Hybrid Particle Petri Net Based Prognosis of a Planetary Rover

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

Hybrid Particle Petri Net Based Prognosis of a Planetary Rover

Pauline Ribot, Elodie Chanthery, Quentin Gaudel, and Matthew J. Daigle, *Senior Member, IEEE*

***Abstract*—This paper describes a model-based prognosis method for the health management of a planetary rover. Using a hybrid model of the rover, including a continuous part and a discrete part, a prognoser is generated that relies on the Hybrid Particle Petri Nets (HPPN) data structure. The prognosis process uses the current diagnosis of the system to predict its future states and to determine its End Of Life (EOL) or its Remaining Useful Life (RUL). The HPPN-based prognoser is initialized with a Stochastic Scaling Algorithm (SSA) that selects the diagnosis hypotheses with the highest beliefs. The SSA provides a compromise between performance and available computational resources through the setting of scaling parameters. The prognoser then uses the future commands to determine the hypotheses over the rover future trajectory and the RUL/EOL. The set of the future hypotheses associated with their belief degrees forms the current rover prognosis. The prognosis method is tested on different scenarios, with different scaling parameters, considering the future commands are known or not. Experimental results show that the approach is robust to real system data and computational performance constraints.**

***Index Terms*—prognosis, hybrid systems, diagnosis, model-based monitoring, health management, uncertainty.**

I. Introduction

Real systems, such as planetary rovers, have become so complex that it is often impossible for humans to capture and explain their behaviors, especially when they are exposed to failures. Prognostics and System Health Management (PHM) including an efficient health monitoring technique has to be adopted to detect, isolate and predict faults leading to failures. A diagnosis method is used to determine the current state from observations and identify the possible causes of failures that lead to this state. Prognosis is the prediction of the future states and the times of the fault occurrences that lead to these states. It is related to the determination of the End Of Life (EOL) of the system, that is the time at which the system is not operational anymore and of the Remaining Useful Life (RUL), that is the remaining period before it reaches the EOL [1].

A system is considered as a hybrid system if it exhibits both discrete and continuous dynamics [2]. Hybrid systems are convenient to describe systems that have multiple operating modes. Modes of such multimode systems represent their continuous evolutions (continuous dynamics) under different conditions [3]. A discrete-event system (DES) defines the changes of modes (discrete dynamics) with occurrences of events. The system discrete state is the current discrete state of the DES. The evolution of the system continuous state depends on continuous dynamics associated with the current system mode. Here, we are interested in modeling changes in system dynamics when one or several anticipated faults occur. As long as the system does not encounter any fault, it is in nominal 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), in which the system realizes its mission either partially or under special conditions. Without repair, the system ends in a failure mode, in which it cannot realize its mission anymore.

Finally and for the sake of generality, we consider that a system can work with discrete and continuous variables (state variables, parameters, inputs and outputs), and that these variables are subject to uncertainty, particularly for real system studies. In such cases, the gap between models and reality must be taken into account, as for the discrete part (symbolic uncertainty) as the continuous part (numerical uncertainty) of the system. Symbolic uncertainty relates to inexact or improbable event sequences in the DES and to missing or false event observations. Numerical uncertainty relates to imprecise continuous dynamics and to noisy numerical data.

Our previous works introduced a data structure called Hybrid Particle Petri Nets (HPPN) that aims at capturing all the system knowledge. It is inspired by Petri nets and state representations used in particle filters. In particular, the HPPN marking contains all the necessary knowledge on the system to determine its diagnosis and prognosis. In [4], we proposed to use HPPN to generate a diagnoser from a multimode description of a hybrid system. The diagnoser tracks the system current state under uncertainty. Its structure is a HPPN and its process is based on particle filters. System observations (inputs and outputs) are used to update the diagnoser marking and determine the diagnosis. A diagnosis at any time contains the hypotheses over its past trajectories. Each hypothesis is valued with a belief degree and includes estimates of the system discrete and continuous states, as well as the set of faults that occurred on the system until the current time. In [5], we tested the diagnosis approach on a simulated three-tank system.

In this paper, we enrich the health monitoring approach with a prognoser, that aims at computing the prognosis of the system under uncertainty, based on the current diagnosis and future inputs. The prognoser structure is a HPPN generated from a multimode description of the hybrid system.

This paper has four main technical contributions:

P. Ribot and E. Chanthery are with LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse F-31400, France (e-mail: pauline.ribot; elodie.chanthery@laas.fr).

Q. Gaudel is with EasyMile, 8 Rue des Trente Six Ponts, 31400 Toulouse, France (email: quentin.gaudel@easymile.com).

M.J. Daigle is with NIO, 3200 N 1st St, San Jose, CA 95134, USA (e-mail: matthew.daigle@nio.io).

1) We defined a prognosis methodology that meets the following criteria. (i) It manages hybrid systems that exhibit both discrete and continuous dynamics. (ii) It manages uncertainty related to inexact or improbable event sequences, to missing or false event observation, to continuous dynamics (imprecise dynamics and noisy numerical data), to current state and to future inputs. (iii) It takes into account an estimation of the current health state of the system given by a diagnosis process.
2) We implemented the methodology and quantitatively computed a RUL for a real system. This implementation faces the challenge of computational performances by proposing an original algorithm named the Stochastic Scaling Algorithm (SSA) that balances precision and computation time. All the codes were written in Python and are available upon request.
3) We experimented the methodology on the K11 planetary rover prototype of the NASA Ames Research Center. The K11 hybrid model proposed in [6] is used to generate the HPPN-based prognoser. The model and the two scenarios data are available on the web[1].
4) We proved that the methodology is relevant for real system data and constraints by evaluating the performance of the method with three metrics: diagnosis and prognosis computation times, and the maximum RAM used are studied for each scenario. Different scenarios with different types of future inputs are tested.

This paper is organized as follows. Section II gives some related work on diagnosis, extended diagnosis and prognosis for hybrid systems. Section III presents the hybrid model of the K11 health evolution. Section IV explains the prognosis method including the use of the SSA, an algorithm that is used to initialize the prognoser given the current system diagnosis and three scaling parameters. Section V provides results obtained by testing the proposed method on the K11 on two scenarios studied in [7]. Conclusions and future works are discussed in the final section.

II. Related Work

Hybrid systems are the center of interest of numerous researches in many areas, such as modeling, verification, control, and monitoring. In system modeling, different formalisms have been introduced to represent hybrid dynamics: hybrid automata (HA) [3], hybrid bond graphs (HBG) [8], [9], hybrid Petri nets (HPN) [10], etc. Such formalisms have largely been used or extended for hybrid system diagnosis. Structural approaches [11]–[13] or conflict-based approaches [14] have also been used for diagnosis purpose.

Figure 1 presents some references to these works and compare the performances of the other approaches cited in this Section.

Some works particularly focus on diagnosis with the intent to use it for prognosis purposes, and then generally consider degradation monitoring. This is what we call "Extended Diagnosis" in Figure 1. In [15], isolation of faults is dynamically proceeded with a hybrid bond graph (HBG). A mode

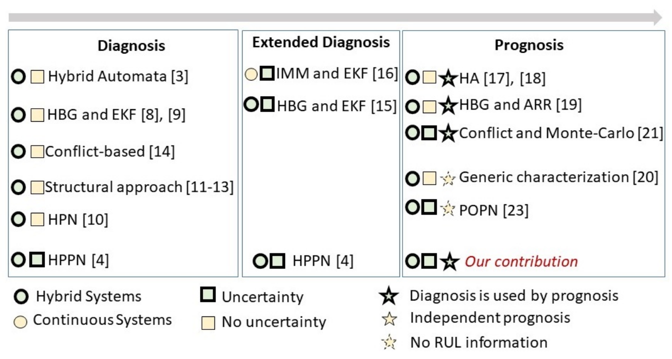


Fig. 1. From Diagnosis to Prognosis.

dependent fault signature matrix is proposed and a waiting time is used to allow all faults to exhibit their symptoms on residuals, especially faults that are only detected with continuous signals. In [16], the diagnosis technique monitors both the system behavior and its degradation in order to have a better estimate as a start for the prognosis process via the Interactive Multiple-Model (IMM) algorithm, but the approach is limited to continuous systems. We aim at extending the diagnosis and the prognosis processes to hybrid systems. With the same purpose, we propose in [4] to use HPPN to monitor the hybrid system degradation in addition to its behavior, considering many sources of uncertainty. However, this previous work only deals with extended diagnosis and do not address the prognosis problem.

In [17], hybrid automata are used to model the system and to generate a prognoser that determines the system RUL. The method is demonstrated on a simulated study case in [18]. Even if both the behavior and the degradation are monitored, only the degradation knowledge is used for the prognosis process, making the approach limited. The approach does not take into account uncertainties. In [19], the authors propose to develop a model-based sequential failure prognosis for hybrid systems where some faults are detectable only after a waiting time. Dynamic fault isolation is performed with Hybrid Bond graph. The degradation behavior of each component is mode dependent and estimated by a hybrid differential evolution algorithm. The RUL of the component is computed by the estimation of the degradation and a threshold given by the user. A sequential prognosis algorithm, including a standard prognosis module and auxiliary module, is proposed. These works do not take into account uncertainty about observations and models. In [20], a generic characterization of the diagnosis and prognosis processes is proposed. The approach does not take into account uncertainties and does not provide any quantitative value for the RUL. In [21], the authors extend the model-based prognostics paradigm to hybrid systems. It relies on previously established methods for hybrid state estimation, and provides an approach to predict the RUL/EOL given a hybrid model, a hybrid state estimate, and a specification of future unknown input. It describes how the resulting probability distribution for the RUL/EOL may be multi-modal due to mode-switching in the predicted future behavior. Petri nets-based approaches

[1]https://homepages.laas.fr/echanthe/K11.

often deal with the prediction of event occurrences from a predictability perspective, i.e. the system monitoring indicates either a fault can still occur, or not. The authors of [22] use a generalized marking to consider unobservable event occurrences while minimizing the state space explosion, whilst the problem is approached with Partially Observed Petri Nets (POPN) in [23]. These works do not however provide any quantitative information about the RUL estimates.

