HAL open science

Apprentissage de modèles pour la gestion de santé de systèmes hybrides sous incertitudes

Amaury Vignolles

To cite this version:

Amaury Vignolles. Apprentissage de modèles pour la gestion de santé de systèmes hybrides sous incertitudes. Automatique. INSA de Toulouse, 2022. Français. NNT : . tel-03880227v1

HAL Id: tel-03880227

https://laas.hal.science/tel-03880227v1

Submitted on 1 Dec 2022 (v1), last revised 4 Jan 2023 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *15/11/2022* **par :**

Amaury Vignolles

Apprentissage de modèles pour la gestion de santé de systèmes hybrides sous incertitudes.

JURY

Dimitri LEFEBVRE	Professeur, Université Le Havre	Rapporteur
Zineb SIMEU-ABAZI	Professeure, Polytech Grenoble-UGA	Rapporteur
Philippe DAGUE	Professeur Emerite, Université Paris-Saclay	Examinateur
Kamal MEDJAHER	Professeur, Ecole Nationale d'Ingénieurs de Tarbes	Examinateur
Elodie CHANTHERY	MCF - INSA Toulouse - HDR	Directrice de thèse
Pauline RIBOT	MCF - Université Toulouse 3	Co-directrice de thèse

École doctorale et spécialité :

EDSYS : Automatique 4200046

Unité de Recherche :

LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes (UPR 8001)

Directrices de Thèse :

Elodie Chanthery et *Pauline Ribot*

Rapporteurs :

Dimitri LEFEBVRE et *Zineb SIMEU-ABAZI*

Sommaire

Liste des figures **v**

Liste des tableaux **viii**

Liste des algorithmes **xi**

1 Introduction **5**

1.1 Contexte 6

1.2 Notions nécessaires 7

1.3 Contributions et plan du manuscrit 12

1.4 Publications 15

2 Cadre de modélisation **17**

2.1 Introduction 18

2.2 État de l'art sur les formalismes existants 19

2.2.1 Les automates hybrides 19

2.2.2 Les réseaux de Petri 19

2.2.3 Synthèse 23

2.3 Le formalisme des HtPN 25

2.3.1 Présentation générale 25

2.3.2 Règles de tirage 31

2.4 Application du formalisme des HtPN à différents systèmes 33

2.4.1 Démarche de modélisation en HtPN 33

2.4.2 Application à un système de production 36

2.4.3 Application à un système de trois cuves 40

2.5 Conclusion 44

2.5.1 Résumé du chapitre 44

2.5.2 Limitations et discussions 45

3 Obtention d'un modèle par apprentissage **47**

3.1 Introduction 48

3.2 État de l'art sur les méthodes d'apprentissage de modèles 48

3.2.1 Apprentissage d'un modèle à événements discrets 49
3.2.2 Apprentissage d'un modèle continu 53
3.2.3 Apprentissage d'un modèle hybride 55
3.2.4 Synthèse 57
3.3 Processus d'apprentissage d'un HtPN 58
3.3.1 Processus d'apprentissage : vue d'ensemble 58
3.3.2 Données d'entrée 59
3.3.3 Prétraitement 59
3.3.4 Apprentissage structurel 61
3.3.5 Apprentissage des dynamiques continues 64
3.3.6 Déclenchement de l'apprentissage pour améliorer un modèle 65
3.4 Application de la méthode d'apprentissage 69
3.4.1 Description du système 69
3.4.2 Données d'entrée 70
3.4.3 Prétraitement des données 71
3.4.4 Apprentissage structurel 71
3.4.5 Apprentissage des dynamiques continues 73
3.5 Conclusion 77
3.5.1 Résumé du chapitre 77
3.5.2 Limitations et discussion 78

4 Méthode de suivi de santé 79
4.1 Introduction 80
4.2 État de l'art sur les méthodes de diagnostic 80
4.2.1 Approches discrètes 81
4.2.2 Approches continues 83
4.2.3 Approches hybrides 84
4.2.4 Synthèse 85
4.3 Méthode de diagnostic avancé 86
4.3.1 Vue globale 86
4.3.2 Concept de métaplace 86
4.3.3 Génération du diagnostiqueur 88
4.3.4 Processus de diagnostic 91
4.4 Application de la méthode de diagnostic à différents systèmes 102
4.4.1 Système à événements discrets 102
4.4.2 Système continu 104
4.4.3 Système hybride 107
4.5 Conclusion 109
4.5.1 Résumé du chapitre 109

4.5.2 Limitations et discussion 109

5 Application **111**

5.1 Introduction 112

5.2 Description du système 112

5.3 Modélisation d'un système hybride vieillissant pour la gestion de santé . . 115

5.3.1 Modélisation du système des panneaux en HtPN 115

5.3.2 Simulation du système 118

5.4 Application de la méthode de diagnostic avancé 122

5.4.1 Scénario 1 : cas nominal 122

5.4.2 Scénario 2 : insertion d'une faute menant à la défaillance 123

5.4.3 Scénario 3 : vieillissement des panneaux 123

5.4.4 Scénario 4 : fusion des trois scénarios précédents 124

5.5 Application de la méthode d'apprentissage 125

5.5.1 Données d'entrées 126

5.5.2 Prétraitement des données 127

5.5.3 Apprentissage structurel du modèle HtPN 127

5.5.4 Apprentissage des dynamiques continues 128

5.5.5 Application de la méthode de diagnostic au modèle appris 130

5.6 Conclusion 134

6 Conclusion et perspectives **135**

Bibliographie **141**

Liste des figures

1.1 Exemple de SED : un automate (Verwer et al. 2006) 8
1.2 Exemple de SED : un réseau de Petri 9
1.3 Exemple d'une représentation multimodale 11

2.1 Exemple de réseau de Petri hybride (Alla et al. 1998) 20
2.2 Exemple de réseau de Petri mixte (a) et des ensembles EP (b) et EM (c) . 21
2.3 Exemple de réseau de Petri particulaire (Gaudel, Chanthery, Ribot et Daigle 2018) 23
2.4 Exemple d'HtPN (a) et les dynamiques associées à ses places (b) 25
2.5 Exemples de destruction (a-b) et duplication (c-d) de jetons 32
2.6 HtPN modélisant le comportement du système de production de Motorola 37
2.7 Représentation du système des trois cuves 40
2.8 Représentation multimode du système des trois cuves 43
2.9 Représentation HtPN des modes $Nom1$ et $Nom2$ 43
2.10 Simulation du système des trois cuves 43
2.11 Evolution de la dégradation pour les cuves d'eau 44

3.1 Réseau de neurones correspondant à un réseau de Petri (Leclercq et al. 2008) 50
3.2 Principe de fonctionnement de DyClee (Roa et al. 2019) 51
3.3 Exemple de fonctionnement de la deuxième étape de DyClee (Roa et al. 2019) 52
3.4 Représentation de l'idée de la méthode SVR 54
3.5 Représentation de l'idée de la méthode RFR 55
3.6 Exemple de régression polynomiale 56
3.7 Vue globale du processus d'apprentissage 58
3.8 s_f calculé pour différents t_f et $\hat{t_f}$ 67
3.9 Processus de validation du modèle 69
3.10 Représentation multimode réduite du système des trois cuves d'eau 70
3.11 Scénario simulé 70
3.12 Exemple de données avant normalisation 71
3.13 Exemple de données après prétraitement 71

3.14 Clusters identifiés par DyClee 72
3.15 Structure HtPN obtenue 72
3.16 Résultats de l'apprentissage pour les échantillons [0 : 309] 73
3.17 Résultats de l'apprentissage pour les échantillons [310 : 329] 74
3.18 Résultats de l'apprentissage pour les échantillons [330 : 370] 74
3.19 Résultats de l'apprentissage pour différents ordres de régression pour les échantillons [0 : 309] 75
3.20 Résultats de l'apprentissage pour différents ordres de régression pour les échantillons [310 : 329] 75
3.21 Résultats de l'apprentissage pour différents ordres de régression pour les échantillons [330 : 370] 76
3.22 Résultats de l'apprentissage du modèle de RP pour l'ensemble des échantillons 76

4.1 Exemples d'un modèle automate (gauche, a) et de son diagnostiqueur (droite, b) (Soldani 2008) 82
4.2 Évolution d'une distribution de probabilité représentant l'état continu du système au cours du temps 83
4.3 Vue globale de la méthode de diagnostic avancé 87
4.4 Exemple d'un système à événements discrets transformé avec des métaplaces 88
4.5 Exemple d'HtPN (a) et de son diagnostiqueur (b) 89
4.6 Étape 1 de la génération du diagnostiqueur 90
4.7 Étape 2 de la génération du diagnostiqueur 91
4.8 Étape 3 de la génération du diagnostiqueur 92
4.9 Étape 4 de la génération du diagnostiqueur 93
4.10 Principe de l'étape de correction 100
4.11 Rééchantillonnage 101
4.12 Système à événement discret considéré 102
4.13 Scénario considéré pour le SED 103
4.14 Diagnostic du scénario considéré pour le SED 103
4.15 Modèle multimode représentant le HtPN du système continu considéré . . 105
4.16 Scénario considéré pour le système continu 105
4.17 Evolution des niveaux d'eau de chaque cuve en fonction du temps et des événements induits 106
4.18 Diagnostic du scénario considéré pour le système continu 107
4.19 Scénario considéré pour le système des cuves d'eau 107
4.20 Diagnostic du scénario considéré pour le système hybride avec 200 jetons maximum 108

5.1 Présentation du système 113
5.2 Système des panneaux devant le soleil artificiel 114
5.3 Schéma récapitulatif de la maquette 115

5.4 Structure HtPN pour un seul panneau 118
5.5 Représentation multimode du modèle HtPN pour les deux panneaux 120
5.6 Courant mesuré par les capteurs pour chaque panneau 121
5.7 Évolution de la dégradation pour chaque panneau jusqu'à la défaillance . . 121
5.8 Simulation et diagnostic pour le scénario 1 (nominal) 122
5.9 Simulation et diagnostic pour le scénario 2 (fautes fatales) 123
5.10 Simulation et diagnostic pour le scénario 3 (usure des panneaux) 124
5.11 Simulation et diagnostic pour le scénario 4 (nominal, fautes fatales et usure des panneaux) 125
5.12 Mode de fonctionnement du jeu de données considéré 126
5.13 Exemple de données disponibles dans le jeu de données 127
5.14 Exemple de données prétraitées 127
5.15 Clusters obtenus par DyClee avec une taille de micro-clusters = 0.8 127
5.16 Structure HtPN obtenue après interprétation des résultats de DyClee . . . 128
5.17 Courant appris pour l'ensemble du jeu de données 129
5.18 Tension apprise pour l'ensemble du jeu de données, sans intervention du mécanisme de ré-apprentissage 129
5.19 Tension apprise pour l'ensemble du jeu de données, avec intervention du mécanisme de ré-apprentissage 130
5.20 Représentation structurelle du HtPN appris avec réapprentissage 131
5.21 Diagnostic du scénario pour le modèle avec réapprentissage 132
5.22 Représentation structurelle du HtPN appris sans réapprentissage 133
5.23 Diagnostic du scénario pour le modèle sans réapprentissage 133

6.1 Exemple d'un Réseau de Petri temporel 138

Liste des tableaux

2.1 Synthèse de l'état de l'art 24

3.1 Comparaison du score R^2 et du temps de calcul requis 73
3.2 Comparaison du score R^2 et du temps de calcul requis pour différents ordre de régression polynomiale 77

5.1 Tableau offrant un aperçu des noms composés dans le modèle dans deux panneaux 119
5.2 Correspondance entre échantillons et place attendue 132

Liste des algorithmes

1 Prétraitement des données 60
2 Obtention de la structure HtPN 62
3 Obtention de la condition numérique associée à une transition t 63
4 Obtention du score discret 95
5 Obtention du score continu 96
6 Normalisation des scores des clans 97
7 Modification du score le plus élevé pour le rééchantillonnage préférencié . . 97
8 Calcul du score de chaque place 99

Glossaire

Abréviations

EOL	End of Life
RUL	Remaining Useful Life
SED	Systèmes à événements discrets
PPN	Réseaux de Petri particulaires – *Particle Petri Nets*
MPPN	Réseaux de Petri particulaires modifiés – *Modified Particle Petri Nets*
HPPN	Réseaux de Petri particulaires hybrides – *Hybrid Particle Petri Nets*
HtPN	Réseaux de Petri hétérogènes – *Heterogeneous Petri Nets*
HeMU	Heterogeneous systems Monitoring under Uncertainty
SVR	Support Vector Regression
SVM	Support Vector Machine
RFR	Random Forest Regression
RP	Régression Polynomiale
DPI	Indicateur de Performance pour la Diagnosticabilité
PPI	Indicateur de Performance pour la Pronosticabilité
ARR	Relations de redondance analytique – *Analytical Redundancy Relations*

Notations mathématiques

$\mathbb{M}$	Marquage d'un HtPN
$\mathbb{M}_0$	Marquage initial d'un HtPN
$\mathbb{M}_k$	Marquage d'un HtPN au temps k
$\hat{\mathbb{M}}_k$	Marquage estimé d'un HtPN au temps k
P	Ensemble des places d'un HtPN
F	L'ensemble des fautes
f	Une faute
p	Une place particulière d'un HtPN
T	Ensemble des transitions d'un HtPN
t	Une transition particulière d'un HtPN
Pre	Matrice d'incidence avant, correspond aux conditions de tir d'un HtPN
$Post$	Matrice d'incidence arrière, correspond aux assignations de tir d'un HtPN
C	Ensemble des dynamiques continues d'un HtPN
D	Ensemble des dynamiques de dégradation d'un HtPN
C_p	Ensemble des dynamiques continues associées à une place p
D_p	Ensemble des dynamiques de dégradation associées à une place p
A	Ensemble des arcs d'un HtPN
$A_{\bullet t}$	Ensemble des arcs en amont de t
E	Ensemble des labels d'événements
E_o	Ensemble des labels d'événements observables
E_{uo}	Ensemble des labels d'événements non observables
X	Espace d'état du vecteur continu
x_k	Vecteur d'état continu
Γ	Espace d'état de dégradation
γ_k	Vecteur d'état de dégradation
$n_H(p)$	Nombre de jetons dans la place p
h	Un jeton du HtPN
δ_k	Configuration d'un jeton au temps k
π_k	Etat d'un jeton au temps k
ϕ_k	Statut d'un jeton au temps k
b_k	Ensemble des événements ayant eu lieu jusqu'au temps k
$a_{p,t}$	Un arc connectant la place p à la transition t
$\Omega^S_{p,t}$	Condition symbolique portée par l'arc $a_{p,t}$
$\Omega^N_{p,t}$	Condition numérique portée par l'arc $a_{p,t}$
$\Omega^D_{p,t}$	Condition de dégradation portée par l'arc $a_{p,t}$
$\rho_{p,t}$	Poids porté par l'arc $a_{p,t}$
$H_a(a_{p,t}, p)$	Ensemble des jetons situés dans p et acceptés par l'arc $a_{p,t}$
$n_{Ha(p,t)}$	Cardinal de l'ensemble $H_a(a_{p,t}, p)$
$a_{t,p}$	Un arc connectant la transition t à la place p
$\Omega^S_{t,p}$	Assignation symbolique portée par l'arc $a_{t,p}$
$\Omega^N_{t,p}$	Assignation numérique portée par l'arc $a_{t,p}$
$\Omega^D_{t,p}$	Assignation de dégradation portée par l'arc $a_{t,p}$
$\rho_{t,p}$	Poids porté par l'arc $a_{t,p}$
$\Psi(p, t)$	Ensemble des jetons sélectionnés pour être tirés
${}^{\bullet}\zeta$	Fonction de sélection aléatoire de jetons
$\Psi(\bullet t)$	Ensemble des jetons tirés par t

S	Ensemble des échantillons de DyClee
Cl	Ensemble des clusters identifiés par DyClee
CG	Ensemble des conditions continues obtenues par apprentissage
RM	Modèle obtenu par application des méthodes de régression
δ_{fi}	Indicateur de diagnosticabilité d'une faute f_i
γ_{fi}	Indicateur de pronosticabilité d'une faute f_i
α_{fi}	Indice de criticité d'une faute f_i
$HtPN_{\Delta}$	Diagnostiqueur HtPN
O	Observations continues et discrètes disponibles
O^S	Observations discrètes disponibles
O^N	Observations continues disponibles
MP	Une métaplace
mp	Une place appartenant à une métaplace
α	Degré de confiance accordé à O^S par rapport à O^N
n_{suff}	Paramètre de la méthode de diagnostic représentant le nombre maximum de duplications pour un jeton
n_{max}	Paramètre de la méthode de diagnostic représentant le nombre maximum de jetons dans le diagnostiqueur
Pr^S	Score discret d'un jeton
Pr^C	Score continu d'un jeton
$Score(h_k)$	Score d'un jeton à un temps k
Δ_k	Résultat de diagnostic à un temps k

Chapitre 1

Introduction

Sommaire

1.1 Contexte **6**
1.2 Notions nécessaires **7**
1.3 Contributions et plan du manuscrit **12**
1.4 Publications **15**

1.1 Contexte

La complexification des systèmes modernes (robots humanoïdes, voitures électriques, satellites, etc.) entraîne de nombreux problèmes. L'un d'entre eux est la difficulté de modéliser, c.-à-d. de représenter, sous un formalisme mathématique, le système concerné de manière précise et correcte. Un modèle précis d'un système permet de prévoir/estimer la réaction du système à diverses commandes, ou encore de vérifier ou d'étudier des propriétés sur le système. Au contraire, un modèle non précis ou erroné peut entraîner des difficultés à anticiper le comportement du système et peut même engendrer des pertes financières, ou pire, humaines, en cas d'apparition de comportements non prévus, comme une instabilité ou une perte de contrôle. Un exemple peut être une surchauffe ou explosion d'une machine ou encore un manque d'eau dans un réservoir suite à une fuite.

Dans cette thèse, on s'intéresse particulièrement à la prédiction/estimation des réactions du système. Développer la possibilité de simuler diverses "lignes de vie" du système et les comparer à la réalité, peut nous permettre d'identifier la cause d'un comportement non voulu (une faute) et de la réparer avant qu'elle n'engendre des pertes.

Il est donc nécessaire, d'une part, d'être capable de déterminer un *modèle* de ces systèmes complexes, et de l'autre d'être capable de détecter ou d'anticiper les fautes sur ces systèmes. Pour répondre au premier point, de nombreux formalismes existent et permettent de représenter des systèmes relativement simples, mais les systèmes plus complexes soulèvent encore de nombreux challenges. Le second point fait quant à lui intervenir la *gestion de santé*, qui vise au développement d'outils adaptés pour réduire les coûts liés aux indisponibilités et aux pannes qui peuvent survenir sur le système.

La gestion de santé d'un système passe par la surveillance de la santé d'un système, c.-à-d. la capacité d'estimer (diagnostic) et de prédire (pronostic) l'état de santé d'un système en vue d'effectuer des actions (décision, planification) pour augmenter sa durée de vie. C'est dans ce cadre qu'interviennent les méthodes de *diagnostic*, qui déterminent l'état courant du système et identifient les causes probables (interaction avec l'environnement, fautes, etc) qui peuvent conduire à l'état déterminé. Les méthodes de *pronostic*, elles, utilisent l'état courant du système, la connaissance sur la dégradation du système ainsi que les actions futures connues sur le système pour anticiper les fautes et ainsi prédire les futurs états de santé du système. Généralement, le pronostic est associé à la prédiction de la fin de vie, *End of Life* (EOL), d'un système, date à laquelle le système n'est plus opérationnel, ou à la durée de vie résiduelle, *Remaining Useful Life* (RUL), qui est la durée restante jusqu'à la fin de vie. Les méthodes de diagnostic et de pronostic permettent donc d'avoir une idée de l'état de santé actuel du système et une prédiction sur l'état de santé futur du système : c'est ce qu'on appelle le *suivi de santé*. Il est ainsi possible pour le processus de décision (ou pour l'opérateur) de se baser sur ces états pour maintenir le système en bon fonctionnement, d'où l'importance de la qualité du diagnostic et du pronostic. C'est ce que l'on appellera la *gestion de santé*.

Il existe deux principales écoles pour appliquer ces méthodes de gestion de santé.

La première est *basée modèle*. En se basant sur le modèle du système, et à condition que le modèle soit de bonne qualité, les méthodes de diagnostic et de pronostic peuvent fournir des résultats précis. Le problème vient du fait que ces méthodes nécessitent, comme leur nom l'indique, un modèle du système. Pour avoir des résultats précis, le modèle doit

aussi l'être, ce qui n'est pas toujours le cas.

La deuxième est *basée données*. En se basant sur les données, telles que des informations issues des capteurs par exemple, les méthodes de diagnostic et de pronostic peuvent fournir des indications intéressantes. Le problème ici est, d'une part, la nécessité d'avoir de nombreuses données, y compris des données sur les fautes dont on cherche à identifier l'occurrence, et, d'autre part, l'explicabilité du résultat.

C'est dans ce cadre qu'interviennent les travaux de cette thèse. Le but est d'arriver à obtenir des modèles de systèmes complexes par des fonctions d'apprentissage à partir de données afin de pouvoir appliquer des fonctions de gestion de santé à base de modèles.

