# Diagnosability Planning for Controllable Discrete Event Systems

**Hassan Ibrahim**[1] and **Philippe Dague**[1] and **Alban Grastien**[2] and **Lina Ye**[1] and **Laurent Simon**[3]
[1] LRI, Univ. Paris-Sud and CNRS, Univ. Paris-Saclay, Orsay, France
Email: firstname.name@lri.fr
[2] Data61 and Australian National University, Canberra, Australia
Email: alban.grastien@data61.csiro.au
[3] LaBRI, Univ. Bordeaux and CNRS, Bordeaux, France
Email: lsimon@labri.fr

## Abstract

In this paper, we propose an approach to ensure the diagnosability of a partially controllable system. Given a model of correct and faulty behaviors of a partially observable discrete event system, equipped with a set of elementary actions that do not intertwine with autonomous events, we search a diagnosability plan, i.e., a sequence of applicable actions that leads the system from an initial belief state (a set of potentially current states) to a *diagnosable* belief state, in which the system is then left to run freely. This helps in reducing the diagnosis interaction with running systems and can be applied, e.g., on the output of a repair plan, like in power networks. The two successive stages of this approach keep diagnosability planning, including diagnosability tests, in PSPACE in comparison to the EXPTIME test for the more complex active diagnosability used usually in such cases. For this, we propose to construct incrementally the twin plant structure of the given system and to exploit its parts already constructed while testing the candidate plans and constructing its next parts. This helps in pruning the twin plant constructions and many non-diagnosability plan tests. We have created a special benchmark and tested three proposed methods, according to the recycling level of twin plants construction, with one cost function used for plan optimality and an optional heuristics.

## Introduction

In the recent years, discrete event systems (DES) have been widely used to model and reason about large and complex systems because of their simplicity and ability to represent at a certain abstraction level real-world problems in various domains (Cassandras and Lafortune 2008). These systems are usually only partially observable and often subject to faults. In this context, diagnosability (the property that the faults can *always* be detected and identified (Sampath et al. 1995)) is an important property to guarantee.

We are interested in the problem of ensuring diagnosability of a partially observable DES. Our setting is different from the classical literature (cf. Section on Related Work). We assume that the system is partially controllable through "actions" that can change its state, but we want to minimize the interactions between the controller and the system, e.g., in an electrical power system, to reduce the outage time and the wear of expensive equipment. To this end, we assume that only one sequence of actions (a plan) is allowed to be performed on a paused system in order to bring it in a diagnosable state and let it run freely after that. Our problem therefore translates as follows: find a plan that leads the system from an initial given belief state to a belief state where diagnosability holds again. Typically, the initial belief could be the output of a previous operation like a repair plan following a fault occurrence identified thanks to previous diagnosability property, but suffering thus in general from uncertainty and from being different from the initial nominal state for which diagnosability was ensured by design.

We provide a formal definition of Diagnosability Planning. We analyze the problem and show that it is PSPACE-COMPLETE, compared to EXPTIME-COMPLETENESS of active diagnosability (where control is authorized during execution (Haar et al. 2013)). Then we propose a method to solve it: while searching for a plan, we build a twin plant (Jiang et al. 2001; Yoo and Lafortune 2002) to verify that the current belief state is diagnosable and we show how the twin plant constructed for a belief state can be partially reused for the diagnosability test of the next one and also prune useless plan tests. We present experimental results on a scalable benchmark. Finally we present related work and conclude with perspectives for future research.

## Diagnosability Planning in DES

### Discrete Event Systems

This work is done in the context of DES. We assume that the system runs in two distinct modes that do not intertwine: the active one, when the system runs freely, i.e., its states are changed autonomously through partially observable transitions without any controlled exogenous event, and the reactive one, in which only *feasible* exogenous actions are applied to change the system state through reactive transitions.

**Definition 1.** A **Controllable Labeled Transition System** (CLTS) is a tuple $G = \langle Q, \Sigma, \delta, I \rangle$, with $\Sigma = \mathcal{A} \cup \mathcal{E}$, where: $Q$ is a finite set of states; $\mathcal{E}$ is a finite set of events; $\mathcal{A}$ is a finite set of actions; $\delta \subseteq Q \times \Sigma \times Q$ is the (active for labels in $\mathcal{E}$, reactive for labels in $\mathcal{A}$) transition relation; $I \subseteq Q$ is the initial belief state.

$\mathcal{E}$ is partitioned into three finite sets: $\Sigma_o$ of observable correct events, $\Sigma_u$ of unobservable correct events and $\Sigma_f$ of un-

observable faulty events. We suppose that every state in $Q$ is actively reachable from a state in $I$. We assume completeness for reactive transitions, i.e., every action is applicable in every state (which can be easily made by modeling every action missing in a state $q$ as a loop transition in $q$).