This paper aims at providing a quantitative information about the RUL estimation. Prognosis process will be proceed in a highly uncertain framework, taking into account among others uncertainty on the health state of the system, on the observations and on the future actions. The health state of the system will be estimated by the diagnosis process and used by the prognosis process for the initialization step. By comparison with [21], we use a graphical formalism that looks like a Petri Net in order to be more compact and expressive. The HPPN data structure also aims at differentiating the continuous part and the discrete one from the degradation part of the system.

III. K11 Model for Prognostics

The K11 is a four-wheeled electric rover that is used by NASA Ames Research Center for diagnostics and prognostics-enabled decision making research [7], [24], [25]. It is a test bed on which some fault and failures can be simulated. In this work, it is studied as an operating rover exposed to failures and executing missions. Its inputs are the wheel speed commands and its outputs are sensor measurements. This section presents the K11 hybrid model proposed in [6] based on the discretization of its health evolution.

A. Platform 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 way-points and joining its charging station before the complete discharge of its battery. The rover wheels are 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 motor.The rover is a skid-steered vehicle, meaning that the wheels cannot be steered and it is rotated by commanding the wheel speeds on the left and right sides to different values. An onboard laptop computer runs a proportional-integral-derivative controller and a data acquisition software. Available data are provided by 65 continuous signals: the 4 commanded wheel speeds and the 61 sensor measurements that include the actual wheel speeds, the total current, the motor currents and temperatures and the voltages and temperatures of each of the 24 battery cells. More details on the K11 model can be found in [24], in which it is studied as a continuous system.

Several fault types can occurred on the rover. These fault occurrences are considered as unobservable discrete events and their effects are summarized in the following subsection.

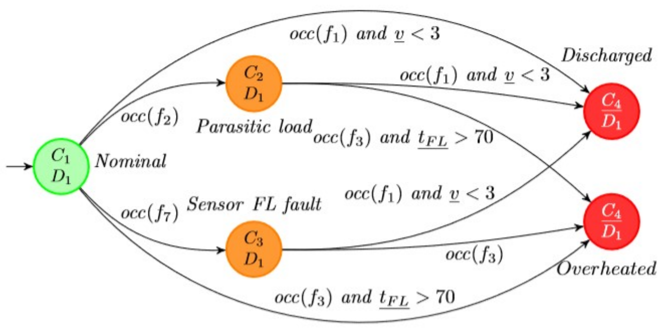

C_1 D_1
Nominal
C_2 D_1
Parasitic load
C_3 D_1
Sensor FL fault
C_4 D_1
Discharged
C_4 D_1
Overheated
$occ(f_1)$ and $\underline{v} < 3$
$occ(f_2)$
$occ(f_1)$ and $\underline{v} < 3$
$occ(f_3)$ and $\underline{t_{FL}} > 70$
$occ(f_7)$
$occ(f_1)$ and $\underline{v} < 3$
$occ(f_3)$
$occ(f_3)$ and $\underline{t_{FL}} > 70$

Fig. 2. Streamlined description of the K11 health evolution.

B. Modeling for Health Monitoring

A simplified multimode description of the rover health evolution, limited to the FL motor part, is presented in Figure 2. The entire rover health evolution can obviously be deduced from this description as the four motors have the same behavior. This simplified view of the system does not include multiple-fault modes. Graphically, the underlying graph vertices represent the modes and the edges represent the mode switches. Variables that can be observed or estimated with observations are underlined. In each mode, associated continuous dynamics and degradation dynamics are indicated.

As long as no fault has occurred, the rover is in mode $Nominal$ with continuous dynamics C_1. Fault f_1 occurrence represents the End Of Discharge (EOD) of the battery, i.e. the time when the battery is too discharged to power the system. This fault is assumed to occur when the battery voltage is lower than 3 V (approximately) and it leads to the mission failure (mode $Discharged$). Fault f_2 represents the emergence of a parasitic battery load.The parasitic load increases the total current and thus the battery drain (mode $Parasitic\ load$ with continuous dynamics C_2), which causes the system to reach the EOD prematurely. The most feared scenario for a motor is an overheating: the heat will eventually destroy the insulation of the windings, causing electrical shorts and leading to motor failure. The overheating of the FL (resp. FR, BL and BR) motor is represented by fault f_3 (resp. f_4, f_5 and f_6). The occurrence of any one of these faults leads to the rover failure (mode $Overheated$ with continuous dynamics C_4) and thus represents the rover 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_7 f_8, f_9 and f_{10}. We consider that the temperature model is not accurate enough without a correction step with observations. As a consequence, once fault f_7 (resp. f_8, f_9 and f_{10}) has occurred, the occurrence of fault f_3 (resp. f_4, f_5 and f_6) does not match with any condition on the FL (resp. FR, BL and BR) motor temperature (see the arc between $Sensor\ FL\ fault$ and $Overheated$). In Figure 2, mode $Sensor\ FL\ fault$ with continuous dynamics C_3 represents the mode where the temperature sensor of the FL motor has failed.

As no degradation law is *a priori* known for this rover, all modes have the same degradation dynamics D_1.

Considering all the motors and multiple faults, 192 modes and 240 mode switches can be identified[2].

Continuous dynamics and degradation dynamics can be represented by differential equations. Because of the lack of knowledge about the rover components degradation in the NASA Ames Research Center model, all modes have the same degradation dynamics D_1 that has been defined as the identity function. C_1 is a set of differential equations that unifies the battery model with the rover motion model and the temperature models in the nominal case. It can be converted into a discrete-time representation and solved with a sample time of 1/20s, while the continuous observation sampling time is about 1s:

$$C_1 = \begin{cases} x_{k+1} = \mathbf{f}(x_k, u_k) + \mathbf{v}(x_k, u_k) \\ y_k = \mathbf{h}(x_k, u_k) + \mathbf{w}(x_k, u_k) \end{cases}, \quad (1)$$

where $x_k \in \mathbb{R}^{n_x}$ is the vector of the n_x continuous state variables, $u_k \in \mathbb{R}^{n_u}$ is the vector of n_u continuous input variables ($n_u = 4$), $\mathbf{f}$ is the noiseless continuous state equation, $\mathbf{v}$ is the continuous process noise equation, $y_k \in \mathbb{R}^{n_y}$ is a vector of n_y continuous output variables ($n_y = 61$), $\mathbf{h}$ is the noiseless continuous output equation, and $\mathbf{w}$ is the continuous output noise equation.

We consider 32 state variables ($n_x = 32$), including the rover 2-dimensional position (x, y), 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. Unifying the battery, motion, and temperature models, however, highly increases complexity and uncertainty.

Fault f_2 occurrence and effect 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 f_2 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 and 0.01 as 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. Faults f_7, f_8, f_9, f_{10} are modeled by increasing significantly the motor temperature sensor noise because failed sensors only send inconsistent large values.

IV. Hybrid System Prognosis

Prognosis aims at predicting the system future states and its RUL/EOL from diagnosis and future inputs available from a mission scenario for example. During an arbitrary prediction horizon τ, the goal is to determine if and when the system will enter a failure mode and will not be operational anymore.

[2]The complete underlying DES of the K11 hybrid model is available at https://homepages.laas.fr/echanthe/K11.

We propose to use the Hybrid Particle Petri Nets (HPPN) data structure to generate three different objects: a model, a diagnoser and a prognoser from a description of the hybrid system. An overview of the health monitoring architecture is given in Algorithm 1 and illustrated in Figure 3.

Algorithm 1 HPPN-based monitoring methodology

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

Although the health monitoring methodology obviously includes diagnosis and prognosis, this paper focuses on the prognoser generation and on the prognosis computation (lines 3 and 7). Functions $GenerateHPPNDiagoser$ and $Update$ are covered in previous work [6] and are assumed to be given.

The first offline step (line 1) is the generation of the HPPN model $HPPN_\Phi$ of the system. It can be directly built from a multimode description or created from expert knowledge. For example, for the K11, it is generated from the K11 hybrid multimode description given in Section III. The second offline step (line 2) is the automatic generation of a HPPN-based diagnoser $HPPN_\Delta$ from the system model $HPPN_\Phi$. The last offline step (line 3) is the automatic generation of a HPPN-based prognoser $HPPN_\Pi$ from the system model $HPPN_\Phi$.

The online process (line 4-8) uses the system consecutive observations O_k (discrete and continuous inputs and outputs) to update the diagnoser marking [6]. This marking contains all diagnosis hypotheses. The diagnosis Δ_k is given by the marking of the HPPN-based diagnoser $HPPN_\Delta$ and represents a distribution of beliefs obtained by particle filtering. To compute the system prognosis Π_k at time k (line 7), the prognoser is initialized with Δ_k and its marking evolves according to a given set of future inputs U_k^+. The set $U_k^+ = \{u_\kappa | \kappa \in \{k, ..., k+\tau\}\}$ includes the system future inputs during the prediction horizon $\tau \in \mathbb{N}$, where $u_\kappa \in \mathbb{R}^{n_u}$ is the continuous input vector at future time κ. The prognosis Π_k is defined as the marking of the prognoser at the end of this process.

Uncertainty and computational performance are two substantial challenges for prognostics that cannot be ignored. The first one is related to the result accuracy and precision, whereas the second one is related to calculation time and computational resource management. Those challenges for prognostics are explained in the next subsections.

