Diagnosis of hybrid systems using Hybrid Particle Petri nets: theory and application on a planetary rover

Quentin Gaudel, Elodie Chanthery, Pauline Ribot, Matthew J. Daigle

Diagnosis of hybrid systems using Hybrid Particle Petri nets: theory and application on a planetary rover

Quentin Gaudel and Elodie Chanthery and Pauline Ribot and Matthew J. Daigle

Abstract This chapter presents a new methodology to perform health monitoring of hybrid systems under uncertainty. Hybrid systems can be represented as multi-mode systems with hybrid automata. Diagnosers are generated from these hybrid automata using a new data structure in order to monitor both the behavior and degradation of such systems. After a review of the state of the art on different existing solutions for diagnosis of hybrid systems under uncertainty, we propose to introduce the Hybrid Particle Petri Nets (HPPN) modeling framework. The main advantage of HPPN is that they take into account knowledge-based uncertainty in the system representation and uncertainty in the diagnosis process. The HPPN-based diagnoser deals with occurrences of unobservable discrete events (such as fault events) and it is robust to false observations. It also estimates the continuous state of the system by using particle filtering. A methodology is proposed to perform model-based diagnosis on hybrid systems by using the HPPN modeling framework. The system diagnosis is computed at any time from a HPPN-based diagnoser and contains all the hypotheses over its past mode trajectory. Each hypothesis is valued with a belief degree and includes discrete and continuous state estimates, as well as the set of faults that occurred on the system up to the current time. The HPPN-based methodology is demonstrated with an application on the K11 planetary rover prototype developed by NASA Ames Research Center. A hybrid model of the K11 is proposed and experimental results show that the approach is robust to real system data and constraints.

Q. Gaudel
EasyMile SAS, Toulouse, France, e-mail: `quentin.gaudel@easymile.com`

E. Chanthery and P. Ribot
LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse, France, e-mail: `{elodie.chanthery}@laas.fr,pauline.ribot`

M.J. Daigle
NIO USA, Inc., San Jose, CA, USA, e-mail: `matthew.daigle@nio.io`

1 Introduction

Real systems have become so complex that it is often impossible for humans to capture and explain their behaviors as a whole, especially when they are exposed to failures. System health management (SHM) or prognostics and health management (PHM) aims at developing tools that can support maintenance and repair tasks, reducing the global costs due to unavailability and repair actions, but also optimizing the mission reward by replanning or reconfiguring the system [Sweet et al. (2014)]. An efficient health monitoring technique has to be adopted to determine the health state of the system at any time by using diagnostics and prognostics techniques. A diagnosis method is used to determine the current health state and identify the possible causes of failures that lead to this state by reasoning on observations. Prognosis is used to predict the future health states and the times of the occurrences of the faults that lead to these states. Hybrid systems have been defined by Henzinger (1996) as follows.

Definition 1 (Hybrid System). A *hybrid system* exhibits both discrete and continuous dynamics.

Sensor data and commands are designated as continuous or discrete observations on the system. Hybrid systems are usually described as a multi-mode system composed of an underlying discrete-event system (DES) representing the mode changes and various underlying continuous dynamics associated with each mode [Bayoudh et al. (2008)].

Definition 2 (Discrete State, Continuous State). The system *discrete state* is defined as the current discrete state of the DES. The evolution of the system *continuous state* depends on continuous dynamics associated with the current system mode.

In most industrial systems, if the degradation is not observable, it is estimated as fault occurrence probabilities. The degradation thus depends on the stress level of the current health mode of the system and, in some cases, also relies on the current continuous state and also on the analysis of the events that occurred on the system [Gaudel et al. (2015)]. Because of these dependencies and its importance for PHM, we choose to evaluate the degradation separately from the discrete state and the continuous state of the system.

Definition 3 (Degradation state). The system *degradation state* is the current value of the degradation whose evolution is represented by degradation dynamics.

We extend the multi-mode system by associating underlying degradation dynamics (e.g. degradation laws) with each mode.

Definition 4 (Mode, Event, State). A *mode* is defined as a combination of a discrete state of the DES with continuous dynamics and degradation dynamics. The changes of modes are associated with occurrences of discrete *events*. The *state* of the hybrid system is defined as the combination of its discrete, continuous and degradation states.

Our previous works introduced a framework called *Hybrid Particle Petri Nets* (HPPN). Gaudel et al. (2014a) proposed to use HPPN to model an uncertain hybrid system and track its current health state by generating a diagnoser. The methodology uses information about the system degradation that is a significant advantage to compute a more accurate diagnosis and to perform prognosis. In Gaudel et al. (2015), we tested the proposed approach on a simulated three-tank system.

This chapter presents in detail the HPPN-based health monitoring method exposed in Gaudel et al. (2015) that has been improved concerning computation performance. The method is hence recalled and new notions are precised, such as the definition of discrete events, the calculation of mode scores or the choice of scale parameters for the diagnoser process. This chapter then exposes results of the implemented health monitoring method on the K11 planetary rover prototype. The K11 is a testbed developed by NASA Ames Research Center which is used for diagnostics and prognostics purposes [Balaban et al. (2013); Daigle et al. (2014); Sweet et al. (2014); Daigle et al. (2015b)]. A hybrid model of the rover is proposed, based on the discretization of its health evolution. Experimental results are given, illustrating how the methodology is robust to real system data and constraints.

The chapter is organized as follows. Section 2 presents related works on diagnosis of hybrid systems. Section 3 recalls and deepens the health monitoring methodology based on the modeling of the hybrid system and the generation of a diagnoser in the HPPN framework. Section 6 focuses on the application of the proposed methodology on the K11 planetary rover prototype. It provides the K11 hybrid model and exposes the experimental results and performance metrics. Conclusions and future works are discussed in the final section.

2 Related Work

Hybrid systems are a very important subject in many fields, such as modeling, verification, control, and monitoring.

Some models, initially purely continuous have been extended with the integration of events [Narasimhan and Biswas (2007)]. Similarly, discrete event models have been extended with some continuous aspects, such as Continuous Petri Nets (CPN) [David and Alla (2005)] and Hybrid Petri Nets (HPN) [Zaidi et al. (2006); Dotoli et al. (2008)], which introduce a new type of place (continuous place) with a rational marking. Finally, some other hybrid models have been build by the explicit combination of a discrete event model and a continuous model, such as Hybrid Automaton [Bayoudh et al. (2008); Hofbaur and Williams (2002)] or particle Petri nets [Lesire and Tessier (2005)]. All of these models have been widely used and extended for monitoring hybrid systems.

Zhao et al. (2005) propose an approach based on timed Petri nets and mode estimation. The Petri nets use is justified by significant computational advantages over concurrent automata. Fault detection and estimation are done sequentially. Uncertainty about discrete events is not considered. Bayoudh et al. (2008) model the system with a hybrid automaton. The hybrid system is described as a multi-mode system in which each mode is associated with a continuous dynamic. These works then exploit the analytical redundancy relations of the continuous models and the parity space approach to generate a DES diagnosis that recognizes the signatures associated with each operational modes. Horton et al. (1998) introduce fluid stochastic Petri nets. The arcs of a fluid stochastic Petri net carry fluid flow which limits the passage of tokens, and create continuous marking. However, the continuous dynamics is limited to a speed and is not appropriate to represent any hybrid system. Lesire and Tessier (2005) combine a discrete event model (Petri nets) and a continuous model (dynamic equations) in an extension of the hybrid Petri nets called Particle Petri Nets (PPN). They propose to distribute the rational marking of the continuous spaces in a set of particles. The tokens of the discrete places (named configurations) and these particles are then used in a monitoring mechanism combining the possibilistic firing [Cardoso et al. (1999)] with a particle filter to manage all the uncertainties relative to the system and discrete and continuous observations. This work is oriented towards mission monitoring, not health monitoring.

The Modified Particle Petri Nets (MPPN) formalism [Zouaghi et al. (2011)] is proposed as an extension of the PPN. The main advantage of MPPN is that they propose to use transitions associated with conditions that deal with both the configuration and particle values. The application is essentially oriented towards mission monitoring, not health management. The different health states of the system are not considered. Moreover, there is no correspondence with the diagnoser object defined in the literature and the problem of ambiguity in the model is not addressed. The diagnoser approach was introduced in Sampath et al. (1995). The diagnoser is basically a monitor that is able to process any possible observable event that occurs in the system. It consists in recording these observations and providing the set of possible faults whose occurrence is consistent with the observations. However, this approach is restricted to DES and does not manage uncertainty. Some approaches extend the diagnoser to DES modelled by Petri nets. However, none of these approaches take into account continuous aspects, nor consider uncertainty in the system. In Soldani et al. (2007), an approach for the localization of intermittent faults by dealing with partial observability in the discrete event framework is proposed. The method is based on Petri nets that model the normal functioning of the system observable behavior. A localization mechanism, based on the diagnoser approach, points out the set of events potentially responsible for the faults.

Some studies are particularly focused on the diagnosis of systems with the intention of using it for prognosis purposes. These approaches consider monitoring the degradation of the system. This is called the advanced diagnosis.

In Vianna and Yoneyama (2015), the diagnosis method uses an extended Kalman filter and an Interactive Multiple-Model (IMM) algorithm to monitor both the behavior of the system and its degradation in order to obtain a better system state estimation as a starting point for the prognosis process. However, the approach is limited to continuous systems.

Chanthery and Ribot (2013) propose to extend the diagnosis approach proposed in Bayoudh et al. (2008) by associating to each mode of the hybrid system a degradation dynamics. However, dynamics are limited to aging laws which estimate the probabilities of occurrences of anticipated faults. The approach does not take into account the uncertainties on the system model and the observations, both for the continuous and discrete part.

In Yu et al. (2011), fault isolation is performed dynamically with a Hybrid Bond Graph (HBG). The method proposes to use a fault signature matrix for each mode and introduces a delay to allow each fault to express its symptoms on the residuals (especially the only detectable faults with the continuous signals). However, no indication of the waiting time is given. In parallel with the monitoring of the evolution of these new faults, the degradations of each component depend on the current operational mode and are estimated with a hybrid differential evolution algorithm.

This paper focuses on the application of the health monitoring methodology on the K11 rover, that is subject to inherent uncertainty of real systems.

Uncertainty has been widely studied for state estimation of continuous systems. Concerning hybrid systems, Koutsoukos et al. (2002) use a particle filtering technique to estimate the state of a hybrid system modeled as a hybrid automaton. Uncertainty related to discrete events is not taken into account and the system degradation is not considered. In Narasimhan et al. (2004), a consistency-based approach combined with particle filters is proposed. Noise and uncertainty are taken into account, but only discrete faults are addressed. Biswas et al. (2003) and Wang et al. (2007) both propose a robust state estimation and fault diagnosis for uncertain hybrid nonlinear systems, where the discrete dynamics has unknown transition functions. However, they only consider discrete faults. Ru and Hadjicostis (2009) use partially observed Petri nets. Partially observed Petri nets are transformed into an equivalent labelled Petri net and an online monitor is built to diagnose faults and provide beliefs (degrees of confidence) regarding the occurrences of faults. However, this approach is limited because it only takes into account uncertainty in the diagnosis results, not about the model or the event observations. Basile et al. (2009) propose to reduce the explosion of the state space by introducing generalized markings (negative tokens) to take into account uncertainty about the firing of transitions. The stochastic Petri nets are used by Jianxiong et al. (2013) to build a formal model of each component of an integrated modular avionics architecture. However, for all these approaches, no continuous aspect in the model is taken into account.