Le premier verrou de cette thèse concerne le **formalisme mathématique associé aux modèles de systèmes complexes**. Les systèmes complexes que nous considérons sont des systèmes composés de sous-parties présentant des dynamiques hétérogènes. Il n'existe à notre connaissance aucun formalisme permettant de représenter des modèles pour de tels systèmes. De plus, le formalisme recherché doit pouvoir représenter et simuler des incertitudes sur la modélisation et sur les observations. Enfin, il serait intéressant pour le formalisme recherché d'être capable de représenter le vieillissement du système, qui est une des causes principales d'occurrence de fautes.

Le deuxième verrou porte sur **l'obtention desdits modèles**. En effet, un expert n'est pas toujours disponible pour fournir un modèle précis du système. Dans certains cas, il n'existe peut-être même pas d'expert capable de fournir un modèle représentant précisément le comportement du système. Il est par contre possible d'obtenir de nombreuses données issues des observations du système considéré. L'idée est d'utiliser ces données pour obtenir un modèle par apprentissage. La méthode d'apprentissage désirée doit être capable de gérer en entrée des données hétérogènes (variables continues, qualitatives ou booléennes) et entachées d'incertitudes. De plus, elle ne doit pas présenter d'hypothèses strictes quant à la forme du modèle appris. Enfin, ce serait un plus si la méthode pouvait s'avérer assez simple à prendre en main, notamment au niveau du choix des hyper-paramètres.

Le troisième et dernier verrou porte sur **le développement de fonctions de gestion de santé sur les modèles obtenus**. Les fonctions de gestion de santé dépendant du formalisme considéré, il faut les adapter pour pouvoir les appliquer aux systèmes complexes considérés. Ces fonctions de gestion de santé doivent être capable de donner une estimation de l'état de santé du système à un temps donné. De plus, elles doivent être capables de prendre en compte des incertitudes afin de pouvoir donner une estimation robuste face à de fausses observations ou du bruit, par exemple.

1.2 Notions nécessaires

Cette section présente les différentes notions nécessaires à la bonne compréhension du manuscrit. Tout d'abord, le terme *système*, récurrent dans les prochains chapitres, doit être défini. Dans notre cas, un système est défini comme suit.

Définition 1 (Système)**.** *Un système est une combinaison d'éléments, cohérente vis-à-vis d'un certain objectif, qui interagissent entre eux et avec l'environnement.*

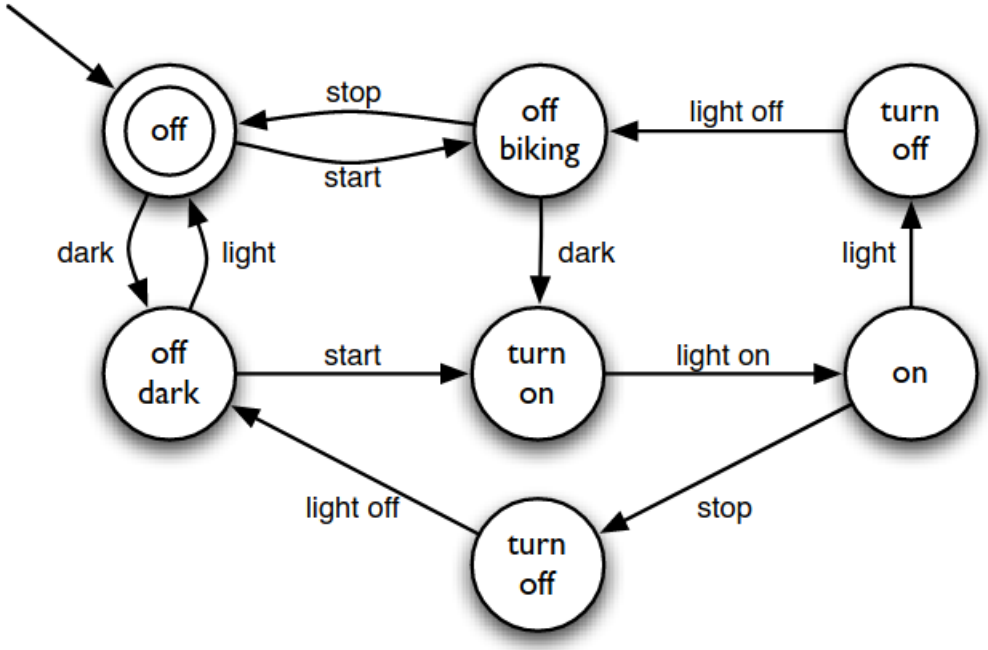


FIGURE 1.1 – Exemple de SED : un automate (VERWER et al. 2006)

Les systèmes sont généralement regroupés dans trois catégories différentes : les systèmes à événements discrets, les systèmes continus et les systèmes hybrides. Les systèmes à événements discrets sont définis dans (CASSANDRAS et al. 2009) comme suit.

Définition 2 (Système à événements discrets (SED))**.** *Un système à événements discrets est un système dont l'espace d'état est discret et dont l'évolution de l'état dépend de l'occurrence d'événements au fil du temps.*

Divers formalismes sont utilisés pour représenter les SED. On peut par exemple citer les automates (SAMPATH et al. 1995), les réseaux de Petri (DAVID et al. 2010) ou encore les chroniques (DOUSSON et al. 1993). Dans ce manuscrit, on se concentre sur les formalismes des automates et des réseaux de Petri. Un automate est défini formellement comme suit.

Définition 3 (Automate)**.** *Un automate (aussi appelé un système de transitions étiquetées) est un 4-uplet $G =< S, E, T, s_0 >$ dans lequel :*

- *S est un ensemble fini d'états,*
- *E est l'ensemble des labels d'événements apparaissant dans G,*
- *$T \subseteq S \times E \times S$ est l'ensemble des transitions,*
- *s_0 est l'état initial.*

La figure 1.1 présente un système à événements discrets sous la forme d'un automate. Un automate est un graphe orienté composé d'états, représentés sous formes de ronds, et de transitions entre ces états, représentées sous forme de flèches.

De manière formelle, un réseau de Petri est défini comme suit.

Définition 4 (Réseau de Petri)**.** *Un réseau de Petri est un 5-uplet $R =< P, T, I, O, M_0 >$, où :*

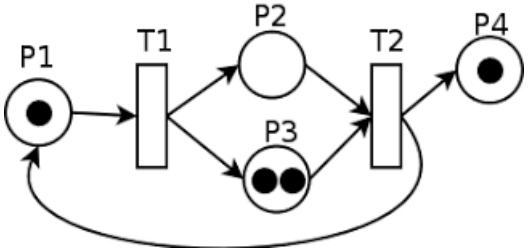


FIGURE 1.2 – Exemple de SED : un réseau de Petri

— *P est l'ensemble des places,*
— *T est l'ensemble des transitions,*
— *M_0 est le marquage initial des places : $M : P \to \mathbb{N}$ où $M_0(p)$ est le nombre de jetons dans la place p.*
— *I est la fonction d'entrée : $I : P \times T \to \mathbb{N}$. La valeur $I(p,t)$ est le poids de l'arc de la place p à la transition t.*
— *O est la fonction de sortie : $O : P \times T \to \mathbb{N}$. La valeur $O(p,t)$ est le poids de l'arc de la transition t à la place p.*

La figure 1.2 présente un système à événements discrets sous la forme d'un réseau de Petri. Graphiquement, un réseau de Petri est un graphe biparti pour lequel un nœud peut être soit une place (un rond), soit une transition (un rectangle). Les places et les transitions sont connectées entre elles par des arcs orientés mais un arc ne peut pas lier deux places ou deux transitions. Les places peuvent contenir des jetons et la distribution des jetons dans les places est appelée le marquage du réseau. L'évolution de l'état, ici représenté sous forme du marquage, dépend seulement de l'occurrence d'événements discrets associés aux transitions.

La définition des systèmes continus, présente dans CASSANDRAS et al. 2009, est la suivante :

Définition 5 (Système continu)**.** *Un système continu est un système dont l'évolution est dictée par des dynamiques continues.*

L'évolution d'un système continu à temps discret peut s'expliciter par une équation dynamique C de cette forme :

$$C = \left\{ \begin{array}{rcl} x_{k+1} & = & \mathbf{f}(x_k, u_k) \quad + \quad \mathbf{v}(x_k, u_k) \\ y_k & = & \mathbf{g}(x_k, u_k) \quad + \quad \mathbf{w}(x_k, u_k) \end{array} \right. \tag{1.1}$$

avec $x_k \in \mathbb{R}^{n_x}$ le vecteur d'état continu composé de n_x variables d'état au temps k, $u_k \in \mathbb{R}^{n_u}$ le vecteur des n_u variables d'entrées continues au temps k, $\mathbf{f}$ la fonction d'évolution non bruitée, $\mathbf{v}$ une fonction de bruit, $y_k \in \mathbb{R}^{n_y}$ le vecteur composé des n_y variables continues de sortie au temps k, $\mathbf{g}$ la fonction non bruitée de sortie et $\mathbf{w}$ la fonction de bruit associée aux observations.

Un système continu peut par exemple représenter l'évolution du niveau d'eau dans des cuves connectées. Dans ce cas, le vecteur d'état continu x représenterait le niveau d'eau dans chaque cuve, le vecteur d'entrée u le flux entrant d'eau dans les cuves et le vecteur de sortie y le flux sortant des cuves.

Les systèmes hybrides sont définis dans HENZINGER 2000 comme suit :

Définition 6 (Système hybride classique)**.** *Un système hybride est un système soumis en permanence à la fois à des dynamiques continues et des informations discrètes.*

Cependant, cette définition de système hybride paraît trop restrictive. En effet, cette définition de système hybride n'inclut pas la possibilité de représenter des systèmes dont des parties discrètes ne présentant aucune dynamique continue communiquent avec des parties présentant des dynamiques continues.

La définition de système hybride dans ce manuscrit est donc élargie de la manière suivante.

Définition 7 (Système hybride)**.** *Un système hybride est un système qui peut être divisé en différents sous-systèmes communiquant entre eux. Ces sous-systèmes peuvent être soit des SED, soit des systèmes continus, soit des systèmes hybrides classiques avec des dynamiques discrètes et continues.*

Nous allons donc travailler sur des systèmes hybrides comme définis ci-dessus. L'objectif est d'effectuer le suivi de santé de tels systèmes. Pour pouvoir suivre la santé d'un système, il faut être capable de représenter son état de santé. Pour ce faire, la notion de modes et une représentation du système de manière multimodale ont été choisies.

Définition 8 (Mode)**.** *Un mode est la combinaison d'un état de la structure SED, d'une dynamique continue et d'une dynamique de dégradation, qui peut représenter le vieillissement ou l'usure d'un système, par exemple.*

Un mode décrit l'évolution du système dans des conditions opérationnelles spécifiques.

La santé du système se définit, entre autres, par rapport à l'occurrence d'une ou plusieurs fautes. Une faute est définie dans Isermann 1997.

Définition 9 (Faute)**.** *Une faute représente une déviation non acceptable d'au moins une propriété caractéristique ou d'un paramètre du système.*

Cette définition nous permet de définir différents types de modes de santé pour un système.

Définition 10 (Mode nominal, mode dégradé, mode de défaillance)**.** *Tant qu'aucune faute n'est survenue, le système est dans des modes appelé modes nominaux. Lorsqu'une faute survient, si le système est encore capable de remplir sa tâche principale, il est dans des modes dégradés. Lorsque le système n'est plus capable de remplir sa fonction principale, on dit qu'il est dans un mode de défaillance.*

Définition 11 (Représentation graphique d'un système multimode)**.**
L'évolution de la santé d'un système est décrite avec un graphe orienté, dans lequel les nœuds représentent les modes et les arcs représentent les changements de mode. Les changements de mode sont provoqués soit par des gardes sur des variables continues, soit par l'occurrence d'événements discrets.

Par exemple, sur la figure 1.3, trois modes sont présents : un mode nominal (Nom, en vert) qui possède une dynamique continue $C1$ et une dynamique de dégradation $D1$, un mode dégradé (Deg, en orange) et un mode de défaillance (Def, en rouge) qui possèdent

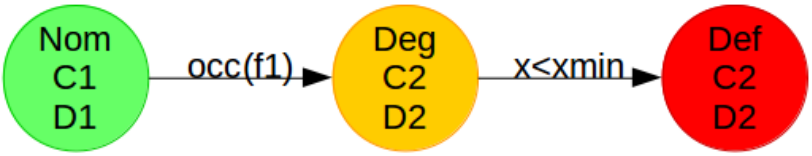


FIGURE 1.3 – Exemple d'une représentation multimodale

tous les deux les mêmes dynamiques continues ($C2$) et de dégradation ($D2$). Initialement en Nom, sur occurrence de la faute f_1, le système passe dans le mode Deg. Puis, une fois que le système ne respecte plus son but (ici maintenir x au-dessus de $xmin$), il passe en mode Def.

La notion de faute apparait donc dans la définition de mode. La majorité du temps, l'apparition d'une faute n'est pas observable directement sur le système. Il est nécessaire, afin de déterminer l'occurrence d'une faute, de s'appuyer sur des fonctions de gestion de santé.

Le suivi de santé de système s'appuie sur les fonctions de diagnostic et de pronostic. La fonction de diagnostic cherche à déterminer l'état de santé courant du système. Dans les travaux de ZWINGELSTEIN 1995, la fonction de diagnostic est définie ainsi :

Définition 12 (Diagnostic)**.** *Le diagnostic est l'identification de la cause probable de la (ou des) défaillance(s) à l'aide d'un raisonnement logique fondé sur un ensemble d'informations obtenu via une inspection, un contrôle ou un test.*

Cette définition de diagnostic se base sur le concept de défaillance, défini comme suit.

Définition 13 (Défaillance)**.** *Une défaillance correspond à une cessation de l'aptitude d'une entité à accomplir une ou plusieurs fonctions requises.*

Les informations obtenues via un test, un contrôle ou une inspection sont dites *observables.*

Définition 14 (Élément observable)**.** *Un élément, une variable continue ou l'occurrence d'un événement, par exemple, est dit observable si son occurrence est détectée par des capteurs.*

La méthode de diagnostic proposée dans ce manuscrit étend la définition proposée ci-dessus à l'état de santé. Le diagnostic effectué est donc qualifié d'avancé, car il consiste à suivre l'état de santé courant du système, et ne se contente pas d'identifier la cause d'une défaillance. L'état de santé d'un système est défini comme suit.

Définition 15 (État de santé, mode de fonctionnement)**.** *L'état de santé est défini par l'état discret du système, appelé mode de fonctionnement, associé à son état continu et à sa dégradation. Il est lié à la fois à l'occurrence de fautes, événements discrets non observables et à l'état de dégradation, pouvant représenter le vieillissement du système.*

Le pronostic, de son côté, sert à anticiper les comportements anormaux ou les fautes pouvant arriver sur le système. Deux définitions sont souvent présentes dans la littérature. La première, issue des travaux de BROTHERTON et al. 2000, est une définition très générale.

Définition 16. *Le pronostic consiste à calculer une prédiction de l'état d'un composant ou d'un système dans le futur.*

La deuxième définition, de Goh et al. 2006, met plus en avant les objectifs applicatifs de la fonction de pronostic. Elle met en évidence la notion temporelle liée à la prédiction, en mettant l'accent sur la durée de vie résiduelle, qui est le temps restant avant que le système ne puisse plus réaliser ses fonctions requises (Engel et al. 2000).

Définition 17. *Le pronostic est la capacité de prédire la durée de vie résiduelle de composants ou systèmes en service.*

Afin d'estimer les probabilités d'occurrence des fautes anticipées, des lois de vieillissement, sous forme de dynamiques de dégradation, sont proposées par les auteurs de Chanthery et Ribot 2013 pour étendre les méthodes de diagnostic et pronostic. Différentes lois sont associées à chaque mode du système et vont faire évoluer les probabilités différemment en fonction des facteurs de stress associés à chaque mode de fonctionnement (Wilkinson et al. 2004, Kirkland et al. 2004).

Définition 18 (Facteur de stress)**.** *Les facteurs de stress d'un système représentent toutes les sollicitations (normales et anormales) qui peuvent influencer l'état de santé du système.*

Parmi l'éventail de facteurs de stress possibles, on peut par exemple citer les conditions opérationnelles du système (ouvrir et fermer une vanne va avoir tendance à l'abîmer plus rapidement), les conditions environnementales (humidité, température, etc) ou encore l'occurrence de fautes et leurs conséquences sur les composants du système.

1.3 Contributions et plan du manuscrit

Ce travail de thèse propose quatre contributions majeures, présentées dans les quatre principaux chapitres de la thèse.

La première contribution répond au premier verrou identifié. Elle consiste en la définition d'**un cadre de modélisation pour les systèmes hybrides sous incertitudes**, présenté dans le chapitre 2. Tout d'abord, un état de l'art de divers formalismes existants montre que les formalismes de la littérature ne conviennent pas à nos besoins. En effet, les formalismes existants ne sont majoritairement pas capables de représenter tous les types de systèmes ou ne permettent pas de représenter des incertitudes sur le modèle. La plupart d'entre eux offrent cependant la possibilité de représenter du parallélisme ou des incertitudes sur les observations. Un formalisme se rapproche le plus de nos besoins, le formalisme des réseaux de Petri particulaires hybrides, *Hybrid Particle Petri Nets* (HPPN) défini dans Gaudel, Chanthery et Ribot 2015. Le formalisme proposé dans cette thèse va donc s'appuyer sur le formalisme des HPPN et l'étendre pour qu'il soit capable de répondre à nos besoins. Le formalisme des réseaux de Petri hétérogènes, *Heterogeneous Petri Nets* (HtPN), et sa sémantique sont donc présentés. Le formalisme des HtPN possède une structure discrète à laquelle sont associées des

dynamiques continues et de dégradation. Ces dernières représentent le vieillissement du système et la probabilité d'occurrence de fautes. Les dynamiques continues représentent quant à elles l'évolution des variables continues lorsque le système est dans cet état de fonctionnement. Les HtPN étant une extension des réseaux de Petri classiques, la notion de parallélisme est présente. Une méthodologie proposant une marche à suivre pour modéliser un système sous le formalisme des HtPN est proposée. Deux systèmes distincts sont modélisés en utilisant ce formalisme. Le premier est un système de production issu de la littérature, afin de montrer que le formalisme est capable d'effectuer du contrôle de système et de représenter les mêmes informations qu'un réseau de Petri classique. Le second est un système hybride composé de cuves d'eau communicantes. Ce deuxième cas d'étude est axé vers la gestion de santé et contient des dynamiques de dégradation.

Pour répondre au deuxième verrou identifié, l'obtention des modèles, une **méthode d'apprentissage d'un modèle haut niveau** est présentée dans le chapitre 3. Ce modèle haut niveau est composé d'une structure discrète et des dynamiques continues associées. Un état de l'art des différentes manières d'obtenir un modèle par apprentissage est effectué. Cet état de l'art se découpe en trois parties et étudie les méthodes d'apprentissage de modèles discrets, de modèles continus et de modèles hybrides. Un mélange de méthodes appliquées pour l'apprentissage de modèles discrets et continus a été choisi. La première partie de la méthode d'apprentissage se focalise sur l'obtention de la structure discrète du modèle, obtenue par l'interprétation des résultats d'un algorithme de clustering. Chaque cluster identifié est associé à un mode de fonctionnement du système et les transitions entre ces modes sont identifiées. La deuxième partie de la méthode d'apprentissage se concentre quant à elle sur l'obtention des dynamiques continues. Pour chaque mode de fonctionnement, deux algorithmes de régression sont appliqués afin d'obtenir les dynamiques continues associées au mode considéré. La méthode d'apprentissage proposée est ensuite appliquée à un jeu de données tiré d'une simulation du cas d'étude des cuves d'eau présenté dans le chapitre 2. Cette application permet de vérifier le bon comportement de la méthode d'apprentissage proposée en l'appliquant sur un système dont le modèle est *a priori* connu.

Pour répondre au troisième verrou, une **méthode de diagnostic avancé** a été définie dans le chapitre 4. Premièrement, un état de l'art de différentes méthodes de diagnostic a été effectué. Des méthodes appliquées aux systèmes discrets, aux systèmes continus et aux systèmes hybrides ont été étudiées. L'idée de la méthode appliquée aux HPPN étant la plus proche de nos objectifs, cette dernière a été adaptée au formalisme développé. La méthode développée se base sur un diagnostiqueur HtPN, généré à partir du modèle HtPN du système. Ce diagnostiqueur est capable de prendre en compte les incertitudes, qu'elles soient issues du modèle ou des observations. Le marquage du diagnostiqueur évolue en fonction des observations effectuées sur le système en se basant sur les principes de prédiction, dans un premier temps, puis de correction. Le diagnostic obtenu est représenté par le marquage du diagnostiqueur. La méthode de diagnostic proposée est mise à l'épreuve sur trois systèmes distincts : un système à événements discrets, un système continu et un système hybride. Ces trois applications se basent sur une simulation des systèmes considérés et permettent de mettre en avant la correspondance entre le diagnostic obtenu et les simulations effectuées.