A. Uncertainty: a Challenge for Prognostics

1) Uncertainty about Diagnosis: The knowledge about the current state is a significant source of uncertainty for the prognosis process. In this work, we use the current diagnosis that contains all the hypotheses on the system. Diagnosis is performed by combining pseudo firing of transitions in the

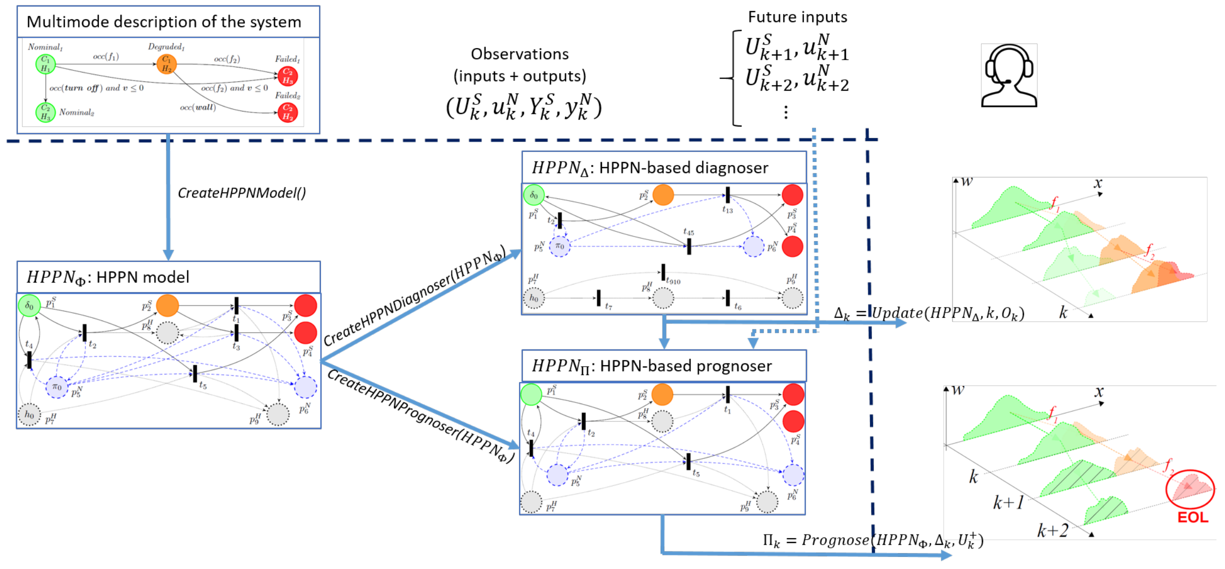

Multimode description of the system
Future inputs
Observations
(inputs + outputs)
$(U_k^S, u_k^N, Y_k^S, y_k^N)$
U_{k+1}^S, u_{k+1}^N
U_{k+2}^S, u_{k+2}^N
CreateHPPNModel()
$HPPN_\Delta$: HPPN-based diagnoser
$HPPN_\Phi$: HPPN model
CreateHPPNDiagnoser($HPPN_\Phi$)
CreateHPPNPrognoser($HPPN_\Phi$)
$\Delta_k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

Fig. 3. Overview of the health monitoring architecture.

HPPN-diagnoser and particle filtering for continuous dynamics. Each diagnostic hypothesis provides a possible continuous state (represented with a distribution of particles), a possible discrete state (a mode of the DES) and a set of events that leads to the discrete state. The prognosis process predicts the system future states by simulating the future evolution of the diagnosis hypotheses. It also predicts events that will occur on the system. Particularly, from a hypothesis in Δ_k, we predict the possible sets of events that would lead the system to failure modes. Mode switches are simulated when some conditions on the continuous state vector are satisfied. The time of occurrence of an event that leads to a failure mode is a possible EOL of this hypothesis.

2) Uncertainty about Continuous Dynamics: Another source of uncertainty is continuous dynamics noise. In the K11 case study, fault f_2 time varying parameters are considered as state variables (see Section III-B) and are estimated during the diagnosis process. Process noise is negligible relative to other sources of uncertainty in prognostics so we choose to ignore it.

3) Uncertainty about Future Inputs: In real systems, determining the set of future inputs U_k^+ is another source of uncertainty. Low level commands are difficult to predict accurately, even if the high level mission plan is known. Considering noise on mission plan and future commands is investigated as a solution to that problem. For the rover case study, we then consider three kinds of future input scenarios: (case 1) future commands are accurately known, (case 2) future commands are set to the maximum speed (the K11 is believed to go straight), (case 3) future commands are stochastically generated (the K11 is believed to move randomly).

B. Computational Performances Challenge

Sequential prognosis (based on simulation) for continuous systems are computationally expensive. In particle filter-based prognostics, the computation time is proportional to the number n of particles to simulate, where $n \in \mathbb{N}$ can be considered as the precision of the predictions. In HPPN-based monitoring, each simulated diagnosis hypothesis has its own set of particles and the difficulty is to select which hypotheses to select as initial condition for prognosis. In order to scale the prognosis computation time, we propose an original algorithm named the Stochastic Scaling Algorithm (SSA), to select which hypotheses to simulate and the precision with which they will be simulated. By setting three scaling parameters ρ_Π^{min}, ρ_Π^{max} and ρ_Π^{tot}, the operator is able to influence the prognoser performances. Parameters ρ_Π^{min} and ρ_Π^{max} are respectively the minimum and maximum precision needed to simulate the evolution of a hypothesis. Parameter ρ_Π^{tot} is the total particle number available to simulate the evolution of all the hypotheses.

The SSA determines, from a given finite set of values distributed on a starting range, a new set of values distributed on a specific arrival range. The objective is on one hand to change values range and one the other hand to change their cardinal. This scaling is a Monte Carlo process and uses three scaling parameters ρ^{min}, ρ^{max} and ρ^{tot}. Parameters ρ^{min} and ρ^{max} are respectively the minimum and maximum bounds of the arrival range, and ρ^{tot} indicates the maximum sum of the new set values. As the SSA is also used for diagnosis, these parameters are denoted ρ_Δ^{min}, ρ_Δ^{max} and ρ_Δ^{tot} for the diagnoser and ρ_Π^{min}, ρ_Π^{max} and ρ_Π^{tot} for the prognoser.

The SSA is given in Algorithm 2. It can be presented as a resource allocation problem solver where there are $N_\mathbb{J}$ jobs to realize, here $N_\mathbb{J}$ hypotheses to simulate. Belief degrees are associated with hypotheses to indicate their importance and are given in input as weights. Each hypothesis has a precision, considered here as resource, that is the number of particles used to simulate it. The higher the number of resources is allocated to a job, the better the job will be realized: in other words, the higher the number of particles is allocated to a

Algorithm 2 Stochastic Scaling Algorithm

```
Input: $\mathbb{B}$   ▷ list of the $N_\mathbb{J}$ hypotheses belief degrees
Input: $\rho^{min}, \rho^{max}, \rho^{tot}$   ▷ scaling parameters
Output: $R$   ▷ list of the particle numbers allocated to the $N_\mathbb{J}$ hypotheses
 1: $J \leftarrow (1, ..., N_\mathbb{J})$   ▷ list of IDs of hypotheses to provide (all at start)
 2: $R \leftarrow (0|\forall i \in t)$   ▷ list of particles numbers allocated to hypotheses
 3: $r \leftarrow \rho^{tot}$   ▷ remaining particle number (all available at start)
 4:
 5: while $r > 0$ and $|J| > 0$ do
 6:                                  ▷ it remains resources and jobs to provide
 7:     $\mathbb{B}^J \leftarrow (\mathbb{B}_i|\forall i \in J)$   ▷ belief degrees of the hypotheses to provide
 8:     for $o \in \{1, ..., r\}$ do   ▷ repeat r times
 9:         $J_c \leftarrow Pick(J, \mathbb{B}^J)$   ▷ pick over a normal distribution
10:         $R_{J_c} \leftarrow R_{J_c} + 1$   ▷ allocate a particle to the hypothesis
11:     end for
12:     for all $R_i \in R$ do
13:         if $R_i > \rho^{max}$ then   ▷ constraint 2 not satisfied
14:             $R_i \leftarrow \rho^{max}$   ▷ handover the overload
15:         end if
16:         if $R_i < \rho^{min}$ then   ▷ constraint 1 not satisfied
17:             $R_i \leftarrow 0$   ▷ handover the underload
18:         end if
19:     end for
20:
21:     $J \leftarrow (J_i|\forall J_i \in J, R_{J_i} < \rho^{max})$   ▷ keep hypotheses to provide
22:     $r^n \leftarrow \rho^{tot} - \sum_{R_i \in R} R_i$   ▷ new remaining particle number
23:     if $r^n = r$ then   ▷ no new allocation
24:         $r \leftarrow 0$   ▷ stop algorithm
25:     else
26:         $r \leftarrow r^n$   ▷ update remaining particle number
27:     end if
28: end while
```

hypothesis, the better the hypothesis will be simulated. The problem consists in allocating the ρ^{tot} available particles to the $N_\mathbb{J}$ hypotheses depending on their belief degrees.

Two constraints are added to the problem. Firstly, a hypothesis can only be simulated if a sufficient number of particles ρ^{min} is allocated to it (constraint 1). Secondly, it is forbidden to allocate more than ρ^{max} particles to a hypothesis because it does not improve the simulation (constraint 2).

The proposed solution to the problem is based on two steps: (1) the random allocation of all the available particles using a normal distribution depending on the hypotheses belief degrees (lines 8-11), (2) the handover of all the particles allocated to hypotheses that do not satisfy constraint 1 (underload) (lines 16-18) and the handover of the unnecessary particles allocated to hypotheses that do not satisfy constraint 2 (overload) (lines 13-15). These two steps are repeated until no more particles are available (line 5) or that the allocation remains unchanged after the execution of the two steps (line 24).