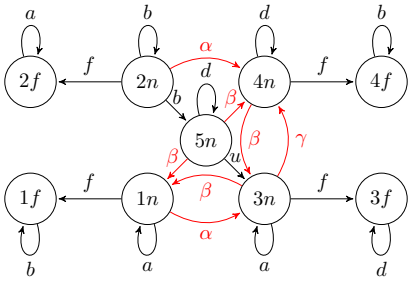


Figure 1: Illustrative example of a CLTS

Figure 1 shows a CLTS that comprises 9 states with $I = \{1n, 2n, 3n, 4n, 5n\}$, $\mathcal{A} = \{\alpha, \beta, \gamma\}$, $\Sigma_f = \{f\}$, $\Sigma_o = \{a, b, d\}$ and $\Sigma_u = \{u\}$. Any action that is missing in a state is assumed to leave it unchanged (for instance, applying $\beta$ in $1n$ leads to state $1n$). Notice that actions, here, do not affect the possibility of the fault occurrence.

## Diagnosability in DES

**Definition 2.** An **active path** (or path) is a sequence of active transitions $\rho = q_0 \xrightarrow{e_1} \cdots \xrightarrow{e_n} q_n$, with $e_1, \ldots, e_n \in \mathcal{E}$, $q_0, \ldots, q_n \in Q$, and $\forall i \in \{0, \ldots, n-1\}, (q_i, e_{i+1}, q_{i+1}) \in \delta$. We call $e_1 \ldots e_n \in \mathcal{E}^*$ the trajectory of $\rho$.

$L_I(G)$ will denote the prefix-closed language of $G$ from the initial belief state $I$, i.e., the set of words from $\mathcal{E}^*$ that are the trajectories of some active paths in $G$ that start from a state in $I$. Elements of $L_I(G)$ are called $I$-trajectories. In the following, $s^f$ denotes an $I$-trajectory ending by the fault $f$, $L_I(G)/s$ denotes the set of all extensions of $s$ as $I$-trajectories and $P$ is the projection of $\mathcal{E}^*$ on $\Sigma_o^*$. We denote the concatenation of two trajectories $s_1$ and $s_2$ by $s_1.s_2$.

**Definition 3.** A fault $f \in \Sigma_f$ is **diagnosable** in a system $G$ with initial belief state $I$ if

$$\exists k \in \mathbb{N}, \forall s^f \in L_I(G), \forall t \in L_I(G)/s^f, |t| \geq k \Rightarrow$$
$$\forall \rho \in L_I(G), (P(\rho) = P(s^f.t) \Rightarrow f \in \rho).$$

Definition 3 (Sampath et al. 1995) states that for each $I$-trajectory $s^f$, and each $t$ that is an extension of $s^f$ in $G$ with enough (depending only on $f$) events, every $I$-trajectory $\rho$ in $G$ that is equivalent to $s^f.t$ in terms of observations should contain in it $f$. As usual, it will be assumed that $L_I(G)$ is live (i.e., for any state, there is at least one active transition issued from this state) and convergent (i.e., there is no active cycle made up only of unobservable events) and $f$ is permanent.

A system $G$ is said to be diagnosable if and only if (iff) any fault $f \in \Sigma_f$ is diagnosable in $G$. We will consider only one fault type at a time, for keeping a linear complexity in the number of fault types, instead of an exponential one if they are all considered simultaneously. It will thus be assumed that there exists only one fault event $f$ ($\Sigma_f = \{f\}$), without restriction on the number of its occurrences. In the general case of $n$ fault types, we will run our algorithm for each tested plan on each fault type, and the plan is accepted iff it achieves the goal in all the $n$ runs. In our example, the initial belief state is not diagnosable as, from its states $1n$ and $2n$, an arbitrary long observable sequence of the event $a$ might represent a faulty or normal $I$-trajectory.

Diagnosability checking of LTS based on the twin plant method (Jiang et al. 2001; Yoo and Lafortune 2002) is known to be polynomial in the number $|Q|$ of states. The first step in this method is the construction from $G$ of a non-deterministic automaton, called pre-diagnoser, designed to preserve all observable information and to append to every state an estimate of failure information. In the pre-diagnoser, we only keep the observable events (the unobservable ones are eliminated as silent transitions) and attach the fault information to each retained state. If the fault has occurred up to a given state $q$ from an initial state, the fault label for $q$ is $f$. Otherwise, it is empty. For the example in Figure 1, in its pre-diagnoser, the fault label for all states ($1n$, $2n$, $3n$, $4n$, $5n$) in the initial belief state is empty since there is no fault occurrence. The fault label for all other states is $f$. The twin plant is then obtained as the product of the pre-diagnoser with itself to get all pairs of trajectories issued from initial states with the same observations. Each state of the twin plant is a pair of pre-diagnoser states that provide two possible diagnoses: if the fault $f$ is contained in exactly one of them, i.e., the occurrence of $f$ is not certain up to this state, it is called an ambiguous state w.r.t. $f$. An ambiguous state cycle is a cycle containing only ambiguous states. A *critical path* is a path in the twin plant issued from a pair of initial states with a prefix followed by an ambiguous state cycle.

**Lemma 1.** *(Jiang et al. 2001) A system is non-diagnosable iff its twin plant contains a critical path.*

In the twin plant of our example, $((1n, \emptyset)(2n, \emptyset)) \xrightarrow{a} ((1n, \emptyset)(2f, f)) \xrightarrow{a} ((1n, \emptyset)(2f, f))$ is a critical path since it is issued from a pair of initial states $1n$, $2n$ and contains a self cycle with an ambiguous state $((1n, \emptyset)(2f, f))$ associated with an observable event $a$. Thus, this system is not diagnosable since one can never be sure about the fault occurrence with an arbitrary number of $a$ observed.