Les contributions théoriques qui ont été implémentées sont regroupées dans un logiciel développé en Python : HeMU (pour *Heterogeneous systems Monitoring under Uncertainty*). Ce logiciel est disponible sur Git [1] et permet la définition d'un système sous le formalisme des HtPN, sa simulation et l'application de la méthode de suivi de santé. Le logiciel permet aussi l'obtention par apprentissage du modèle haut niveau présenté dans le chapitre 3.

Les chapitres 2, 3 et 4 ont été appliqués sur **un cas d'étude réel** dans le chapitre 5. Cette plateforme de système physique, entièrement développée dans le cadre de la thèse, est composée de deux panneaux photovoltaïques, éclairés par un soleil artificiel et connectés à diverses charges. Tout d'abord, un modèle HtPN des deux panneaux photovoltaïques est proposé à partir d'une composition de modèles représentant un panneau. Ce modèle considère un vieillissement des panneaux ainsi que certaines fautes entraînant une destruction immédiate (représentant un court-circuit, par exemple), appelées fautes fatales. Ensuite, la méthode de diagnostic est appliquée sur des simulations du modèle des deux panneaux proposé. Cette méthode est appliquée dans des cas nominaux et dans des cas présentant soit des fautes dues au vieillissement, soit des fautes fatales. Enfin, la méthode d'apprentissage est appliquée en utilisant des données réelles, obtenues via les capteurs présents sur le système, telles que le courant mesuré dans chaque panneau ou encore la luminosité captée, par exemple. Deux modèles sont obtenus, un modèle en utilisant un mécanisme de réapprentissage et l'autre sans l'utiliser. Ces modèles sont comparés à l'attendu identifié lors de l'acquisition des données après application de la méthode de diagnostic avancé. Cette application s'inscrit dans le contexte socio-écologique actuel et montre l'applicabilité de la méthode à des systèmes complexes réels.

Les applications et les calculs présentés dans ce manuscrit ont été effectués sur Linux, avec un Intel(R) Core(TM) i5-9400H CPU2.50GHZ.

1. https ://gitlab.laas.fr/hymu/hemu.git

1.4 Publications

— Elodie Chanthery, Pauline Ribot et Amaury Vignolles (2019). “HyMU : a software for Hybrid Systems Health Monitoring under Uncertainty”. In : *30th International Workshop on Principles of Diagnosis DX’19*

— Amaury Vignolles, Elodie Chanthery et al. (2020). “An overview on diagnosability and prognosability for system monitoring”. In : *European conference of the Prognostics and Health Management Society (PHM Europe)*

— Amaury Vignolles, Pauline Ribot et al. (2021). “A Holistic Advanced Diagnosis Approach For Systems Under Uncertainty”. In : *32nd International Workshop on Principle of Diagnosis–DX 2021*

— Amaury Vignolles, Elodie Chanthery et al. (2022). “Hybrid Model Learning for System Health Monitoring”. In : *Safeprocess 2022 : 11th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*

— Amaury Vignolles, Pauline Ribot et al. (2022). “Modeling complex systems with Heterogeneous Petri Nets (HtPN)”. In : *33rd International Workshop on Principle of Diagnosis–DX 2022*

Chapitre 2

Cadre de modélisation

Sommaire

2.1 Introduction 18

2.2 État de l'art sur les formalismes existants 19

2.2.1 Les automates hybrides 19

2.2.2 Les réseaux de Petri 19

2.2.3 Synthèse 23

2.3 Le formalisme des HtPN 25

2.3.1 Présentation générale 25

2.3.2 Règles de tirage 31

2.4 Application du formalisme des HtPN à différents systèmes . 33

2.4.1 Démarche de modélisation en HtPN 33

2.4.2 Application à un système de production 36

2.4.3 Application à un système de trois cuves 40

2.5 Conclusion 44

2.5.1 Résumé du chapitre 44

2.5.2 Limitations et discussions 45

2.1 Introduction

Afin de pouvoir contrôler et surveiller de manière précise l'évolution d'un système, il faut être capable de le modéliser de manière précise. De nos jours, avec la complexification des systèmes, il est nécessaire pour les formalismes de modélisation d'évoluer eux aussi. C'est pourquoi un nouveau formalisme, les HtPN, basé sur les réseaux de Petri, sera présenté dans ce chapitre. Le formalisme des HtPN a été créé dans le but de répondre à des besoins particuliers, mais il est aussi capable d'englober ce que font les réseaux de Petri "classiques".

Le formalisme utilisé doit permettre de modéliser, simuler et surveiller la santé d'un système hybride sous incertitudes. Pour rappel, un système hybride est un système dans lequel des sous-parties purement discrètes, continues ou hybrides (mélangeant les deux) sont liées et communiquent (définition 7). Par exemple, les systèmes cyber-physiques (Napoleone et al. 2020) présentent des composants de différentes natures. Ils intègrent donc des parties ayant des dynamiques différentes les unes des autres. Nous souhaitons disposer d'un formalisme capable de représenter et simuler de tels systèmes. Par ailleurs, les systèmes réels sont intrinsèquement entachés d'incertitudes. Nous souhaitons pouvoir représenter et estimer/suivre les différents types d'incertitude, sur le modèle et sur les observations, suite à du bruit ou des problèmes de communications, par exemple. Les solutions pour modéliser l'incertitude sont notamment le parallélisme et des fonctions de bruit. Le parallélisme peut servir à représenter des incertitudes : dans un réseau de Petri, différents jetons peuvent permettre de représenter différentes hypothèses sur l'état de santé du système, mais aussi de représenter certains types de systèmes complexes, comme les systèmes multi-composants. Enfin, nous souhaitons un formalisme sur lequel nous pouvons développer des fonctions de santé qui donnent des informations sur l'état courant et le fonctionnement du système. À cette fin, le formalisme choisi doit être capable de représenter le vieillissement de systèmes à travers des dynamiques de dégradation par exemple. Ces dynamiques de dégradation doivent pouvoir être fonction de l'état de stress ou des sollicitations appliquées sur le système à un moment donné.

Pour résumer, le formalisme choisi doit être capable de :

— modéliser, simuler et surveiller des systèmes hybrides composés de sous-parties ayant des dynamiques différentes ;
— représenter et suivre l'incertitude sur le système, que ce soit au niveau du modèle ou au niveau des observations ;
— suivre l'état de santé d'un système, notamment son vieillissement, à travers une dégradation fonction d'une sollicitation dynamique.

Ce chapitre est organisé de la manière suivante. Tout d'abord, dans la section 2.2, nous proposons un état de l'art des formalismes existants et nous allons voir en quoi ces formalismes ne correspondent pas tout à fait à nos besoins. Le formalisme des HtPN, basé sur le travail de Gaudel, Chanthery et Ribot 2014 est donc proposé dans la section 2.3. Ce formalisme va permettre de modéliser, simuler et surveiller des systèmes hybrides sous incertitudes. La section 2.4 présente l'application de ce formalisme sur deux exemples dans le cadre de la modélisation et de la simulation : un système de production de Motorola issu de la littérature (Alla et al. 1998) et un système composé de deux panneaux photovoltaïques que nous avons conçu pour cette étude.

2.2 État de l'art sur les formalismes existants

Le but de cette section est d'étudier les différentes solutions et formalismes existants pour représenter des systèmes hybrides composés de parties ayant des dynamiques différentes. Nous essaierons d'identifier des solutions qui satisfassent *a priori* les besoins explicités dans l'introduction : possibilité d'avoir du parallélisme, de représenter et suivre l'incertitude sur le modèle et/ou les observations ainsi que de représenter et surveiller le vieillissement du système. Ce chapitre se concentre sur les automates hybrides et les réseaux de Petri. En effet, dans la littérature, ces deux formalismes sont les principaux utilisés pour représenter des systèmes hybrides.

2.2.1 Les automates hybrides

Les automates hybrides ont été définis dans Henzinger 2000 mais une version préliminaire a été présentée en 1996 dans les *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS96).*

Dans les automates hybrides (Alur et al. 1995), chaque état discret représente un état possible de l'automate, c.-à-d. du système. Chaque état est systématiquement associé avec des dynamiques continues qui définissent l'évolution d'un vecteur d'état continu. Cependant, dans cette représentation, un seul état de l'automate peut être actif à la fois. Ceci est donc incompatible, par définition, avec notre besoin de parallélisme.

Les transitions sont définies par un ensemble de 5 éléments de la forme $(q, Guard, \sigma, Jump, q')$, avec q l'état précédant la transition, q' l'état la succédant, $Guard$ la condition à remplir pour pouvoir franchir la transition, σ l'événement reçu ou émis lors du franchissement de la transition et $Jump$ les changements ayant lieu sur les variables lors du franchissement de la transition (que ce soit une remise à zéro ou une nouvelle valeur). Les concepts de $Guard$ et de $Jump$ nous paraissent très intéressants et prometteurs.

Par ailleurs, bien que la composition d'automates, représentant un ensemble de composants, soit possible, il est impossible pour eux de partager un même état. Il est donc impossible de représenter une incertitude sur les observations ou sur l'état actuel du système. On peut toutefois représenter une incertitude sur les dynamiques.

2.2.2 Les réseaux de Petri

Les réseaux de Petri sont souvent utilisés pour la gestion et le contrôle de systèmes de production (Zhou et al. 1992 ; DiCesare et al. 1993 ; Balduzzi et al. 2001 ; Alla et al. 1998). Les réseaux de Petri ont l'avantage d'être intuitifs pour modéliser et spécifier le comportement de systèmes. Ils sont reconnus pour leur compacité et leur utilité dans les processus de prise de décision et de surveillance de système. Ils peuvent aussi être utilisés pour prouver certaines propriétés sur des systèmes. C'est pourquoi cette section va se focaliser sur des formalismes basés sur les réseaux de Petri. Tous ont en commun le principe de places, de jetons et de transitions. L'extension aux réseaux de Petri hybrides permet d'intégrer des aspects hybrides que nous allons détailler dans cette section.

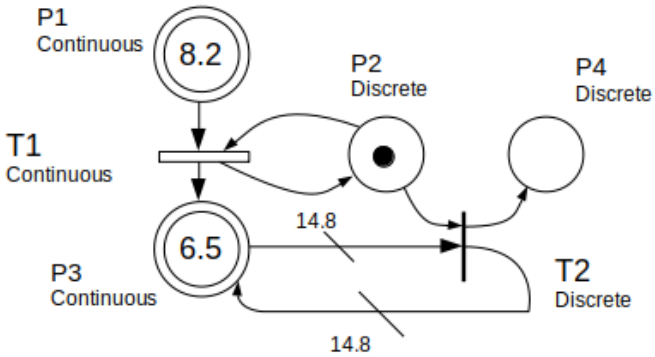


FIGURE 2.1 – Exemple de réseau de Petri hybride (ALLA et al. 1998)

2.2.2.1 Les réseaux de Petri hybrides

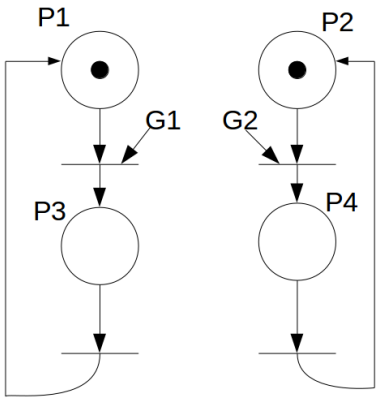
Dans les réseaux de Petri hybrides (ALLA et al. 1998), on distingue deux types de places : des places continues, représentées par une ligne double, et des places discrètes, représentées par une ligne simple, comme on peut le voir sur la figure 2.1.

Les places $P1$ et $P3$ sont des places continues, alors que les places $P2$ et $P4$ sont des places discrètes. Les jetons qui se situent dans les places continues sont des nombres réels ($8,2$ dans $P1$ par exemple), alors que les jetons se trouvant dans les places discrètes sont des nombres entiers (1 jeton dans $P2$, 0 jeton dans $P4$). Comme les places, il existe deux types de transitions dans ce formalisme. Des transitions continues, représentées sous forme de rectangle ($T1$ par exemple) et des transitions discrètes, représentées sous forme de trait noir ($T2$ par exemple). Dans le cas des transitions continues, une quantité de franchissement doit être définie et fonctionne comme un poids porté par les arcs dans des réseaux de Petri "classiques". Dans l'exemple, $T1$ doit être associée à une quantité de franchissement qui définira quelle quantité passera de $P1$ à $P3$ à chaque tic. Il est possible pour une transition donnée d'être connectée en amont aux deux types de places différents (c'est le cas pour $T1$ et $T2$), mais dans ce cas, une place discrète doit toujours être présente en entrée et en sortie de la transition. De plus, les poids sur les arcs entrants et sortants de la transition discrète doivent être identiques. Ceci est visible sur la figure 2.1 avec les transitions $T1$ et $T2$. En cas de conflit entre transitions continues et discrètes, la transition discrète est prioritaire. Le concept de parallélisme est applicable étant donné que plusieurs jetons peuvent évoluer simultanément dans le modèle.

Cependant, les places continues des réseaux de Petri hybrides ne sont pas associées à des dynamiques continues. Il est donc impossible de faire évoluer un vecteur d'état continu en fonction de la place du réseau représentant l'état du système et de le surveiller. De même, il n'est pas possible de représenter une notion de vieillissement ou de dégradation du système en fonction du niveau de stress induit par l'état dans lequel il se trouve.

2.2.2.2 Les réseaux de Petri mixtes

Dans les réseaux de Petri mixtes, présentés par VALENTIN-ROUBINET 1999, une place peut être continue et associée à des dynamiques d'évolution continue, ou discrète et représenter un phénomène discret. Un exemple de réseau de Petri mixte, ne contenant que des places continues, est illustré sur la figure 2.2.



(a)

EP	
P1	x(k) = x(k-1) +1
P2	y(k) = y(k-1) +1
P3	x(k) = x(k-1) -1
P4	y(k) = y(k-1) -1

(b)

EM	
P3 + P4	z(k) = z(k-1)+1

(c)

FIGURE 2.2 – Exemple de réseau de Petri mixte (a) et des ensembles EP (b) et EM (c)

Une ou plusieurs équations différentielles sont associées à une place continue. L'évolution des variables continues, qui sont des variables globales au système, s'effectue donc à travers deux ensembles différents, EP et EM. L'ensemble EP donne les équations activées par le marquage d'une place (P) particulière. Par exemple, dans le tableau de la figure 2.2 (b), un jeton présent dans la place $P1$ va incrémenter la valeur de la variable continue x. Un jeton présent dans la place $P2$ va incrémenter la valeur de la variable continue y. L'ensemble EM donne les équations activées par le marquage ($\mathbb{M}$) d'un ensemble de places. Par exemple, dans le tableau de la figure 2.2(c), si les places $P3$ et $P4$ sont marquées, la variable continue z sera incrémentée à chaque tic.

Les concepts de *Guard* et *Jump* des automates hybrides sont aussi présents dans les réseaux de Petri mixtes ($G1$ et $G2$ sur la figure 2.2 par exemple). Le parallélisme est applicable dans une certaine mesure, car il ne faut pas que deux places marquées en même temps influent sur les mêmes variables continues. En effet, comme dit précédemment, les variables continues sont globales et évoluent selon les ensembles d'équations actives. C'est une des limitations majeures des réseaux de Petri mixtes. Bien que ce formalisme se rapproche de nos besoins, il est tout de même restrictif sur le plan du parallélisme et de la représentation des incertitudes, comme deux places différentes ne peuvent influer sur les mêmes variables. Dans ce formalisme, une dégradation pourrait être représentée par les ensembles EP et EM.

2.2.2.3 Les réseaux de Petri pour systèmes embarqués

Un autre modèle basé sur les réseaux de Petri, nommé PRES, a été présenté par CORTÉS et al. 1999 pour modéliser des systèmes embarqués. Dans ce formalisme, un jeton est une paire $k = (v_k, r_k)$, où v_k est la valeur du jeton et r_k la date qui lui est associée. Un jeton est typé, cela signifie qu'il est de type entier ou réel ou complexe, etc. Les places sont, elles aussi, typées et ne peuvent accueillir que des jetons de leur type. Par exemple, une place définie comme place "entière" ne pourra contenir que des jetons de type "entiers". Le type de la place ne peut pas changer lors de l'évolution du réseau. Dans ce formalisme, une transition est associée à une fonction de sortie, qui fera évoluer la valeur du jeton, une fonction de retard, qui fera évoluer le temps associé au jeton, et une garde, qui représente la condition pour pouvoir tirer la transition. Bien que des idées intéres-

santes puissent être retrouvées dans ce travail, comme les concepts de *fonctions de sortie*, de *fonctions de retard* (similaires au concept de *Jump* dans les automates hybrides), de *gardes* (similaires au concept de *Guard*) et le fait que les jetons portent une information, certains aspects ne correspondent pas vraiment à nos attentes. Par exemple, comme dit précédemment, dans ce formalisme, les places sont typées et ne peuvent contenir que des jetons de ce type. Cela rend la combinaison de comportements discrets et continus assez difficile et peut entraîner une explosion du nombre de places dans des systèmes complexes. De plus, les valeurs des jetons n'évoluent pas en fonction des places dans lesquelles ils se trouvent, mais seulement via les fonctions de sortie et de retard associées aux transitions. Nous souhaiterions que les valeurs des jetons puissent évoluer en fonction des différentes places du réseau. Par contre, le parallélisme est représentable dans le formalisme, et la dégradation pourrait être représentée par un ensemble de places continues. Il faudrait cependant trouver un moyen de gérer son évolution, étant donné que les valeurs des jetons n'évoluent que via le tir de transitions.

2.2.2.4 Les réseaux de Petri hybrides particulaires

Lors de travaux précédents, un formalisme a été présenté dans Gaudel, Chanthery, Ribot et Daigle 2018 ; Gaudel, Chanthery et Ribot 2015, les HPPN. Un HPPN est formellement défini comme un ensemble de 11 éléments $< P, T, A, \mathring{A}, E, X, \Gamma, C, D, \Omega, \mathbb{M}_0 >$ réunissant les informations pour décrire des évolutions discrètes (à travers des places symboliques), des évolutions continues (à travers des places numériques), des mécaniques de dégradation (à travers des places de dégradation) et les relations des unes avec les autres (avec les transitions) :

- P est l'ensemble des places, réunissant les places symboliques P^S, les places numériques P^N et les places de dégradation P^D : $\mathrm{P} = P^S \cup P^N \cup P^D$
- T est l'ensemble des transitions
- $A \subset (P \times T \cup T \times P)$ est l'ensemble des arcs
- $\mathring{A}$ est l'ensemble des annotations des arcs
- E est l'ensemble des labels d'événements. C'est l'union de l'ensemble E_o des labels d'événements observables et de l'ensemble E_{uo} des labels d'événements non observables : $E = E_o \cup E_{uo}$
- $X \subset \mathbb{R}^{n_N}$ est l'espace d'état du vecteur d'état continu, avec $n_N \in \mathbb{N}_+$ le nombre fini de variables d'état continues
- $\Gamma \subset \mathbb{R}^{n_D}$ est l'espace d'état du vecteur d'état de dégradation, avec $n_D \in \mathbb{N}_+$ le nombre fini de variables d'état de dégradation
- C est l'ensemble des dynamiques continues associées aux places numériques
- D est l'ensemble des dynamiques de dégradation associées aux places de dégradation
- Ω est l'ensemble des conditions associées aux transitions
- $\mathbb{M}_0$ est le marquage initial du réseau

Trois types de places sont donc définis dans un HPPN : les places symboliques, numériques et de dégradation. Comme pour le formalisme PRES, une place ne peut contenir que des jetons du même type qu'elle. Un exemple d'un HPPN est visible sur la figure 2.3. Six places, deux symboliques (en vert), deux numériques (en bleu) et deux de dégradation (en gris) ainsi qu'une transition t sont présentées sur la figure. Une transition t doit avoir les mêmes types de places en entrée et en sortie. Ceci implique une difficulté à représenter

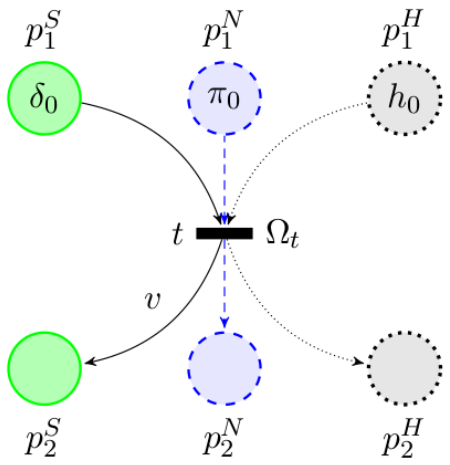


FIGURE 2.3 – Exemple de réseau de Petri particulaire (GAUDEL, CHANTHERY, RIBOT et DAIGLE 2018)

des systèmes ayant des parties aux dynamiques variées et entraîne facilement une explosion du nombre de places nécessaires à la représentation du système.

Une méthode de diagnostic a été développée à partir de ce formalisme. Ce formalisme est capable de représenter des incertitudes, à la fois sur le modèle et sur les observations. Il est aussi capable de représenter et surveiller le vieillissement du système. Cependant, ce formalisme ne s'applique qu'aux systèmes hybrides contenant à la fois des dynamiques continues et discrètes. Nous avons donc considéré ce formalisme et nous avons choisi de l'étendre pour qu'il soit capable de modéliser, simuler et surveiller tout type de système (continu, discret et hybride). De plus, ce premier formalisme était relativement complexe à prendre en main. Un effort de simplification a donc été réalisé.