In previous works, health monitoring and diagnosis was applied to the K11 rover. In Narasimhan et al. (2012), two diagnosis algorithms were applied, Qualitative Event-based Diagnosis (QED) [Daigle et al. (2015a)], and the Hybrid Diagnosis Engine (HyDE) [Narasimhan and Browston (2007)]. QED performs diagnosis based on reasoning over symbols representing qualitative deviations of the sensor signals with respect to model-predicted values. Sensor and process noise are handled by using an observer to estimate the current system state, however no uncertainty in the symbols computed for diagnosis is considered, and all diagnostic hypotheses are viewed as equally likely. HyDE is a consistency-based diagnosis engine that uses hybrid and stochastic models and reasoning. Reasoning is performed by hypothesizing alternative system trajectories inferred from the transition and behavior models of the system, and considers a priori fault probabilities and mode transition probabilities. Both diagnosis algorithms were used to diagnose parasitic load, motor friction, and voltage sensor faults in simulation. In Sweet et al. (2014), QED diagnosed parasitic load faults and voltage sensor faults in real-world scenarios.

3 Health Monitoring Methodology for Hybrid Systems

This section details the methodology proposed in Gaudel et al. (2015) to perform model-based health monitoring of hybrid systems.

We are interested in modeling changes in system dynamics when one or several anticipated faults occur.

Definition 5 (Health Modes). The *health modes* are the hybrid system modes (a discrete state with continuous dynamics and degradation dynamics) and represent different health conditions.

Definition 6 (Nominal mode). As long as the system does not encounter any fault, it is in a *nominal mode*.

Definition 7 (Degraded mode). Tracked faults are assumed to be permanent, i.e. once a fault happens, the system moves from a nominal mode to a *degraded mode* or faulty mode.

Definition 8 (Failure mode). Without repair, the system ends in a *failure mode* in which it is not operational anymore.

The set of health modes is the superset of nominal, degraded and failure modes.

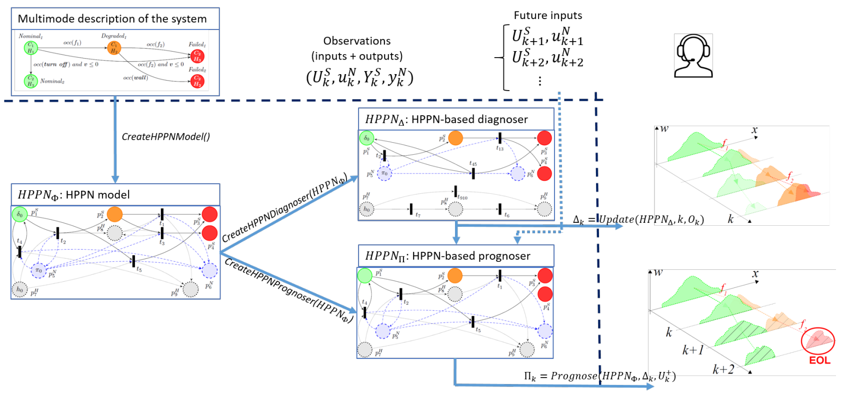
An overview of the health monitoring method is illustrated in Fig. 1 and described by Algorithm 1. Three different objects are defined in the HPPN framework: a hybrid system model $HPPN_{\Phi}$, a HPPN-based diagnoser $HPPN_{\Delta}$ and a HPPN-based prognoser $HPPN_{\Pi}$. Note that the generation of the prognoser object for prognosis purpose is not detailed in this chapter.

The first offline step is the modeling of the hybrid system (line 1) using the HPPN framework, as described in Section 4). The system model $HPPN_{\Phi}$ can be built either from a multimode description of the system or directly from expert knowledge. The second offline step (line 2) is the generation of a HPPN-based diagnoser $HPPN_{\Delta}$ from the system model $HPPN_{\Phi}$ described in Section 5.2. Then the online diagnoser process (line 3-6) uses the system consecutive observations O_k (inputs and outputs) to update the diagnoser result and compute the diagnosis Δ_k (see Section 5).

Example 1. Throughout Section 3, an example of a mobile robot, described in Fig. 2, is used to illustrate the definitions and concepts.

The system is described with an oriented graph, in which the nodes represent the health modes and the arcs represent the mode changes. Variables that can be observed or estimated with observations are in bold.

The robot mission is to move without encountering an obstacle or failure, until it reaches a specific area and is turned off. The initial mode is *Nominal*$_1$: the



Multimode description of the system
Observations (inputs + outputs)
$(U_k^S, u_k^N, Y_k^S, y_k^N)$
Future inputs
U_{k+1}^S, u_{k+1}^N
U_{k+2}^S, u_{k+2}^N
CreateHPPNModel()
$HPPN_\Phi$: HPPN model
$HPPN_\Delta$: HPPN-based diagnoser
$HPPN_\Pi$: HPPN-based prognoser
CreateHPPNDiagnoser($HPPN_\Phi$)
CreateHPPNPrognoser($HPPN_\Phi$)
$\Delta_k = Update(HPPN_\Delta, k, O_k)$
$\Pi_k = Prognose(HPPN_\Phi, \Delta_k, U_k^+)$
EOL

Fig. 1. Overview of the health monitoring methodology for hybrid systems.

Algorithm 1 HPPN-based monitoring methodology

```
1: $HPPN_\Phi \leftarrow CreateHPPNModel()$
2: $HPPN_\Delta \leftarrow GenerateHPPNDiagnoser(HPPN_\Phi)$
3: for all $k$ do
4:     $O_k \leftarrow (U_k^S, u_k^N, Y_k^S, y_k^N)$
5:     $\Delta_k \leftarrow Update(HPPN_\Delta, k, O_k)$
6: end for
```

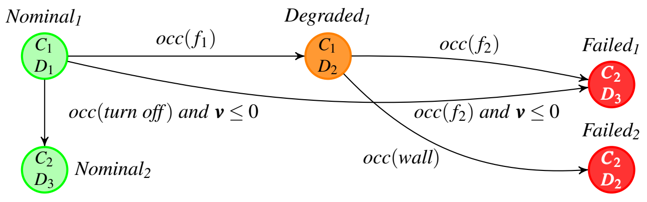


Fig. 2. Mobile robot description.

robot is not degraded and is moving in a non-hostile zone. Its velocity v can be estimated with continuous dynamics C_1 and continuous observations, and is positive. Two faults are expected and the robot degradation is estimated as fault occurrence probabilities with degradation dynamics D_1, in which the probabilities increase with time.

When the (discrete and observable) on-off command *turn off* occurs, the robot stops and its velocity decreasing to 0. The robot enters mode $Nominal_2$, where its motor is turned off and its velocity thus stays 0 (continuous dynamics C_2). Because the robot is turned off, the fault occurrence probabilities stagnate, following degradation dynamics D_3.

Fault f_2 represents a disconnection of the robot motor. Its occurrence leads the system to the failure mode *Failed$_1$*. The occurrence of f_2 implies the robot stops, so its velocity decreases to 0. Once the motor is disconnected, the robot has the same continuous and degradation dynamics (C_2 and D_3) as if it was turned off.

Fault f_1 represents the robot entrance in a hostile zone where it is degrading faster due to environmental conditions. The robot is still moving at the same velocity (C_1). The physical conditions in mode *Degraded$_1$* imply that the probability of f_2 increases more significantly than in mode *Nominal$_1$*. This is defined with degradation dynamics D_2.

From mode *Degraded$_1$*, the robot can still enter in mode *Failed$_1$* with fault f_2 occurrence but it does not match with any condition on the velocity in that case (see arc between *Degraded$_1$* and *Failed$_1$*). The velocity estimation is considered less accurate in the hostile zone than in the non-hostile zone, indeed.

Finally, the hostile zone contains obstacles. The robot can encounter a wall, that stops the robot but not its motor. In that case, the mission fails and the robot enters in failure mode *Failed$_2$*. This event *wall* is not predictable (not estimated with probabilities) but is observable with an environmental on-off sensor. Even if the mission is compromised and the robot is not moving anymore (C_2), its motor is still on so the degradation laws remain the same (D_2).

4 Hybrid System Modeling

We propose to model the system by using the *Hybrid Particle Petri Nets* (HPPN) framework introduced in Gaudel et al. (2014a).

4.1 Hybrid Particle Petri Nets

The HPPN formalism is an extension of Petri nets. Definition 9 gives the complete structure of a Hybrid Particle Petri Net before explaining each notation.

Definition 9. A HPPN is defined as a 11-tuple $\langle P, T, A, \mathscr{A}, E, X, D, \mathscr{C}, \mathscr{D}, \Omega, M_0 \rangle$ which describes discrete evolutions (with symbolic places), continuous evolutions (with numerical places) and degradation evolutions (with degradation places) and relations between them:

- P is the set of places, partitioned into numerical places P^N, symbolic places P^S and degradation places P^D, $P = P^S \cup P^N \cup P^D$,
- T is the set of transitions,
- $A \subseteq P \times T \cup T \times P$ is the set of arcs,
- $\mathscr{A}$ is the set of arc annotations,
- E is the set of event labels,

- $X \subseteq \mathbb{R}^{n_N}$ is the state space of the continuous state vector, with $n_N \in \mathbb{N}_+$ the number of continuous state variables,
- $D \subseteq \mathbb{R}^{n_D}$ is the state space of the degradation state vector, with $n_D \in \mathbb{N}_+$ the number of degradation state variables,
- $\mathscr{C}$ is the set of dynamic equation sets associated with numerical places,
- $\mathscr{D}$ is the set of dynamic equation sets associated with degradation places,
- Ω is the set of conditions associated with transitions,
- $\mathbb{M}_0$ is the initial marking of the Petri net.

An example of a simple HPPN is illustrated in Fig. 3. Symbolic places are represented by thin green circles, numerical places are represented by blue circles and degradation places are represented by thick grey circles. Transitions are represented by black lines. Arcs connecting transitions and symbolic places (resp. numerical and degradation places) are represented by plain arrows (resp. discontinuous and dotted arrows).

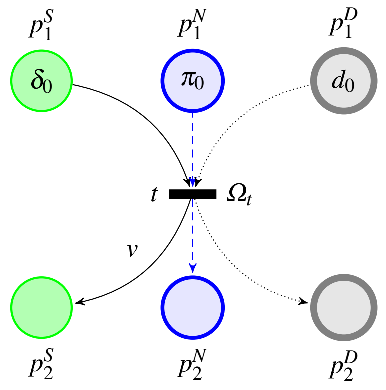


Fig. 3. Example of a simple HPPN at time $k = 0$.

The set $E = E_o \cup E_{uo}$ is the set of event labels that is partitioned into observable event labels E_o and unobservable event labels E_{uo}. Fo example, an anticipated fault in the system model is represented by an unobservable event $f \in E_{uo} \subset E$. An event e is a couple $e = (v,k)$ where $v \in E$ is an event label (or type) and $k \in \mathbb{R}$ the time of occurrence of e. For example, $(z,4)$ means an event type z has occurred at time 4. An event (v,k) is unobservable if for all k, v does not belong to the set of discrete observations of the system.

The places of a HPPN are marked by tokens that carry different types of information.

Symbolic places P^S model the discrete states of the system and are marked by symbolic tokens called configurations.The set of configurations at time k is denoted M_k^S. Each configuration in a HPPN carries the trace of events that occurred on the system until time k. A configuration $\delta_k \in M_k^S$ is a token at time k whose value is a set of events b_k that occurred on the system until time k: $b_k = \{(v,\kappa) \| \kappa \leq k\}$.