The SSA can return undesirable results with some extreme scaling parameter setups and belief distributions. For instance, the scaling of the sample example with parameters $(\rho^{min}, \rho^{max}, \rho^{tot}) = (20, 120, 1000)$ never assigns any particle to the hypotheses, because the total number of available particles is too low (1000) compared to the number of hypotheses (200) and the minimum number of particles to allocate to each one of them (20). In such cases, a recovery mechanism allocates the maximum number of particles ρ^{max} to the hypotheses with the highest belief degrees until no more particle is available. A computational performance study comparing different values of the scaling parameters is provided in Section V-B. The choice of these parameters highly depends on the constraints on the computation time and the precision required on the results.

C. Prognoser Generation

The prognoser is defined with the HPPN framework. The HPPN structure, introduced in [4], is an extension of Petri nets. The main advantage of the HPPN structure is that it can capture heterogeneous knowledge about the system, such as discrete evolutions with symbolic places, continuous evolutions with numerical places, and relations between each others with transitions via degradation.

Let us consider the HPPN model $HPPN_\Phi$ of the system. The prognoser $HPPN_\Pi$ derived from $HPPN_\Phi$ is a tuple $\langle P, T, A, \mathcal{A}, E, X, D, \mathcal{C}, \mathcal{D}, \Omega, \mathbb{M}_0 \rangle$ where:

- P is the set of places, partitioned into symbolic places P^S, numerical places P^N and degradation places P^D,
- T is the set of transitions,
- $A \subset P \times T \cup T \times P$ is the set of arcs,
- $\mathcal{A}$ is the set of arc annotations,
- E is the set of event labels,
- $X \subset \mathbb{R}^{n_N}$ is the state space of the continuous state vector, with $n_N \in \mathbb{N}_+$ the number of continuous state variables,
- $D \subset \mathbb{R}^{n_D}$ is the state space of the degradation state vector, with $n_D \in \mathbb{N}_+$ the number of degradation state variables,
- $\mathcal{C}$ is the set of dynamic equation sets associated with numerical places, representing continuous dynamics,
- $\mathcal{D}$ 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.

The different steps of the generation of the HPPN-prognoser $HPPN_\Pi$ are described in detail in [26].

The set of the event labels $E = E_o \cup E_{uo}$ is partitioned into observable event labels E_o and unobservable event labels E_{uo}. An event is defined as a couple $e = (v, k)$ where $v \in E$ is an event label and k the time of occurrence of e.

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

Numerical places P^N model the different continuous evolutions of the system. A numerical place $p^N \in P^N$ is associated with a set of equations $C \in \mathcal{C}$ modeling system continuous dynamics without their associated numerical uncertainty. Noise functions $\mathbf{v}$ and $\mathbf{w}$ in Equation (1) are removed in the prognoser. Numerical places are marked with particles. The set of all particles at time k is noted M_k^N. A particle $\pi_k \in M_k^N$ is a

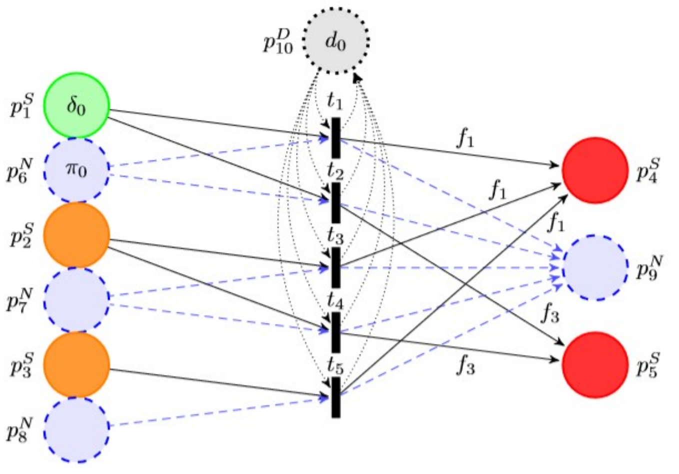


Fig. 4. Prognoser $HPPN_\Pi$ of the K11 rover.

numerical token at time k whose value is a possible continuous state $x_k \in X$ of the system at time k.

A couple 'symbolic place' and 'numerical place' models a system operating mode. A symbolic or a numerical place can be used to model several modes (for example, if two modes have the same continuous evolution, they can be modeled with two symbolic places and only one numerical place). Figure 4 shows the HPPN-based prognoser generated with the simplified hybrid model of the K11 rover presented in Figure 2. The 5 symbolic places (plain places p_1^S,...,p_5^S) represent the 5 discrete states of the hybrid automaton ($Nominal$,...,$Overheated$). The 4 numerical places (dashed places p_6^N,..., p_9^N) represent the 4 continuous evolutions (C_1,..., C_4). Mode $Nominal$ is thus fully represented with places p_1^S and p_6^N. Modes $Discharged$ and $Overheated$ are symbolically represented with respectively places p_4^S and p_5^S, and they are both numerically represented with the same numerical place p_9^N that is associated with continuous dynamics C_4.

Degradation places P^D and dynamics $\mathcal{D}$ model the system degradation evolutions associated to modes [5] and are marked with degradation tokens M^D. In this work, our knowledge on the rover does not encompass such parallel evolution. As a consequence, we use only one degradation place (dotted place p_{10}^D in Figure 4) in $HPPN_\Pi$ that contains all the degradation tokens at any time k. Any degradation token $d_k \in M^D$ links one configuration with one particle. These links are a particularity of the HPPN data structure. We note $link(\delta_k, \pi_k, d_k) = \top$, where $\top$ is the true value, if the degradation token d_k links the configuration δ_k with the particle π_k.

These links are required to represent hypotheses on the system trajectories, i.e. its discrete and continuous states at time k and the set of events that occurred on it until time k. We define formally a diagnosis hypothesis at time k as a set of tokens $\{\delta_k, \pi_k^1, ..., \pi_k^{n_k}, d_k^1, ..., d_k^{n_k} | \forall i \in \{1, ..., n_k\}, link(\delta_k, \pi_k^i, d_k^i) = \top\}$ that contains all the knowledge on the continuous state at time k with the particles and the events occurred until time k with the configuration. The subset of particles represents the distribution of probability over the continuous state (as in particle filters). The number of particles n_k is called the precision of the hypothesis at time k. Here, degradation tokens ensure the consistency of the hypothesis by linking a configuration with particles.

The marking $\mathbb{M}_k$ of the HPPN-based prognoser is the distribution of the configurations, the particles and the degradation tokens in the different places at time k. The set M_k denoted the set of all tokens at time k:

$$M_k = M_k^S \cup M_k^N \cup M_k^D. \quad (2)$$

Transitions in the prognoser model mode switches. A transition $t \in T$ will have as input (resp. output) the set of places representing the outgoing (resp. ingoing) mode of the switch.

A transition $t \in T$ may be fired:

- if there is at least one token in any of its input places,
- these tokens are linked together,
- with respect to its associated set of conditions $\Omega_t \in \Omega$ describing the mode switches.

A condition $\omega_t : M_k \to \mathbb{B}$, with $\omega_t \in \Omega_t$ and $\mathbb{B} = \{\top, \bot\}$ can be either a test on a token value, always satisfied ($\top$), or never satisfied ($\bot$).

The prognosis process works with future inputs only (discrete input events and continuous input vectors). As a consequence, only the mode switches triggered by continuous variables or input events are modeled in the prognoser. For the K11 case study, there is no discrete input event, so a transition is created only for the mode switches triggered by conditions on continuous state variables. It means that any set Ω_t is composed of only one numerical condition ω_t^N. For example in Figure 4, only switches triggered by $\underline{v} < 3$ or $\underline{t_{FL}} < 70$ are considered in the prognoser of the K11 as transitions t_1, t_2, t_3, t_4 and t_5. With this reasoning, fault occurrence conditions are ignored in the prognoser.

Arc annotations $\mathcal{A}$, however, are used in the prognoser to update the configuration values during the transition firing and thus record fault occurrences. An arc $a \in A$ that connects a transition t to a symbolic place p^S, may be annotated with an event label $v \in E$. In such a case, a configuration δ that is moved to p^S after the firing of t at time k, sees its event set b updated with the event (v, k). In Figure 4, each arc that connects a transition to a symbolic place is annotated with the fault label that conditions the mode switch in Figure 2.

D. Prognosis Process

The prognosis process may be performed at any time k and provides a prognosis Π_k, as shown in Algorithm 1, line 7. Its inputs are the HPPN-based prognoser $HPPN_\Pi$ generated from a hybrid model $HPPN_\Phi$, the diagnosis Δ_k and the set U_k^+ of the future system inputs at time k. Diagnosis Δ_k is provided by the marking of HPPN-based diagnoser $HPPN_\Delta$ at time k [5], [6] and thus contains hypotheses over the system past trajectories. Each diagnosis 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 until time k. The prognosis process steps are given in Algorithm 3.

Algorithm 3 Prognose

```
Input: $HPPN_\Pi, \Delta_k, U_k^+$
Output: $\hat{M}_{k_{\mathrm{EOP}}|k}$
1: $Initialize(HPPN_\Pi, \Delta_k)$
2: for all $U_\kappa \in U_k^+$ do
3:     if $AllHypothesesFailed(\hat{M}_{\kappa-1|k})$==0 then
4:         $\hat{M}_{\kappa|k} \leftarrow Update(HPPN_\Pi, \kappa, u_\kappa)$
5:     else
6:         break for loop
7:     end if
8: end for
9: $k_{\mathrm{EOP}} \leftarrow \kappa$
```

1) Prognoser Initialization: The prognoser is initialized from the diagnosis Δ_k (line 1). To keep diagnosis uncertainty, the initial marking $\mathbb{M}_k = \mathbb{M}_0$ of the prognoser $HPPN_\Pi$ is based on Δ_k, the set of configurations, particles and degradation tokens representing all the diagnosis hypotheses on the system at time k.