## Planning

**Definition 4.** A **plan** $\pi$ for a CLTS $G$ is a sequence of actions $a_i \in \mathcal{A}$.

Applying the plan $\pi = a_1 \ldots a_m$ from a current belief state (set of states) $S \subseteq Q$ leads to the belief state $\pi(S)$ defined recursively by $\pi(S) = S$ if $m = 0$ and $\pi(S) = \pi'(S')$ if $m > 0$, where $S' = \{q' | \exists q \in S\ (q, a_1, q') \in \delta\}$ and $\pi' = a_2 \ldots a_m$. In our example, $\alpha\beta(I) = \beta(\alpha(I)) = \beta(\{3n, 4n, 5n\}) = \{1n, 3n, 4n\}$.

In planning, the objective is generally to find a plan $\pi$ that leads the system into a belief state that is included in one among a given set of "acceptable" states (e.g., states where the system is safe or where the system provides the service that we expect from it): $\pi(I) \subseteq B$ for some $B \subseteq Q$, $B$ acceptable. Here the definition of the objective is at the level of

a belief state instead of an individual state and moreover $B$ is not explicitly given however it is specified by a property.

**Definition 5.** A **planning problem** is a pair $\langle G, O\rangle$, where $G = \langle Q, \Sigma, \delta, I\rangle$ is a CLTS and $O \subseteq 2^Q$ is a collection of goal belief states. The solution to the planning problem is a plan $\pi$ such that $\pi(I) \subseteq B$, with $B \in O$.

### Problem Definition

In our problem, the objective of planning is to find a plan that leads the system to a diagnosable belief state.

**Definition 6.** Given a CLTS $G = \langle Q, \Sigma, \delta, I\rangle$, **diagnosability planning** is the problem of finding a plan $\pi$ such that $\langle Q, \Sigma, \delta, \pi(I)\rangle$ is diagnosable.

Diagnosability planning can be equivalently phrased as the problem of solving the planning problem $\langle G, O\rangle$, where $O$ is the set of maximal belief states from where $G$ is diagnosable: $O = \{I' \subseteq Q | \langle Q, \Sigma, \delta, I'\rangle$ is diagnosable and $I'$ maximal$\}$. Notice that this framework allows easily the specification of vacuous solutions that would ensure diagnosability but of no practical interest and thus from which one wants to stay away. We have just to remove from $O$ the corresponding states (which may be explicitly defined as well as implicitly by a property to be checked as done for diagnosability). Our goal is to find an optimal plan for a given criterion, that will be here to minimize the length (or cost) of the plan (number of actions in the sequence or sum of their costs). We denote by $\pi^\star$ an optimal plan. In our example, the plan $\pi^\star = \alpha\beta\beta$ is an optimal diagnosability plan that leads the system to the belief state $\pi^\star(I) = \{1n, 3n\}$ from which no critical path can be constructed.

## Complexity

In this section, we determine the complexity of the decision problem associated with diagnosability planning.

**Theorem 1.** *Diagnosability planning with explicit states representation is* PSPACE-COMPLETE.

**Hardness** can be shown by reducing classical propositional planning with succinct representation to Conformant Planning (i.e., with undeterminism of initial state and/or actions, thus belief-space planning as defined in definition 5) with eXplicit States (CPXS), and then CPXS to diagnosability planning. Classical propositional planning with succinct representation is known to be PSPACE-COMPLETE (Bylander 1994), which will give the result. A classical propositional planning is defined in a succinct way by a set of propositional variables and a set of actions. Each action has a precondition (subset of variables that must be $True$ for the action to be applicable) and two sets of positive/negative effects (variables that become $True/False$ upon the application of the action). Furthermore, the problem specifies the list of variables $True$ in the initial state and the list of variables $True$ in the goal states. The objective is to find a sequence that leads from the initial state to one goal state. For each variable $v$, we create two states $v_0$ and $v_1$ in the CPXS problem that represent respectively the fact that the variable $v$ has $False$ or $True$ assignment in the classical planning problem. A state in the classical planning problem is therefore represented by a belief state in the CPXS problem. The initial belief state and the final goal belief states are set accordingly. For any action $a$ and any variable $v$, we create in the CPXS problem a transition labeled by $a$ from $v_i$ to $v_j$ that models the effect of $a$ from the perspective of variable $v$, so according to the precondition/effects of $a$ and the semantics of $v_0/v_1$ described above. Thus, the solutions for the reduced CPXS problem are the same as the solutions for the original classical planning problem. The second reduction can be done by adding active transitions labeled by events, in such a way that no ambiguous state cycle can be reached from any CPXS goal belief state while any other belief state gives birth to an ambiguous state cycle. A solution plan to the CPXS problem corresponds thus to a diagnosability plan for the controllable DES obtained.