Numerical places P^N represent continuous dynamics of the system and related uncertainty. Each numerical place $p^N \in P^N$ is associated to a set of dynamic equations $C_{p^N} \in \mathscr{C}$ modeling system continuous dynamic and its corresponding model noise and measurement noise:

$$C_{p^N} = \begin{cases} x_{k+1} = \mathrm{f}(x_k,u_k) \ + \mathrm{v}(x_k,u_k) \\ \quad y_k = \mathtt{h}(x_k,u_k) + \mathrm{w}(x_k,u_k) \end{cases}, \tag{1}$$

where $x_k \in X$ is the continuous state vector, $u_k \in \mathbb{R}^{n_u}$ is the vector of n_u continuous input variables, f is the noise-free continuous evolution function, v is a noise function, $y_k \in \mathbb{R}^{n_y}$ is the vector of n_y continuous output variables, $\mathtt{h}$ is the noise-free output function and w is the noise function associated to observation. Functions f, v, $\mathtt{h}$ and w depend on the considered place p^N. Numerical places are marked by numerical tokens called particles. The set of particles at time k is denoted M_k^N. More precisely, a particle $\pi_k \in M_k^N$ is a token whose value represents a possible continuous state $x_k \in X$ of the system at time k.

Degradation places P^D represent degradation dynamic of the system and related uncertainty. Each degradation place $p^D \in P^D$ is associated with a set of equations $\mathtt{D}_{p^D} \in \mathscr{D}$ modeling system degradation dynamic:

$$\mathtt{D}_{p^D} = \left\{ \mathtt{d}_{k+1} = \mathtt{g}(\mathtt{d}_k,b_k,x_k,u_k) + \mathtt{z}(\mathtt{d}_k,b_k,x_k,u_k) \right., \tag{2}$$

where $\mathtt{d}_k \in D$ is the degradation state vector, g is the noise-free degradation evolution function and z is a noise function. Functions g and z depend on the considered place p^D. It has to be noticed that, as said earlier, the difference between continuous and degradation places is that degradation system states are function of the continuous state and the set of events b_k that have occurred time k.

Degradation places are marked by degradation tokens. The set of degradation tokens at time k is denoted M_k^D. A degradation token $d_k \in M_k^D$ links a configuration δ_k to a particle π_k and its value is a possible degradation state $\mathtt{d}_k \in D$ of the system at time k.

The set of places P of the HPPN is:

$$P = P^S \cup P^N \cup P^D = \{p_1^S,...,p_s^S\} \cup \{p_1^N,...,p_n^N\} \cup \{p_1^D,...,p_d^D\} \tag{3}$$

where s, n and d are respectively the number of symbolic, numerical and degradation places. In Fig. 3, we have for example, $P = \{p_1^S, p_2^S, p_1^N, p_2^N, p_1^D, p_2^D\}$.

Let M_k denote the set of tokens of the HPPN at time k:

$$M_k = M_k^S \cup M_k^N \cup M_k^D, \tag{4}$$

where M_k^S, M_k^N and M_k^D are respectively the set of configurations, particles and degradation tokens at time k. In Fig. 3, we have for example, $M_0 = \{\delta_0, \pi_0, d_0\}$.

The marking $\mathbb{M}_k$ of a HPPN at time k is the distribution of tokens in the different places:

$$\mathbb{M}_k = \mathbb{M}_k^S \cup \mathbb{M}_k^N \cup \mathbb{M}_k^D, \tag{5}$$

where $\mathbb{M}_k^S \in (2^{M_k^S})^{\mathrm{s}}$, $\mathbb{M}_k^N \in (2^{M_k^N})^{\mathrm{n}}$ and $\mathbb{M}_k^D \in (2^{M_k^D})^{\mathrm{h}}$ are respectively symbolic, numerical and degradation markings at time k. For the example illustrated in Fig. 3:

$$\begin{aligned} \mathbb{M}_0^S &= [[\delta_0]\ \emptyset], \\ \mathbb{M}_0^N &= [[\pi_0]\ \emptyset], \\ \mathbb{M}_0^D &= [[d_0]\ \emptyset]. \end{aligned}$$

Initial marking $\mathbb{M}_0$ represents the initial conditions of the system (the initial continuous and degradation states and the set of events that have occurred until time 0).

Definition 10 (Hypothesis). A *hypothesis* on the system contains all knowledge about the system state at time k and the events that have occurred on the system until time k. A *hypothesis* $\{\delta_k, \pi_k^1, ..., \pi_k^{n_k}, d_k^1, ..., d_k^{n_k}\}$ at time k is composed of a configuration δ_k, a set of particles $\{\pi_k^i | i \in \{1, ..., n_k\}\}$ and a set of degradation tokens $\{d_k^i | i \in \{1, ..., n_k\}\}$, where each degradation token d_k^i links the particle π_k^i to the configuration δ_k.

For example, if the event set is b_0, the continuous state x_0 and the degradation state $\mathtt{d}_0$ are precisely known, the initial set of tokens $M_0 = \{\delta_0, \pi_0, d_0\}$, where d_0 links δ_0 and π_0, is the unique hypothesis. A hypothesis at time k may contain several particles and degradation tokens to represent imprecise knowledge on continuous and degradation states, e.g. $\{\delta_k^1, \pi_k^1, ..., \pi_k^{n_k}, d_k^1, ..., d_k^{n_k}\}$, where $n_k \in \mathbb{N}_+$ is the number of particles used to represent the continuous state, and where n_k degradation tokens links n_k particles to the configuration δ_k^1. The number n_k of particles and degradation tokens is representative of the hypothesis precision at time k.

Definition 11 (Particle cluster). The set of n_k particles linked to the same configuration with n_k degradation tokens is called a *particle cluster*.

In Fig. 4, d^1 and d^2 links π^1 and π^2 to δ^1, and d^3 and d^4 links π^3 and π^4 to δ^2. Two hypotheses are represented $\{\delta^1, \pi^1, \pi^2, d^1, d^2\}$ and $\{\delta^2, \pi^3, \pi^4, d^3, d^4\}$, with two particle clusters $\{\pi^1, \pi^2\}$ and $\{\pi^3, \pi^4\}$. A each time k, the set of particle clusters defines a partition of the particle set M_k^N of the HPPN.

4.2 *Illustration example*

The HPPN model of the mobile robot is presented in Fig. 5. Symbolic places are represented by places with regular thicknesses, while numerical and degradation places are represented by places with medium and large thicknesses, respectively. Arcs that connect transitions and symbolic (numerical and degradation) places are represented by solid (dashed and dotted) arrows.

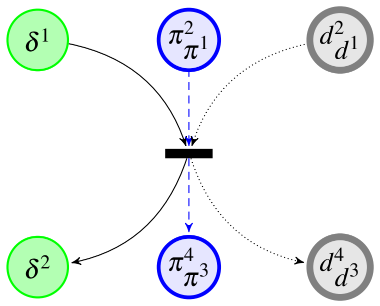


Fig. 4. Illustration of particle clusters.

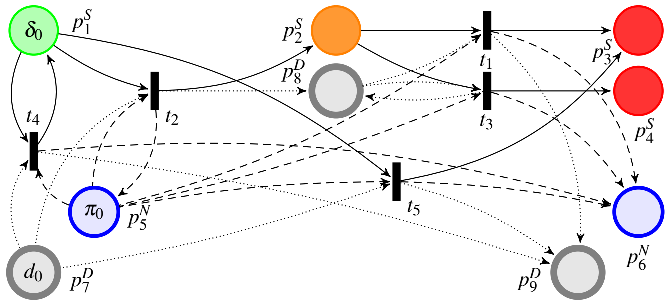


Fig. 5. HPPN model of the mobile robot.

Health modes of a hybrid system (nominal, degraded and failure modes). are represented by combinations of discrete states, continuous dynamics and degradation dynamics. Transitions model changes of health modes, so any transition have three places (one of each type) in its sets of input places and three places in its set of output places. Two transitions cannot have both the same set of input places and the same set of output places.

We decompose the five health modes of the robot into four symbolic places, two numerical places and three degradation places. Four discrete health states are identified from the robot description (Fig. 2). One nominal state, one degraded state, and two different failure states are represented by the four symbolic places p_1^S, p_2^S, p_3^S and p_4^S, respectively. The two numerical places p_5^N and p_6^N represent the continuous dynamics C_1 and C_2. The three degradation places p_7^D, p_8^D and p_9^D represent the degradation dynamics D_1, D_2 and D_3, respectively. Five transitions represent the health mode changes. For example, transition t_4 represents the change from mode $Nominal_1$ to mode $Nominal_2$ so ${}^\circ t_4 = \{p_1^S, p_5^N, p_7^D\}$ and $t_4^\circ = \{p_1^S, p_6^N, p_9^D\}$.

The initial mode is $Nominal_1$ so the tokens δ_0, π_0 and d_0 are in p_1^S, p_5^N and p_7^D, respectively. At time $k = 0$, no event has occurred, $b_0 = \{\}$. The only estimated state is the velocity, $x_0 = [v_0]^T$ with $v_0 > 0$ because the velocity is initially positive. The initial fault occurrence probabilities $\rho_0^{f_1}$ and $\rho_0^{f_2}$ are very low. Thus, $d_0 = [\rho_0^{f_1}, \rho_0^{f_2}]^T$ with $\rho_0^{f_1} = 0.01$ and $\rho_0^{f_2} = 0.05$.

4.3 Marking evolution rules in HPPN for diagnosis

Firing rules in a HPPN may be different depending on the utilization for model simulation, diagnosis or prognosis purposes. Semantics of transition firing are proposed here only for diagnosis purpose.

The following assumptions are considered. The set of input places of a transition is composed of at least one place of any type and at most a place of each type. The set of output places of a transition is composed of at least as many places of each type contained in its set of input places.

Let $^\circ t$ (resp. t°) denote the set of input (resp. output) places of t. The firing of a transition $t \in T$ depends on its associated condition set $\Omega_t \in \Omega$. This condition set Ω_t contains as many conditions as there are input places in $^\circ t$:

$$\forall t \in T, |\Omega_t| = |^\circ t|. \tag{6}$$

For example, if t has a place of each type in $^\circ t$, its condition set is $\Omega_t = \langle \omega_t^S, \omega_t^N, \omega_t^D \rangle$. A condition $\omega : M_k \to \mathbb{B}$, with $\mathbb{B} = \{\top, \bot\}$ (set of logic values TRUE and FALSE), can be a test on the token value, always satisfied ($\top$), or never satisfied ($\bot$).

A symbolic condition ω_t^S can be $\top$ or $\bot$, or it can test the occurrence of an event $v \in E$ (as fault, mission event, interaction with environment, etc.). In this case, the condition $\omega_t^S(\delta_k) = occ(b_k, v)$ tests if the event set b_k of the configuration δ_k contains the event (v, k).

A numerical condition ω_t^N (resp. degradation condition ω_t^D) can be $\top$ or $\bot$ or it can test a constraint on the continuous state (resp. degradation state) of the system. In this case, the condition $\omega_t^N(\pi_k) = c(x_k)$ tests the value of the continuous state vector x_k of the particle π_k.

Example 2. For the example of the mobile robot illustrated on Fig. 5, condition $\Omega(t_4)(\delta_k, \pi_k, d_k) = occ(b_k, turn\ off) \wedge (x_k^0 \leq 0)$ tests if an event labeled with $turn\ off$ occurred at time k and if v_k is 0. We assume that a fault occurs if its probability of occurrence is greater than a predefined threshold 0.9. Consequently, the condition associated with transition t_2 is $\Omega(t_2)(\delta_k, \pi_k, d_k) = occ(b_k, f_1) \vee (d_k^0 > 0.9)$. With the same reasoning, we have $\Omega(t_1)(\delta_k, \pi_k, d_k) = occ(b_k, f_2) \vee (d_k^1 > 0.9)$, $\Omega(t_3)(\delta_k, \pi_k, d_k) = occ(b_k, f_2) \wedge (x_k^0 \leq 0) \vee (d_k^1 > 0.9)$ and $\Omega(t_5)(\delta_k, \pi_k, d_k) = occ(b_k, wall)$.