During the *Initialize* procedure (line 1), the tokens of Δ_k are duplicated in a way that any diagnosis hypothesis of precision n is represented in the prognoser by m prognosis hypotheses with precision 1, where $m \in \mathbb{N}_+$ is determined with the SSA (see Algorithm 2 in Section IV-B). The n jobs to realize are the simulations in the future of the n diagnosis hypotheses included in Δ_k. The resources are the hypotheses with precision 1.

More formally, any diagnosis hypothesis $\{\delta_k, \pi_k^1, ..., \pi_k^n, d_k^1, ..., d_k^n\} \subset \Delta_k$, where the n degradation tokens link the n particles to the configuration δ_k, is reproduced in the prognoser initial marking M_k and takes the form of an equivalent distribution $\{\delta_k^1, \pi_k^1, d_k^1, ..., \delta_k^m, \pi_k^m, d_k^m\}$, in which, for any $i \in \{1, ..., m\}$, d_k^i links δ_k^i and π_k^i, and any δ_k^i has the same value b_k as δ_k. Once initialized, the prognoser has as many configurations as particles and degradation tokens:

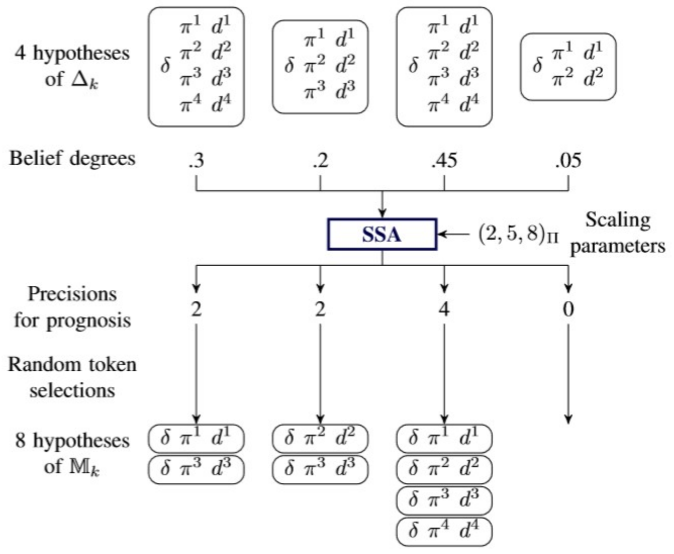
$$|M_k^S| \quad = \quad |M_k^N| \quad = \quad |M_k^D|. \tag{3}$$

Without performance constraints, a diagnosis hypothesis is completely reproduced in the prognoser, i.e. $m = n$: any particle π_k or degradation token d_k in Δ_k is duplicated. To improve computational performance of the prognosis process, however, the hypotheses in Δ_k can be partially reproduced. The SSA determines the precisions to associate to all the hypotheses, based on their belief degrees, and by using three scaling parameters ρ_Π^{min}, ρ_Π^{max} and ρ_Π^{tot}. It means that any m is chosen to satisfy the predicate $\rho_\Pi^{min} \leq m \leq \rho_\Pi^{max}$. It also means the total number of tokens in the prognoser $|M_k|$ always verifies:

$$|M_k| \quad \leq \quad 3 \times \rho_\Pi^{tot}, \tag{4}$$

because there are three types of tokens. Then, for any hypothesis in Δ_k, m particles and m degradation tokens are randomly selected and duplicated to form $\mathbb{M}_k$.

The initialization process is illustrated in Figure 5. It shows how the initialization process creates 8 hypotheses of precision 1 in $\mathbb{M}_k$ from the 4 hypotheses in Δ_k.



4 hypotheses of Δ_k
δ π^1 d^1 π^2 d^2 π^3 d^3 π^4 d^4
Belief degrees
.3
.2
.45
.05
SSA
$(2,5,8)_\Pi$
Scaling parameters
Precisions for prognosis
2
2
4
0
Random token selections
8 hypotheses of $\mathbb{M}_k$

Fig. 5. Illustration of $HPPN_\Pi$ initialization.

Parameters ρ_Π^{min} and ρ_Π^{max} are respectively the minimum and the sufficient precisions granted to the prognoser to simulate a hypothesis in the future. The prognoser scaling parameters thus provide a compromise between the number of hypotheses to project in the future and the precision granted to each one of them, relative to a maximum number of tokens to simulate. They are set up to fulfill performance constraints.

2) Prognoser Marking Evolution: Once the prognoser marking is initialized, the future evolutions of the selected hypotheses are simulated according to U_k^+ (line 4). The prognoser marking evolves according to the future inputs $U_k^+ = \{u_\kappa | \kappa \in \{k, ..., k+\tau\}\}$, where $\tau + 1 \in \mathbb{N}$ is the prediction horizon. The prediction is stopped when all the inputs have been simulated (line 6) or before if all the hypotheses have reached a failure mode (lines 3-4), in order to reduce computation time.

The goal of the *Update* function described by Algorithm 4 is to predict the evolutions of the system from the generated prognoser and the inputs at time κ. For each continuous input vector $u_\kappa \in \mathbb{R}^{n_u}$ at future time κ, the marking $\hat{M}_{\kappa-1|k}$ is updated to $\hat{M}_{\kappa|k}$ in two steps, the transition firing and the tokens value update. Three rules are given hereafter to define the transition firing step.

a) Transition Firing: All enabled transitions are simultaneously fired using the following definitions and rules.

When a transition $t \in T$ is fired, all the hypotheses in its input places satisfying conditions Ω_t are moved to the output places of t.

Let $^\circ t$ and t° respectively denote the set of input and output places of t, and $M_\kappa(p)$ denote the set of the tokens in the place $p \in P$ at time κ.

Let define the set $\mathcal{S}_\kappa^t$ as

$$\mathcal{S}_\kappa^t = \{(\delta_\kappa, \pi_\kappa, d_\kappa) \in (M_\kappa(p^S) \times M_\kappa(p^N) \times M_\kappa(p^D)) | \\ link(\delta_\kappa, \pi_\kappa, d_\kappa) \wedge \omega_t^N(\pi_\kappa) \qquad \}, \tag{5}$$

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

Definition 1 (Enabled transition): A transition $t \in T$ is enabled at time κ if

$$|\mathcal{S}_\kappa^t| \quad > \quad 0. \tag{6}$$

Rule 1 (Transition firing): Let $t \in T$ be an enabled transition. For each type of places in input and output of t, the firing of t at time κ is defined by
$\forall P^o \in \{P^S, P^N, P^D\}, p \in P^o \cap^\circ t, p' \in P^o \cap t^\circ$,

$$\begin{aligned} M_{\kappa+1}(p) &= M_\kappa(p) \backslash \mathcal{S}_\kappa^t(p), \\ M_{\kappa+1}(p') &= M_\kappa(p') \cup \mathcal{S}_\kappa^t(p), \end{aligned} \tag{7}$$

where $\mathcal{S}_\kappa^t(p)$ is the set of tokens of $\mathcal{S}_\kappa^t$ that are in the place p.

b) Token Value Update: During the transition firing, configuration values are updated with the arc annotations $\mathcal{A}$ with Rule 2.

Rule 2 (Configuration value update): When a configuration moves through an arc a during the firing of a transition t at time κ, its event set b_κ is updated with the event label $v \in E$ that annotates a: $b_{\kappa+1} \leftarrow b_\kappa \cup \{(v, \kappa)\}$.

After the transition firing, all particle values are updated with Rule 3.

Rule 3 (Particle value update): The continuous state vector x_κ of a particle is updated with the continuous state equation associated with the numerical place the particle belongs: $x_{\kappa+1} = \mathbf{f}(x_\kappa, u_\kappa)$, where u_κ is the continuous input vector at time κ.

Algorithm 4 Update

```
Input: $HPPN_\Pi, \kappa, u_\kappa$
Output: $\hat{M}_{\kappa|k}$
1: for all $t \in T$ do
2:     if $t$ is enabled at time $\kappa - 1$ then
3:         $\hat{M}_{\kappa|k} \leftarrow$ update the marking applying Rule 1
4:         $b_\kappa \leftarrow$ update configuration values applying Rule 2
5:         $x_\kappa \leftarrow$ update particle values with $u_{\kappa-1}$ to applying Rule 3
6:     end if
7: end for
```

Concerning the complexity, the prognoser is initialized with a maximum of $3 \times \rho_\Pi^{tot}$ tokens (see Section IV-D1) and the marking evolution does not create new token. So, the prognoser $HPPN_\Pi$ is $3 \times \rho_\Pi^{tot}$-bounded.

3) Prognosis Result: The output of the prognosis process is the marking of the HPPN-based prognoser: $\Pi_k = \hat{M}_{k_{\text{EOP}}|k}$, where k_{EOP} is the End Of Prediction: $k_{\text{EOP}} \leq k + \tau$. The prognosis Π_k is a distribution of beliefs over the system future modes until time k_{EOP}. It especially contains the event occurrences that will lead to failure modes, and particularly the faults and their times of occurrences. It thus contains all necessary data to compute a belief distribution over the system RUL/EOL.

Whatever the value of the prediction horizon τ, k_{EOP} always satisfies $k_{\text{EOP}} \leq k + \tau$.