**Membership** to PSPACE has a proof similar to that for classical propositional planning. We iterate over all the possible belief states (PSPACE), verify that this belief state is diagnosable (PTIME), and search for a conformant plan from the initial belief state to this belief state (PSPACE).

This result has to be compared to the EXPTIME-COMPLETENESS of the active diagnosability decision problem with explicit states (Haar et al. 2013). And instead of having to synthesize an active diagnoser of exponential size w.r.t. the system size, a passive diagnoser is enough to recognize online the observable sequences that will signal the presence of a fault basing on its diagnosability fulfillment.

## Solving the Diagnosability Planning Problem

### Analyzing the Problem

Solving the Diagnosability Planning problem requires finding a plan that leads the system from a given belief state into a diagnosable belief state (goal belief state). This consists in alternating the generation of candidate plans and the diagnosability test of the belief state reached by each one. Thus, one must ensure the absence of any critical path issued from this final belief state. The traditional way to test diagnosability is just applying from scratch the twin plant approach: we call this approach the *Normal method*. Regarding the planning steps, the candidate plan is generated by traversing the candidates search space with an algorithm such as Breadth First Search (BFS) which ensures plan size minimality. The search space size can vary depending on the initial belief state and the type of available actions. For example, if we had a single state initial belief state with deterministic actions, then the worst case for a solution plan size would be $|Q|$. If the initial belief state contains more than one state or the actions are not deterministic, and we assume both cases in this paper, this worst case is $2^{|Q|}$.

From another hand, during the search of the intended plan, the different twin plants constructed starting from different belief states will generally share some states with each other. This makes interesting the idea of recycling previously built parts of these twin plants. In particular, if we find any critical path during the test of a candidate plan, we know that each time we will meet again its starting state, which is the root of this constructed twin plant, we will be sure to recover a crit-

ical path. This state represents a pair of states in the source belief state. Hence, the idea is to label such a pair as a *bad pair* after each test in order to avoid re-testing it later if it occurs in another planned belief state. More interestingly, these pairs can be used to prune the next twin plants construction and even to guide the search of a diagnosability plan.

**Definition 7.** We call $\{q_1, q_2\} \in Q \times Q$ a **bad pair** iff $((q_1, \emptyset), (q_2, \emptyset))$ is the starting state of a critical path in the twin plant of $G$. If $q_1 = q_2$, then it is called a bad unit. We denote the set of all currently known bad pairs as $\mathcal{B}$.

Actually, when a critical path is discovered, not only its starting state but all its non-ambiguous states (i.e., before the fault occurrence) give rise to bad pairs. Notice that the system $G = \langle Q, \Sigma, \delta, I \rangle$ is not diagnosable iff $I \times I$ contains a bad pair. So, for a current state of knowledge, a sufficient condition for the non-diagnosability of $G$ is $(I \times I) \cap \mathcal{B} \neq \emptyset$.

In a similar way if the diagnosability test failed to find any critical path issued from a non-ambiguous pair of states we call this pair a *good pair*.

**Definition 8.** We call $\{q_1, q_2\} \in Q \times Q$ a **good pair** iff there is no critical path in the twin plant of $G$ issued from $((q_1, \emptyset), (q_2, \emptyset))$. If $q_1 = q_2$, then it is called a good unit. We denote the set of all currently known good pairs as $\mathcal{G}$.

Actually, when no critical path exists, not only the starting state of the twin plant but all its non-ambiguous states give rise to good pairs. Notice that the system $G = \langle Q, \Sigma, \delta, I \rangle$ is diagnosable iff $I \times I$ contains only good pairs. So, for a current state of knowledge, a sufficient condition for the diagnosability of $G$ is $I \times I \subseteq \mathcal{G}$.

In our example, $\{1n, 2n\}$ is a bad pair, since from it a critical path can be constructed as shown before, and $\{1n, 3n\}$ is a good pair, because no critical path is issued from it.

One natural way to build the twin plant for any belief state $S$ is to process $S$ globally by reducing it to only one virtual initial state $q^S$ related by unobservable transitions to any state in $S$ and to take into account learned labeled pairs while constructing the traditional twin plant from $((q^S, \emptyset), (q^S, \emptyset))$. We call this approach the *Lazy Learning method*, as it will learn only bad pairs and so will not exploit the good pairs. Nevertheless, this approach may lead to small size constructed twin plants by concentrating only on bad pairs. Another way to a better exploitation of the labeled pairs is to construct the twin plants for each pair of states in the belief state, i.e. for each element of $S \times S$. Thus we stop the process immediately if we meet any known bad pair and, before this, we prune developing any branch of the twin plants each time we meet a known good pair. We call this approach, allowing learning both good and bad pairs and a useful recycling of both, the *Eager Learning method*.