The firing of a transition t at time k is illustrated in Fig. 6. A token is accepted by the conditions Ω_t at time k if it satisfies the condition of its type. Let $M_k(p)$ be the set of tokens in the place $p \in P$ at time k. $\mathscr{S}_k^t$ is the set of tokens in the input places of the transition t that are accepted by the conditions Ω_t at time k :

$$\begin{array}{rl} \mathscr{S}_k^t = & \{\ \delta_k \in M_k(p^S)\ |\ \omega_t^S(\delta_k) = \top\ \} \cup \\ & \{\ \pi_k \in M_k(p^N)\ |\ \omega_t^N(\pi_k) = \top\ \} \cup \\ & \{\ d_k \in M_k(p^D)\ |\ \omega_t^D(d_k) = \top\ \}\ , \end{array} \tag{7}$$

where $(p^S, p^N, p^D) \in (P^S \cap {}^\circ t) \times (P^N \cap {}^\circ t) \times (P^D \cap {}^\circ t)$ and $\omega_t^S \in \Omega_t$, $\omega_t^N \in \Omega_t$, $\omega_t^D \in \Omega_t$.

Definition 12 (Fireable transition). A transition $t \in T$ is fireable at time k if it exists at least one token in each of its input places which are accepted by the conditions Ω_t:

$$\forall p \in^\circ t,\ |\mathscr{S}_k^t(p)| > 0. \tag{8}$$

In Fig. 6, we suppose at time k that $\omega_t^S(\delta^1) = \top$, $\omega_t^N(\pi^1) = \top$, $\omega_t^D(d^1) = \top$, $\omega_t^S(\delta^2) = \bot$ and $\omega_t^N(\pi^2) = \bot$, then the transition t is fireable.

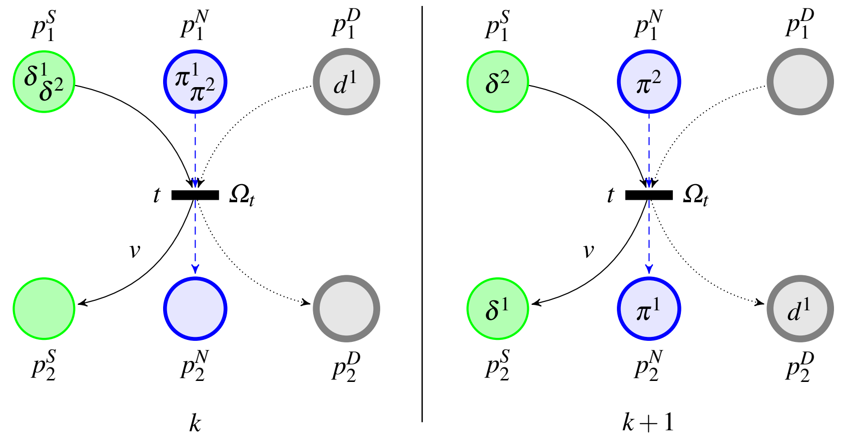


Fig. 6. Illustration of a transition firing at time k.

Definition 13 (Transition firing). The firing of a transition $t \in T$ at time k is defined as follows:
$\forall P^o \in \{P^S, P^N, P^D\}, p \in P^o \cap^\circ t, p' \in P^o \cap t^\circ$,

$$\begin{array}{r} M_{k+1}(p) = M_k(p) \backslash \mathscr{S}_k^t(p), \\ M_{k+1}(p') = M_k(p') \cup \mathscr{S}_k^t(p), \end{array} \tag{9}$$

where $\mathscr{S}_k^t(p)$ is the set of tokens of $\mathscr{S}_k^t$ which are in the place p.

In Fig. 6, after firing the transition t, the three accepted token are in the output places of t. During the transition firing, accepted tokens are moved, their links are conserved and their values are either conserved or updated. This property is the main difference from ordinary Petri nets in which tokens are consumed and new tokens are created in the output places of the transition. The conservation of token values exists in some extensions of Petri nets, like in colored Petri nets for example but the existence of links between tokens and their conservation during the transition firing is specific to HPPN.

An arc $a \in A$ connecting a transition t to a symbolic place p^S, can be annotated with an event label $v \in E$. In this case, the set of event b of a configuration δ which has moved in p^S after the firing of t at time k is updated with the event (v,k). The values of configurations evolve with the annotations $\mathscr{A} \subset A \times E$ during the firing of transitions. In Fig. 6, we suppose that d^1 links δ^1 and π^1 at time k. After the firing of t, the value of δ^1 is $b^1_{k+1} = b^1_k \cup (v,k)$, the value of π^1 is $x^1_{k+1} = x^1_k$, the value of d^1 is $d^1_{k+1} = d^1_k$, and d^1 still links δ^1 and π^1.

5 Hybrid System Diagnosis

Diagnosis aims at tracking the system current health state. The system health state is the combination of its discrete, continuous and degradation states. We propose to build a diagnoser from the HPPN model of a hybrid system [Gaudel et al. (2014b)]. The HPPN-based diagnoser monitors both the system behavior and degradation under uncertainty. Its online process takes as inputs the set of discrete and continuous observations on the system. The output of the diagnoser process at any time k is an estimation of the system health state that takes the form of the marking of the HPPN-based diagnoser $\Delta_k = \hat{M}_k$.

5.1 Uncertainty

Several types of uncertainty are taken into account by using HPPN. Knowledge-based uncertainty must be taken into account because the model does not reflect perfectly reality, as for the symbolic part of the model than the numerical one. Due to the inherent imprecision of sensors, we also consider uncertainty about observations. Two types of uncertainty are then considered: the symbolic uncertainty dealing with the discrete model and observations; and the numerical uncertainty dealing with the imprecision on the continuous model and numerical values.

Regarding the symbolic aspects, the discrete model of the system may include symbolic uncertainty as impossible or incomplete event sequences. Concerning the discrete observations, an event may occur without being observed: this is a missing observation. Dually, an event may be observed whereas it has not really occurred: we talk about false observation.

Symbolic uncertainty is managed at two levels in the HPPN-based diagnoser:

- Every symbolic condition of transitions is replaced by a TRUE condition during the diagnoser generation. It means that pseudo-firing is used for these transitions with modified symbolic conditions.
- During the prediction step of the online diagnoser process, the diagnoser uses pseudo-firing of transitions [Lesire and Tessier (2005); Zouaghi et al. (2011)], introduced in Cardoso et al. (1999) to consider the occurrences of each event consistent with the discrete dynamic. Pseudo-firing creates new hypotheses.

Transition pseudo-firing duplicates tokens: tokens in the input places of the transition are not moved but duplicated and their duplicates are moved in the output places of the transition.

Definition 14 (Transition pseudo-firing). Let $t \in T$ an enabled transition. For each type of input and output place of t, the pseudo-firing of $t \in T$ at time $k-1$ is formally defined by:
$\forall P^o \in \{P^S, P^N, P^D\}, p \in P^o \cap {}^\circ t, p' \in P^o \cap t^\circ$,

$$\begin{array}{l} M_k(p) = M_{k-1}(p), \\ M_k(p') = M_{k-1}(p') \cup \mathscr{S}^t_{k-1}(p), \end{array} \tag{10}$$

where $\mathscr{S}^t_{k-1}(p)$ is the token set of $\mathscr{S}^t_{k-1}$ that are in place p.

Example 3. Fig. 7 illustrates the pseudo-firing of a transition t. At time k, d^1 is supposed to link δ^1 and π^1 and transition t is supposed to be enabled. After pseudo-firing t, tokens δ^1, π^1 and d^1 are not moved and tokens δ^2, π^2 and d^2 are created and moved in the output places of t. Moreover, d^1 links δ^1 and π^1, and d^2 links δ^2 and π^2.

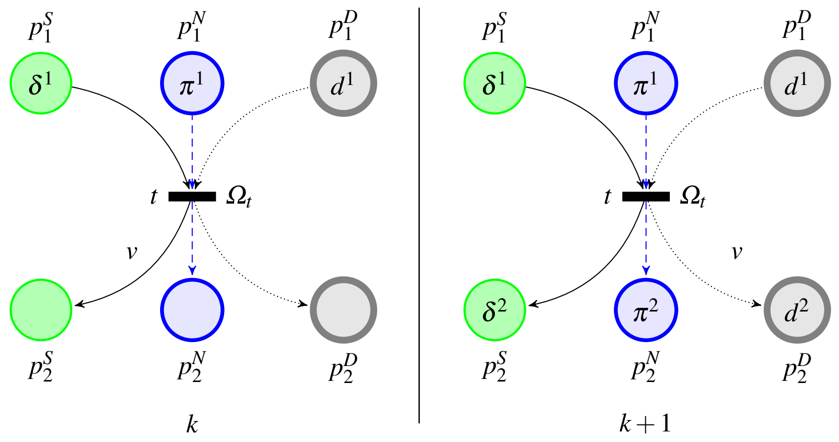


Fig. 7. Pseudo-firing of a transition t at time k.

Besides the intrinsic deviation between reality and continuous model of a system, numerical uncertainty embodies the fact that the numerical values are imprecise. This is an inevitable problem in real case studies. For example, in Fig. 8, we can see the difference between the measured data of a battery voltage and its non-noisy discharge model.

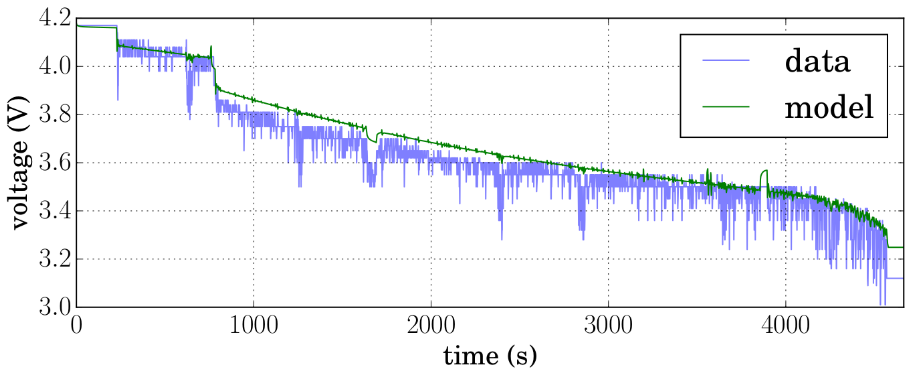


Fig. 8. Comparison between measured data of a battery voltage and its non-noisy discharge model.

Numerical uncertainty is often dealt with through an estimator [Ding (2014)], that aims at estimating the continuous state according to model noise and measurement noise. We use particle filters [van der Merwe et al. (2000)] to estimate the continuous state through the set of particles of the HPPN. The use of particulate filters is relevant for estimating discrete, continuous and degradation states since the representation of the continuous state estimate is already discretized into particles.

For example, in this work, a particle filter is applied independently to each cluster of particles thanks to the links between the configurations and the particles, provided by the degradation tokens. During the diagnosis prediction step, the values of the particles evolve as a function of the continuous dynamics associated with the numerical places to which the particles belong. Then, during the correcting step of the online diagnoser process, each particle cluster is resampled independently. The links between the configurations and the particles, provided by the degradation tokens, are thus used to prevent the particle distribution to be disturbed by pseudo-firing.

