with a larger state space. Thus for bounded systems we do not expect this approach to be particularly efficient with respect to other two considered in this paper. Note, however, that once the reachability/coverability graph of the VN is computed, we need not compute a diagnoser but only need to check this graph as summarized in the last but one column of Table 1.

The MATLAB tool that implements this approach is downloadable from [18].

6 The considered benchmark

In this section we introduce the benchmark we use to compare the above three tools.

The proposed benchmark describes a parametric manufacturing system. It consists of two symmetric working groups that process different components of the same product. Each working group is characterized by β production lines, each one devoted to the processing of a component. Each production line executes $\eta + 1$ operations: when the first η operations are executed, no output signal is produced; on the contrary, the last operation of each line can be observed in the sense that an output signal is produced whenever the corresponding operation is completed. However, such operations are not completely distinguishable in the sense that operations corresponding to a given line produce an output signal that is the same for the two groups. An only exception exists to this that is related to the first line. In particular, two different cases may occur: (a) the last operations of the first lines of both groups produce the same output signal; (b) the last operations of the first lines of both groups produce different output signals. Finally, faults may occur in the system. In particular, a part of a generic line i may be moved accidentally to production line $i + 1$ of the same group, for $i = 1, \dots, \beta - 1$. Note that this may only occur after the first η operations in the line have already been executed.

The labeled PN model of such a manufacturing system is shown in Fig. 1. Operations in the generic line i of the first group are modeled via unobservable transitions and are denoted $\varepsilon'_{i,j}$, with $i = 1, \dots, \eta$ and $j = 1, \dots, \beta$. Similarly, operations in the generic line i of the second group are modeled as unobservable transitions and are denoted $\varepsilon''_{i,j}$, with $i = 1, \dots, \eta$ and $j = 1, \dots, \beta$. For the sake of clarity all such transitions are drawn in blue in Fig. 1.

Fault events are modeled as unobservable transitions as well but are drawn in red to clearly distinguish them from the previous unobservable but regular transitions.

Finally, green transitions model the last operation in each line. All such transitions are observable. In particular, the same label a_i, with $i = 2, \dots, \beta$, is shared by transitions in line i in the first and in the second group. The last transition in the first line of the second group is labeled a_1, while two different labels may be associated with the last transition in the first line of the first group, namely, a_0 or a_1, depending on the value assigned to α in Fig. 1 that may either take value 0 or 1. Note that such a distinction is really significant because the value of α affects the diagnosability of the system. In particular, using any of the above tools, it can be proved that the following holds:

$$\alpha = \begin{cases} 1, & \text{the system is diagnosable,} \\ 0, & \text{the system is not diagnosable.} \end{cases}$$

7 Numerical simulations

In this section we compare the MATLAB tools PN_DIAG and PN_DIAG_UNBOUNDED, and the UMDES library. Such a comparison is carried out on the benchmark model introduced in the previous Section 6, whose PN model is sketched in Fig. 1. The automaton model used by UMDES corresponds to the reachability graph of the PN system.

Several cases have been studied for different values of β and η, considering both $\alpha = 0$ and $\alpha = 1$