## Learning and exploiting Bad and Good pairs

**In Diagnosability Test:** As said above, meeting a known bad pair during a twin plant construction predicts the existence of a critical path and allows us to stop this construction and learn at least one new bad pair (the one corresponding to the starting state of the twin plant). Many other bad pairs can actually be learned in general, corresponding to all non-ambiguous states in the critical path discovered. Moreover, we can avoid testing any candidate plan that leads to a belief state that contains a known bad pair. The *Lazy Learning method* concentrates only on exploiting those bad pairs. But the *Eager Learning method* exploits also the good pairs generated after each twin plant construction to guide the plan generation and to prune the further parts of twin plants construction. Discovering a good pair means that the twin plant constructed starting from this pair does not contain any critical path. This can be exploited in two ways. The first is that each non-ambiguous state in the constructed twin plant represents a good pair, that can be learned. The second is that in any next twin plant, meeting a good pair allows pruning the construction of branches from this state (construction in other branches has to be continued). Moreover, for a candidate plan, we can avoid testing all known good pairs in its reached belief state.

**In Planning:** We can use all discovered pairs in guiding a greedy algorithm for the plan generation. Let call $g(\pi)$ the cost of plan $\pi$, which is the sum of its elementary actions costs (e.g., its length if all actions have cost 1), that we want to minimize. The idea is to order the possible plans according not only to their cost but also to the chance of reaching a diagnosable belief state, heuristically evaluated by the ratio of good pairs in this belief state, denoted by $h(\pi)$. For this, we classify, in a priority queue, the candidate plans $\pi$ (those obtained by adding one action to a previous failed candidate plan) by partitioning $\pi(I) \times \pi(I)$ into, a structure of three classes $\langle \mathcal{B}_\pi, \mathcal{G}_\pi, \mathcal{U}_\pi \rangle$, which represent respectively known bad, known good and unlabeled pairs. We rank first those $\pi$ with $\mathcal{B}_\pi = \emptyset$ and the best candidate plan, i.e., the one that optimizes the global objective function (combination of $g(\pi)$ and $h(\pi)$ minimizing the first and maximizing the second) is chosen among them. Then we test for diagnosability $\mathcal{U}_\pi$ for the plan $\pi$ chosen, by the same previous iterative twin plant construction. The new bad and good pairs discovered at this occasion are used to update or compute the classes $\langle \mathcal{B}_\pi, \mathcal{G}_\pi, \mathcal{U}_\pi \rangle$ for the next candidate plans $\pi$ considered in order to proceed to their ranking. The process is repeated until finding a diagnosability plan or proving its absence. The labeled pairs (good and bad) are propagated into all the classes, so any pair is never tested more than once with the *Eager Learning method* and is used to prune or stop the construction of the further twin plants.

One can learn three things by observing iteratively this structure: the first is the existence of a diagnosability plan $\pi$ that can be directly deduced if ($\mathcal{B}_\pi = \emptyset \wedge \mathcal{U}_\pi = \emptyset$), the second is the absence of a diagnosability plan deduced if no more candidate plans exist and the third is that, if none of the two stopping conditions is satisfied, the next possible candidate to test is the one on the top of the priority queue.

## General algorithm

We describe the algorithm 1 which contains the general procedure that learns bad and good pairs and uses them to prune the search space, i.e., the *Eager Learning method*. This pruning is twofold: pruning the search space of the candidate plans and pruning the construction of the twin plants.

The procedure *genCandidatePlan* closes candidate plans

**Algorithm 1** General algorithm

**procedure** FINDDIAGNOSABILITYPLAN($G, I, g, h$)
  $\pi \leftarrow \emptyset$; $\Pi \leftarrow \emptyset$; $EBS \leftarrow \{I\}$; $\mathcal{B} \leftarrow knownBad$; $\mathcal{G} \leftarrow knownGood$
  ▷ Initialization of the last candidate explored, the set of current candidates, the belief states explored, the bad and good pairs (known from test of $G_I$)
  $\Pi \leftarrow genCandidatePlan(\Pi, \pi, EBS, G, I, g)$
  **while** $\Pi \neq \emptyset$ **do**
    $\pi \leftarrow getBestCandidate(\Pi, \mathcal{B}, \mathcal{G}, g, h)$
    $EBS \leftarrow EBS \cup \{\pi(I)\}$
    $Pairs \leftarrow getAllPossiblePairs(\pi(I))$
    **while** $(Pairs \cap \mathcal{B} = \emptyset) \wedge (Pairs \setminus \mathcal{G} \neq \emptyset)$ **do**
      $\{q_1, q_2\} \leftarrow chooseOne(Pairs \setminus \mathcal{G})$
      $DG^{\pi}_{(q_1,q_2)} \leftarrow getPreDiag(G_{\{q_1,q_2\}})$
      $TP^{\pi}_{(q_1,q_2)} \leftarrow getTP(DG^{\pi}_{(q_1,q_2)}, \Sigma_o, \mathcal{B}, \mathcal{G})$
      ▷ $\mathcal{B}$ and $\mathcal{G}$ are used to prune TP building
      $newBad \leftarrow CPPairs(TP^{\pi}_{(q_1,q_2)})$
      **if** $newBad \neq \emptyset$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} \cup newBad$
        ▷ unambiguous states of the found critical path give bad pairs
      **else**
        $\mathcal{G} \leftarrow \mathcal{G} \cup StatesOf(TP^{\pi}_{(q_1,q_2)})$
        ▷ all TP states give good pairs
    **if** $Pairs \setminus \mathcal{G} = \emptyset$ **then**
      **return** $\pi$
      ▷ $\pi$ is a diagnosability plan
    **else**
      $\Pi \leftarrow genCandidatePlan(\Pi, \pi, EBS, G, I, g)$
  **return** $\emptyset$
  ▷ there is no diagnosability plan
**procedure** GETBESTCANDIDATE($\Pi, \mathcal{B}, \mathcal{G}, g, h$)
  $\Pi \leftarrow sortByCostandGoodness(\Pi, \mathcal{B}, \mathcal{G}, g, h)$
  ▷ those $\rho \in \Pi$ with $(\rho(I) \times \rho(I)) \cap \mathcal{B} \neq \emptyset$ are the last in the sort; others are sorted first according to cost function $g$ and $\mathcal{G}$-based heuristics $h$
  $\rho \leftarrow removeTop(\Pi)$
  **return** $\rho$
**procedure** GENCANDIDATEPLAN($\Pi, \pi, EBS, G, I, g$)
  $CurrentActions \leftarrow Actions(G)$; $new\Pi \leftarrow \emptyset$
  **while** $CurrentActions \neq \emptyset$ **do**
    $a \leftarrow removeOne(CurrentActions)$
    $\rho \leftarrow \pi a$
    **if** $\nexists BS \in EBS \mid BS \subseteq \rho(I)$ **then**
      ▷ if $\rho(I)$ contains an already explored belief state, then $\rho$ is closed
      $new\Pi \leftarrow new\Pi \cup \{\rho\}$
  **while** $new\Pi \neq \emptyset$ **do**
    $\rho \leftarrow removeOne(new\Pi)$
    **if** $\nexists \rho' \in new\Pi \mid \rho'(I) \subseteq \rho(I) \wedge g(\rho') \leq g(\rho)$ **then**
      ▷ if a plan candidate $\rho'$ is better than $\rho$, then $\rho$ is closed
      $\Pi \leftarrow \Pi \cup \{\rho\}$
  **return** $\Pi$