5.2 Diagnoser Generation

The system model $HPPN_{\Phi}$ is a tuple $\langle P_{\Phi}, T_{\Phi}, A_{\Phi}, \mathscr{A}_{\Phi}, E_{\Phi}, X_{\Phi}, D_{\Phi}, \mathscr{C}_{\Phi}, \mathscr{D}_{\Phi}, \Omega_{\Phi}, \mathbb{M}_{0\Phi}\rangle$ as defined in Section 4. The diagnoser $HPPN_{\Delta}$ is a tuple:

$$HPPN_{\Delta} = \langle P_{\Delta}, T_{\Delta}, A_{\Delta}, \mathscr{A}_{\Delta}, E_{\Delta}, X_{\Delta}, D_{\Delta}, \mathscr{C}_{\Delta}, \mathscr{D}_{\Delta}, \Omega_{\Delta}, \mathbb{M}_{0\Delta}\rangle, \tag{11}$$

which is generated from the system model $HPPN_{\Phi}$ in 6 steps that are described hereafter.

The diagnoser has to estimate the discrete, continuous and degradation states of the system. Step 1 consists in copying the HPPN system model. Indeed, discrete, continuous and degradation state spaces, as well as continuous and degradation dynamics are the same as those of the model. As a result, all the places, event labels, state spaces and diagnosis dynamics remain the same as those of the model $HPPN_{\Phi}$:

$$P_{\Delta} = P_{\Phi}, E_{\Delta} = E_{\Phi}, X_{\Delta} = X_{\Phi}, D_{\Delta} = D_{\Phi}, \mathscr{C}_{\Delta} = \mathscr{C}_{\Phi}, \mathscr{D}_{\Delta} = \mathscr{D}_{\Phi}. \tag{12}$$

The initial marking $\mathbb{M}_{0\Delta}$ of the HPPN-based diagnoser $HPPN_{\Delta}$ corresponds to the initial marking $\mathbb{M}_{0\Phi}$ of the system model, which contains knowledge about mode, state and events that occurred on the system at time 0 (usually none): $\mathbb{M}_{0\Delta} = \mathbb{M}_{0\Phi}$.

Step 2 consists in separating the HPPN-based diagnoser into two levels: the *behavioral level* manages the observable part of the system whereas the *degradation* level includes the unobservable part. Thus, the behavioral level contains only the symbolic and numerical places, while the degradation level contains the degradation places.

Each transition transition $t \in T_{\Phi}$ thus generates a pair of transitions (t', t'') during the diagnoser generation. Transition t' inherits arcs linking t to symbolic and numerical places, as well as symbolic and numerical conditions. Transition t'' inherits arcs linking t to the degradation places, as well as the degradation condition. t' and t'' are then defined by:

$$^{\circ}t' = ^{\circ}t \cap (P^S \cup P^N), \qquad t'^{\circ} = t^{\circ} \cap (P^S \cup P^N), \tag{13}$$

and:

$$^{\circ}t'' = ^{\circ}t \cap P^D, \qquad t''^{\circ} = t^{\circ} \cap P^D, \tag{14}$$

with the following conditions $\Omega_{t'}$ and $\Omega_{t''}$:

$$\Omega_{t'} = \langle \omega_t^S, \omega_t^N \rangle, \qquad \Omega_{t''} = \langle \omega_t^D \rangle. \tag{15}$$

The set of all transitions in the behavioral level and in the degradation level is denoted T_{Δ}.

Example 4. Fig. 9 illustrates the second step of the diagnoser generation $HPPN_{\Delta}$, the separation into 2 levels, for the example of the mobile robot. Degradation places and associated transitions are put in the degradation level.

Step 3 consists in setting to TRUE all the symbolic conditions, in agreement with the uncertainty management concerning the occurrences of events (see Section 5.1):

$$\forall t \in T_{\Delta},\ \omega_t^S \in \Omega_t \Rightarrow \omega_t^S \leftarrow \top. \tag{16}$$

Thus, all configurations in the HPPN-based diagnoser satisfy the symbolic conditions. This means that the diagnoser considers at any time the occurrence of each

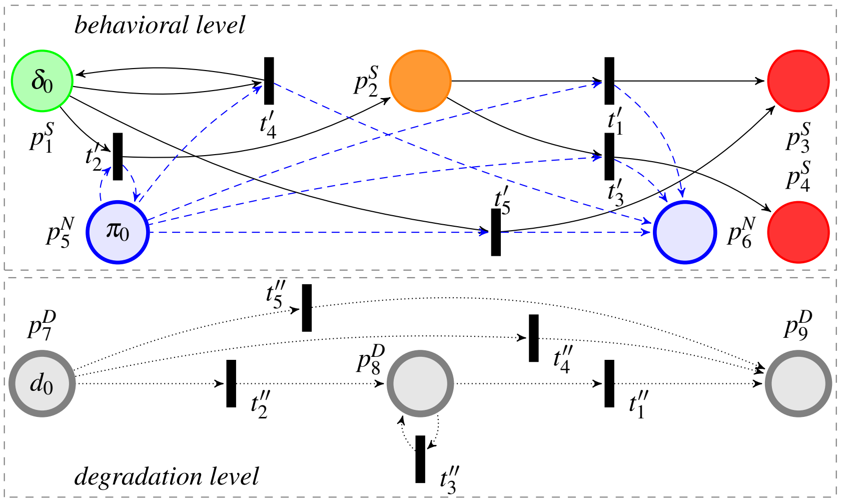


Fig. 9. Diagnoser generation of the mobile robot - Step 2 : Separation into 2 levels

event that can occur from the estimated current mode. The arc annotations remain unchanged:

$$\mathscr{A}_\Delta = \mathscr{A}_\Phi. \tag{17}$$

Example 5. In the example of the mobile robot, conditions $\Omega(t_1) = \Omega(t_2) = \Omega(t_5) = \top$ because of the OR logical expression. Transitions t_3 and t_4 keep their numerical conditions because of the AND logical expression.

Step 4 consists in removing degradation conditions in order to disconnect the marking evolution of the degradation level from the degradation state:

$$\forall t \in T_\Delta,\ \omega_t^D \in \Omega_t \Rightarrow \Omega_t \leftarrow \Omega_t \backslash \{\omega_t^D\}. \tag{18}$$

This step allows to manage computation performance and to keep focus on observations during the diagnosis process.

Example 6. In the example of the mobile robot, condition $\Omega(t_3) = \top$ after Step 4.

Step 5 also improves the computation performance by merging transitions having the same sets of input and output places in order. It reduces the size of the possible state space. As a consequence, in the behavioral level, hypotheses sharing the same cluster of particles are created during the prediction step of the online diagnoser. In other words, several hypotheses are monitored according to the same continuous dynamics with a single cluster of particles instead of having as many clusters as hypotheses.

In the degradation level, this step eliminates concurrent transitions which have the same degradation place as input and the same degradation place as output.

Two transitions are mergeable if they represent the same change in continuous dynamics (the same numerical places as input and as output, and the same numerical condition) and have the same symbolic input place.

Definition 15 (meargeable transitions). Two transitions $(t', t'') \in T^2$ are mergeable if and only if:

$$({}^\circ t' = {}^\circ t'') \wedge (t'^\circ \cap P^N = t''^\circ \cap P^D) \wedge (t'^\circ \cap P^D = t''^\circ \cap P^N) \wedge (\Omega_{t'} = \Omega_{t''}). \quad (19)$$

Step 5 of the diagnoser generation consists in merging every pair of mergeable transitions as long as there are at least two mergeable transitions using the following definition.

Definition 16 (Merging of two transitions). Merging two meargeable transitions $(t', t'') \in (T)^2$ is defined by:

1. Create a new transition t such as:

$$^\circ t \leftarrow {}^\circ t', \qquad t^\circ \leftarrow t'^\circ \cup t''^\circ, \qquad \Omega_t \leftarrow \Omega_{t'}. \quad (20)$$

2. Update T_δ:

$$T_\delta \leftarrow (T_\delta \backslash \{t', t''\}) \cup \{t\}. \quad (21)$$

Example 7. Fig. 10 illustrates the merging step of the diagnoser generation for the mobile robot. To simplify the reading, transitions of Fig. 9 have been renamed according to the following correspondence table.

Fig. 9	t'_1	t'_2	t'_3	t'_4	t'_5	t''_1	t''_2	t''_3	t''_4	t''_5
Fig. 10	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}

In the behavioral level, transitions t_1 and t_3 (t_4 and t_5) are merged into t_{13} (respectively t_{45}) as they have the same set of input places $\{p^S_2, p^N_5\}$ (respectively $\{p^S_1, p^N_5\}$) and the same numerical place p^N_6 as output. In the degradation level, the transitions t_9 and t_{10} are merged into t_{910} as they were concurrent.

Step 6 consists in removing the transitions resulting in an elementary loop in the degradation level (pure Petri net).

$$T_\Delta \leftarrow T_\Delta \setminus \{t | {}^\circ t \cap P^D = t^\circ \cap P^D\}. \quad (22)$$

The goal is to improve the computational performance by avoiding the displacement of the degradation tokens through a transition that loops on the same degradation place. This step has no impact on the tracking quality of the degradation.

Example 8. Fig. 11 shows the diagnoser of the mobile robot. Transition t_8 is removed because it formed an elementary loop with p^D_8.

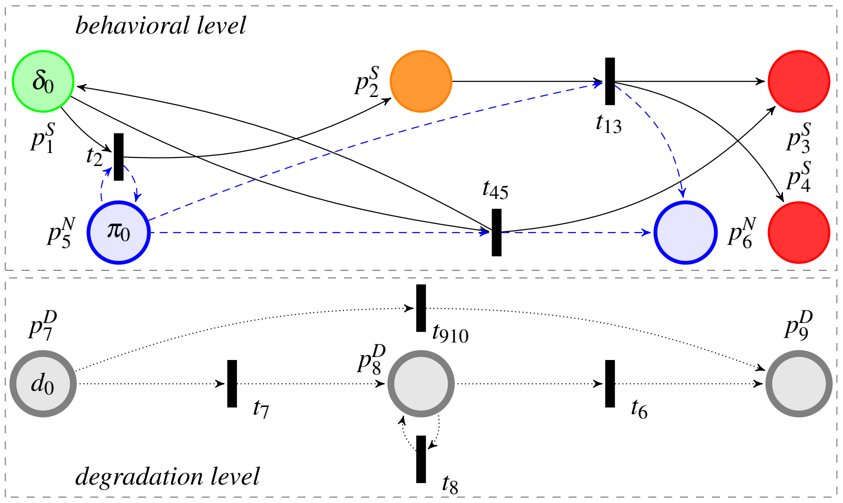


Fig. 10. Diagnoser generation of the mobile robot - Step 5 : Transition merging

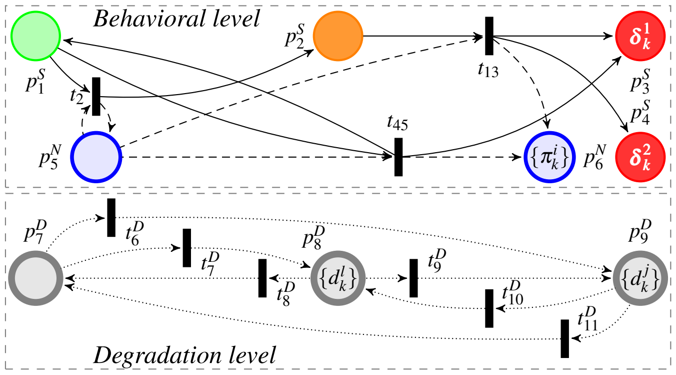


Fig. 11. HPPN-based diagnoser $HPPN_\Delta$ for the mobile robot

5.3 Diagnoser Process

The initial marking $M_0 = \{M_0^S, M_0^N, M_0^D\}$ of the HPPN-based diagnoser represents the system's initial mode. It is composed of one configuration with value b_0, n_0^N particles with value x_0 and n_0^N degradation tokens with value d_0, where n_0^N is the initial number of particles. As long as only one hypothesis is considered in the initial marking, two hypotheses cannot share the same configuration. However, two hypotheses can share the same set of particles if they have the same continuous dynamics but different discrete states (see Example 7). From the initial marking and the initial commands, the diagnoser marking $\hat{M}_k$ evolves at time k according to the observations $O_k = O_k^S \cup O_k^N$, where O^S and O^N respectively represent the observations corresponding to the symbolic part and the numerical part.