Let $\{\delta, \pi, d\}$ be a hypothesis in Π_k and p^S, p^N and p^D be the places where the tokens δ, π and d are. If places p^S, p^N and p^D represent a failure mode, then the EOL of this hypothesis is the occurrence time of the last event contained in the event set b of the configuration δ:

$$EOL(\{\delta, \pi, d\}) \quad \triangleq \quad max\{\kappa | (v, \kappa) \in b\}. \tag{8}$$

With the EOL, we can retrieve the RUL:

$$RUL(\{\delta, \pi, d\}) \quad \triangleq \quad EOL(\{\delta, \pi, d\}) - k. \tag{9}$$

V. Experimental Results

This section provides results obtained by testing the proposed prognosis method on the K11. Two scenarios fully described in [7] are considered in this section: a nominal scenario and a scenario during which a battery parasitic load fault (i.e. f_2) is injected. For the two scenarios, the rover mission is to visit 5 way-points and to go back to its starting position. It starts at 0s with batteries fully charged and with all components at the ambient temperature.

The K11 rover has actually two motor temperature sensors (FL and BL) failed. The sensors faults are diagnosed in one sampling period by the diagnoser if we consider the initial mode to be unknown. In the rest of the paper, we assume to know the rover initial mode as the degraded mode *Sensor BL FL fault*. For the sake of clarity, we also consider the End Of Discharge (EOD) as a Rover end Of Life (EOL).

The proposed methodology 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, x86_64). In order to reduce computation time, the token value updates are multi-threaded on the 4 physical cores. The rest of the implementation uses one core.

The scaling parameters are chosen for the HPPN-based diagnoser in order to monitor between 18 and 37 hypotheses. In order to only simulate the future of hypotheses with high beliefs, but also to limit the prognosis computation time, the HPPN-based prognoser scaling parameters are firstly set to:

$$(\rho_\Pi^{min}, \rho_\Pi^{max}, \rho_\Pi^{tot}) \quad = \quad (1, 5, 100)_\Pi.$$

Thereby, 20 to 100 hypothesis evolutions will be simulated, with precisions varying between 1 and 5.

The prediction horizon is $\tau = 10000$, because missions last a maximum of 150 minutes. For the two scenarios, future commands are based on recorded data, then the last command is repeated to fulfill the prediction horizon. Finally, the prognoser computes a prognosis result every 30 diagnoses.

A. Scenarios and Results

1) Nominal Scenario: In the nominal scenario, no fault is injected. The rover successfully visits the 5 way-points and returns to the charging station at 5064s.

Figure 6 shows the prognoser RUL estimates, i.e. the rover remaining times before entering a failure mode, all along the scenario. These RUL estimates are obtained from the HPPN-based prognoser marking and Equation (9). The RUL associated to all hypotheses simulated by the prognoser are drawn with different shades of gray; the ones with the highest

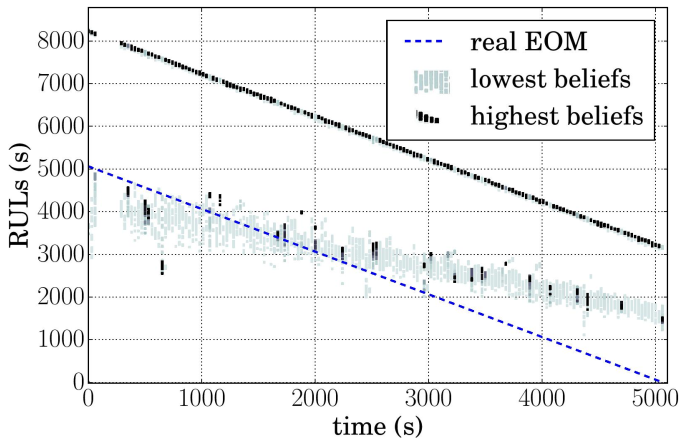


Fig. 6. Nominal scenario - RUL estimates with $(40, 80, 1500)_{\triangle}$ and $(1, 5, 100)_{\Pi}$.

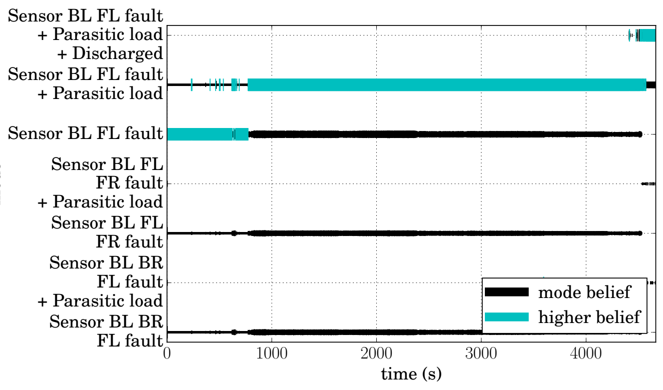


Fig. 7. Battery parasitic load fault scenario - Mode beliefs at any time with $(40, 80, 1500)_{\triangle}$.

belief degrees are black, while the ones with the lowest belief degrees are light gray. The real time of the End Of Mission (EOM) is drawn with a dashed line. The gap between 81s and 281s corresponds to a break during the experiment.

From Figure 6, it is possible to verify that the EOL with the highest belief degree is almost always around 8150s. These results are consistent with our expectations because there is a constant difference of about 3000s between the EOM and these EOL. EOM occurs before EOD. This illustrates that the diagnoser keeps the true mode *Sensor BL FL fault* in its set of possible current modes and assigns it a high belief all along the scenario.

Other modes are also considered by the diagnoser at any time because of the model-based uncertainty, leading to shorter RUL in Figure 6.

2) Parasitic Load Fault Scenario: In this scenario, a 2 A battery parasitic load fault (fault f_2) is injected between 772s and 784s and the rover is fully discharged at 4571s before returning to its starting point.

Figure 7 shows the belief degrees at any time of some modes, giving an overview of the system diagnosis. The 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 belief degree of a mode at time k is the maximum of the belief degrees of all the hypotheses being in that mode at time k.

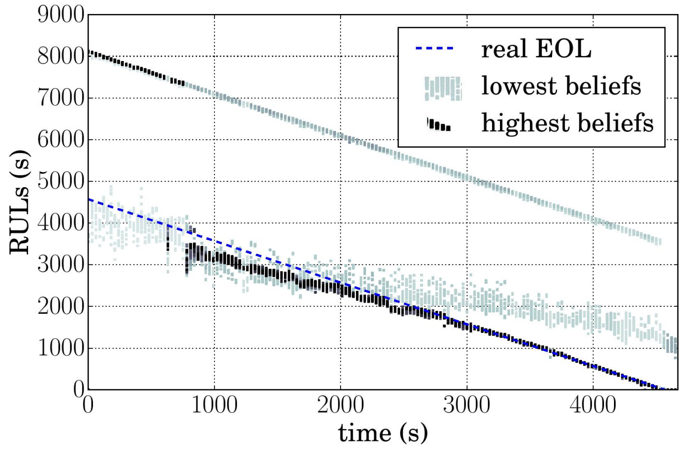


Fig. 8. Battery parasitic load fault scenario - RUL estimates with $(40, 80, 1500)_{\triangle}$ and $(1, 5, 100)_{\Pi}$.

Figure 8 shows the prognoser RUL estimates and the real EOL (dashed line) for the faulty scenario. It illustrates that the HPPN-based prognoser succeeds to compute the expected RUL estimates and thus validates the model. Before the fault occurrence, we can see the same scheme as for the nominal scenario.

After the fault occurrence, the RUL estimates for hypotheses with the highest beliefs (in dark) underestimate the real EOL. After 3000s, estimates and real value coincide.

More precisely, between 750s and 2800s, two clusters of RUL stand: they correspond to the diagnosis hypotheses considering the system is still in the initial mode and the ones considering that f_2 has occurred.

After 2800s, a third cluster appears in the RUL estimates in Figure 8. It corresponds to diagnosis hypotheses stemming from the initial hypothesis and that consider a parasitic load recently appeared. These diagnosis hypotheses have however lower belief degrees than the others ones (light gray).

3) Comparison with Other Approaches and Exploitation: In comparison with other prognosis works on the rover [25], the main advantage of our approach is that the RUL estimates are not presented as an unique distribution but are clustered in several distributions, where each cluster corresponds to one of the main possible futures of the system. In case of decision making in a health management context, if the operator is pessimistic, the shortest RUL estimate will be obviously taken into account.

Some indicators on RUL estimates can qualify the results. A relevant indicator is the average relative error between the shortest RUL estimate and the real RUL. In the faulty scenario, between 760s and 4571s, it is -14.12%, which indicates the shortest RUL is most of the time shorter than the real one. Between 760s and 2676s, the error is -15.90%, and between 2677s and 4571s, it is -12.31%, indicating it is quite constant along the scenario. These results are summed up in Table II.

A real-time estimation of the system past and future trajectory can be more valuable than an estimation of the current

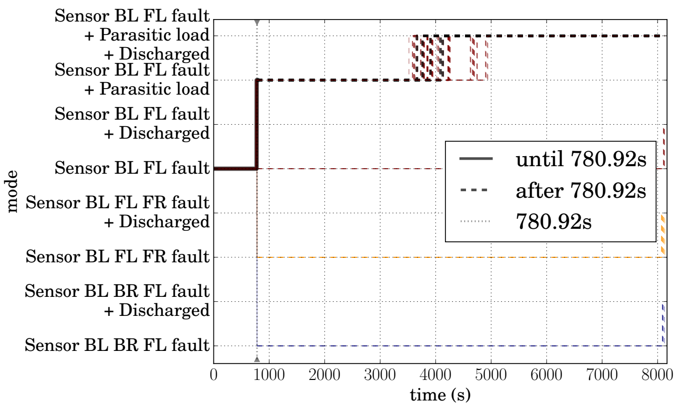


Fig. 9. Battery parasitic load fault scenario - Hypotheses on the system trajectories at 781s with $(40, 80, 1500)_\Delta$ and $(1, 5, 100)_\Pi$.

TABLE I
COMPUTATIONAL PERFORMANCES OF THE HPPN-BASED HEALTH MONITORING FOR DIFFERENT SCALING PARAMETERS.