2.2.3 Synthèse

Le tableau 2.1 synthétise l'état de l'art réalisé dans cette section, en se focalisant sur nos besoins.

Il indique si le formalisme étudié est capable de représenter tout type de systèmes ou non (afin de pouvoir représenter des systèmes hybrides composés de sous parties variées : discrètes, continues ou hybrides), si le concept de parallélisme est applicable ou non (pour représenter des systèmes multi-composants ou certaines incertitudes), si des incertitudes sur le modèle et/ou les observations sont représentables ainsi que la possibilité de représenter le vieillissement du système, à travers une dynamique de dégradation, par exemple. Le symbole $\simeq$ signifie que la propriété, bien que présente dans le formalisme, ne correspond pas complètement à nos besoins définis préalablement et ne permet pas de faire de la gestion de santé sur n'importe quel type de système.

Ce tableau met en évidence que, bien que la plupart de ces formalismes présentent des concepts très intéressants pour notre approche, il est nécessaire d'en créer un nouveau.

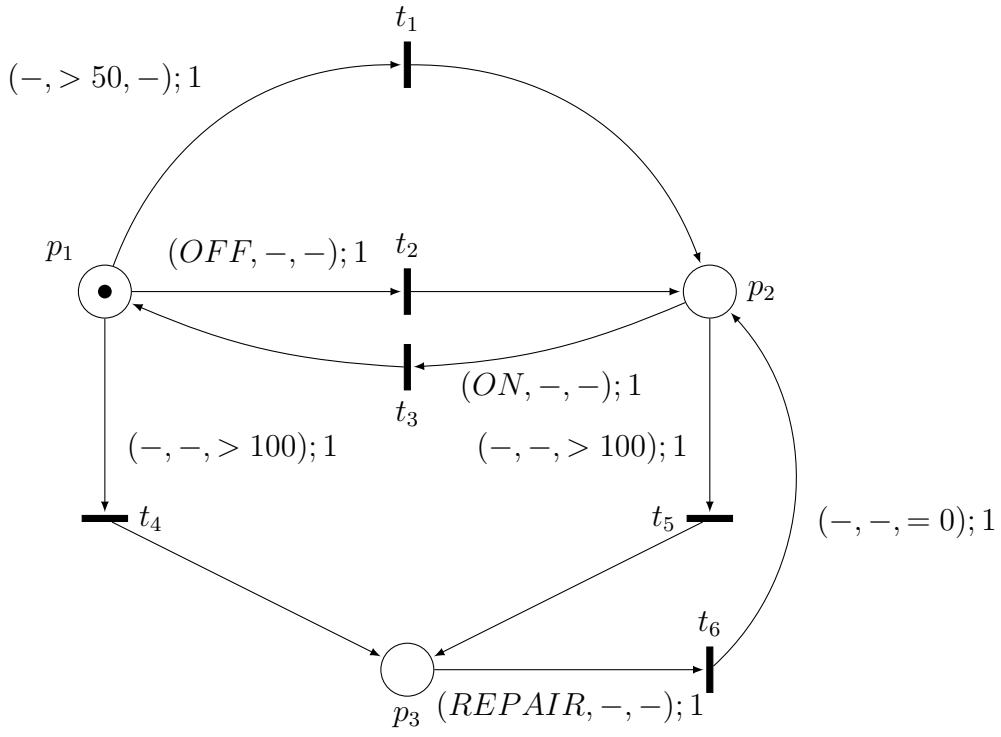
TABLEAU 2.1 – Synthèse de l'état de l'art

Reférences	Formalisme	Tout type de système ?	Parallélisme ?	Incertitudes sur le modèle ?	Incertitudes sur les observations ?	Représentation du vieillissement ?
Henzinger 2000, Alur et al. 1995	Automates hybrides	non	non	non	non	non
Zhou et al. 1992, DiCesare et al. 1993, Balduzzi et al. 2001	Réseaux de Petri	non	✓	non	non	non
Alla et al. 1998	Réseaux de Petri hybrides	non	✓	non	✓	non
Valentin-Roubinet 1999	Réseaux de Petri mixtes	non	≀	non	✓	✓
Cortés et al. 1999	PRES	≀	✓	non	non	≀
Gaudel, Chanthery, Ribot et Daigle 2018, Gaudel, Chanthery et Ribot 2015	HPPN	non	≀	✓	✓	✓

2.3 Le formalisme des HtPN

Comme nous avons pu le voir à travers l'état de l'art, les formalismes existants ne satisfont jamais parfaitement nos besoins. Il est donc pertinent de définir un nouveau formalisme étendant les réseaux de Petri classiques. Cette section présente ce nouveau formalisme, les réseaux de Petri hétérogènes (HtPN) et leur sémantique.

Un système hybride tel que nous l'avons défini dans l'introduction (Définition 7) a été modélisé en utilisant le formalisme des HtPN. Cet exemple sera utilisé en tant que fil rouge dans cette section afin d'illustrer les différentes notions introduites.



(a)

Places	Dynamiques continues	Dynamiques de dégradation
p_1	$x_{k+1} = x_k + 1$	$\gamma_{k+1} = \gamma_k + 3$
p_2	-	$\gamma_{k+1} = \gamma_k + 1$
p_3	-	-

(b)

FIGURE 2.4 – Exemple d'HtPN (a) et les dynamiques associées à ses places (b)

2.3.1 Présentation générale

La définition formelle d'un HtPN est la suivante.

Définition 19 (HtPN)**.** *Un HtPN est un ensemble composé des 5 éléments suivants :* $< P, T, Pre, Post, \mathbb{M}_0 >$ *avec*

— *P l'ensemble des places ;*

— *T l'ensemble des transitions ;*
— *Pre : $P \times T$ la matrice d'incidence avant, correspondant aux conditions de tir du système ;*
— *$Post$: $P \times T$ la matrice d'incidence arrière, correspondant aux assignations de tir du système ;*
— *$\mathbb{M}_0$ le marquage initial du réseau.*

Les 5 éléments de la définition 19 permettent de décrire les informations discrètes ainsi que les dynamiques continues et de dégradation du modèle. Les relations entre les places et les transitions sont explicitées à travers les matrices Pre et $Post$.

2.3.1.1 Places

Dans les HtPN, une place est un objet pouvant être associé à une dynamique continue et une dynamique de dégradation. L'information discrète est représentée par la place elle-même. On définit C et D comme les ensembles des dynamiques continues et de dégradation du système. Un ensemble d'équations bruitées $C_p \in C$ modélisant les dynamiques continues du système et les incertitudes sur les mesures et l'évolution (à travers la partie bruitée) et un ensemble d'équations $D_p \in D$ modélisant les dynamiques de dégradation du système sont associés à chaque place $p \in P$:

$$p = \begin{Bmatrix} C_p \\ D_p \end{Bmatrix} \tag{2.1}$$

L'ensemble d'équations C_p est défini comme suit :

$$C_p = \begin{cases} x_{k+1} = \mathbf{f}(x_k, u_k) + \mathbf{v}(x_k, u_k) \\ \quad y_k = \mathbf{g}(x_k, u_k) + \mathbf{w}(x_k, u_k) \end{cases} \tag{2.2}$$

avec $x_k \in \mathbb{R}^{n_x}$ le vecteur d'état continu composé de n variables d'état au temps k (X étant l'espace d'état continu), $u_k \in \mathbb{R}^{n_u}$ le vecteur des n_u variables d'entrée continues au temps k, $\mathbf{f}$ la fonction d'évolution non bruitée, $\mathbf{v}$ une fonction de bruit sur l'évolution continue, $y_k \in \mathbb{R}^{n_y}$ le vecteur composé des n_y variables continues de sortie au temps k, $\mathbf{g}$ la fonction non bruitée de sortie et $\mathbf{w}$ la fonction de bruit associée aux observations.

Les fonctions f, v, g et w sont dépendantes de la place p considérée. L'ensemble d'équations D_p est quant à lui défini comme :

$$D_p = \Big\{ \gamma_{k+1} = d(\gamma_k, b_k, x_k, u_k) + z(\gamma_k, b_k, x_k, u_k) \tag{2.3}$$

avec $\gamma_k \in \Gamma$ le vecteur d'état de dégradation (Γ étant l'espace d'état de dégradation) et d la fonction hybride non-bruitée de dégradation. Cette fonction dépend à la fois des événements discrets représentés par b_k et du vecteur d'état continu x_k. La fonction z représente le bruit sur l'évolution de la dégradation. Les fonctions d et z sont dépendantes de la place p considérée. Dans l'exemple de la figure 2.4, on distingue 3 places : p_1, p_2 and p_3. Leurs dynamiques continues et de dégradation sont explicitées dans le tableau 2.4 (b).

Dans le formalisme des HtPN, il n'est pas nécessaire d'associer des dynamiques continues et/ou de dégradation à une place. Ceci est illustré dans le tableau 2.4 (b) à travers

l'absence de dynamique continue pour p_2 et p_3 ou l'absence de dynamique de dégradation pour p_3.

Par défaut, une place p est définie comme n'étant associée à aucune dynamique continue ni aucune dynamique de dégradation. Elle est notée comme suit :

$$p = \begin{Bmatrix} - \\ - \end{Bmatrix} \tag{2.4}$$

2.3.1.2 Jetons

Une place p contient toujours $n_H(p)$ jetons ($n_H(p) \geq 0$). Chaque jeton h a trois attributs : un attribut discret, un attribut continu et un attribut de dégradation. Ces trois attributs sont représentés comme un ensemble $< \delta_k, \pi_k, \phi_k >$. δ_k représente l'information discrète au temps k, π_k représente l'information continue au temps k et ϕ_k l'information de dégradation au temps k. Ces attributs évoluent selon les événements discrets observés et les dynamiques associées à la place p dans laquelle se trouve le jeton considéré.

L'information discrète portée par un jeton h est appelée une **configuration**. La configuration d'un jeton est l'ensemble des événements qui ont eu lieu lors de la vie du système jusqu'au temps k et dont l'occurrence explique la situation courante du jeton. De manière formelle, δ_k est l'ensemble b_k des événements ayant eu lieu jusqu'au temps k :

$$b_k = \{(v, \kappa) | \kappa \leq k\}$$

où (v, κ) est un événement $v \in E$ ayant eu lieu à $t = \kappa$. E est l'ensemble des événements du système correspondant à l'union des ensembles E_o, des événements observables et E_{uo} des événements non observables.

L'information continue portée par un jeton est appelée **l'état du jeton**. L'état π_k représente le vecteur d'état continu $x_k \in X$ du système au temps k. L'état d'un jeton h va évoluer selon les dynamiques continues C_p (voir l'équation 2.2) associées à la place dans laquelle il est situé. Si aucune dynamique n'est spécifiée, l'état du jeton ne va pas évoluer et va donc rester constant.

L'information de dégradation portée par un jeton est appelée **le statut du jeton**. Le statut ϕ_k représente le vecteur d'état de dégradation $\gamma_k \in \Gamma$ au temps k. Le statut d'un jeton h va évoluer selon les dynamiques de dégradation D_p associées à la place dans laquelle il est situé. Si aucune dynamique n'est spécifiée, le statut du jeton ne va pas évoluer et va donc rester constant.

Dans notre contexte d'application à de la gestion de santé, il est possible d'associer un jeton à l'état de santé d'un système. La définition 15 est donc étendue de la manière suivante :

Définition 20 (Etat de santé)**.** ***Dans un modèle sous le formalisme des HtPN, il est représenté à travers un jeton présent dans une place et l'information qu'il porte : sa configuration, son état continu et son statut.*** *Ce concept est lié à la fois à l'occurrence de fautes, événements discrets non observables, et à l'état de dégradation, pouvant représenter le vieillissement du système.*

Définition 21 (Marquage)**.** *Le marquage $\mathbb{M}_k$ d'un HtPN est la distribution des jetons dans les différentes places du réseau au temps k.*

Le marquage initial $\mathbb{M}_0$ représente les conditions initiales du système. Chaque jeton porte ainsi une configuration initiale (l'ensemble des événements ayant eu lieu jusqu'au temps 0), son état continu initial et son statut de dégradation initial.

2.3.1.3 Arcs

Notons $A \subset (P \times T \cup T \times P)$ l'ensemble des arcs permettant de connecter les places P aux transitions T du réseau (et inversement).

Arcs entrants Pre est la matrice contenant l'ensemble des conditions de tirs portées par les arcs qui connectent les places aux transitions, de dimension $P \times T$. $Pre(t)$ est donc un vecteur contenant les conditions de tir des arcs liant un ensemble de places entrantes à une transition donnée t. Un arc qui connecte une place entrante p à une transition t est noté $a_{p,t}$. En se basant sur le concept de *Guard* des automates hybrides, $Pre(p,t)$ est défini comme suit :

$$Pre(p,t) = \begin{cases} ((\Omega^S_{p,t}, \Omega^N_{p,t}, \Omega^D_{p,t}); \rho_{p,t}) \text{ ssi } \exists a_{p,t} \in A \\ \emptyset \text{ sinon} \end{cases} \tag{2.5}$$

avec

— $(\Omega^S_{p,t}, \Omega^N_{p,t}, \Omega^D_{p,t})$ un triplet de conditions (respectivement une condition symbolique [1], une condition numérique [2] et une condition de dégradation).

— $\rho_{p,t} \in \mathbb{N}^{+*}$ le poids de l'arc.

Par défaut, si un arc connecte une place p à une transition t, cet ensemble est défini comme $Pre(p,t) = \{(\top, \top, \bot); 1\}$, ce qui signifie que s'il n'est pas spécifié, les conditions symbolique et numérique sont mises à VRAI (c.-à-d. qu'elles sont satisfaites par défaut), la condition de dégradation est définie à FAUX et le poids de l'arc est égal à 1. Si un élément n'est pas précisé lors de la définition de l'arc, qu'il s'agisse d'une condition du triplet ou du poids, cet élément prendra sa valeur par défaut.

La **condition symbolique** $\Omega^S_{p,t}$ est une condition liée à la configuration des jetons situés dans une place entrante p d'une transition t. Cette condition peut être mise à VRAI ($\top$), FAUX ($\bot$) ou bien tester l'occurrence d'un événement de E. Dans ce cas, cette condition est de la forme $\Omega^S_{p,t}(\delta_k) = occ(v)$ (VRAI s'il y a eu une occurrence de l'événement v).

La **condition numérique** $\Omega^N_{p,t}$ est une condition liée à l'état des jetons situés dans une place entrante p d'une transition t. Cette condition peut être mise à VRAI ($\top$), FAUX ($\bot$) ou représenter une, ou plusieurs contraintes portant sur le vecteur d'état continu. Dans ce cas, $\Omega^N_{p,t}(\pi_k) = c(x_k)$ est un test portant sur le vecteur d'état continu x_k. On peut distinguer différents types de conditions numériques : des conditions numériques dites simples et des conditions numériques dites complexes.

Définition 22 (Condition numérique simple)**.** $\Omega^N_{p,t}(\pi_k) = c(x_k)$ *est considérée comme simple si et seulement si* $c(x_k)$ *est une phrase logique ne contenant que des "ET" et dont les prédicats consistent chacun en une inégalité portant sur une seule variable* x_i *du vecteur d'état* x *du système.*

1. Dans ce manuscrit, nous utiliserons indifféremment symbolique ou discret
2. Dans ce manuscrit, nous utiliserons indifféremment numérique ou continu

Par exemple, la condition $x_1 \geq a \wedge x_2 \leq b$, avec b et a des valeurs réelles est considérée comme simple.

Définition 23 (Condition numérique complexe)**.** $\Omega^N_{p,t}(\pi_k) = c(x_k)$ *est considérée comme complexe si et seulement si elle n'est pas simple. En d'autre terme,* $c(x_k)$ *est une phrase logique pouvant contenir autre chose que des "ET" ("OU", "OU exclusif", etc.) et dont les prédicats consistent chacun en une inégalité qui peut porter sur une composition de plusieurs variables du vecteur d'état* x *du système.*

Par exemple, la condition $(x_1 + x_2) \geq a \vee (x_2 * x_4) \leq b$, avec b et a des valeurs réelles est considérée comme complexe. Ces définitions seront réutilisées dans le chapitre 3 en page 63.

La **condition de dégradation** $\Omega^D_{p,t}$ est une condition liée au statut des jetons situés dans une place entrante p d'une transition t. Cette condition peut être mise à VRAI ($\top$), FAUX ($\perp$) ou représenter une contrainte portant sur le vecteur d'état de dégradation. Dans ce cas $\Omega^D_{p,t}(\phi_k) = c(\gamma_k)$ est un test portant sur le vecteur d'état de dégradation γ_k.

Des exemples sont donnés sur la figure 2.4 (a) :

1. $Pre(p_1, t_1)$ porte une condition numérique qui vérifie si le vecteur d'état continu est supérieur à 50 ($x_k > 50$) et n'a ni condition symbolique ni condition de dégradation,
2. $Pre(p_1, t_2)$ porte une condition symbolique qui vérifie si l'événement OFF est apparu et n'a ni condition numérique ni condition de dégradation,
3. $Pre(p_1, t_4)$ porte une condition de dégradation qui vérifie si le vecteur d'état de dégradation est supérieur à 100 ($\gamma_k > 100$) et n'a ni condition symbolique ni condition numérique.

Définition 24 (Jeton accepté)**.** *Un jeton* h *est dit accepté par un arc entrant s'il satisfait :*

— *soit à la fois les conditions symboliques et numériques de l'arc,*
— *soit la condition de dégradation de l'arc.*

De manière formelle, en considérant une place p *telle que* $p \in P \wedge Pre(p,t) \neq \emptyset$ *:*

$$\begin{aligned} &\forall h =< \delta_k, \pi_k, \phi_k >\in p, Accept(h, a_{p,t}) \equiv \\ &((\Omega^S_{p,t}(\delta_k) = \top) \wedge (\Omega^N_{p,t}(\pi_k) = \top)) \vee (\Omega^D_{p,t}(\phi_k) = \top) \end{aligned} \tag{2.6}$$

L'ensemble des jetons situés dans la place p et acceptés par l'arc $a_{p,t}$ est noté $H_a(a_{p,t}, p)$:

$$h \in H_a(a_{p,t}, p) \equiv (Accept(h, a_{p,t}) = \top) \tag{2.7}$$

Le poids $\rho_{p,t}$ d'un arc qui connecte une place p à une transition t représente le nombre minimum de jetons acceptés requis pour pouvoir valider l'arc $a_{p,t}$.

Définition 25 (Arc validé)**.** *Considérons un arc* $a_{p,t}$*,* $n_{Ha(p,t)}$ *le nombre de jetons acceptés par l'arc* $a_{p,t}$ *(c.-à-d. le cardinal de l'ensemble* $H_a(a_{p,t}, p)$*) et* $\rho_{p,t}$ *le poids de l'arc, l'arc* $a_{p,t}$ *est dit validé si* $n_{Ha(p,t)} \geq \rho_{p,t}$*.*

Arcs sortants *Post* est la matrice contenant l'ensemble des assignations de tir portées par les arcs qui connectent les transitions aux places, de dimension $P \times T$. $Post(t)$ est

donc un vecteur contenant les assignations de tir des arcs qui connectent la transition t à ses places de sortie. Un arc connectant une transition t à une place de sortie p est noté $a_{t,p}$. En se basant sur le concept de *Jump* des automates hybrides, $Post(t,p)$ est défini comme suit :

$$Post(t,p) = \begin{cases} ((\Omega^S_{t,p}, \Omega^N_{t,p}, \Omega^D_{t,p}); \rho_{t,p}) \text{ ssi } \exists a_{t,p} \in A \\ \emptyset \text{ sinon} \end{cases} \tag{2.8}$$

avec

— $(\Omega^S_{t,p}, \Omega^N_{t,p}, \Omega^D_{t,p})$ un triplet d'assignations (respectivement une symbolique, une numérique et une de dégradation).

— $\rho_{t,p} \in \mathbb{N}^{+*}$ le poids de l'arc.

Le symbole $-$ pour une assignation signifie qu'aucune modification de l'attribut concerné n'aura lieu. Par défaut, $Post(t,p) = \{(-,-,-);1\}$, signifie qu'aucune assignation n'est spécifiée. Un poids égal à 1 implique qu'un seul jeton sera placé dans la place de sortie de t.

L'assignation symbolique $\Omega^S_{t,p}$ concerne la configuration des jetons qui passent par l'arc sortant $a_{t,p}$. Soit δ_k la configuration d'un jeton h passant par l'arc $a_{t,p}$ à un temps k et ayant comme valeur b_k :

— si $\Omega^S_{t,p} = v$, avec $v \in E$, alors l'événement v est concaténé avec la configuration actuelle du jeton passant par l'arc :

$$b_{k+1} \longleftarrow b_k \cup (v, k+1) \tag{2.9}$$

— si $\Omega^S_{t,p} = b_{new}$, avec b_{new} un nouvel ensemble d'événements datés, la configuration est effacée et contient dorénavant b_{new} (dans le cas d'une réparation, par exemple, où l'on souhaite "réinitialiser" la configuration à une valeur donnée) :

$$b_{k+1} \longleftarrow b_{new}, \tag{2.10}$$

— si $\Omega^S_{t,p} = -$, alors la configuration du jeton passant par l'arc reste inchangée :

$$b_{k+1} \longleftarrow b_k. \tag{2.11}$$

L'assignation numérique $\Omega^N_{t,p}$ concerne l'état des jetons qui passent par l'arc sortant $a_{t,p}$. Soit π_k l'état d'un jeton h traversant l'arc $a_{t,p}$ à un temps k et π_k portant une valeur du vecteur d'état continu x_k.