The estimated marking at time k, $\hat{M}_k = \{\hat{M}_k^S, \hat{M}_k^N, \hat{M}_k^D\}$ where $\hat{M}_k = \hat{M}_{k|k}$, represents all the possible hypotheses on the system mode at time k.

The marking evolution in the HPPN-based diagnoser is based on two steps, prediction and correction, which combine the transition pseudo-firing, particle filters and an algorithm called the Stochastic Scaling Algorithm (SSA).

In particle filtering, the number of particles defines the precision of the filter. The goal of the SSA is to avoid the combinatory explosion and to limit the number of tokens at each step of the algorithm. It dynamically adapts the hypotheses precision. This algorithm is not described in this chapter, but the reader could refer to Douc and Cappé (2005), Li et al. (2015) or Doucet and Johansen (2009) to obtain more information about resampling methods for particle filtering.

The prediction step of the online diagnoser process aims at determining all possible next states of the diagnoser $\hat{M}_{k+1|k}$. It is based on the firing of the enabled transitions and on the update of the token values. All the enabled transitions are fired according to the rules described in section 4.3. This implies the assumption that a single event can occur at time k. The event set b_k of a configuration δ_k moved through an arc $a \in A$ during the transition firing, is updated according to the annotation $\mathscr{A}(a)$. The value x of a particle π is updated according to the continuous dynamics associated to the numerical place $p^N \in P^N$ in which π belongs after the transition firing. Noise is added during the particle value update to take into account uncertainty about model continuous dynamics. The value $\mathtt{d}$ of a degradation token d is updated according to the degradation dynamics associated to the degradation place $p^D \in P^D$ in which d belongs after the transition firing.

The correction step of the online diagnoser process updates the predicted marking $\hat{M}_{k+1|k}$ to the estimated marking $\hat{M}_{k+1|k+1}$ according to new observations O_{k+1}. It is based on the computation of the scores of all hypotheses contained in the marking and on the resampling of the tokens depending on the scores of the hypotheses they represent. The scores of hypotheses are computed with Pr^S and Pr^N, the probability distributions over the symbolic and the continuous states, respectively. Pr^S gives the configuration weights. A configuration weight is computed as the inverse of exponential of the distance between the configuration event set and $O_{k+1}^- = \{O_\kappa | \kappa \le k+1\}$, the set of symbolic observations until $k+1$. Pr^N gives the normalized particle weights, calculated according to the distance between the particle values and numerical observations O_{k+1}^N. Then, the score of one hypothese is computed using a weighted function of the sum of its particle weights and its configuration weight:

$$Score(\delta_k^i, \{\pi_k^j\}, \{d_k^l\}) = \alpha \times Pr^S(\delta_k^i) + (1-\alpha) \times \sum_{j=1}^{n_k^N} Pr^N(\pi_k^j), \tag{23}$$

where $\alpha \in [0,1]$ is the coefficient indicating the global confidence of the symbolic part relatively to the numerical part and $n_k^N = |\{\pi_k^j\}|$ is the number of particles considered for the hypothesis. The score of a hypothesis is always between 0 and 1. A decision making process associates a new number of parti-cles n_{k+1}^N to each set of particles, according to the best score of all the possible modes it belongs and three scale parameters, denoted n_{min}^N, n_{suff}^N and n_{max}^N. Each set of particles is then resampled with its associated n_{k+1}^N particles, like in classical particle filtering. Parameters n_{min}^N and n_{suff}^N are respectively the minimum and the sufficient numbers of particles (but also the number of degradation tokens) to monitor a hypothesis. It means that any n_{k+1}^N is chosen to satisfy the predicate $n_{min}^N \leq n_{k+1}^N \leq n_{suff}^N$. Parameter n_{max}^N is the maximum number of particles (or degradation tokens) available to monitor all hypotheses. It means the total number of particles after the resampling is always less than or equal to n_{max}^N. During the resampling, degradation tokens linked to duplicated particles are duplicated while those linked to deleted particles are deleted. Finally, configurations that are no longer linked with any degradation tokens are deleted. The correction mechanism highlights that the degradation tokens, in addition to estimate the degradation state, prevent the particle distribution of one hypothesis to be disturbed by the particle distributions of the other hypotheses. In particle filtering, the number of particles defines the precision of the filter but is also a computational performance factor. The scale parameters of the diagnoser process thus compromise the number of hypotheses to monitor and the precision granted to each one of them, relative to the available computational power (n_{max}^N can be set up to fulfill performance constraints).

The diagnosis Δ_k is deduced from the marking of the HPPN-based diagnoser $HPPN_\Delta$ at time k:

$$\Delta_k = \hat{M}_k = \{\hat{M}_k^S, \hat{M}_k^N, \hat{M}_k^D\}. \tag{24}$$

It represents all diagnosis hypotheses as a distribution of beliefs over the current health mode and how this mode has been reached. In other words, the marking $\hat{M}_k$ indicates the belief over the continuous state, the fault occurrences and the degradation state. The HPPN-based diagnoser results include the results of a classical diagnoser in terms of fault occurrences. In a classical diagnoser, however, every diagnosis hypotheses has the same belief degree. A HPPN-based diagnoser handles more uncertainty and evaluates the ambiguity according to the tokens places and values.

6 Case Study

This section focuses on the application of the proposed methodology on the K11 planetary rover prototype. The K11 is a four-wheeled rover designed as a platform for testing power-efficient rover architectures in Antarctic conditions [Lachat et al. (2006)].
The K11 has then been redesigned by NASA Ames Research Center for diagnostics and Prognostics-enabled Decision Making research [Balaban et al. (2013); Sweet et al. (2014); Daigle et al. (2014)]. It has been transformed into a testbed to simulate

some fault occurrences and failures. In this work, it is studied as a functional rover exposed to failures and executing missions.

6.1 Rover Description

The K11 rover is powered by twenty-four 2.2 Ah lithium-ion single cell batteries. A typical mission of the rover consists in visiting and performing desired science functions at a set of waypoints, before joining its charging station. A decision making module (DM) is responsible for determining the order in which to visit the waypoints according to the terrain map, the waypoint positions and rewards, and the rover conditions. The rover has four wheels, denominated by their location: the front-left (FL) wheel, the front-right (FR) wheel, the back-left (BL) wheel and the back-right (BR) wheel. Each wheel is driven by an independent 250 W graphite-brush motor, with control performed by a single-axis digital motion controller. An onboard laptop computer runs the control and data acquisition software. The rover is a skid-steered vehicle, meaning that the wheels cannot be steered and the rover is rotated by commanding the wheel speeds on the left and right sides to different values. The battery management system provides battery charging and load balancing capabilities. It also sends voltage and temperature measurements for each of the individual cells to the onboard computer. The data acquisition module collects current and motor temperature measurements and sends them to the onboard computer. The motor controllers send back motion data such as commanded speeds and actual speeds. More details on the rover can be found in Balaban et al. (2013).

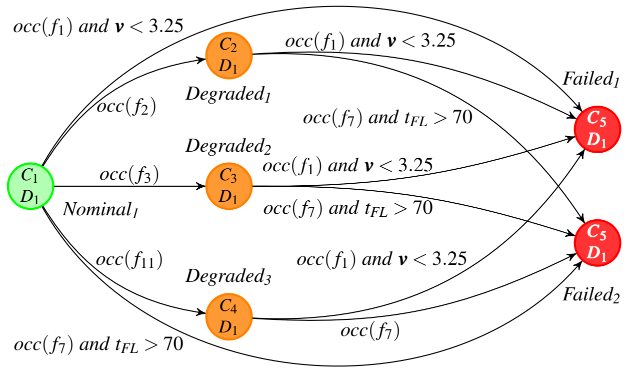
All the continuous observations on the rover and the list of faults we consider in this study are presented in Table 1. Four signals command the wheels with a proportional-integral-derivative controller and the set of sensors returns 61 measurement signals. Several fault types have been implemented on the testbed and are related to the power system (battery), the electro-mechanical system (motors, controller), and the sensors (drift, bias, scaling or failure).

The K11 rover has no discrete actuator or discrete sensor and thus has mostly been studied as a continuous system, where faults were defined as constraints on the continuous state. We propose to abstract anticipated faults into unobservable events. The multi-mode system that describes the rover health evolution is presented in Fig. 12. To simplify the description, only a part of the multi-mode system is shown. The modes corresponding to consecutive fault occurrences are not included and only the front-left motor is considered.

The rover is in mode *Nominal*$_1$ with continuous dynamics C_1 as long as no fault has occurred. Fault f_1 occurrence represents the *end of discharge* (EOD) of the battery, i.e. the date when the battery is too discharged to power the system. This is assumed to occur when the battery voltage is lower than 3.25 V and it leads to the mission failure (mode *Failed*$_1$ with continuous dynamics C_5). Fault f_2 represents the emergence of a parasitic battery load arising from an electrical

Table 1. Continuous commands, continuous measurements, and fault types on the K11

Command type	**Comments**	**Units**
Wheel speed	Commanded speeds for wheels on the same side are the same	rad/s
Measurement type	**Comments**	**Units**
Wheel speed	One for each wheel	rad/s
Total current	A current sensor on the power bus	A
Motor current	One for each motor	A
Motor temperature	One for each motor	°C
Battery temperature	One for each battery cell	°C
Battery voltage	One for each battery cell	V
Fault event labels	**Fault descriptions**	**Effects**
f_1	Battery charge depletion	Lead to failure
f_2	Parasitic electric load	Increase battery drain
f_3, f_4, f_5, f_6	Increased motor frictions	Increase battery drain and motor temperatures
f_7, f_8, f_9, f_{10}	Motor overheating	Lead to failure
$f_{11}, f_{12}, f_{13}, f_{14}$	Failed motor temperature sensors	Unable to sense motor temperatures



$occ(f_1)$ and $\mathbf{v} < 3.25$
C_2 D_1
$occ(f_1)$ and $\mathbf{v} < 3.25$
$Failed_1$
$occ(f_2)$
$Degraded_1$
$occ(f_7)$ and $t_{FL} > 70$
C_5 D_1
$Degraded_2$
C_1 D_1
$occ(f_3)$
C_3 D_1
$occ(f_1)$ and $\mathbf{v} < 3.25$
$Nominal_1$
$occ(f_7)$ and $t_{FL} > 70$
C_5 D_1
$occ(f_{11})$
$occ(f_1)$ and $\mathbf{v} < 3.25$
$Degraded_3$
$Failed_2$
C_4 D_1
$occ(f_7)$
$occ(f_7)$ and $t_{FL} > 70$

Fig. 12. Streamlined description of the rover health evolution.

submodule continuously engaged, for example. The parasitic load increases the total current and thus the battery drain (mode *Degraded$_1$* with continuous dynamics C_2), which causes the system to reach the EOD prematurely. Fault f_3 (f_4, f_5 and f_6) represents an increased friction of the FL (FR, BL and BR) motor. The increased friction induces the need for a larger amount of current to satisfy the same speed (mode *Degraded$_2$* with continuous dynamics C_3). Furthermore, the load demands will be higher, raising the motor temperature. The most critical scenario for a motor is an overheating. In such case, the heat will eventually destroy the insulation of the windings, causing electrical shorts and leading to motor failure. The overheating of the FL (FR, BL and BR) motor is represented by fault f_7 (f_8, f_9 and f_{10}). The occurrence of any one of these faults leads to the rover failure (mode *Failed$_2$* with continuous dynamics C_5) and thus represents the rover *end of life* (EOL). A motor is assumed to overheat when its temperature exceeds 70 °C. The motor temperatures are measured by four sensors. These sensors, however, are known to fail unexpectedly, sending inconsistent values. These failures are represented by faults f_{11} f_{12}, f_{13} and f_{14}. We consider that the temperature model is not accurate enough without a correction step with observations. As a consequence, once f_{11} (f_{12}, f_{13} and f_{14}) has occurred, the occurrence of fault f_7 (f_8, f_9 and f_{10}) does not match with any condition on the FL (FR, BL and BR) motor temperature (see the arc between *Degraded₃* and *Failed₂*). In Fig. 12, mode *Degraded₃* with continuous dynamics C_4 represents the mode where the temperature sensor of the FL motor has failed. The rover degradation state can be monitored with degradation dynamics D_1, which corresponds to the identity dynamics.