Scaling parameters		Δ_k time (s)	Π_k time (s)	max. RAM (MB)
$(40, 80, 1500)_\Delta$, $(1, 5, 100)_\Pi$	min.	0.28	52.12	
	max.	4.54	1188.48	
	ave.	3.35	605.32	126.73
$(40, 80, 400)_\Delta$, $(1, 5, 100)_\Pi$	min.	0.28	0.03	
	max.	1.00	444.73	
	ave.	0.56	108.00	122.15
$(20, 60, 400)_\Delta$, $(1, 1, 1)_\Pi$	min.	0.22	0.03	
	max.	0.98	33.50	
	ave.	0.74	12.35	112.18

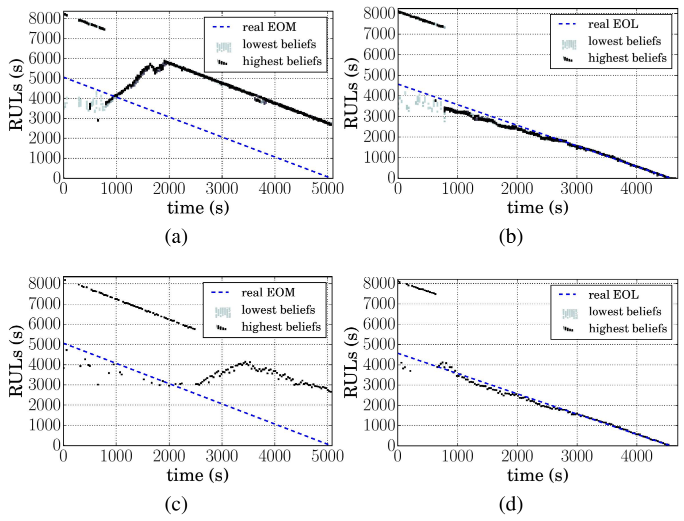


Fig. 10. RUL estimates - (a),(c): nominal scenario; (b),(d): battery parasitic load fault scenario; (a),(b): $(40, 80, 400)_\Delta$ and $(1, 5, 100)_\Pi$; (c),(d): $(20, 60, 400)_\Delta$ and $(1, 1, 1)_\Pi$.

TABLE II
BATTERY PARASITIC LOAD FAULT SCENARIO - AVERAGE RELATIVE ERRORS BETWEEN SHORTEST RUL ESTIMATE AND REAL RUL (%).

Scaling parameters	760s-4571s	760s-2676s	2677s-4571s
$(40, 80, 1500)_\Delta$, $(1, 5, 100)_\Pi$	-14.12	-15.9	-12.31
$(40, 80, 400)_\Delta$, $(1, 5, 100)_\Pi$	-9.32	-10.29	-8.34
$(20, 60, 400)_\Delta$, $(1, 1, 1)_\Pi$	-1.55	-4.56	1.51

mode and RUL/EOL. Figure 9 shows the distribution over the system trajectories at 781s. It combines the past hypotheses in the diagnosis and future hypotheses in the prognosis. Concerning the prognoses, it clearly details four possible futures of the system. This representation can be useful for the operator in case of complex decision making.

B. Performance Analysis

Diagnosis and prognosis computation times, and the maximum RAM used for different sets of scaling parameters, are given in Table I. Tests have been performed on three scenarios (including the nominal and faulty scenarios presented above) and run 12 times. 54403 diagnoses and 1822 prognoses were computed.

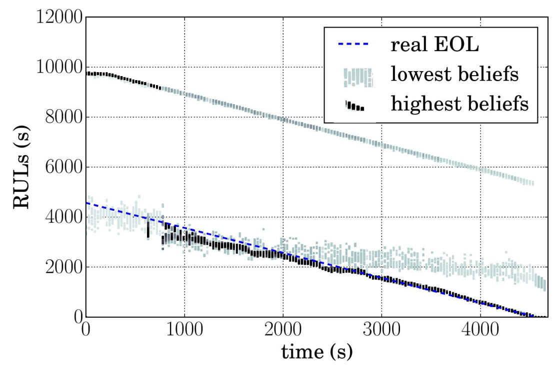
These metrics point out that computation times with the initial scaling parameters remain acceptable but do not respect real-time constraints; observations sampling is about 1s and the average diagnosis computation time is 4.54s. A prognosis process is launched every 30s, but the minimum prognosis computation time is 52.12s. This is mainly due to that both processes rely on parallel step-by-step simulations but it is also due to the rover model computational complexity.

The methodology 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.

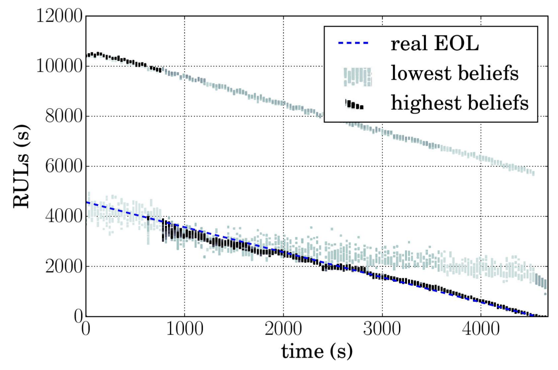
We can see in Table I that reducing the scaling parameters reduces computation times and memory usages. RUL estimates (see Figure 10) are still acceptable, even if they are not as detailed as with the initially parameter sets $(40, 80, 1500)_\Delta$ and $(1, 5, 100)_\Pi$.

Regarding the average relative errors between the shortest RUL estimate and the real RUL for the battery parasitic load fault scenario, we can see in Table II that reducing the scale parameters reduces the error. It also highlights that the error decreases along the scenario because the parasitic load estimation becomes more consistent with the model as explained is Section V. These results are consistent because less uncertainty is taken into account.

This kind of performance analysis needs to be performed during the design of the health monitoring phase for choosing the relevant scale parameters. An interesting perspective of this work is to develop some solutions for automatically find the best scale parameters, knowing the required precision and the computation time constraints.



(a) Maximum speed commands.



(b) Stochastic commands.

Fig. 11. Battery parasitic load fault scenario - RUL estimates with $(40, 80, 1500)_\Delta$ and $(1, 5, 100)_\Pi$, and unknown future commands.

C. Unknown Future Commands

In this section, results of RUL estimation are given in the case where future inputs are unknown. In such a case, we propose to dynamically construct the set of future inputs U_k^+ following a given rule. The construction consists in determining the wheel speed commands at any future time κ. Wheel speeds are commanded with value between 0 and 100, where 100 is the maximum speed value.

We propose two kinds of rules, inspired by [27]. The first one considers the rover will go straight at the highest speed. The 4 command variables of u_κ are thus set to the maximum speed value 100, at any future time κ. The second rule considers the rover will move randomly and thus the future wheel speed commands are then randomly chosen. More precisely, for each time κ, the left-side (resp. right-side) wheel speed is randomly chosen between 0, 50 and 100.

Figures 11 (a) and (b) show the results of the use of the two rules for the battery parasitic load fault scenario. They show quite similar results than Figure 8. By looking at the average relative errors between the shortest RUL estimate and real RUL (Table III), we can see that both rules provide slightly smaller errors than those obtained with known future commands (between 1.45% and 9.57%). Using stochastic future commands divides by almost 2 the average error between 760s and 4571s, and by almost 4.5 the average error between 2677s and 4571s. We can deduce that in this particular case, the use of the stochastic rule corrects the model inaccuracy during the simulation of the future behavior of the K11.

TABLE III
BATTERY PARASITIC LOAD FAULT SCENARIO - AVERAGE RELATIVE ERRORS BETWEEN SHORTEST RUL ESTIMATE AND REAL RUL (%).

Future command types	760s-4571s	760s-2676s	2677s-4571s
Known commands	-14.12	-15.9	-12.31
Maximum speed commands	-11.33	-14.45	-8.16
Stochastic commands	-7.66	-12.5	-2.74

This case study leads to several conclusions on the HPPN-based prognosis method:

- it is adaptable to systems without discrete observations nor degradation knowledge,
- it is robust to real system data and provides results that are consistent,
- it is tunable to fulfill performance constraints (both the diagnoser and prognoser scaling parameter sets have an effect on the prognoser performance),
- reducing the scaling parameters still provides acceptable results but does not respect prognosis real-time constraints for the rover with the current implementation,
- it can be used without knowledge on the future commands, and it particularly provides interesting results when using stochastic future commands.

VI. Conclusion

This work proposes an original prognosis approach of hybrid systems based on Hybrid Particle Petri Nets, and its application on a real case study, the K11 planetary rover prototype. The main contributions are the uncertainty management and the compromise between performance and available computational resources, through the setting of scaling parameters in the SSA. Two scenarios are tested on the rover to illustrate the usability of the proposed methodology. Results are consistent and show that the prognosis method is robust to real system data. The metrics show its computational performance.

As said in Section V-B, an interesting perspective is first of all to develop some solutions for automatically finding the best scale parameters, knowing the required precision and the computation time constraints. Learning approaches or fuzzy logic could be investigated. Future works will also focus on the HPPN-based diagnoser and prognoser verifications. We aim at investigating how the HPPN-based prognoser results may influence the HPPN-based diagnoser results (and vice versa). The prognoser could, for example, increases the precision of the monitoring of a system hypothesis whose predicted trajectory is particularly critical.

References

[1] S. Engel, B. Gilmartin, K. Bongort, and A. Hess, “Prognostics, the real issues involved with predicting life remaining,” in *Aerospace Conf., IEEE*, vol. 6, 2000, pp. 457–469.

[2] T. Henzinger, “The theory of hybrid automata,” in 11^{th} *Annual IEEE Symposium on Logic in Computer Science*, 1996, pp. 278–292.