— si $\Omega^N_{t,p} = x_{new}$, alors x_{new} représente la nouvelle valeur pour le vecteur d'état continu x_k porté par l'état du jeton π_k :

$$x_{k+1} = x_{new}, \tag{2.12}$$

— si $\Omega^N_{t,p} = -$, alors la valeur du vecteur d'état continu porté par π_k reste inchangée :

$$x_{k+1} = x_k. \tag{2.13}$$

L'assignation numérique $\Omega^N_{t,p}$ donne une condition initiale pour l'état du jeton qui traverse l'arc. Cet état va ensuite évoluer selon l'ensemble d'équations continues $C_p \in C$ défini dans l'équation 2.2 de la place p dans laquelle se trouve le jeton.

L'assignation de dégradation $\Omega^D_{t,p}$ concerne le statut des jetons qui passent par l'arc sortant $a_{t,p}$. Soit ϕ_k le statut d'un jeton h traversant l'arc $a_{t,p}$ à un temps k et γ_k la valeur du vecteur d'état de dégradation porté par ϕ_k,

— si $\Omega^D_{t,p} = \gamma_{new}$, γ_{new} représente la nouvelle valeur pour le vecteur d'état de dégradation :

$$\gamma_{k+1} = \gamma_{new}, \tag{2.14}$$

— si $\Omega^D_{t,p} = -$, alors la valeur du vecteur d'état de dégradation porté par ϕ_k reste inchangée :

$$\gamma_{k+1} = \gamma_k. \tag{2.15}$$

L'assignation de dégradation $\Omega^D_{t,p}$ donne une condition initiale pour le statut du jeton qui traverse l'arc. Ce statut va ensuite évoluer selon l'ensemble d'équations de dégradation $D_p \in D$ de la place p dans laquelle se trouve le jeton. Un exemple d'assignation de dégradation est visible sur la figure 2.4 (a) : $Post(t_6, p_2)$ va donner la valeur 0 au vecteur d'état de dégradation lorsque le jeton passe par cet arc.

Le poids $\rho_{t,p}$ définit le nombre de jetons à placer dans la place de sortie p de l'arc $a_{t,p}$. Selon le poids, des jetons peuvent se trouver dupliqués et/ou détruits. Un poids $\rho_{t,p}$ inférieur au nombre de jetons tirés, c.-à-d. au nombre de jetons déplacés par le tir de la transition t considérée, va entraîner la destruction de certains de ces jetons. Un poids $\rho_{t,p}$ plus grand que le nombre de jetons tirés va à l'inverse entraîner la duplication de certains de ces jetons. Ces destructions ou duplications sont effectuées aléatoirement. Ce phénomène sera formellement expliqué dans la section 2.3.2.2. Dans la figure 2.5, ces deux cas peuvent être observés : la destruction (a-b) et la duplication (c-d) de jetons. Dans le premier cas, trois jetons h_1, h_2 et h_3 sont présents dans p_1 et vont être tirés par la transition t. Cependant, l'arc sortant a_{t,p_2} a un poids ρ_{t,p_2} égal à 1. Deux jetons parmi les trois seront donc détruits lors du tir de cette transition pour n'avoir qu'un seul jeton dans la place de sortie p_1. Dans le second cas, un seul jeton h est présent dans p_1 et tiré par la transition t. Cependant, l'arc a_{t,p_2} a un poids ρ_{t,p_2} égal à 3. Le jeton h sera alors dupliqué jusqu'à ce que 3 jetons h_1, h_2 et h_3 soient présents dans la place de sortie p_2.

2.3.2 Règles de tirage

2.3.2.1 Transition activée

Une transition $t \in T$ est activée (*enabled*) à un temps k si tous les arcs entrants dans cette transition sont validés (voir la définition 25)

$$enabled(t) \equiv \Big(\forall p \text{ t.q } a_{p,t} \in A_{\bullet t},\ n_{Ha(p,t)} \geq \rho_{p,t}\Big) \tag{2.16}$$

avec $A_{\bullet t}$ l'ensemble des arcs situés en amont de la transition t, $n_{Ha(p,t)}$ le nombre de jetons acceptés par l'arc entrant $a_{p,t}$ et $\rho_{p,t}$ le poids de l'ensemble de conditions $Pre(p,t)$.

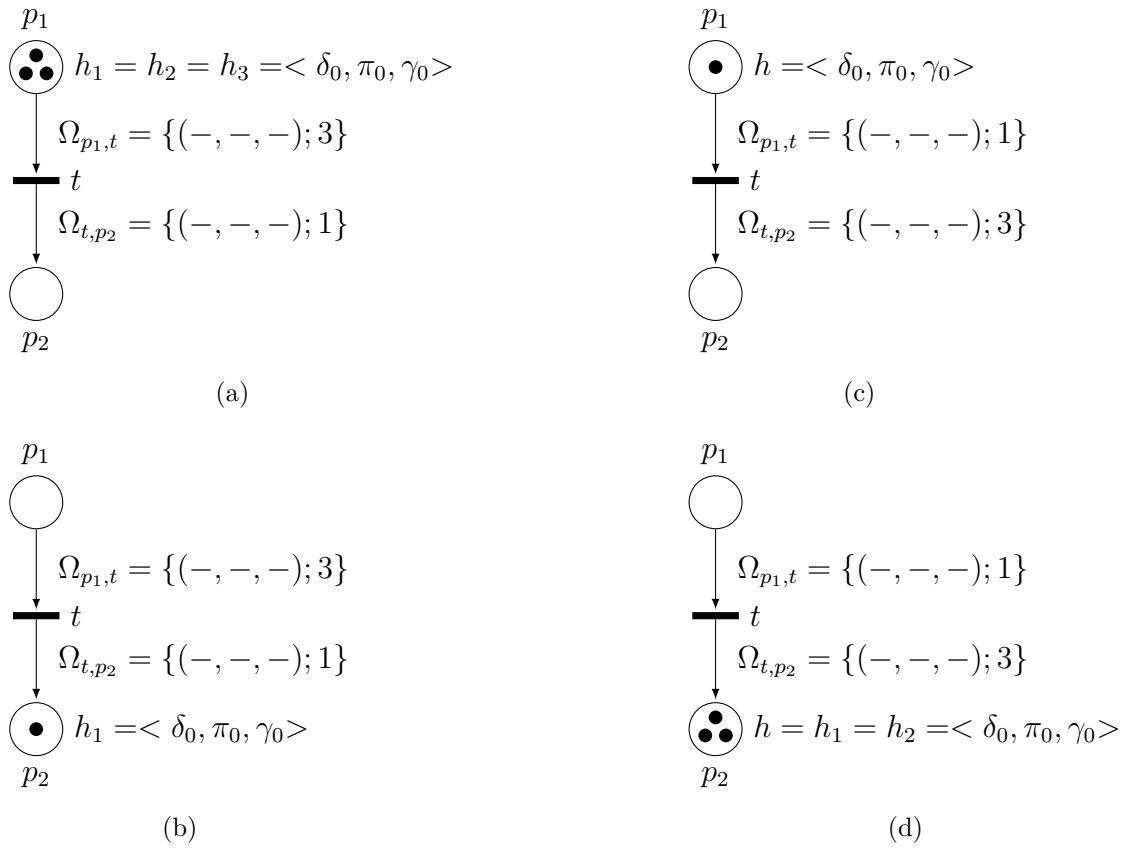


FIGURE 2.5 – Exemples de destruction (a-b) et duplication (c-d) de jetons

2.3.2.2 Ensemble de jetons tirés

Quand $n_{Ha(p,t)}$, le nombre de jetons acceptés par l'arc $a_{p,t}$, est supérieur au poids de l'arc $\rho_{p,t}$, une fonction de sélection aléatoire ${}^{\bullet}\zeta : \mathbb{N}^+ \times H_a \rightarrow H_a$ est définie pour choisir les $\rho_{p,t}$ jetons qui vont être tirés parmi l'ensemble de jetons acceptés $H_a(a_{p,t}, p)$.

Soit p telle que $p \subset P \wedge Pre(p,t) \neq \emptyset$. L'ensemble des jetons sélectionnés pour être tirés parmi les jetons acceptés est noté $\Psi(p,t)$ et est formellement défini comme suit :

$$\Psi(p,t) = \begin{cases} {}^{\bullet}\zeta(\rho_{p,t}, H_a(a_{p,t}, p)) & \text{si } n_{Ha(p,t)} > \rho_{p,t} \\ H_a(a_{p,t}, p) & \text{sinon.} \end{cases} \tag{2.17}$$

À partir de $\Psi(p,t)$, il est possible de définir $\Psi(\bullet t)$ comme l'ensemble des jetons présents dans toutes les places $p_1, p_2, ..., p_i$ connectées en amont de t qui seront tirés par la transition t :

$$\Psi(\bullet t) = \Psi(p_1, t) \cup \Psi(p_2, t) \cup ... \cup \Psi(p_i, t) \tag{2.18}$$

Une autre fonction de sélection aléatoire $\zeta^{\bullet} : \mathbb{N}^+ \times \Psi(\bullet t) \rightarrow \Psi(\bullet t)$ est définie pour sélectionner les jetons qui seront gardés, dupliqués ou détruits parmi l'ensemble de jetons tirés $\Psi(\bullet t)$. Cette fonction dépend de l'ensemble d'assignations $Post(t,p)$ porté par les arcs sortants de la transition. Si aucun poids n'est défini, il n'y aura pas de duplication de jetons. Cependant, il est toujours possible qu'une destruction ait lieu si le nombre de jetons tirés est plus grand que le poids $\rho_{t,p}$.

2.3.2.3 Tir de transitions

Durant le tir d'une transition t, les jetons tirés $\Psi(\bullet t)$ sont déplacés dans les places de sorties de t. Les attributs des jetons sont soit conservés, soit modifiés par les assignations $\Omega_{t,p}$. Comme précisé dans la section 2.3.1.3, cette modification, ainsi que les possibles destructions ou duplications de jetons sont définies par l'ensemble d'assignations $\Omega_{t,p}$ spécifiées dans $Post(t,p)$ pour les arcs sortants d'une transition t considérée. Pour rappel, si $Post(t,p)$ ne contient aucune information, les attributs des jetons ne sont pas modifiés.

Le tir d'une transition t à un temps k est formellement défini comme suit :

$\forall p \in P \wedge Pre(p,t) \neq \emptyset$ and $\forall p' \in P \wedge Post(t,p') \neq \emptyset$,

$$\begin{array}{rcl} n_{H,k+1}(p) & = & n_{H,k}(p) - \rho_{p,t} \\ n_{H,k+1}(p') & = & n_{H,k}(p') + \rho_{t,p'} \end{array} \tag{2.19}$$

avec $\rho_{p,t}$ le poids de l'arc entrant $a_{p,t}$ (reliant la place p à la transition t), $\rho_{t,p'}$ le poids de l'arc sortant $a_{t,p'}$ (reliant la transition t à la place p') et $n_{H,k}(p)$ le nombre de jetons présents dans la place p au temps k. $\mathbb{M}_k(p)$ représente l'ensemble des jetons présents dans la place p au temps k :

$$\begin{array}{rcl} \mathbb{M}_{k+1}(p) & = & \mathbb{M}_k(p) \setminus \Psi(p,t) \\ \mathbb{M}_{k+1}(p') & = & \mathbb{M}_k(p') \cup \zeta^{\bullet}(\rho_{t,p'}, \Psi(\bullet t)) \end{array} \tag{2.20}$$

2.4 Application du formalisme des HtPN à différents systèmes

Le formalisme des HtPN ayant été défini, il est désormais possible de l'utiliser pour modéliser et simuler le comportement de différents systèmes.

Une méthodologie de modélisation est tout d'abord présentée dans un cadre général. Ensuite, pour illustrer l'utilisation du formalisme des HtPN, deux applications sont proposées. Dans un premier temps, un système de production de Motorola issu de la littérature est modélisé en HtPN, afin de montrer que le formalisme permet de représenter les mêmes informations que les réseaux de Petri classiques. Puis, un système hybride du monde académique est à son tour modélisé.

Les représentations HtPN de ces deux systèmes ont été implémentées en python et fournies au logiciel HeMU pour pouvoir simuler leur comportement.

2.4.1 Démarche de modélisation en HtPN

L'objectif de cette première sous-section est de proposer une démarche générale pour modéliser un système en utilisant le formalisme des HtPN. Il est important de noter que cette démarche n'est pas unique. Elle n'est qu'un guide amendable destiné au modélisateur novice en HtPN pour guider la démarche de modélisation.

Cette démarche peut se découper en trois étapes. La première étape consiste à identifier des modes de fonctionnement nominaux. La deuxième étape se focalise sur le problème de gestion de santé à travers l'identification des fautes et des modes de fonctionnement dégradés. La troisième étape est l'ajout de places ne correspondant pas à des modes

de fonctionnement, elles permettent de représenter des états particuliers ou de compter l'occurrence d'un événement par exemple. Dans le cas d'une modélisation axée contrôle, sans s'intéresser à la gestion de santé, les première et troisième étapes suffisent donc.

Ces 3 étapes sont généralisables à d'autres formalismes, comme les automates hybrides. Nous spécifions par la suite chacune des étapes en utilisant le formalisme des HtPN.

Étape 1 : identification des modes de fonctionnement nominaux et des conditions d'évolution La première étape pour modéliser un système consiste à identifier ses modes de fonctionnement nominaux, les liens entre ces modes (l'occurrence d'un événement, le franchissement d'un seuil sur une variable continue ou une combinaison des deux), leurs dynamiques continues (c.-à-d. comment évoluent les variables continues dans ces modes de fonctionnement) et leurs dynamiques de dégradation (c.-à-d. comment vieillit le système dans ces modes de fonctionnement).

Création des places P et identification des dynamiques continues C et de dégradation D Pour chaque mode de fonctionnement identifié, une place p_i est créée. Les dynamiques continues qui ont été identifiées pour chaque mode de fonctionnement correspondent aux dynamiques continues C_i associées à la place p_i. Les dynamiques de dégradation identifiées correspondent quant à elles aux dynamiques de dégradation D_i associées à la place p_i.

Création des transitions T Les liens entre les modes correspondent aux transitions entre les modes. À chaque fois qu'il est possible de passer d'un mode i (associé à une place p_i) à un mode j (associé à une place p_j), une transition $t_{i,j}$ est créée.

Création de l'ensemble Pre La création de l'ensemble Pre dépend des conditions pour changer de mode :

1. si ce changement de mode s'opère via l'occurrence d'un événement observable $e \in E_o$, une condition symbolique vérifiant l'occurrence de e, $\Omega^S_{p_i,t_{i,j}} = occ(e)$ est associée à l'arc entrant $a_{p_i,t_{i,j}}$, reliant la place p_i à la transition $t_{i,j}$.
2. si ce changement de mode s'opère via le franchissement d'un seuil $seuil_{i,j}$ sur une variable x_i continue, une condition numérique vérifiant le franchissement de ce seuil $\Omega^N_{p_i,t_{i,j}} = x_i > seuil_{i,j}$ est associée à l'arc entrant $a_{p_i,t_{i,j}}$.
3. si ce changement de mode est une combinaison des deux cas précédents, une condition symbolique vérifiant l'occurrence de e, $\Omega^S_{p_i,t_{i,j}} = occ(e)$ et une condition numérique vérifiant le franchissement de ce seuil $\Omega^N_{p_i,t_{i,j}} = x_i > seuil_{i,j}$ doivent être associées à l'arc entrant $a_{p_i,t_{i,j}}$.
4. si ce changement de mode de fonctionnement nécessite la présence d'un certain nombre de composants N, alors un poids $\rho_{p_i,t_{i,j}} = N$ doit être associé à l'arc entrant $a_{p_i,t_{i,j}}$.

Création de l'ensemble $Post$ La création de l'ensemble $Post$ dépend des variables modifiées et des événements émis par le système à l'issue du changement de mode :

1. si un événement observable $e \in E_o$ est émis lors du changement de mode, une assignation symbolique, représentant l'occurrence de e, $\Omega^S_{t_{i,j},p_j} = (e)$ est associée à l'arc sortant $a_{t_{i,j},p_j}$, reliant la transition $t_{i,j}$ à la place p_j.
2. si le changement de mode induit une évolution des valeurs continues vers une nouvelle valeur x_{new}, une assignation numérique représentant cette évolution $\Omega^N_{t_{i,j},p_j} = x_{new}$ est associée à l'arc sortant $a_{t_{i,j},p_j}$.
3. si ce changement de mode induit une combinaison des deux cas précédents, une assignation symbolique représentant l'occurrence de e, $\Omega^S_{t_{i,j},p_j} = (e)$ et une assignation numérique représentant l'évolution des variables continues $\Omega^N_{t_{i,j},p_j} = x_{new}$ doivent être associées à l'arc sortant $a_{t_{i,j},p_j}$.
4. si ce changement de mode de fonctionnement implique l'apparition ou la destruction d'un certain nombre de composants N, alors un poids $\rho_{t_{i,j},p_j} = N$ doit être associé à l'arc sortant $a_{t_{i,j},p_j}$.

Étape 2 : identification des modes de fonctionnement dégradé La deuxième étape pour modéliser un système est d'identifier des modes de fonctionnement dégradé. Il s'agit d'identifier les différentes fautes pouvant avoir lieu sur le système, leurs conditions et/ou probabilités d'occurrence, ainsi que leurs possibles impacts sur les dynamiques continues et de dégradation du système. Une fois les modes de fonctionnement dégradé identifiés, de nouvelles places peuvent être créées dans le modèle HtPN. Les fautes les induisant sont représentées par un événement discret non observable ou le franchissement d'un seuil par une variable de dégradation, par exemple. Ces fautes sont indiquées dans les conditions associées aux arcs entrants du modèle. La représentation des modes dégradés identifiés par des places p_i et des liens entre les modes par des transitions $t_{i,j}$ s'opère de manière similaire à l'étape 1.

Création de l'ensemble Pre La création de l'ensemble Pre dépend des différentes manières de représenter l'occurrence d'une faute f impliquant le passage d'un mode nominal vers le mode dégradé considéré :

1. si f est représentée par l'occurrence d'un événement discret non observable $f \in E_{uo}$, une condition symbolique vérifiant l'occurrence de f, $\Omega^S_{p_i,t_{i,j}} = occ(f)$ est associée à l'arc entrant $a_{p_i,t_{i,j}}$, reliant la place p_i à la transition $t_{i,j}$.
2. si f est représentée par le franchissement d'un seuil $seuil_f$ sur une variable γ_i de dégradation, une condition de dégradation vérifiant le franchissement de ce seuil $\Omega^N_{p_i,t_{i,j}} = \gamma_i > seuil_f$ est associée à l'arc entrant $a_{p_i,t_{i,j}}$.
3. si l'occurrence de f nécessite la présence d'un certain nombre de composants N, alors un poids $\rho_{p_i,t_{i,j}} = N$ doit être associé à l'arc entrant $a_{p_i,t_{i,j}}$.

La partie de l'ensemble Pre ne portant pas sur l'occurrence des fautes se définit de la même manière que pour l'ensemble Pre de l'étape 1.

Création de l'ensemble $Post$ La création de l'ensemble $Post$ dépend de ce qu'il se passe lors de l'occurrence de la faute f (voir remarque 1) :

1. si un événement observable $e \in E_o$ est émis lors de l'occurrence de f, une assignation symbolique représentant l'occurrence de e, $\Omega^S_{t_{i,j},p_j} = (e)$ est associée à l'arc sortant $a_{t_{i,j},p_j}$, reliant la transition $t_{i,j}$ à la place p_j.

2. si l'occurrence de f induit une évolution des variables continues vers une nouvelle valeur x_{new}, une assignation numérique représentant cette évolution $\Omega^N_{t_{i,j},p_j} = x_{new}$ est associée à l'arc sortant $a_{t_{i,j},p_j}$.
3. si l'occurrence de f induit une combinaison des deux cas précédents, une assignation symbolique représentant l'occurrence de e, $\Omega^S_{t_{i,j},p_j} = (e)$ et une assignation numérique représentant l'évolution des variables continues $\Omega^N_{t_{i,j},p_j} = x_{new}$ doivent être associées à l'arc sortant $a_{t_{i,j},p_j}$.
4. si l'occurrence de f implique l'apparition ou la destruction d'un certain nombre de composants N, alors un poids $\rho_{t_{i,j},p_j} = N$ doit être associé à l'arc sortant $a_{t_{i,j},p_j}$.

La partie de l'ensemble *Post* ne portant pas sur l'effet des occurrences des fautes se définit de la même manière que pour l'ensemble *Post* de l'étape 1.