6.2 Rover Modeling

Considering all the motors and the consecutive fault combinations, we identified 192 modes and 240 mode changes. The HPPN-based model of the rover has 241 places (192 symbolic places, 48 numerical places, 1 degradation place) and 240 transitions. The HPPN-based diagnoser has the same number of places and transitions because the merging step of the diagnoser generation (Step 5) does not reduce the number of transitions (it is specific to this case study). Actually, the merging step merges transitions having exactly the same sets of input and output places. This kind of transition does not exist in this real application, but it may be very useful in other cases. The degradation place is removed from the transition inputs and outputs, reducing the complexity of the net. Because there is only one degradation place, all transitions in the degradation level are removed by Step 6 of the diagnoser generation. The underlying DES of the multi-mode system and HPPN-based model and diagnoser of the K11 rover are available at https://homepages.laas.fr/echanthe/PetriNets2016.

The nominal continuous dynamics is represented as a set of differential equations that unifies the battery model with the rover motion model and the temperature models. It can be converted to a discrete-time representation and solved with a sample time of $1/20s$, while continuous observation sampling is about $1s$. We consider 30 state variables for the rover, including the rover 3-dimensional position, its relative angle position, the wheel control errors, the motor temperatures and motor winding temperatures. The 24 batteries are lumped into a single one to only consider 5 battery state variables (3 charges, the temperature and the voltage) instead of 120. The battery model has been validated with experimental data in previous works [Daigle et al. (2014); Sweet et al. (2014)]. Unifying the battery model with motion and temperatures, however, increases uncertainty about the rover model.

Fault f_2 occurrence and effect on the system behavior are modeled as a time varying parameter. The parasitic battery load is captured as an additional current reaching a value between 1.5 A and 4.5 A from value 0 A in a few seconds after the fault occurrence. First, two parameters are added to the continuous state vector to monitor both the duration since the fault occurrence and the additional current value. Then, the uncertain rise of the additional current is modeled by adding a Gaussian noise, with a mean and standard deviation values starting respectively at 3 and 0.3, and decreasing to 0 while the duration since the fault occurrence increases.

Finally, the temperature model is quite uncertain so temperature measurements are assumed to be reliable when sensors are not failed. We model fault f_{11}, f_{12}, f_{13} and f_{14} by increasing significantly the motor temperature sensor noise because failed sensors only send inconsistent large values with no pattern. Fault f_3, f_4, f_5 and f_6 and increased motor frictions can be modeled with time varying parameters (as additional motor resistances) like f_2 but are not monitored in this study.

6.3 Simulation Results

The HPPN framework is implemented in Python 3.4. The tests were performed on a 4 Intel(R) Core(TM) i5-4590 CPU at 3.30 GHz with 16 GB of RAM and running GNU/Linux (Linux $3.13.0-74$, x86_64). In order to reduce computation time, the token value update step is multithreaded on the 4 physical cores. The rest of this implementation only uses one core.

Two scenarios studied in Sweet et al. (2014) are considered in this work. The rover mission is to visit a maximum of 12 waypoints and to go back to its starting position. All waypoints have different associated rewards. In nominal conditions, the rover DM system returns a 5-waypoints path, starting and finishing at the same position. For all scenarios, the K11 rover starts at 0s with batteries fully charged and with all components at the ambient temperature. The K11 rover currently has, however, 2 motor temperature sensors (FL and BL) failed. These faults do affect the monitoring but not the physical system, so the DM returns the same path as in nominal conditions. These sensor faults are diagnosed in one sampling period by the diagnoser if we consider the initial mode to be unknown, so we assume to know the rover initial degraded mode, and we have only one hypothesis in the initial diagnoser marking.

For the sake of clarity, in the rest of the paper, health modes are designated with representative keywords of the rover state. For example, the initial mode is designated as *Sensor BL FL fault*. The initial number of particles and degradation tokens is $n_0^N = 100$. The scale parameters of the diagnoser process are set to $(n_{min}^N, n_{suff}^N, n_{max}^N) = (40, 80, 6000)$.

6.3.1 Scenario 1

In Scenario 1, no fault occurs. The rover successfully executes its mission. Fig. 13 presents tthe diagnosis hypotheses as the distribution of beliefs over the current health mode at any time.

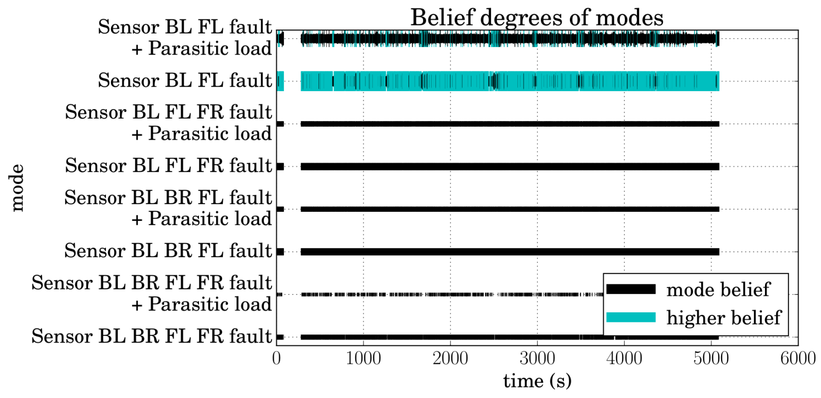


Fig. 13. Scenario 1: Mode belief at any time.

The belief degree of a possible mode is the score computed for the related hypothesis with Eq. 23 and α set to 0.5. Any belief degree is between 0 and 1, this represents a score, so the sum of the belief degrees of all possible modes is not 1. In Fig. 13, the maximum belief degree of a mode at any time is represented by the thickness of the line and the highest belief degree of all the modes is plotted in blue. The gap between 81 s and 281 s corresponds to a break during the experiment. The figure shows that the diagnoser keeps the real mode *Sensor BL FL fault* in its set of hypotheses and assigns it the highest belief degree almost all along the scenario. Other modes are also highly considered by the diagnoser at any time because of the model-based uncertainty. The combination of continuous and discrete evolutions is explained by the marking rules in the HPPN. As the diagnoser generation process replaces every symbolic condition by a TRUE condition, the emphasis is put on the continuous evolution of the system. It means that faults are essentially detected by continuous clues. In the same way, if a discrete event occurs before a degradation is detected, the discrete evolution of the system will always be followed thanks to the pseudo-firing process, and then the degradation will confirm or not this discrete evoluton.

6.3.2 Scenario 2.

In Scenario 2, a battery parasitic load occurs between 660 s and 695 s, and the DM system cancels the visit of the farthest waypoint. Fault f_2 occurrence is immediately detected by the diagnoser (Fig. 14). After 678 s, the possibility of being in mode *Sensor BL FL fault* + *Parasitic load* is the highest until the end of the mission. The fault load is estimated (most likely) at 1.39 A at 678 s, 1.73 A at 679 s, 2.16 A at 683 s and 2.16 A at 3906 s. A zoom between 570 s and 760 s on the trajectories of the modes that are still possible at 3906s (Fig. 15) shows that fault f_2 is believed

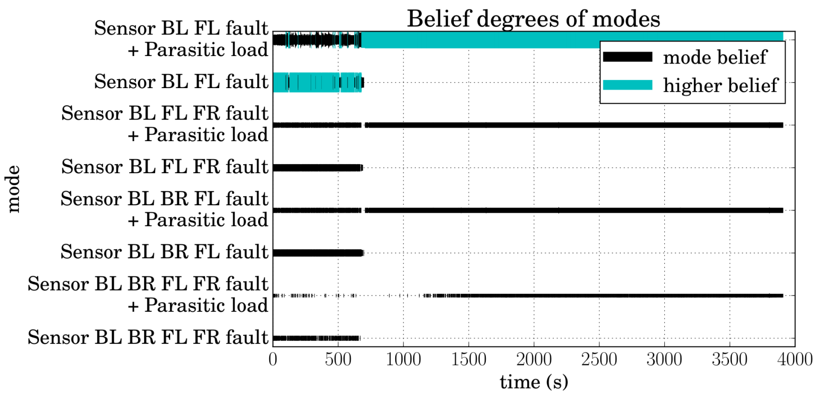


Fig. 14. Scenario 2: Mode belief at any time.

to occur between 631 s and 694 s, and most likely between 677 s and 689 s. These results are consistent with our analysis of the measured total current.

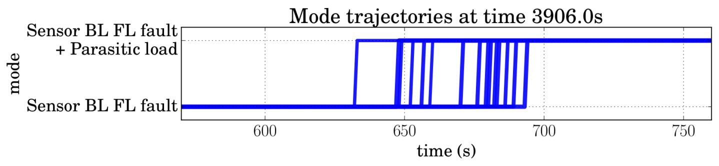


Fig. 15. Scenario 2: Trajectories of possible modes at time 3906s.

Faults are always detected in one sampling period because the HPPN-based diagnoser considers all hypotheses (including the hypotheses concerning faults with slow degradation) during the prediction step due to pseudo-firing. Moreover it keeps the matching marking during the correction step. However, the isolation may be longer than one sampling period. The results show that the diagnoser grants most of the time, but not always, the highest belief to the real mode. The diagnosis, however, carries all the explanation of the observations as a distribution of beliefs, and then the real mode is always considered in the set of diagnosis hypotheses. This illustrates the robustness of the HPPN-based diagnoser to the rover model and data. The average diagnosis computation time and token number are 13.3 s and 8801.4, respectively. These metrics point out the diagnosis computation time remains acceptable compared to the system model computational complexity. The maximum RAM used by Scenario 1 and 2 are 140.7 MB and 141.8 MB. More extended performance analyses are proposed in the next section.

The case study results show that HPPN-based diagnosis is robust to real system data and constraints and adaptable to systems without discrete observations nor degradation knowledge.

6.3.3 Performance Analysis, Comparison with other Approaches

Diagnosis computation times, and the maximum RAM used for different sets of scaling parameters, are given in Table 2. Tests have been performed on three scenarios (including the nominal and faulty scenarios presented above) and run 12 times. 54403 diagnoses are computed.

Table 2. Computational performances of the HPPN-based diagnosis method for different scaling parameters.

Scaling parameters		Δ_k time (s)	max. RAM (MB)
	minimum	0.28	
$(40,80,1500)_\Delta$	maximum	4.54	
	average	3.35	126.73
	minimum	0.28	
$(40,80,400)_\Delta$	maximum	1.00	
	average	0.56	122.15
	minimum	0.22	
$(20,60,400)_\Delta$	maximum	0.98	
	average	0.74	112.18

These metrics point out that computation times with the initial scaling parameters remain acceptable but do not respect real-time constraints; observations sampling is about 1 s and the average diagnosis computation time is 3.35 s. This is mainly due to that diagnosis process relies on parallel step-by-step simulations but it is also due to the rover model computational complexity.