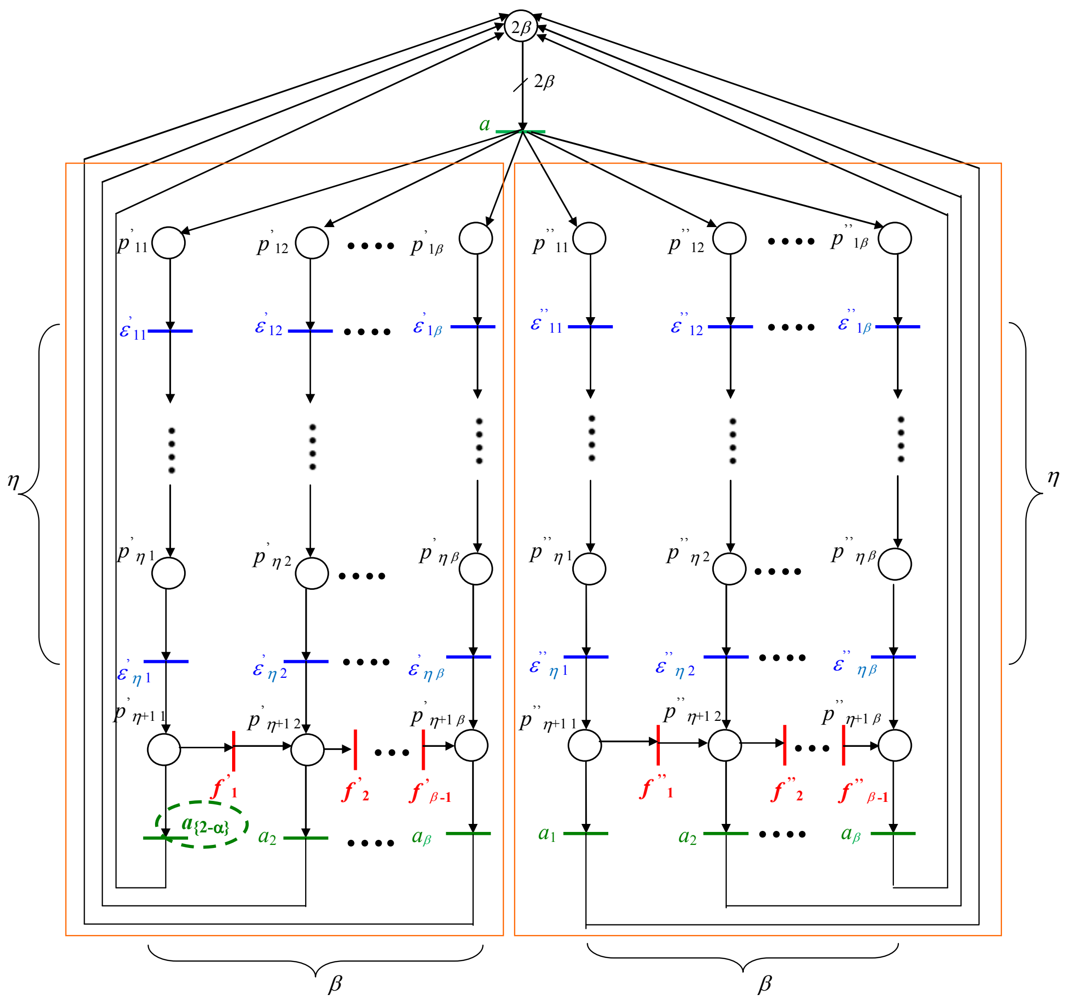

Figure 1: The considered benchmark

and they are summarized in Table 2.

For the sake of simplicity we assumed that all faults belong to the same class.

All tests have been run on a PC Intel with a clock of 2.27 GHz, RAM 4 GB.

- Columns 1, 2 and 3 show the values of β, η and α.
- Column 4 shows the time in seconds required to analyze the diagnosability using the tool PN_DIAG. This is the sum of times to compute the MBRG, the BRD and to verify the presence of indeterminate cycles.
- Column 5 shows the time in seconds required to analyze the diagnosability using the tool PN_DIAG_UNBOUNDED. This is the sum of times to compute the VN, the reachability graph of the VN and to look for cycles enabled after the firing of a fault transition.
- Column 6 shows the time in seconds required to analyze the diagnosability using the library UMDES. This is the sum of times to compute the diagnoser, starting from the automaton that represents the reachability graph of the benchmark net, and to look for indeterminate cycles. Obviously, we did not take into account the time spent to convert the PN into the corresponding automaton.

Some cells in the table contain non numerical values.

- *o.t.* (*out of time*): denotes that the tool did not halt within 24 hours;
- *o.m.* (*out of memory*): denotes that the virtual memory of the calculator has run out.

Table 2 shows that the time required to analyze diagnosability highly increases with the dimension of the net and of its initial marking, namely with β and η. From the table one can also conclude that the best results are always obtained using PN_DIAG, while the other tools in most of the cases cannot determine a solution within the maximum time allotted (24 hours) or run out of memory.

It is not surprising that the best performances are obtained by a tool that uses the notion of basis marking. Indeed, for the considered example, the number of basis markings is significantly smaller than the number of reachable markings and such a difference becomes more relevant as the values of the parameters increase, as summarized in Table 3. In particular, from this table it can be seen that while the number of basis markings does not depend on η, the number of reachable markings is highly dependent on it.

Table 3 shows that for certain values of the parameters β and η the state space becomes so large that the two tools PN_DIAG_UNBOUNDED and UMDES that require an exhaustive enumeration of it either go out of memory or out of time.

From the above comparison PN_DIAG_UNBOUNDED is the less effective tool, even when compared to UMDES. This is due to the fact that PN_DIAG_UNBOUNDED requires an exhaustive

β	η	α	**PN_DIAG**	**PN_DIAG_UNBOUNDED**	**UMDES**	**Diagnosable?**
2	1	0	$5.1 \cdot 10^{-1}$	$7.9 \cdot 10^{1}$	$1.1 \cdot 10^{0}$	No
2	1	1	$3.7 \cdot 10^{-1}$	$6.3 \cdot 10^{1}$	$1.1 \cdot 10^{0}$	Yes
2	2	0	$1.7 \cdot 10^{0}$	o.m.	$6.0 \cdot 10^{1}$	No
2	2	1	$1.6 \cdot 10^{0}$	$1.0 \cdot 10^{5}$	$3.0 \cdot 10^{1}$	Yes
2	3	0	$8.9 \cdot 10^{0}$	o.m.	o.t.	No
2	3	1	$8.8 \cdot 10^{0}$	o.m.	o.t.	Yes
3	1	0	$6.7 \cdot 10^{1}$	o.m.	o.t.	No
3	1	1	$6.3 \cdot 10^{1}$	o.m.	o.t.	Yes
3	2	0	$2.0 \cdot 10^{3}$	o.m.	o.t.	No
3	2	1	$2.1 \cdot 10^{3}$	o.m.	o.t.	Yes
3	3	0	$3.4 \cdot 10^{5}$	o.m.	o.t.	No
3	3	1	$3.4 \cdot 10^{5}$	o.m.	o.t.	Yes