Remarque 1. *Le fait que la faute f ait un impact directement observable rend la faute observable, ce qui n'est généralement pas le cas et rendrait le problème de diagnostic trivial et inutile. L'ensemble Post est généralement laissé vide après l'occurrence d'une faute f.*

Bien évidemment, plusieurs modes de fautes peuvent s'enchaîner, et mener à des modes de défaillance. De même, il serait possible de définir des transitions permettant de revenir dans des modes nominaux à partir de modes de faute ou de défaillance en cas de réparation du système.

Étape 3 : compléments de modélisation La troisième étape pour modéliser un système est l'ajout de tout élément nécessaire à spécifier le comportement du système. On peut par exemple ajouter des places qui ne représentent pas un mode de fonctionnement du système. Ces places peuvent être utilisées pour représenter l'arrivée du système dans un état particulier, compter l'occurrence d'un événement ou du franchissement d'un seuil par exemple, ou représenter des régimes transitoires présents avant d'atteindre un régime permanent dans un mode de fonctionnement donné.

2.4.2 Application à un système de production

2.4.2.1 Description du système

Pour illustrer le formalisme et montrer qu'il est capable de représenter les mêmes informations que les réseaux de Petri classiques, un exemple d'un réseau de Petri hybride tiré de Alla et al. 1998 a été représenté et simulé. Afin de réduire le temps de calcul, les nombres présents dans l'exemple d'origine (poids, nombre de jetons, etc.) ont été divisés par 10. Il est cependant important de noter que le logiciel est capable de gérer les valeurs d'origine. Le système avec les valeurs d'origine est disponible sur notre GitLab. Le système étudié représente un système de production de Motorola. Ce système est capable de traiter deux types de pièces, type-L et type-R, arrivant en lot. Quand un lot de type-L arrive, il est immédiatement décomposé en 3000 pièces, qui seront traitées individuellement sans interruption et placées en attente. Quand 50 pièces sont en attente, le traitement des pièces de type-L commence après une attente de 30 unités de temps (ce qui correspond au réglage du système pour un type-L). Une fois toutes les pièces du lot traitées, le système se met en veille et est disponible pour traiter un autre lot.
Pour un lot de type-R, le processus est presque identique, aux différences suivantes près.

Tout d'abord, un lot de pièces de type-R ne contient que 2000 pièces. Ensuite, au niveau du début du processus, une attente de 100 unités de temps a lieu. Le délai avant le traitement est quant à lui fixé à 36 unités de temps. De plus, il faut que 60 pièces soient en attente avant que le processus de traitement des pièces de type-R ne commence. Le système considéré ici n'est ni purement discret, étant donné que le temps qui passe est représenté par des dynamiques continues, ni purement continu, car le système ne peut être représenté en utilisant uniquement des dynamiques continues : il est donc considéré comme un système hybride. Étant donné que le système a été développé dans le but de faire du contrôle, l'aspect gestion de santé avec les dynamiques de dégradation n'est pas considéré et donc non représenté dans le modèle.

2.4.2.2 Modélisation en HtPN

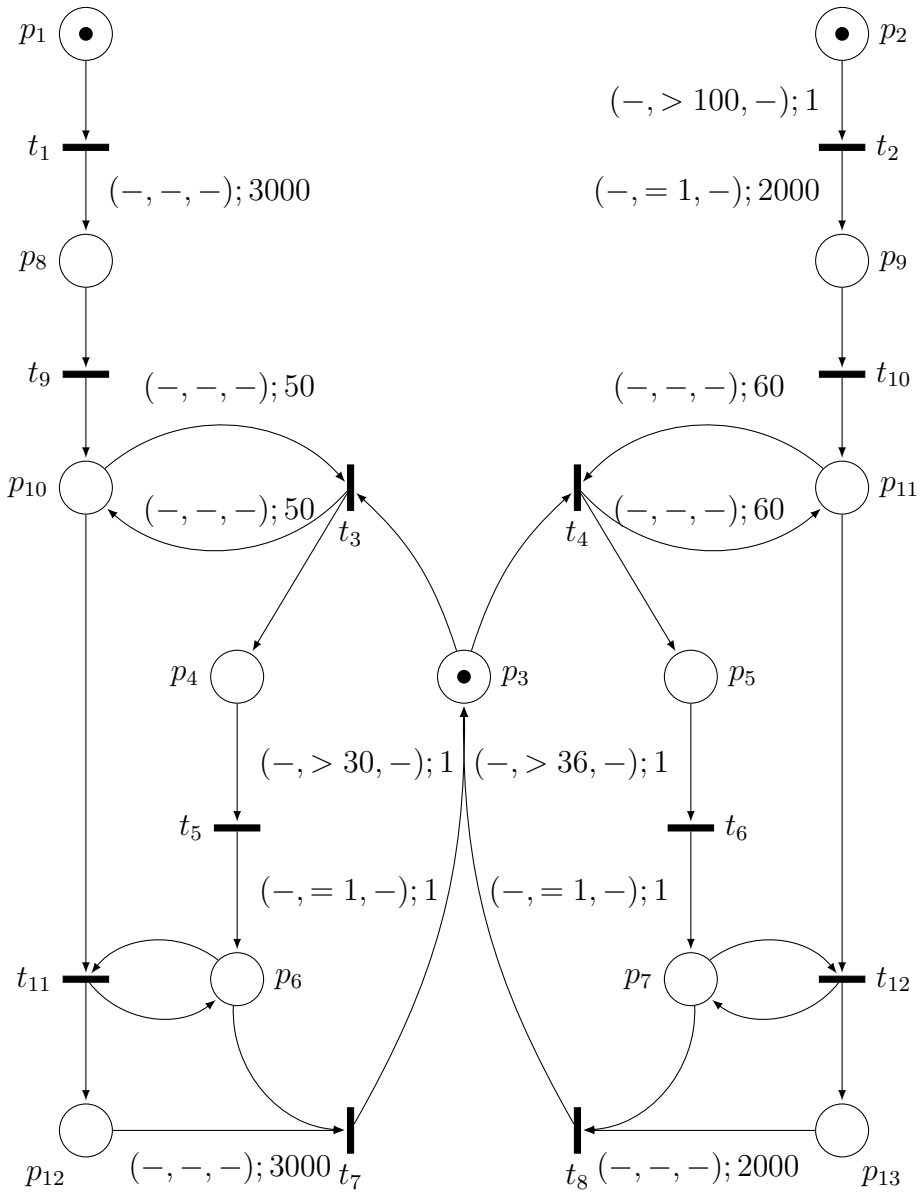


FIGURE 2.6 – HtPN modélisant le comportement du système de production de Motorola

Le système modélisé en utilisant le formalisme des HtPN est illustré sur la figure 2.6. La partie gauche de la figure représente les pièces de type-L et la partie droite de la figure les pièces de type-R. L'ensemble P est composé de 13 places : $P = \{p_1, \ldots, p_{13}\}$. Ce modèle correspond à notre définition de système hybride, car il est composé de places n'ayant aucune dynamique continue (purement discrète) qui communiquent avec des places présentant des dynamiques continues (p_2, p_4 et p_5).

Le marquage initial de cet HtPN est $\mathbb{M}_0 = \{p_1, p_2, p_3\}$. Dans ce marquage initial, les jetons ont une configuration nulle (qui le restera étant donné que l'ensemble des événements E est vide), un état continu $\pi = [1]$ qui représente le temps passé dans chaque place : x_k^h représente le temps que le jeton h a passé dans la place p à un temps k donné. Comme il n'y a pas de dynamique de dégradation, le statut ϕ du jeton est mis à 0.

Les dynamiques continues de l'ensemble C incrémentent l'état continu du jeton de 1 dans les places p_2, p_4 et p_5, qui sont les places concernées par le temps dans le système étudié :

$$C_{p_i} = \Big\{x_{k+1} = x_k + 1, \quad i = \{2, 4, 5\}\Big\} \tag{2.21}$$

L'ensemble des transitions T est composé de 12 transitions : $T = \{t_1, \ldots, t_{12}\}$.

Une condition numérique $\Omega_{p,t}^N$ a été utilisée pour représenter le temps écoulé depuis l'arrivée du jeton dans la place. C'est par exemple visible pour les ensembles de conditions $Pre(p_2, t_2)$, $Pre(p_4, t_5)$, $Pre(p_5, t_6)$ qui seront détaillés un peu plus tard dans cette section.

Pour simuler le nombre de pièces, les poids sur les arcs ont été fixés en accord avec la description du système. Par exemple, ρ_{t_1,p_8} a une valeur de 3000, pour simuler le fait que le lot est décomposé en 3000 pièces.

Les éléments non nuls des matrices Pre et $Post$ sont détaillés ci-dessous :

- $Pre(p_2, t_2) = \{(-, \pi_k > 100, -); 1\}$ pour représenter les 100 unités de temps à attendre en p_2,
- $Pre(p_{10}, t_3) = \{(-, -, -); 50\}$ pour représenter la présence de 50 pièces en p_{10} avant le début du traitement des pièces de type-L,
- $Pre(p_{11}, t_4) = \{(-, -, -); 60\}$ pour représenter la présence de 60 pièces en p_{11} avant le début du traitement des pièces de type-R,
- $Pre(p_4, t_5) = \{(-, \pi_k > 30, -); 1\}$ pour représenter les 30 unités de temps à attendre en p_4,
- $Pre(p_5, t_6) = \{(-, \pi_k > 36, -); 1\}$ pour représenter les 36 unités de temps à attendre en p_5,
- $Pre(p_{12}, t_7) = \{(-, -, -); 3000\}$ pour représenter la fin du traitement des 3000 pièces de type-L,
- $Pre(p_{13}, t_8) = \{(-, -, -); 2000\}$ pour représenter la fin du traitement des 2000 pièces de type-R.
- $Post(t_1, p_8) = \{(-, -, -); 3000\}$ pour représenter la transformation du lot de type-L en 3000 pièces de type-L,
- $Post(t_2, p_9) = \{(-, \pi = 1, -); 2000\}$ pour représenter la transformation du lot de type-R en 2000 pièces de type-R et réinitialiser l'état continu des jetons franchissant la transition,
- $Post(t_3, p_{10}) = \{(-, -, -); 50\}$ est associé à $Pre(p_{10}, t_3)$ dans la vérification du nombre de pièces en p_{10},

— $Post(t_4, p_{11}) = \{(-,-,-); 60\}$ est associé à $Pre(p_{11}, t_4)$ dans la vérification du nombre de pièces en p_{11},
— $Post(t_5, p_6) = \{(-, \pi = 1, -); 1\}$ pour réinitialiser l'état continu des jetons franchissant la transition,
— $Post(t_6, p_7) = \{(-, \pi = 1, -); 1\}$ pour réinitialiser l'état continu des jetons franchissant la transition.

2.4.2.3 Résultats de simulation

La durée totale de simulation est de 6000s, cela est suffisant pour que le système traite un lot de type-R et un lot de type-L, puis pour qu'il se remette en veille. Quand le système est en veille et disponible, un jeton est présent dans la place p_3. Un jeton en p_1 représente le fait qu'un lot de type-L est disponible et attend d'être traité. Ce lot est décomposé en 3000 pièces, représentées par 3000 jetons dans la place p_8. Cette décomposition est représentée par le tir de la transition t_1. La transition t_9 est ensuite tirée et représente la mise en attente de pièces en p_{10}. Quand 50 jetons sont disponibles en p_{10}, la transition t_3 est tirée, ce qui entraîne l'initialisation du système pour des pièces de type-L, représentée par la place p_4. Ensuite, au bout de 30 unités de temps, t_5 est tirée, ce qui correspond à la fin de la phase d'initialisation pour les pièces de type-L. Un jeton en p_6 montre que l'initialisation pour les pièces de type-L est terminée et que le système est prêt pour leur traitement. La transition t_{11} est alors activée, ce qui marque le début du traitement des pièces de type-L. Une fois que les 3000 pièces ont été traitées, t_7 est tirée et le système se remet en état de disponibilité. Pour le lot de type-R, le processus est similaire, les quelques différences explicitées plus haut mises à part.
Une vidéo montrant l'évolution des places marquées et du nombre de jetons présents dans chaque place est disponible au lien suivant : `https://www.youtube.com/watch?v=RZ8yhZum_Pw&feature=youtu.be`.
Dès le lancement de la simulation, les places p_1, p_2 et p_3 sont marquées ; cela représente un lot de type-L et un lot de type-R en attente de traitement, et la disponibilité du système. Ensuite, le traitement du lot de type-L commence, avec la division du lot en 3000 pièces, placées en p_8. Les pièces sont ensuite mises, une par une, en attente de traitement, dans la place p_{10}. Une fois que 50 pièces sont en attente, t_3 est tirée et un jeton est placé en p_4 (à 0min04s sur la vidéo). 30 unités de temps après le marquage de p_4, t_5 est tirée, et un jeton est placé dans p_6, entraînant le tir de t_{11}. Les pièces de type-L sont en train d'être traitées. Après 100 unités de temps, t_2 est tirée (à 0min06s), symbolisant la décomposition du lot de type-R en 2000 pièces, placées dans p_9. Les pièces sont mises en attente une par une et placées dans p_{11}. Le système n'étant pas disponible, les pièces de type-R ne sont pas traitées et restent en attente. A 2min15s, toutes les pièces de type-R sont en attente de la disponibilité du système, qui a encore 900 pièces de type-L à traiter. A 3min12s, toutes les pièces de type-L ont été traitées. t_7 est tirée, et le système est de nouveau disponible. t_4 est tirée immédiatement et un jeton est placé en p_5, correspondant à l'initialisation des pièces de type-R. Après 36 unités de temps, t_6 est tirée et le traitement des pièces de type-R commence, représenté par les tirs successifs de la transition t_{12}. Une fois les pièces de type-R traitées, t_8 est tirée et le système est de nouveau disponible. La simulation s'arrête, après 6000 unités de temps.

2.4.3 Application à un système de trois cuves

2.4.3.1 Description du système

Le système hybride considéré est un système de trois cuves d'eau communiquant entre elles via un système de vannes, tiré de la thèse de Q. Gaudel (Gaudel 2016), qui a été adapté pour le formalisme des HtPN. Le système est représenté sur la figure 2.7. Ce système est étudié pour faire de la gestion de santé. Des dynamiques de dégradation et d'événements de fautes pouvant perturber le fonctionnement nominal du système sont donc représentés dans le modèle.

Les cuves d'eau sont placées en série. Le flot d'arrivée d'eau $Q_1(k)$ délivré par la pompe dans la cuve T_1 est supposé constant. La cuve T_2 se vide avec un flot $Q_{20}(k)$. Les niveaux d'eau h_i de chaque cuve T_i sont mesurés et disponibles à chaque date k donnée. Les vannes v_{13} et v_{32} permettent la circulation de l'eau entre les différentes cuves. Dans notre cas, seule la vanne v_{13} est contrôlée via des événements discrets de contrôle $open_{v_{13}}$ et $close_{v_{13}}$. Six fautes, considérées comme des événements non observables, peuvent apparaître dans le système : f_1, f_2 et f_3 représentent une fuite pour les cuves T_1, T_2 et T_3, respectivement. Les fautes f_4 et f_5 représentent le fait que les valves v_{13} et v_{32} se bloquent en position fermée. Enfin, la faute f_0 correspond à un niveau d'eau l_2 dans la cuve numéro 2 inférieur au seuil l_{2min} et entraîne la défaillance du système. Afin de simplifier les illustrations, seuls les événements associés à la valve v_{13} et les fautes f_1, f_4 et f_0 seront considérés dans cet exemple. Le but du système est de maintenir le niveau d'eau dans la cuve T_2 au-dessus d'une valeur minimale $h_{2_{min}}$. La fuite dans la cuve T_1 est connue et de trop grande amplitude pour que le système arrive à maintenir le niveau d'eau désiré.

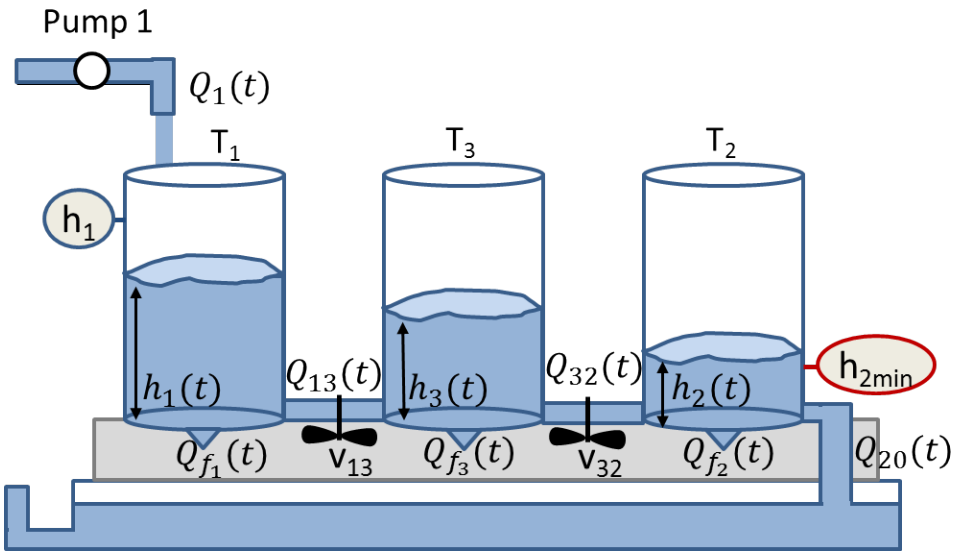


Figure 2.7 – Représentation du système des trois cuves

Dans le cadre de la simulation, seules les actions effectuées sur la vanne entre la première cuve et la troisième cuve sont observées : $close_{v_{13}}$ et $open_{v_{13}}$. Les fautes considérées sont : f_1, l'apparition d'une fuite sur la cuve T_1, f_4, le blocage de la vanne en position fermée, et f_0, la faute indiquant l'échec du maintien du niveau d'eau (et menant donc à un mode défaillant).

2.4.3.2 Modélisation en HtPN

Afin de modéliser le système considéré, la méthode de modélisation proposée dans la section 2.4.1 va être appliquée. Les niveaux d'eau dans chacune des cuves sont représentés dans le vecteur d'état continu. La probabilité d'occurrence de chaque faute est représentée par le vecteur d'état de dégradation. Les dynamiques continues et de dégradation associées à chacune des places identifiées sont disponibles sur internet [3].

Étape 1 : identification des modes de fonctionnement nominaux et des dynamiques d'évolution On distingue deux modes nominaux : $Nom1$, pour lequel les deux vannes sont ouvertes et les cuves ont un niveau de stress faible (le niveau de stress, élevé, moyen ou faible, est un niveau de sollicitation entraînant une dégradation plus ou moins rapide du composant concerné), et le mode $Nom2$, pour lequel la vanne v_{13} est fermée et la cuve T_1 subit un niveau de stress élevé. Les dynamiques continues associées au mode $Nom1$ sont représentées par C_1, et le niveau de stress des cuves par les dynamiques de dégradation D_1. Les dynamiques continues associées à Nom_2 sont représentées par C_2. Le niveau de stress des cuves est représenté par les dynamiques de dégradation D_2.

Le système évolue de $Nom1$ à $Nom2$ sur l'occurrence de l'événement $close_{v_{13}}$ et de $Nom2$ à $Nom1$ sur l'occurrence de l'événement $open_{v_{13}}$. L'évolution de $Nom1$ à $Nom2$ se traduit donc par les ensembles de conditions et d'assignations suivants :

— $Pre(Nom1, t_1) = \{(close_{v_{13}}, -, -); 1\}$ pour représenter l'occurrence de l'événement $close_{v_{13}}$,
— $Post(t_1, Nom2) = \{(-, -, -); 1\}$ pour représenter l'évolution du système en $Nom2$, sans assignation.

Étape 2 : identification des modes de fonctionnement dégradé Depuis $Nom1$ (resp. $Nom2$) le système évolue dans le mode dégradé $Deg2$ (resp. $Deg3$) sur l'occurrence de la faute f_1, ou dans le mode dégradé $Deg1$ sur l'occurrence de la faute f_4. Ces modes ont leurs dynamiques continues et de dégradation propres. Les dynamiques continues associées au mode dégradé $Deg2$ sont représentées par C_3, celles associées au mode $Deg3$ sont représentées par C_4. Les deux modes dégradés $Deg2$ et $Deg3$ ont les mêmes dynamiques de dégradation, représentées par D_4, où les niveaux de stress des cuves T_2 et T_3 sont faibles et où la partie du vecteur d'état de dégradation représentant la probabilité d'occurrence de f_1 n'évolue plus et n'est plus évalué, f_1 ayant déjà eu lieu. Les dynamiques continues associées au mode dégradé $Deg1$ sont représentées par C_2. Les dynamiques de dégradation associées au mode $Deg1$ sont représentées par D_3 et correspondent à un niveau de stress élevé pour T_1 et faible pour T_2 et T_3. La différence avec D_2 est que la partie du vecteur d'état de dégradation représentant la probabilité d'occurrence de la faute f_4 n'évolue plus, vu que f_4 a déjà eu lieu. Depuis $Deg2$ et $Deg3$, la vanne v_{13} peut se bloquer en position fermée (faute f_4). Ceci est représenté par le passage du système dans le mode dégradé $Deg4$, ayant les mêmes dynamiques continues C_4 que $Deg3$ et des dynamiques de dégradation représentées par D_5 dans lesquelles les parties du vecteur d'état de dégradation correspondant aux probabilités d'occurrence des fautes f_1 et f_4 ne sont plus évaluées, étant donné leurs occurrences passées.