The methodology theoretical complexity is difficult to evaluate because it depends on the continuous equations, the DES structure and the token number, among others. Moreover, software implementation, compilation optimization or virtual machine execution are also other performance factors difficult to evaluate in practice. This is why we propose in this work to approach performance constraints by tuning the scaling parameters.

Other diagnosis works have been conducted on the rover [Balaban et al. (2013)], such as the QED algorithm, described in Daigle et al. (2015a) or the HyDe (Hybrid Diagnostic Engine) [Narasimhan and Brownston (2007)]. The main advantage of our approach is that the diagnosis estimates are not presented as a set of candidates, but as a distribution of candidates. In case of decision making in a health management context, the operator may take a more justified decision. In terms of detection delay, QED and HyDe detect the fault in less than 1 sec. QED isolates it in 26 s and has a good estimate of the parasitic load (3 % of relative error) in about 50 s. Our method detects the fault after the first diagnosis. The estimation of the worse isolation time is about 18 s. The estimation of the parasitic load is good (8% of relative error) after 23 s.

7 Conclusion

This work applies the approach of health monitoring based on Hybrid Particle Petri Nets to a real case study, the K11 planetary rover prototype. The HPPN framework is particularly useful to take into account knowledge-based and observation-based uncertainty. The HPPN-based diagnoser deals with event occurrence possibility and knowledge imprecision. It monitors both discrete and continuous dynamics, as well as degradation evolution, in order to introduce concepts that will be useful to perform prognosis and health management of hybrid systems under uncertainty. In addition, diagnosis results can be used as probability distributions for decision making.

Then, the methodology was applied on the K11 rover. A hybrid model of the rover has been proposed by discretizing its health evolution and defining fault events. The system model and diagnoser have been generated in the HPPN framework and two scenarios have been tested to illustrate the proposed method advantages. The diagnoser results are consistent with the expected ones and show that HPPN-based diagnosis is robust to real system data and constraints and adaptable to systems without discrete observations nor degradation knowledge.

Other works aim at formalizing and developing a prognosis process that will interleave diagnosis and prognosis methods to obtain more accurate results. The HPPN-based prognostics methodology has been defined and tested on a three-tank system as well as on the K11 rover.

This work has a lot of interesting perspectives. The first one is the extension of the work to very large systems. To be applied on real large scale systems, the proposed methodology could be adapted in the context of decentralized diagnosis structures as the approaches developed in Sayed-Mouchaweh and Lughofer (2015). The second perspective deals with the major hypothesis on the HPPN model-based approach: the system model is assumed to be correct and complete. Machine learning techniques may be used to adapt this predefined model with new collected data [Khamassi et al. (2016); Leclercq et al. (2008)]. Another perspective is to use machine learning methods to improve the detection of small drift in the system parameters. The combination of model-based and data-driven approaches for diagnosis is under investigation [Jung et al. (2016), Tidriri et al. (2016), Toubakh and Sayed-Mouchaweh (2016)].

References

Balaban E., Narasimhan S., Daigle M. J., Roychoudhury I., Sweet A., Bond C., Celaya J. R., Gorospe G. (2013) Development of a mobile robot test platform and methods for validation of prognostics-enabled decision making algorithms. International Journal of Prognostics and Health Management **4**(006):1–19

Basile F., Chiacchio P., Tommasi G. D. (2009) Fault diagnosis and prognosis in Petri Nets by using a single generalized marking estimation. In: 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Spain

Bayoudh M., Travé-Massuyes L., Olive X. (2008) Hybrid systems diagnosis by coupling continuous and discrete event techniques. In: IFAC World Congress, Korea, pp. 7265–7270

Biswas G., Simon G., Mahadevan N., Narasimhan S., Ramirez J., Karsai G. (2003) A robust method for hybrid diagnosis of complex systems. IFAC Proceedings Volumes 36(5):1023–1028

Cardoso J., Valette R., Dubois D. (1999) Possibilistic Petri nets. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29(5):573–582

Chanthery E., Ribot P. (2013) An integrated framework for diagnosis and prognosis of hybrid systems. In: 3rd Workshop on Hybrid Autonomous System, Italy

Daigle M., Roychoudhury I., Bregon A. (2014) Integrated diagnostics and prognostics for the electrical power system of a planetary rover. In: Annual Conference of the PHM Society, USA

Daigle M., Roychoudhury I., Bregon A. (2015a) Qualitative event-based diagnosis applied to a spacecraft electrical power distribution system. Control Engineering Practice 38:75–91

Daigle M., Sankararaman S., Kulkarni C. S. (2015b) Stochastic prediction of remaining driving time and distance for a planetary rover. In: IEEE Aerospace Conference

David R., Alla H. (2005) Discrete, Continuous, and Hybrid Petri Nets. Springer

Ding S. X. (2014) Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems. Springer

Dotoli M., Giua A., Seatzu C., Fanti M. P. (2008) Modelling systems by hybrid petri nets: an application to supply chains. Petri Net, Theory and Application Advanced Robotic Systems

Douc R., Cappé O. (2005) Comparison of resampling schemes for particle filtering. In: Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on, IEEE, pp. 64–69

Doucet A., Johansen A. M. (2009) A tutorial on particle filtering and smoothing: Fifteen years later. Handbook of nonlinear filtering 12(656-704):3

Gaudel Q., Chanthery E., Ribot P. (2014a) Health monitoring of hybrid systems using hybrid particle Petri Nets. In: Annual Conference of the PHM Society, USA

Gaudel Q., Chanthery E., Ribot P., Le Corronc E. (2014b) Hybrid systems diagnosis using modified particle Petri nets. In: 25th International Workshop on Principles of Diagnosis, Austria

Gaudel Q., Chanthery E., Ribot P. (2015) Hybrid particle Petri Nets for systems health monitoring under uncertainty. International Journal of Prognostics and Health Management **6**(022):1–20

Henzinger T. (1996) The theory of hybrid automata. In: 11th Annual IEEE Symposium on Logic in Computer Science, pp. 278–292

Hofbaur M., Williams B. (2002) Mode estimation of probabilistic hybrid system. Hybrid Systems: Computation and Control Lecture Notes in Computer Science 2289:253–266

Horton G., Kulkarni V. G., Nicol D. M., Trivedi K. S. (1998) Fluid stochastic petri nets: Theory, applications, and solution techniques. European Journal of Operational Research 105(1):184–201

Jianxiong W., Xudong X., Xiaoying B., Chuang L., Xiangzhen K., Jianxiang L. (2013) Performability analysis of avionics system with multilayer hm/fm using stochastic Petri Nets. Chinese Journal of Aeronautics **26**(2):363–377

Jung D., Ng K. Y., Frisk E., Krysander M. (2016) A combined diagnosis system design using model-based and data-driven methods. In: 3rd Conference on Control and Fault-Tolerant Systems (SysTol), IEEE, pp. 177–182

Khamassi I., Sayed-Mouchaweh M., Hammami M., Ghédira K. (2016) Discussion and review on evolving data streams and concept drift adapting. Evolving Systems pp. 1–23

Koutsoukos X., Kurien J., Zhao F. (2002) Monitoring and diagnosis of hybrid systems using particle filtering methods. In: 15th International Symposium on Mathematical Theory of Networks and Systems, USA

Lachat D., Krebs A., Thueer T., Siegwart R. (2006) Antarctica rover design and optimization for limited power consumption. In: 4th IFAC Symposium on Mechatronic Systems

Leclercq E., el Medhi S. O., Lefebvre D. (2008) Petri nets design based on neural networks. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 529–534

Lesire C., Tessier C. (2005) Particle Petri Nets for aircraft procedure monitoring under uncertainty. In: Cardio, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol 3536, Springer, Heidelberg, pp. 329–348

Li T., Bolic M., Djuric P. M. (2015) Resampling methods for particle filtering: classification, implementation, and strategies. IEEE Signal processing magazine 32(3):70–86

Narasimhan S., Biswas G. (2007) Model-based diagnosis of hybrid systems. IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans 37(3):348–361

Narasimhan S., Brownston L. (2007) Hyde–a general framework for stochastic and hybrid model-based diagnosis. 18th International Workshop on Principles of Diagnosis 7:162–169

Narasimhan S., Browston L. (2007) HyDE - a general framework for stochastic and hybrid modelbased diagnosis. In: 18th International Workshop on Principles of Diagnosis, pp. 162–169

Narasimhan S., Dearden R., Benazera E. (2004) Combining particle filters and consistency-based approaches for monitoring and diagnosis of stochastic hybrid systems. In: 15th International Workshop on Principles of Diagnosis

Narasimhan S., Balaban E., Daigle M., Roychoudhury I., Sweet A., Celaya J., Goebel K. (2012) Autonomous decision making for planetary rovers using diagnostic and prognostic information. In: 8th IFAC Symposium on Fault Dectection, Supervision and Safety of Technical Processes, Mexico, pp. 289–294

Ru Y., Hadjicostis C. N. (2009) Fault diagnosis in discrete event systems modeled by partially observed Petri Nets. Discrete Event Dynamic Systems **19**(4):551–575

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

Sayed-Mouchaweh M., Lughofer E. (2015) Decentralized approach without a global model for fault diagnosis of discrete event systems. International Journal of Control 88(11):2228–2241, DOI 10.1080/00207179.2015.1

Soldani S., Combacau M., Subias A., Thomas J. (2007) On-board diagnosis system for intermittent fault: Application in automotive industry. In: 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems, vol 7-1, pp. 151–158, DOI 10.3182/20071107-3-FR-3907.00021

Sweet A., Gorospe G., Daigle M., Celaya J. R., Balaban E., Roychoudhury I., Narasimhan S. (2014) Demonstration of prognostics-enabled decision making algorithms on a hardware mobile robot test platform. In: Annual Conference of the PHM Society, USA

Tidriri K., Chatti N., Verron S., Tiplica T. (2016) Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. Annual Reviews in Control 42:63–81

Toubakh H., Sayed-Mouchaweh M. (2016) Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. Neurocomputing 171:1496–1516

van der Merwe R., Doucet A., De Freitas N., Wan E. (2000) The unscented particle filter. In: Annual Conference on Neural Information Processing Systems, vol 2000, pp. 584–590

Vianna W. O. L., Yoneyama T. (2015) Interactive Multiple-Model Application for Hydraulic Servovalve Health Monitoring. In: Annual Conference of the PHM Society, USA

Wang W., Li L., Zhou D., Liu K. (2007) Robust state estimation and fault diagnosis for uncertain hybrid nonlinear systems. Nonlinear analysis: Hybrid systems 1(1):2–15

Yu M., Wang D., Luo M., Huang L. (2011) Prognosis of Hybrid Systems With Multiple Incipient Faults: Augmented Global Analytical Redundancy Relations Approach. IEEE Transactions on Systems, Man, and Cybernetics 41, Part A(3):540–551

Zaidi A., Zanzouri N., Tagina N. (2006) Modelling and monitoring of hybrid systems by hybrid Petri nets. In: 10^{th} WSEAS International Conference on Systems

Zhao F., Koutsoukos X., Haussecker H., Reich J., Cheung P. (2005) Monitoring and fault diagnosis of hybrid systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35(6):1225–1240

Zouaghi L., Alexopoulos A., Wagner A., Badreddin E. (2011) Modified particle Petri Nets for hybrid dynamical systems monitoring under environmental uncertainties. In: IEEE/SICE International Symposium on System Integration, pp. 497–502