that lead to a belief state which is a superset of the belief state of a previously explored plan or of another candidate plan generated with a lower cost. Therefore no candidate plan will be generated if all nodes are closed, which is the stopping criterion for the algorithm (independently of the use or not of an heuristics $h$), meaning the nonexistence of a diagnosability plan. Then, the procedure *getBestCandidate* ranks the candidate plans according to their costs $g$ (i.e., using BFS for example if all elementary actions have the same cost) and optionally favoring the goodness-based heuristics $h$ (ratio of good pairs in the belief state), and ranking last those whose belief space contains a bad pair. It returns the best candidate plan $\pi$ for this ranking, which gives the current belief state $\pi(I)$ to be tested for diagnosability. Then if $\pi(I)$ does not contain any known bad pair and contains at least one unlabeled pair, iteration is done on these *unlabeled* pairs of states to check their goodness. That is, a pair $\{q_1, q_2\}$ is chosen and (part of, in the light of known labeled pairs) the twin plant $TP^{\pi}_{(q_1,q_2)}$ of the subsystem $G_{\{q_1,q_2\}}$ (having $q_1$ and $q_2$ as initial states) is built. If a critical path (which may be predicted just by meeting a bad pair) is found in $TP^{\pi}_{(q_1,q_2)}$, all the pairs of states corresponding to the new non-ambiguous states of this critical path (so, at least $\{q_1, q_2\}$) are added to the bad pairs list. In case $TP^{\pi}_{(q_1,q_2)}$ does not contain any critical path, all pairs represented by its non-ambiguous states are learned as good pairs and the iteration continues. The iteration stops either because all pairs of $\pi(I) \times \pi(I)$ are found to be good, and then $\pi$ is a diagnosability plan (necessarily optimal for the cost $g$ if $h$ has not been used), which ends the algorithm, or because a bad pair is found in $\pi(I)$. In this last case, $\pi$ is failed and its candidate successors are generated by *genCandidatePlan* and the next best candidate plan is chosen by *getBestCandidate*. So the algorithm always terminates and returns a diagnosability plan if it exists (guaranteed optimal if $h$ is not used, the role of $h$ is to guide the search in order to get a solution in less steps at the expense of its optimality).

Notice that handling multiple faults is straightforward. For each candidate plan, we build successively twin plants for each unlabeled pair and for each fault (all other faults being thus considered as unobservable correct events) until a critical path is found for one pair and one fault (and we learn bad pairs, and possibly good pairs for successful tries before) or all pairs are diagnosable for all faults, i.e., we found a diagnosability plan. Good pairs have to be stored with their related fault for being reused properly (differently from bad pairs which still bad even for at least one fault).