3. https ://homepages.laas.fr/echanthe/hymu/water_tanks

L'évolution du système de $Nom1$ à $Deg2$ se traduit par les ensembles de conditions et d'assignations suivants :

— $Pre(Nom1, t_2) = \{(-, -, (\gamma_k[0] > seuil_{f1})); 1\}$ pour représenter le fait que la première valeur du vecteur γ_k dépasse le seuil fixé pour la faute f_1,
— $Post(t_2, Deg2) = \{(-, -, -); 1\}$ pour représenter l'évolution du système en $Deg2$, sans assignation.

Étape 3 : compléments de modélisation Depuis les modes dégradés identifiés, le système peut entrer dans un mode de défaillance ($FailX$) sur l'occurrence de la faute f_0, qui correspond au moment où le niveau d'eau est passé sous la valeur seuil $h_{2_{min}}$. L'évolution du système de $Deg2$ à $Fail2$, par exemple, va se traduire par les ensembles de conditions et d'assignations suivants :

— $Pre(Deg2, t_3) = \{(-, x_k[1] < h_{2_{min}}, -); 1\}$ pour représenter le fait que le niveau d'eau dans la seconde cuve (représenté par $x_k[1]$ au temps k) passe sous la valeur $h_{2_{min}}$ demandée,
— $Post(t_3, Fail2) = \{(-, -, -); 1\}$ pour représenter l'évolution du système dans $Fail2$, sans assignation.

Pour récapituler, 10 places et 14 transitions ont été définies : 2 places correspondant aux modes nominaux $Nom1$ et $Nom2$ (étape 1 du processus de création), 4 places correspondant aux modes dégradés, $Deg1$, $Deg2$, $Deg3$ et $Deg4$ (étape 2 du processus de création) et 4 places en complément de modélisation, $Fail1$, $Fail2$, $Fail3$, $Fail4$, qui ne correspondent pas à un mode de fonctionnement du système (elles ont des dynamiques identiques aux dynamiques des places $DegX$ en amont) et sont là pour représenter le fait que le système n'a pas réussi à maintenir le niveau d'eau au-dessus du seuil requis.

La représentation multimode du système considéré est visible sur la figure 2.8. Les différents modes de fonctionnement, nominaux en vert, dégradés en orange et de défaillance en rouge, sont visibles sur cette représentation. Pour chaque mode, représenté par une place $p_i \in P$ dans le HtPN, les dynamiques continues C_i et de dégradation D_i associées sont déterminées. Les conditions d'évolution d'un mode à l'autre sont explicitées sur les arcs. L'absence de chiffre sur les arcs implique que tous les poids des arcs ont la valeur de 1 par défaut dans le HtPN. Par souci de lisibilité, seule la représentation HtPN des modes $Nom1$ et $Nom2$ est illustrée sur la figure 2.9.

2.4.3.3 Résultats de simulation

Le scénario de simulation choisi est le suivant : le flux d'eau en entrée est constant et fixé à $3.5 \times 10^{-5} m^3/s$. 3 400 observations sont enregistrées toutes les 60s. Au bout de 310 minutes (18 600 s), la vanne v_{13} est fermée toutes les heures, puis elle est ré-ouverte 20 minutes après. A $t = 191760s$, la cuve T_1 se met à fuir (occurrence de f_1), puis le niveau d'eau h_2 passe sous la valeur seuil, $h_{2_{min}}$ à $t = 195420s$.

Les résultats de simulation obtenus sont représentés sur la figure 2.10. On observe bien les passages répétés entre les modes nominaux $Nom1$ et $Nom2$, en fonction de l'occurrence des événements $close_{v_{13}}$ et $open_{v_{13}}$. Après environ 190000s, on observe bien qu'une fuite

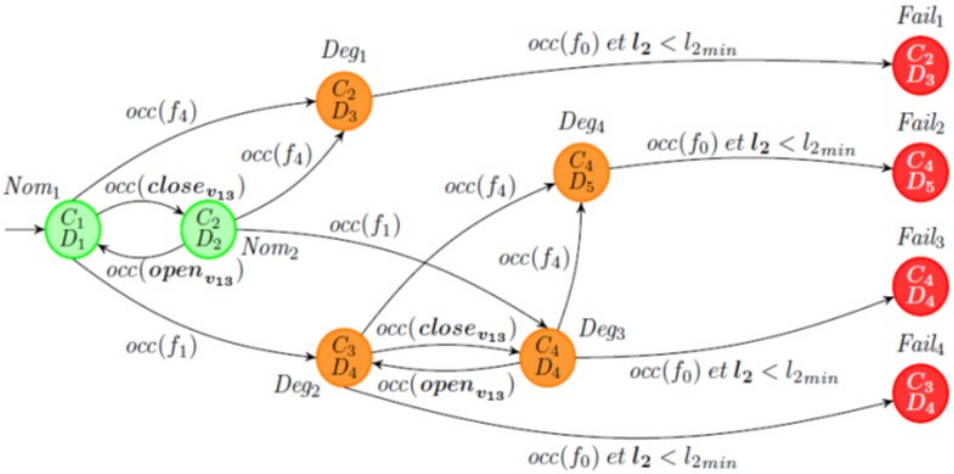


FIGURE 2.8 – Représentation multimode du système des trois cuves

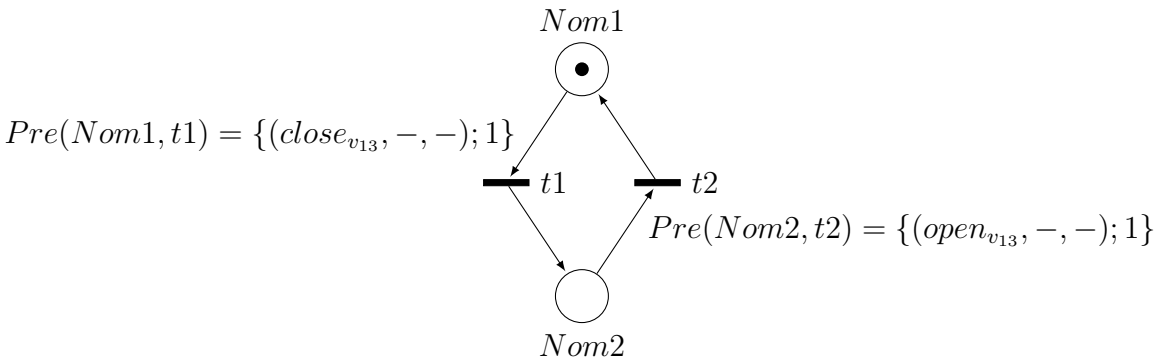


FIGURE 2.9 – Représentation HtPN des modes $Nom1$ et $Nom2$

est apparue dans la cuve T_1, ce qui correspond à l'occurrence de la faute f_1. Le système entre donc dans le mode dégradé $Deg2$. Il alterne ensuite entre les modes dégradés $Deg2$ et $Deg3$ avec les occurrences des événements $open_{v_{13}}$ et $closev_{13}$. Peu de temps après, la fuite entraîne la défaillance du système visible par son entrée dans la place associée au mode $Fail3$.

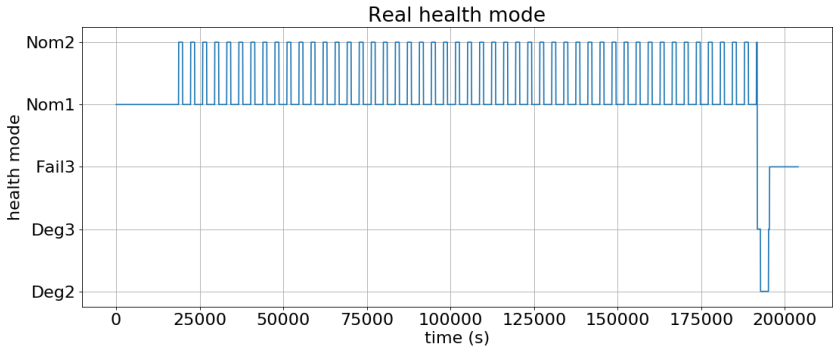


FIGURE 2.10 – Simulation du système des trois cuves

La figure 2.11 illustre l'évolution en fonction du temps du vecteur de dégradation pour les fautes f_1 et f_4 considérées. La dégradation de la faute f_4 a été multipliée par 100 afin de pouvoir l'illustrer convenablement. Une fois le seuil de 0.90 franchi, on considère que

la faute a eu lieu (c'est le cas pour f_1 à environ $190000s$).

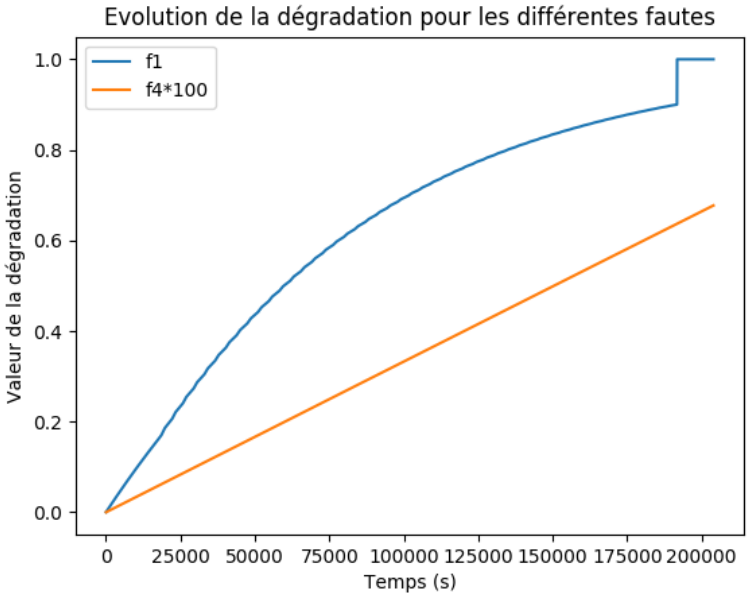


FIGURE 2.11 – Evolution de la dégradation pour les cuves d'eau

À travers ces deux exemples, l'un axé contrôle et l'autre axé gestion de santé, nous pouvons conclure que le formalisme des HtPN peut être utilisé pour modéliser et simuler divers types de système.

2.5 Conclusion

2.5.1 Résumé du chapitre

Dans ce chapitre, le formalisme des HtPN a été présenté. Ce formalisme a été développé dans le but de modéliser et simuler les mêmes informations que les réseaux de Petri classiques et est une extension des réseaux de Petri classiques. Ce formalisme est capable de représenter n'importe quel type de système, qu'il soit purement discret, purement continu ou hybride. Il peut notamment modéliser et simuler des systèmes hybrides composés de sous-parties ayant des dynamiques différentes. Il permet également de représenter et suivre des incertitudes sur le système, que ce soit au niveau du modèle ou des observations, à travers les fonctions de bruit présentes sur les dynamiques continues et de dégradation. Les HtPN permettent enfin de représenter l'état de santé des systèmes, notamment leur vieillissement, par sa capacité à spécifier des fonctions de dégradation.

Une démarche de modélisation en HtPN a été proposée, et deux exemples ont permis d'illustrer ce formalisme. Le premier exemple, un système de production de Motorola issu de la littérature, est axé contrôle. Le deuxième exemple est un système hybride composé de trois cuves d'eau communicantes. Ces deux exemples ont été modélisés et simulés en utilisant le logiciel HeMU, développé dans ce but.

2.5.2 Limitations et discussions

Le formalisme tel que défini actuellement ne peut inclure l'aspect stochastique de certains formalismes ou encore l'incertitude temporelle telle qu'elle est définie dans Berthomieu et al. 1983. En effet, une transition validée est tirée dès sa validation. L'introduction de probabilités de tirage ou de bornes de tirs admissibles pour les transitions paraît donc être une perspective intéressante pour de futurs travaux.

En cas de non disponibilité d'un expert pour définir le modèle du système étudié, une méthode d'apprentissage appliquée au formalisme des HtPN pour obtenir un modèle est présentée dans le chapitre suivant.

Chapitre 3

Obtention d'un modèle par apprentissage

Sommaire

3.1 Introduction 48
3.2 État de l'art sur les méthodes d'apprentissage de modèles . . 48
3.2.1 Apprentissage d'un modèle à événements discrets 49
3.2.2 Apprentissage d'un modèle continu 53
3.2.3 Apprentissage d'un modèle hybride 55
3.2.4 Synthèse 57
3.3 Processus d'apprentissage d'un HtPN 58
3.3.1 Processus d'apprentissage : vue d'ensemble 58
3.3.2 Données d'entrée 59
3.3.3 Prétraitement 59
3.3.4 Apprentissage structurel 61
3.3.5 Apprentissage des dynamiques continues 64
3.3.6 Déclenchement de l'apprentissage pour améliorer un modèle . . 65
3.4 Application de la méthode d'apprentissage 69
3.4.1 Description du système 69
3.4.2 Données d'entrée 70
3.4.3 Prétraitement des données 71
3.4.4 Apprentissage structurel 71
3.4.5 Apprentissage des dynamiques continues 73
3.5 Conclusion 77
3.5.1 Résumé du chapitre 77
3.5.2 Limitations et discussion 78

3.1 Introduction

Quand on parle de méthodes de gestion de santé, il est possible de distinguer deux approches particulières : les méthodes basées modèles et les méthodes basées données. Les méthodes basées modèles sont très efficaces si le modèle du système est précis et correct. Néanmoins, si ce n'est pas le cas, les méthodes basées modèles perdent en efficacité. Les méthodes basées données, quant à elles, sont efficaces quand le modèle n'est pas disponible, mais elles manquent d'explicabilité et ne fournissent pas vraiment de connaissances sur le système. De plus, elles présupposent la possession d'un grand nombre de données sur le système. L'idée développée dans ce chapitre est de combiner ces deux types de méthodes en proposant une méthode d'apprentissage de modèle basée sur les données, le modèle appris pouvant ensuite être utilisé par une méthode de suivi de santé basé modèles qui sera exposée dans le chapitre 4.

La thématique de ce chapitre est donc l'apprentissage de modèles. La procédure présentée pour l'apprentissage est appliquée aux HtPN. Apprendre un modèle sous ce formalisme permettra d'appliquer la méthode de suivi de santé qui sera définie dans le prochain chapitre. Cependant, ce processus d'apprentissage peut tout aussi bien s'appliquer à d'autres formalismes, comme les automates hybrides ou des réseaux de Petri mixtes, par exemple, qui ont été présentés dans le chapitre 2. En effet, tout type de formalisme comprenant une structure discrète et/ou des dynamiques continues est apprenable, c'est-à-dire identifiable à partir des observations par la méthode d'apprentissage proposée dans ce chapitre.

Le processus d'apprentissage est organisé en différentes étapes. La première étape consiste en l'obtention de la structure discrète du réseau, représentée par des places et des transitions, en utilisant une méthode de clustering. Cette étape sera appelée l'*apprentissage structurel*. Ensuite, pour chaque place obtenue lors de l'étape précédente, des algorithmes de régression, comme la régression basée machines à vecteurs supports, *Support Vector Regression* (SVR), ou la régression basée sur des forêts d'arbres aléatoires, *Random Forest Regression* (RFR), ou une régression polynomiale (RP), sont appliqués afin d'apprendre les dynamiques continues associées à la place considérée. Ces algorithmes ont été choisis pour leur bonne capacité de généralisation, c'est-à-dire leur capacité à prédire la valeur d'une donnée non présente dans l'ensemble d'entraînement mais comprise entre le minimum et le maximum.

Ce chapitre est organisé de la manière suivante. La section 3.2 donne un état de l'art des méthodes existantes pour apprendre différents types de modèles (hybrides, discrets ou continus). On en dégagera les méthodes les plus pertinentes pour l'apprentissage de modèles. La section 3.3 présente la méthodologie proposée pour obtenir un modèle sous le formalisme des HtPN. Ensuite, la section 3.4 illustre la méthodologie en l'appliquant sur l'exemple académique des trois cuves d'eau. Enfin, la section 3.5 présente des conclusions et perspectives pour l'apprentissage de modèles.

3.2 État de l'art sur les méthodes d'apprentissage de modèles

La section qui suit présente des méthodes d'apprentissage de modèles pour différents types de systèmes. La première partie se concentre sur les systèmes à événements discrets. La deuxième se focalise sur l'apprentissage de dynamiques continues. Finalement,

la troisième concerne les méthodes d'apprentissage de modèles pour les systèmes hybrides.

De manière à guider cet état de l'art et choisir avec pertinence les méthodes utilisables dans notre approche, les besoins identifiés sur les méthodes d'apprentissage de modèles sont listés ci-après :

— On souhaite apprendre une structure discrète (places, transitions) sur laquelle on ne présume pas du nombre de places et de transitions *a priori*.
— Les échantillons/données d'entrée sont hétérogènes : il peut s'agir d'événements discrets, de valeurs booléennes, de valeurs continues ou même de données qualitatives.
— Les échantillons/données d'entrée sont entachées d'incertitudes, certaines observations pouvant être erronées sur un certain nombre de tics d'horloge ; les méthodes d'apprentissage devront donc être capables d'écarter les outliers et être robustes aux bruits de mesure (tant au niveau discret que continu).
— On ne dispose d'aucune hypothèse sur les dynamiques continues.
— Les dynamiques continues apprises doivent pouvoir se généraliser (interpolation, extrapolation) avec une bonne précision de prédiction.
— Idéalement, le nombre d'hyper-paramètres à choisir doit être faible et les hyper-paramètres doivent être faciles à fixer sans connaissance experte sur le système.

3.2.1 Apprentissage d'un modèle à événements discrets

L'objectif de cette sous-section est de donner un aperçu de différentes méthodes existantes permettant d'apprendre un modèle pour des systèmes à événements discrets.

3.2.1.1 Méthodes basées événements

Pour apprendre la structure d'un SED, des méthodes basées événements sont utilisées, c.-à-d. des méthodes dont les raisonnements sont directement basés sur les événements observés dans le système.

Par exemple, Dotoli et al. 2008 présente une méthode d'apprentissage d'un réseau de Petri. En se basant sur l'observation des événements et les sorties disponibles qui constituent les échantillons d'entrée, un réseau de Petri est obtenu via un problème de programmation linéaire ou un algorithme d'identification, si les ensembles de places et de transitions sont connus en amont ou non. Si ces ensembles sont connus, un problème d'identification linéaire est utilisé pour déterminer le réseau de Petri modélisant le système. Ce problème de programmation linéaire est défini à chaque échantillon pour obtenir le réseau de Petri de manière récursive. Dans le cas où les ensembles de places et de transitions ne sont pas connus (une borne haute du nombre de places est néanmoins nécessaire), le problème d'identification est résolu par un algorithme d'identification qui observe en temps réel les événements ayant eu lieu et les vecteurs de sortie correspondants. Ce travail est intéressant, mais pose la limitation de la connaissance soit des ensembles de places et de transitions, dont on ne dispose pas, soit d'une borne haute sur le nombre de places, ce qui est limitant pour l'apprentissage des systèmes que nous considérons.

Dans Moreira et al. 2019, un automate déterministe contenant les sorties et les conditions sur les transitions est obtenu en se basant sur des chemins non-fautifs observés. L'automate obtenu représente donc uniquement le comportement non fautif du système.

Afin d'obtenir un modèle plus compact, des boucles sont ajoutées au modèle appris. Ceci permet au modèle appris de générer des séquences non observées et ainsi d'augmenter le langage de l'automate. La limitation de l'approche à des chemins non-fautifs ne correspond pas à notre critère selon lequel les observations peuvent être incertaines.

Une autre méthode pour apprendre des réseaux de Petri est d'utiliser des réseaux de neurones. De manière très simplifiée, un réseau de neurones est un estimateur composé d'au moins trois couches de nœuds (ou neurones) : une couche d'entrée, une (ou plusieurs) couche(s) cachée(s) et une couche de sortie qui communiquent entre elles. Chaque nœud, ou neurone artificiel, se connecte à un autre et possède un poids, sur les connexions, et un seuil associés. Si la sortie d'un nœud est supérieure à la valeur de seuil spécifiée, ce nœud est activé et envoie des données à la couche suivante du réseau. Sinon, aucune donnée n'est transmise à la couche suivante du réseau. L'utilisation des réseaux de neurones est variée, ils peuvent tout aussi bien servir à faire de la classification que de la régression. L'utilisation des réseaux de neurones pour apprendre un réseau de Petri est proposée dans Leclercq et al. 2008. L'idée est de considérer un réseau de Petri comme un réseau de neurones multicouches et d'y appliquer un algorithme de propagation arrière de l'erreur pour identifier les paramètres. La structure des réseaux de neurones considérée est donnée sur la figure 3.1, tirée de l'article. La couche cachée correspondent soit à l'ensemble T des transitions, soit à l'ensemble P des places, selon le jeu de données. Les couches d'entrée et de sortie correspondent à l'autre ensemble. Les matrices de poids Q et V correspondent aux matrices Pre ou $Post$, selon les ensembles correspondant aux différentes couches du réseau de neurones. Les ensembles E et S correspondent aux entrées et aux sorties du réseau. Pour chaque donnée d'entrée e correspondant à un événement, le processus va comparer les sorties observées s, correspondant à un état du réseau, et les sorties estimées du réseau et va modifier les ensembles Q et V en fonction de l'erreur entre les deux.