Table 2: Numerical Results (times in seconds)

(β,η)	**Reachable Markings**	**Basis Markings**
(2,1)	121	16
(2,2)	361	16
(2,3)	841	16
(3,1)	2025	64
(3,2)	10000	64
(3,3)	34225	64

Table 3: The number of reachable and basis markings for different values of the parameters

enumeration of the reachability set of the VN which may be significantly larger than the reachability set of the original system. However, it should be stressed that PN_DIAG_UNBOUNDED is the only tool that can manage unbounded nets. Moreover, unlike all other tools, it also applies to nets whose unobservable subnet is acyclic. An interesting open problem is that of finding structural conditions for the analysis of the VN without resorting to the enumeration of its reachability set: this may lead to the development of a more efficient tool.

We finally remark that PN_DIAG computes all possible paths on the BRD, following a depth-first search, starting from the initial node until a cycle is found. It may be possible that other algorithms based on the computation of the cycles without computing the paths may in some cases further improve the performance of the simulator.

8 Conclusions

The main contribution of this paper is to compare the performance of three tools that analyze diagnosability of labeled PNs and point out critical bottlenecks that could lead to better implementation. We have considered as a benchmark model a PN taken from the manufacturing area and we have analyzed the tools both in case of diagnosable and non diagnosable systems.

The benchmark shows that the PN tool using basis markings may, in some cases, greatly outperform tools based on an exhaustive enumeration of the state space.

Acknowledgments

We thank Marco Pocci for the development of the MATLAB tool PN_DIAG and Rita Perria for the development of the MATLAB tool PN_DIAG_UNBOUNDED.

References

[1] F. Basile, P. Chiacchio, and G. De Tommasi. An efficient approach for online diagnosis of discrete event systems. *IEEE Trans. on Automatic Control*, 54(4), 2009.

[2] M.P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu. A New Approach for Diagnosability Analysis of Petri Nets Using Verifier Nets. *IEEE Trans. on Automatic Control*, 2012. to be published.

[3] M.P. Cabasino, A. Giua, M. Pocci, and C. Seatzu. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems. *Control Engineering Practice*, 19(9):989–1001, September 2011.

[4] M.P. Cabasino, A. Giua, and C. Seatzu. Diagnosability of bounded Petri nets. In *Proc. 48th IEEE Conf. on Decision and Control*, Shanghai, China, dec 2009.

[5] M.P. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9):1531–1539, 2010.

[6] S.L. Chung. Diagnosing pn-based models with partial observable transitions. *International Journal of Computer Integrated Manufacturing*, 12 (2):158–169, 2005.

[7] DISC Software Platform webpage. *http://www.disc-project.eu/software_platform.html.*

[8] M. Dotoli, M.P. Fanti, A.M. Mangini, and W. Ukovich. Fault detection of discrete event systems using Petri nets and integer linear programming. *Automatica*, 45:2665–2672, 2009.

[9] Stefan Haar. Qualitative diagnosability of labeled Petri nets revisited. In *Proc. 48th IEEE Conf. on Decision and Control*, Shanghai, China, dec 2009.

[10] S. Jiang, R. Kumar, and H. E. Garcia. Diagnosis of repeated/intermittent failures in discrete event systems. *IEEE Transactions on Robotics and Automation*, 19(2):310–323, 2003.

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

[12] S. Lai, D. Nessi, M.P. Cabasino, A. Giua, and C. Seatzu. A comparison between two diagnostic tools based on automata and Petri nets. In *Proc. IFAC WODES'08: 9th Work. on Discrete Event Systems*, pages 144–149, May 2008.

[13] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems*, 4(2):197–212, 1994.

[14] F. Lin, J. Markee, and B. Rado. Design and test of mixed signal circuits: a discrete event approach. In *Proc. 32rd IEEE Conf. on Decision and Control*, pages 246–251, 1993.

[15] M. Pocci PN_DIAG tool tool available at the webpage. *http://www.diee.unica.it/giua/TESI/09_Marco.Pocci/.*

[16] A. Madalinski, F. Nouioua, and P. Dague. Diagnosability verification with Petri net unfoldings. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 14(2):49–55, 2010.

[17] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[18] R. Perria, PN_DIAG_UNBOUNDED tool downloadable at the webpage. *http://www.disc-project.eu/PN_Diag_bounded_unbounded.zip.*

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

[20] UMDES library. *http://www.eecs.umich.edu/umdes/toolboxes.html.*

[21] T. Ushio, L. Onishi, and K. Okuda. Fault detection based on Petri net models with faulty behaviors. In *Proc. SMC'98: IEEE Int. Conf. on Systems, Man, and Cybernetics (San Diego, CA, USA)*, pages 113–118, October 1998.

[22] Y. Wen and M. Jeng. Diagnosability analysis based on T-invariants of Petri nets. In *Proc. IEEE Networking, Sensing and Control*, Tucson, Arizona, March 2005.