## Experimental Results

In order to test our proposed approaches on a benchmark, we created a rectangular grid of components by repeating the active model (i.e., without its actions) of our running example in Figure 1 and we reconfigured the actions in all components and added global actions between components. We defined two actions models that are applied to a component according to its position in the grid (given by line index $i$ and column index $j$, with the origin at the left top corner). The first one is adopted for all top and bottom border components and allows the planner, for an initial belief state chosen inside one of these components, to find a short plan, of length two or three, made up of local actions inside this component. The second one is adopted for all other (internal) components and does not allow reaching a diagnosable belief state inside this component, but allows reaching a belief state inside the component just below or above it by using global actions that connect any internal component with its two neighbors on the same column. We designed the actions in order the diagnosability plan to be, starting from any internal component belief state, the one obtained

by selecting "downward" global actions to the successive below neighbors until reaching the bottom line of the grid. Moving upward is also always an option but will not give a diagnosability plan. We also connect actively by event transitions each component with its (at most four) neighbors, by adding an observable communication event $c$ that connects each state $3n$ (resp. $1n$) at the position $(i, j)$ in the grid to the state $2n$ (resp. $4n$) at the position $(i+1, j)$, each state $3f$ (resp. $4f$) at the position $(i, j)$ to the state $1f$ (resp. $2f$) at the position $(i, j+1)$ and each state $1f$ (resp. $2f$) at the position $(i, j)$ to the state $3f$ (resp. $4f$) at the position $(i, j-1)$.

We have tested three search algorithms of the intended diagnosability plan. The first algorithm uses the *Normal method*, i.e., without any recycling. The second algorithm uses the *Lazy Learning method* which concentrates on learning only the bad pairs, which it later uses to prune another plan test or to stop another twin plant construction if it meets such a pair again. The third algorithm uses the *Eager Learning method* (the one described Algorithm 1).

Concerning the search strategy, firstly we applied to the planner the BFS strategy (i.e., without $h$ and with equal elementary actions costs in $g$) which guarantees optimal length of the plan, and tested the three methods. Secondly we applied our greedy strategy, limited to the use of the only heuristic function $h$ (without $g$), computed from the learned pairs and maximizing the ratio of good pairs in the belief state reached by the chosen candidate plan. This approach is applicable to the third method, the only one to have the information about good pairs but we adapted it to the second method by minimizing the number of unlabeled pairs.

We have first tested the 5 different methods on increasing grid heights (with width 3) with the initial belief state made up of the five normal states in the central component of the grid. Figure 2 shows the efficiency of recycling learned pairs even if they are not exploited in guiding the planning step. For example, the *Eager Learning method* is 15 times faster, for a grid of height 50 (the length of the optimal diagnosability plan being 74), than the *Normal method*, that tests diagnosability without any learning. But the benefit of Using the greedy heuristics in the two learning methods appears clearly even if the *Eager Learning method* does not give an optimal diagnosability plan (actually it scales well up to a $180 \times 3$ grid with a plan of length 355).

The tests reported Figure 3 are more challenging as we suppose an initial belief state composed of the normal states of two scattered internal components. The largest the height of the grid (with width 5), the farthest they are. The search space of the plan is much bigger, but still, even without guiding the plan search by learned pairs, the performances of the two learning methods are better than the performance of the *Normal method* which explodes for the height 17. Results improve dramatically when exploiting the learned pairs in guiding the planning search and the *Eager Learning method* scales well up to a $100 \times 5$ grid. However, the greedy heuristics used here does not return an optimal length of the intended plan as does BFS strategy. But we can notice that it is very close to the optimal one for the *Lazy Learning method* (which is not the case for the *Eager Learning method*).

Finally, in order to show how changing the size of the initial belief state can affect the results, we fixed the size of the system to a $10 \times 10$ grid and incrementally increased the size of the initial belief state by adding at each increment $i$ the five normal states of the component at the position $(i, i)$, so up to 45 states. Then, the *Normal method* explodes and we continued adding randomly normal states from other components to the initial belief state to compare the *Eager* and *Lazy Learning methods*. As shown Figure 4, the first one scales up to the maximum of 500 states (i.e., the normal states of all the 100 components of the grid, representing complete uncertainty on the belief state).

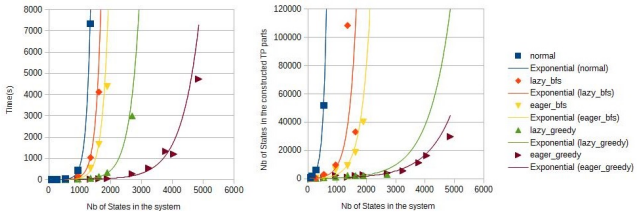


Figure 2: $I$ contains the normal states of one central component.

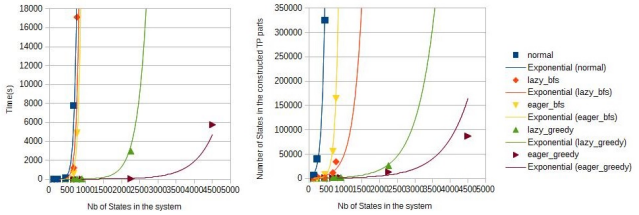


Figure 3: $I$ contains the normal states of two scattered internal components.

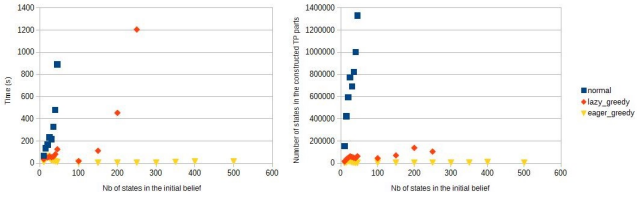


Figure 4: Changing initial belief state size in a fixed system of $(10 \times 10)$ components.