[3] M. Bayoudh, L. Travé-Massuyes, and X. Olive, “Hybrid systems diagnosis by coupling continuous and discrete event techniques,” in *IFAC World Congress*, Korea, 2008, pp. 7265–7270.

[4] Q. Gaudel, E. Chanthery, and P. Ribot, "Health Monitoring of Hybrid Systems Using Hybrid Particle Petri Nets," in *Annual Conf. of the PHM Society*, USA, 2014.

[5] ——, "Hybrid Particle Petri Nets for Systems Health Monitoring under Uncertainty," *Int. Journal of Prognostics and Health Management*, vol. 6, no. 022, 2015.

[6] Q. Gaudel, P. Ribot, E. Chanthery, and M. J. Daigle, "Health Monitoring of a Planetary Rover Using Hybrid Particle Petri Nets," in 37^{th} *International Conference on Application and Theory of Petri Nets and Concurrency*, Poland, 2016.

[7] A. Sweet, G. Gorospe, M. Daigle, J. R. Celaya, E. Balaban, I. Roychoudhury, and S. Narasimhan, "Demonstration of Prognostics-Enabled Decision Making Algorithms on a Hardware Mobile Robot Test Platform," in *Annual Conf. of the PHM Society*, USA, 2014.

[8] S. Narasimhan and G. Biswas, "Model-Based Diagnosis of Hybrid Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 37, Part A, no. 3, pp. 348–361, 2007.

[9] M. Daigle, X. Koutsoukos, and G. Biswas, "An event-based approach to integrated parametric and discrete fault diagnosis in hybrid systems," *Trans. of the Institute of Measurement and Control*, vol. 32, no. 5, pp. 487–510, 2010.

[10] X. Du, S. Zeng, and J. Guo, "Dynamic system reliability modeling using extended hybrid Petri nets," in *Prognostics and System Health Management Conference*. IEEE, 2014.

[11] V. Cocquempot, T. El Mezyani, and M. Staroswiecki, "Fault detection and isolation for hybrid systems using structured parity residuals," in 5^{th} *Asian Control Conf.*, vol. 2. IEEE, 2004, pp. 1204–1212.

[12] F. Cadini, G. Peloni, and E. Zio, "A Structural Model Decomposition Framework for Hybrid Systems Diagnosis," in *Prognostics and System Health Management Conference*. IEEE, 2010.

[13] M. Daigle, A. Bregon, and I. Roychoudhury, "A Structural Model Decomposition Framework for Hybrid Systems Diagnosis," in 26^{th} *Int. Workshop on Principles of Diagnosis*, 2015.

[14] S. Narasimhan and L. Browston, "HyDE - a general framework for stochastic and hybrid modelbased diagnosis," in 18^{th} *Int. Workshop on Principles of Diagnosis*, 2007, pp. 162–169.

[15] M. Yu, D. Wang, M. Luo, and L. Huang, "Prognosis of Hybrid Systems With Multiple Incipient Faults: Augmented Global Analytical Redundancy Relations Approach," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 41, Part A, no. 3, pp. 540–551, 2011.

[16] W. O. L. Vianna and T. Yoneyama, "Interactive Multiple-Model Application for Hydraulic Servovalve Health Monitoring," in *Annual Conf. of the PHM Society*, USA, 2015.

[17] E. Chanthery and P. Ribot, "An Integrated Framework for Diagnosis and Prognosis of Hybrid Systems," in 3^{rd} *Workshop on Hybrid Autonomous System*, Italy, 2013.

[18] S. Zabi, P. Ribot, and E. Chanthery, "Health Monitoring and Prognosis of Hybrid Systems," in *Annual Conf. of the PHM Society*, 2013.

[19] M. Yu, D. Wang, and M. Luo, "Model-Based Prognosis for Hybrid Systems With Mode-Dependent Degradation Behaviors," *IEEE Trans. on Industrial Electronics*, vol. 61, no. 1, pp. 546–554, 2014.

[20] P. Ribot, Y. Pencolé, and M. Combacau, "Generic characterization of diagnosis and prognosis for complex heterogeneous systems," *Int. Journal of Prognostics and Health Management*, vol. 4, no. 023, 2013.

[21] M. Daigle, I. Roychoudhury, and A. Bregon, "Model-based Prognostics of Hybrid Systems," in *Annual Conf. of the PHM Society*, USA, 2015.

[22] F. Basile, P. Chiacchio, and G. D. Tommasi, "Fault diagnosis and prognosis in Petri Nets by using a single generalized marking estimation," in 7^{th} *IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Spain, 2009.

[23] D. Lefebvre, "Fault Diagnosis and Prognosis With Partially Observed Petri Nets," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 44, no. 10, pp. 1413–1424, 2014.

[24] E. Balaban, S. Narasimhan, M. J. Daigle, I. Roychoudhury, A. Sweet, C. Bond, J. R. Celaya, and G. Gorospe, "Development of a Mobile Robot Test Platform and Methods for Validation of Prognostics-Enabled Decision Making Algorithms," *Int. Journal of Prognostics and Health Management*, vol. 4, no. 006, 2013.

[25] M. J. Daigle and S. Sankararaman, "Advanced methods for determining prediction uncertainty in model-based prognostics with application to planetary rovers," *Annual Conference of the Prognostics and Health Management Society*, 2013.

[26] P. Ribot, E. Chanthery, and Q. Gaudel, "Hppn-based prognosis for hybrid systems," *Annual Conference of the Prognostics and Health Management Society*, 2017.

[27] M. Daigle, S. Sankararaman, and C. S. Kulkarni, "Stochastic Prediction of Remaining Driving Time and Distance for a Planetary Rover," in *IEEE Aerospace Conf.*, 2015.

Pauline Ribot received a Master's degree in control engineering in 2996 and a PhD degree in model-based diagnosis and prognosis in 2009 from the Paul Sabatier University, in Toulouse, France. Since 2011 she is Associate Professor in control and discrete event systems at Paul Sabatier University in Toulouse, working at the Laboratory for Analysis and Architecture of Systems of the Centre National de la Recherche Scientifique (LAAS-CNRS) in the Diagnosis and Supervisory Control research group. Her main research interests are in model-based reasoning, supervision of heterogeneous multicomponent systems and diagnosis and prognosis integration.

Elodie Chanthery received her engineering and master's degrees in control engineering from the Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et des Telecommunications, Toulouse, France, in 2002, and the Ph.D. degree in mission planning for autonomous vehicles from the Ecole Nationale Suprieure de l'Aéronautique et de l'EspaceSupAero, Toulouse, France, in 2005. She received the Habilitation à diriger des recherches from the Institut National Polytechnique de Toulouse, Toulouse, France, in 2018. She is currently an Assistant Professor of control with the Institut National des Sciences Appliquées, Toulouse, France. Her research activities are conducted in LAAS-CNRS in the field of complex hybrid systems, focusing on the links between diagnosis and other tasks, such as planning or prognosis.

Quentin Gaudel is a R&D software engineer at EasyMile, Toulouse, France. He graduated in Critical and Embedded Systems from the Institut National des Sciences Appliques (INSA), Toulouse, France, in 2013, and defended in 2016 his PhD works on modeling, diagnosis and prognosis of hybrid systems under uncertainty. In this area, he published several papers in scientific or industrial conferences proceedings. He started his career as a member of the Fleet Management R&D team, dealing with the remote monitoring and control of driverless vehicles, driverless mobility solutions, fleet traffic management, fleet simulation, data analysis and data management.

Matthew J. Daigle received the B.S. degree in computer science and computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2004, and the M.S. and Ph.D. degrees in computer science from Vanderbilt University, Nashville, TN, in 2006 and 2008, respectively. From September 2004 to May 2008, he was a Graduate Research Assistant with the Institute for Software Integrated Systems and Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN. From June 2008 to December 2011, he was an Associate Scientist with the University of California, Santa Cruz, at NASA Ames Research Center. From January 2012 to May 2017, he was a Research Computer Scientist with NASA Ames Research Center, and lead of the Diagnostics and Prognostics Group from 2016 to 20187. Since June 2017 he has been a Principal AI Scientist with NIO USA, Inc. He has published over 100 peer-reviewed papers in the area of systems health management. His current research interests include systems health, autonomous driving, and artificial intelligence.

Figure captions:

Fig. 1. From Diagnosis to Prognosis

Fig. 2. Streamlined Description of the K11 health evolution

Fig. 3. Overview of the health monitoring architecture

Fig. 4. Prognoser $HPPN_{\Pi}$ of the K11 rover

Fig. 5. Illustration of $HPPN_{\Pi}$ initialization

Fig. 6. Nominal scenario - RUL estimates with $(40, 80, 1500)_{\Delta}$ and $(1, 5, 100)_{\Pi}$

Fig. 7. Battery parasitic load fault scenario - Mode beliefs at any time with $(40, 80, 1500)_{\Delta}$

Fig. 8. Battery parasitic load fault scenario - RUL estimates with $(40, 80, 1500)_{\Delta}$ and $(1, 5, 100)_{\Pi}$

Fig. 9. Battery parasitic load fault scenario - Hypotheses on the system trajectories at 781s with $(40, 80, 1500)_{\Delta}$ and $(1, 5, 100)_{\Pi}$

Fig. 10. RUL estimates - (a),(c): nominal scenario; (b),(d): battery parasitic load fault scenario; (a),(b): $(40, 80, 400)_{\Delta}$ and $(1, 5, 100)_{\Pi}$; (c),(d): $(20, 60, 400)_{\Delta}$ and $(1, 1, 1)_{\Pi}$

Fig. 11. Battery parasitic load fault scenario - RUL estimates with $(40, 80, 1500)_{\Delta}$ and $(1, 5, 100)_{\Pi}$, and unknown future commands