## Related Work

In (Sampath et al. 1995), the authors introduced the first definition of diagnosability for DES and proposed a necessary and sufficient condition for testing it by constructing a deterministic diagnoser. The main drawback is its exponential space complexity in the number of system states. Then, the authors of (Jiang et al. 2001; Yoo and Lafortune 2002) proposed new algorithms with polynomial complexity, which is the classical twin plant method. After that, distributed approaches to solve diagnosability problem based on twin plant have been investigated (Pencolé 2004; Schumann and Huang 2008; Schumann and Pencolé 2007). However, all of them return only information about whether the systems are diagnosable or not, and can do nothing for non-diagnosable systems. But diagnosability is a quite strong property, which is often not satisfied in real systems.

On the other hand, planning techniques have been developed over the last decades, whose idea is to find a plan satisfying the desired goals. In (Barveau, Kabanza, and St-Denis 1998), a synthesis method was presented that automatically generates controllers on timed transition graphs, where the specification of control requirements is expressed by metric temporal logic (MTL) formulas. However, the authors assumed the full observability, i.e., every state variable can be observed at each step. Considering that planning domains are often partially observable and non-deterministic, new approaches for planning under partial observability, dealing with uncertainty about the states, were proposed in (Bertoli et al. 2001; 2006). The search space is no longer the set of states of the domain, but its power-set. A close work was presented in (Ciré and Botea 2008), where a goal state represented in LTL is calculated and verified on the fly. In their model, all transitions are deterministic while they are non-deterministic in ours. Furthermore, their online learning component has, for each step, to construct a Boolean formula that can explain the violation of the considered property, which could be quite complex since different rules have to be applied at the atomic level.

The work in (Grastien 2015) addressed the self healing as a combination between conformant planning and diagnosis steps to repair the system without explicitly computing its belief state. Given the goal states, an optimal plan is computed for a sample of the belief state that leads to a goal state, then the plan is refined depending on the result of a special diagnoser that tries to find a behavior of the system which contradicts the current plan. Once a behavior is found, it can enrich the sample to recompute a better plan. Our work can be seen as a continuation of this work, where the goal states are described implicitly by the diagnosability property.

## Conclusion and Future Work

In this paper, we defined the problem of Diagnosability Planning before demonstrating that it is PSPACE-COMPLETE. We provided an algorithm to search for an optimal diagnosability plan: the twin plant is incrementally constructed by updating the set of learned bad and good pairs, which helps in pruning its further construction and in defining an heuristics to guide the plan search. Experimental results demonstrated the efficiency of the approach by exploiting the different learning strategies (code and benchmark are available upon request to the first author). Considering more informative heuristics (if possible finding an admissible one, but this is a difficult problem with non-explicit goals) and encoding the whole problem in SAT (already done for the diagnosability test) is our current work. We will extend our approach to bounded $k$-diagnosability and to other twin plant-based properties, such as predictability (Genc and Lafortune 2009). Merging successive repair and diagnosability plans construction will be also studied to achieve global optimality for these two tasks together.

## References


Barveau, M.; Kabanza, F.; and St-Denis, R. 1998. A method for the synthesis of controllers to handle safety, liveness, and real-time constraints. *IEEE Transactions on Automatic Control (TAC)* 43(11):1543–1559.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, 473–478. Morgan Kaufmann Publishers Inc.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2006. Strong planning under partial observability. *Artif. Intell.* 170(4-5):337–384.

Bylander, T. 1994. The computational complexity of propositional strips planning. *Artificial Intelligence* 69(1):165–204.

Cassandras, C. G., and Lafortune, S. 2008. *Introduction To Discrete Event Systems, Second Edition*. Springer.

Ciré, A., and Botea, A. 2008. Learning in planning with temporally extended goals and uncontrollable events. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, 578–582.

Genc, S., and Lafortune, S. 2009. Predictability of Event Occurrences in Partially-observed Discrete-event Systems. *Automatica* 45(2):301–311.

Grastien, A. 2015. Self-healing as a combination of consistency checks and conformant planning problems. In *Proceedings of the 26th International Workshop on Principles of Diagnosis (DX'15)*, 105–112.

Haar, S.; Haddad, S.; Melliti, T.; and Schwoon, S. 2013. Optimal constructions for active diagnosis. In *Proceedings of the 33rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13)*, 527–539.

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.

Pencolé, Y. 2004. Diagnosability Analysis of Distributed Discrete Event Systems. In *Proceedings of the 16th European Conference on Articifial Intelligent (ECAI04)*, 43–47. Nieuwe Hemweg: IOS Press.

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

Schumann, A., and Huang, J. 2008. A Scalable Jointree Algorithm for Diagnosability. In *Proceedings of the 23rd American National Conference on Artificial Intelligence (AAAI-08)*, 535–540. Menlo Park, Calif.: AAAI Press.

Schumann, A., and Pencolé, Y. 2007. Scalable Diagnosability Checking of Event-driven System. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI07)*, 575–580. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Inc.

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