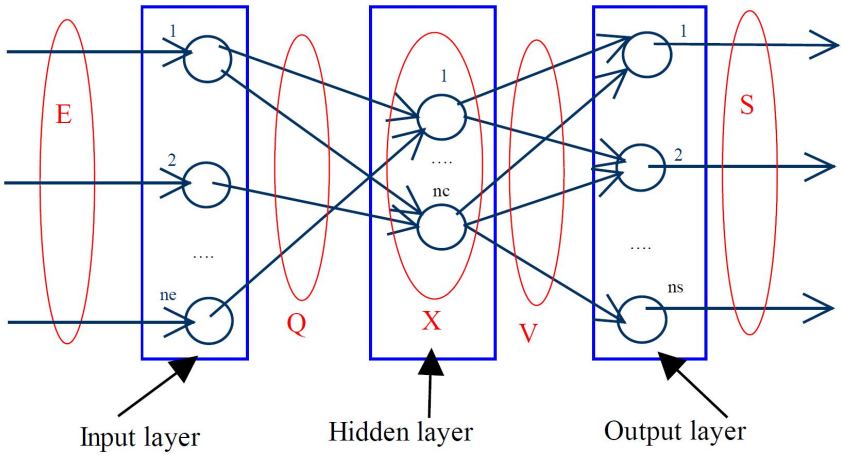


Figure 3.1 – Réseau de neurones correspondant à un réseau de Petri (Leclercq et al. 2008)

Cette idée est intéressante et pourrait s'étendre à des réseaux de Petri présentant des dynamiques continues. Seulement, les réseaux de neurones sont gourmands en termes d'hyper-paramètres et peu explicables. Bien que ça ne soit pas un critère strict, l'explicabilité est un concept que nous aimerions garder.

Ces méthodes d'apprentissage de modèles ne s'appliquent qu'à des systèmes à événements discrets, où tous les échantillons sont purement discrets, c.-à-d. prenant un nombre

fini de valeurs. Les dynamiques continues, qui peuvent amener un changement dans l'état des systèmes hybrides par exemple, ne sont pas prises en compte.

3.2.1.2 Méthodes basées clustering

Une méthode de clustering permet de regrouper les échantillons d'entrées similaires en groupes, appelés clusters. La similarité des échantillons est obtenue en calculant une mesure (souvent une distance ou une norme) entre les échantillons à partir de leur ensemble de caractéristiques. Ceci permet donc d'inclure des caractéristiques quantitatives continues, en plus de caractéristiques booléennes. Des caractéristiques qualitatives peuvent également être prises en compte dans certains algorithmes de clustering.

Les méthodes de clustering permettent de tenir compte des variations de dynamiques continues et des occurrences d'événements discrets dans la création de la structure d'un modèle à événements discrets.

Comme indiqué dans l'introduction, toutes les méthodes de clustering ne nous paraissent pas pour autant intéressantes. Les méthodes de clustering qui nécessitent en hyper-paramètre d'entrée le nombre de clusters, comme c'est le cas pour les méthodes *K-means* (Krishna et al. 1999) ou *BIRCH* (T. Zhang et al. 1996) par exemple, ne nous intéressent pas. Dans d'autres méthodes de clustering, plus intéressantes pour nous, comme *Denstream* (Cao et al. 2006), *DBSCAN* (Ester et al. 1996) ou encore *OPTICS* (Ankerst et al. 1999), la définition du nombre de clusters est indirecte via la taille des clusters. Définir le nombre de clusters via la taille peut entraîner une amélioration de la connaissance du système, ainsi qu'une meilleure explicabilité, contrairement au fait de fixer le nombre de clusters directement.

C'est aussi le cas de l'algorithme intitulé "DyClee", pour *Dynamic Clustering for tracking evolving environments*, développé précédemment dans l'équipe DISCO du LAAS-CNRS et présenté dans Roa et al. 2019. Cet algorithme est adapté pour suivre des systèmes qui évoluent avec du clustering dynamique.

L'algorithme de clustering dynamique se déroule en deux étapes, une première étape basée sur la distance et une seconde étape sur la densité. La figure 3.2 montre le principe de l'algorithme sous forme de schéma bloc.

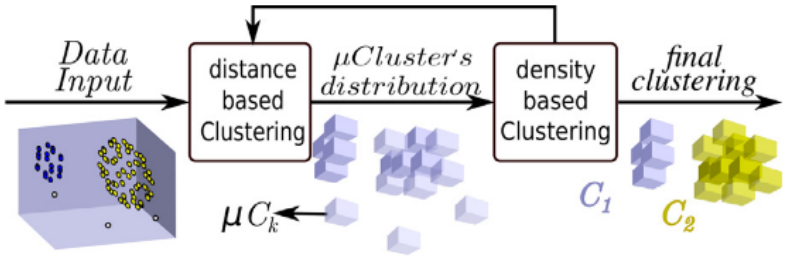


Figure 3.2 – Principe de fonctionnement de DyClee (Roa et al. 2019)

Dans la première étape basée sur la distance, chaque échantillon va être assigné à un micro-cluster, correspondant à un hypercube centré sur un échantillon, selon la taille du micro-cluster définie et la norme L_1. Si la norme L_1 de la différence des caractéristiques entre l'échantillon et le centre d'un micro-cluster est inférieure à la taille des micro-clusters

donnée, alors l'échantillon est associé au micro-cluster. Sinon, un nouveau micro-cluster est créé et centré sur l'échantillon. Pour le premier échantillon, étant donné qu'aucun micro-cluster n'existe, un micro-cluster sera créé et centré sur le premier échantillon. Cette première étape s'exécute à la fréquence d'arrivée des données. Lors de la deuxième étape appelée "étape de densité", ayant lieu à une fréquence moindre par rapport à la première, la densité des micro-clusters est observée et catégorisée en "basse", "moyenne" ou "haute". Les clusters qui vont être fournis par DyClee sont basés sur la densité des micro-clusters. Les clusters sont composés de micro-clusters de densité "haute" à l'intérieur et "haute" ou "moyenne" pour le contour. Les micro-clusters ayant une densité "basse" peuvent être considérés comme des outliers.

Un exemple de la deuxième étape, basée sur la densité, est visible sur la figure 3.3, elle aussi tirée de l'article (Roa et al. 2019). Dans cette figure, de nombreux carrés sont présents. Chaque carré est un micro-cluster. Plus le micro-cluster est dense, plus la couleur du carré va tendre vers le noir. La première étape de la figure est en haut à gauche, c'est le début de l'étape de densité. Cette étape va sélectionner l'ensemble des micro-clusters tendant le plus vers le noir (densité haute) pour définir l'intérieur d'un cluster. Les bords seront composés de micro-clusters noirs (densité "haute") ou gris foncé (densité "moyenne"). Dans la figure, deux clusters sont obtenus : un rouge et un bleu. Le premier cluster, le rouge, est identifié et créé (en haut à droite). Les micro-clusters ayant été affectés au cluster rouge sont enlevés du processus pour ne pas être assignés à deux clusters différents (en bas à gauche). Enfin, le deuxième cluster est identifié, le bleu, et créé (en bas à droite).

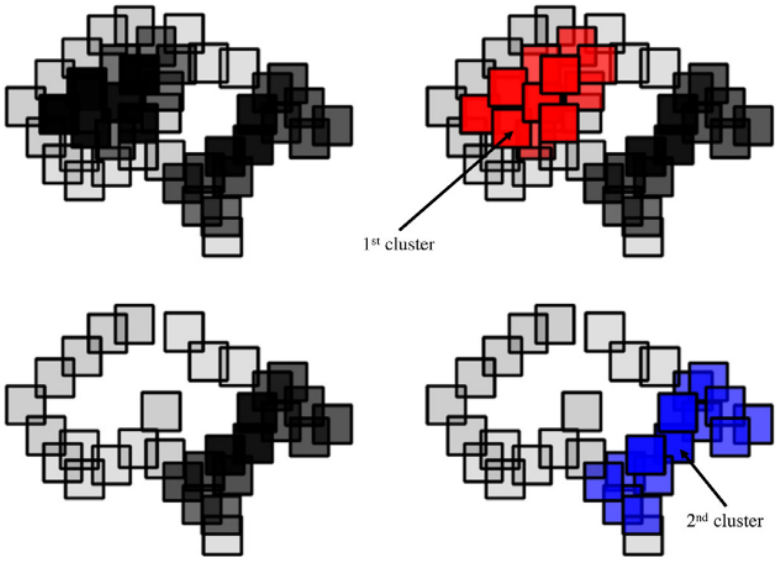


Figure 3.3 – Exemple de fonctionnement de la deuxième étape de DyClee (Roa et al. 2019)

L'obtention d'un automate temporisé a été proposée dans Barbosa et al. 2017 à partir des résultats de DyClee, en effectuant un parallèle entre les clusters obtenus et les états de l'automate. En se basant sur les résultats intéressants obtenus dans ces travaux et la disponibilité de personnes travaillant sur l'implémentation de DyClee, nous avons choisi de l'utiliser pour obtenir la structure discrète de notre modèle. Du point de vue de la structure, DyClee ne présume pas du nombre de places ou de transitions, est capable

de prendre en compte des incertitudes et d'écarter des outliers et a un nombre d'hyper-paramètres faible (la taille des micro-clusters). De plus, l'origine des clusters est explicable, ce qui est un ajout intéressant.

3.2.2 Apprentissage d'un modèle continu

Différentes méthodes d'obtention de modèles continus proposées dans la littérature sont présentées dans cette section.

3.2.2.1 Descente de gradient

Pour apprendre des dynamiques continues, la méthode de la descente de gradient (Ruder 2016) est souvent utilisée afin d'obtenir les paramètres d'une fonction correspondant à un ensemble de données. Le but de cette méthode est de trouver le minimum d'une fonction réelle différentiable de manière itérative. L'idée est de partir d'un point initial aléatoire, de calculer la valeur du gradient de la fonction en ce point, puis de progresser dans la direction opposée au gradient (maximum de pente) afin de converger vers un minimum de la fonction.

La descente de gradient est notamment utilisée dans le travail de Tamssaouet et al. 2020, basé sur un modèle entrée-sortie qui mesure la distance entre l'état courant du système et l'état de défaillance pour effectuer un pronostic du système. Dans ce travail, la descente de gradient est utilisée pour obtenir les paramètres du modèle en comparant la valeur estimée par le modèle et la valeur mesurée. Cependant, la descente de gradient a un problème majeur : pour pouvoir régler les paramètres d'une fonction, la forme de la fonction (comme l'ordre d'un polynôme, la présence d'une partie exponentielle, etc) doit être connue, ce qui n'est pas forcément notre cas. La méthode de descente du gradient n'est donc pas adaptée à notre cas.

3.2.2.2 Réseaux de neurones

Les réseaux de neurones, dont l'idée a été présentée en section 3.2.1.1, peuvent aussi être utilisés pour estimer la sortie d'une fonction complexe ou inconnue. C'est par exemple le cas dans le travail de Djedidi et al. 2020 où un modèle *Nonlinear Autoregressive Network with eXogenous inputs* (NARX) qui mesure la consommation énergétique d'un téléphone est appris via un réseau de neurones. Les réseaux de neurones nécessitent cependant un réglage précis d'hyper-paramètres tels que la forme du réseau (nombre de couches) et sa taille (nombre de nœuds par couches), ce qui rend complexe une généralisation et une automatisation du processus d'apprentissage.

3.2.2.3 Régression basée machines à vecteurs supports (SVR)

La méthode SVR détaillée dans Awad et al. 2015, est souvent utilisée pour apprendre des dynamiques continues, comme dans Drezet et al. 1998 ou L. Zhang et al. 2004, où cette méthode de régression est utilisée pour apprendre des modèles linéaires ou non-linéaires. Dans le cas linéaire, l'algorithme SVR essaie de trouver pour chaque caractéristique x_i du vecteur de caractéristiques x une fonction $w_i x_i$ telle qu'un maximum de données soit contenu dans l'intervalle $[w_i x_i - \epsilon, w_i x_i + \epsilon]$, avec w_i le coefficient du modèle

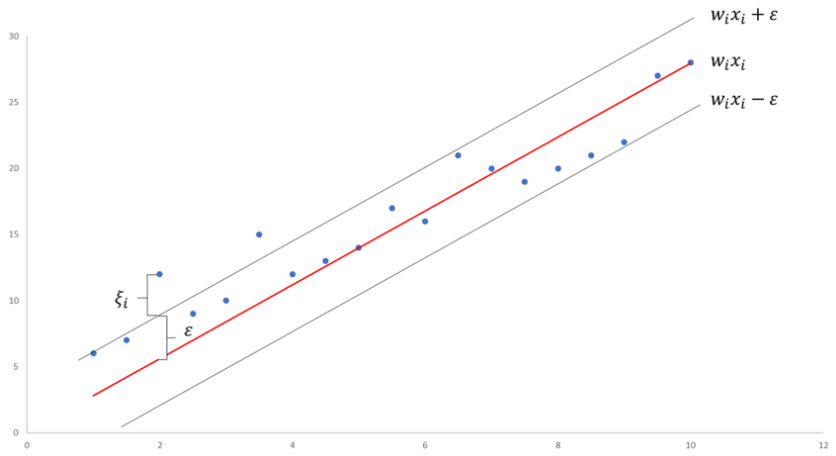


FIGURE 3.4 – Représentation de l'idée de la méthode SVR

et $\epsilon \in \mathbb{R}$ l'erreur voulue, donnée en hyper-paramètre de la méthode d'apprentissage. Cette idée est illustrée sur la figure 3.4, sous la forme d'une droite, mais la méthode est capable d'apprendre des modèles non linéaires.

L'avantage principal de cette méthode est qu'elle est robuste aux outliers, elle se généralise facilement avec une précision de prédiction assez grande et ne nécessite pas une phase de réglage trop complexe ou précise. Par contre, cette méthode est limitée dans le cas de jeux de données importants ou bruités car elle est sensible au bruit et devient lente avec des jeux de données conséquents.

3.2.2.4 Régression basée sur des forêts d'arbres aléatoires (RFR)

La méthode RFR peut également être envisagée. Elle est utilisée dans LIAW et al. 2002, où les forêts d'arbres aléatoires permettent de classifier et de faire une régression sur les données d'entrée. C'est ce qu'on appelle une méthode d'ensemble, c.-à-d. qu'elle consiste en une combinaison de plusieurs techniques de prédiction plus simples. Ici, cela consiste à faire la moyenne des prédictions d'arbres de décision comme illustré sur la figure 3.5.

Les données d'entrée (échantillons) sont réparties de manière aléatoire parmi divers arbres de décision, qui donneront chacun une prédiction. Une moyenne est ensuite obtenue à partir de l'ensemble de ces prédictions et donnée en sortie de l'algorithme d'apprentissage.

La régression basée sur des forêts d'arbres aléatoires est un algorithme robuste face au sur-apprentissage et il est capable de généraliser les résultats obtenus. Il y a seulement deux hyper-paramètres, le nombre d'arbres considérés dans la forêt et le nombre de feuilles pour chaque arbre, et les résultats obtenus n'y sont pas si sensibles. Enfin, ce type de régression n'a pas besoin d'idée préliminaire sur la forme de la fonction que l'on essaie d'apprendre.

Les méthodes proposées plus haut, la RFR, la SVR ainsi que la régression basée réseaux de neurones présentent un problème commun : l'incapacité d'extrapoler. Ceci limite l'usage en temps réel, étant donné qu'un modèle appris sur des bornes données ne sera pas capable de prédire en dehors de ces bornes. C'est pour celà qu'une troisième méthode a été étudiée : la régression polynomiale.

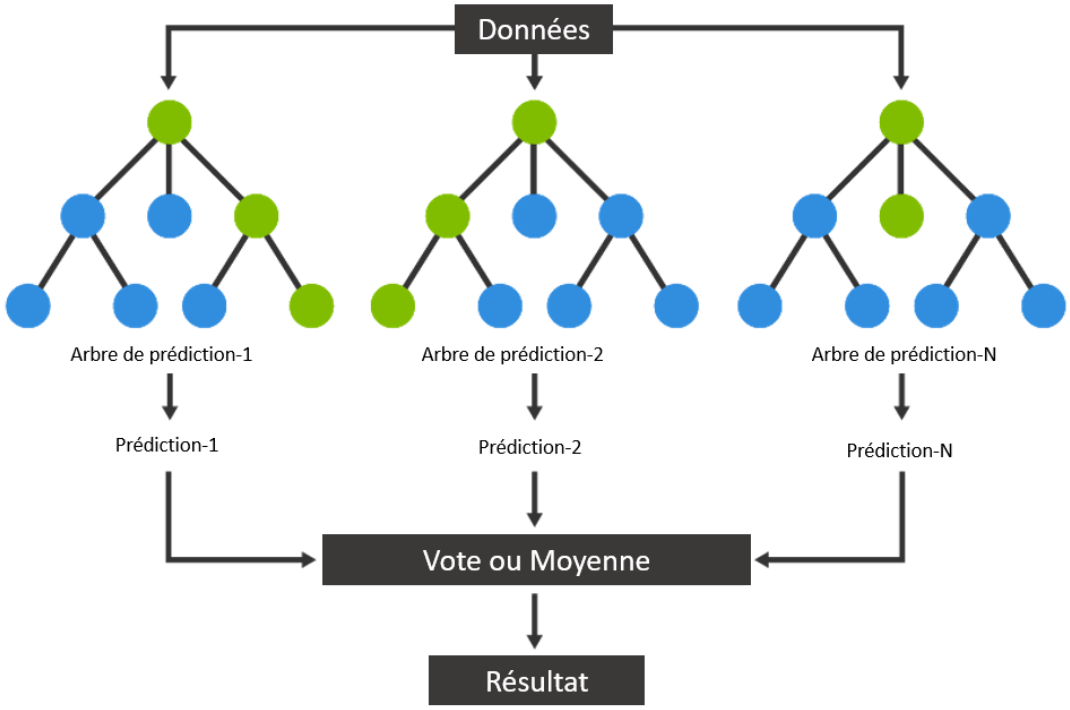


FIGURE 3.5 – Représentation de l'idée de la méthode RFR

3.2.2.5 Régression polynomiale (RP)

La méthode RP est une extension de la régression linéaire classique et permet d'obtenir des modèles non linéaires pour représenter des dynamiques continues. L'idée est d'obtenir une fonction

$$y = \beta_0 + \beta_1.x + \beta_2.x^2 + \ldots \beta_k.x^k \tag{3.1}$$

avec k le degré du polynôme recherché. Ce degré est le seul hyper-paramètre à fournir à l'algorithme de régression polynomiale. Le choix de cet hyper-paramètre n'est néanmoins pas évident, car la méthode est prompte à l'overfitting et sensible aux outliers. La figure 3.6 présente les résultats obtenus par une régression polynomiale pour un jeu de données particulier et trois différents degrés de polynôme : $3, 5$ et 9. Plus l'ordre est important, plus le polynôme obtenu va correspondre aux données, au détriment d'un risque d'overfitting.

La régression polynomiale est présentée et utilisée dans OSTERTAGOVÁ 2012 pour déterminer la relation entre différentes tractions exercées dans différentes directions lors d'un forage.

3.2.3 Apprentissage d'un modèle hybride

3.2.3.1 Fusion de méthodes

Une première solution pour apprendre des modèles hybrides est d'effectuer une fusion des méthodes appliquées aux SED, avec des méthodes utilisées pour apprendre des systèmes continus.

Cette idée a par exemple été appliquée dans NIGGEMANN et al. 2012 pour l'apprentissage d'automates hybrides. La première étape de l'apprentissage est l'apprentissage d'un

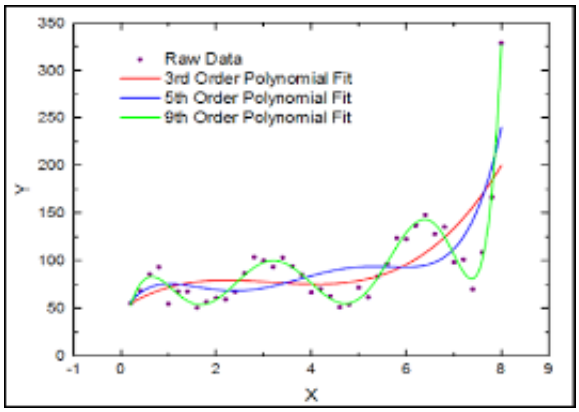


FIGURE 3.6 – Exemple de régression polynomiale

arbre de préfixes acceptable basé sur les différentes séquences d'observations. Chaque séquence d'observation va créer un chemin partant de la racine de l'arbre jusqu'à une feuille. Ensuite, pour chaque nœud de l'arbre, les dynamiques continues sont apprises par régression linéaire ou réseaux de neurones. Si les résultats obtenus par la régression linéaire sont trop imprécis, alors les réseaux de neurones sont utilisés. Dans un second temps, les nœuds ayant des dynamiques continues similaires sont fusionnés. Cette méthode, bien qu'intéressante, ne satisfait pas nos besoins. En effet, l'obtention de la structure discrète de l'automate ne s'effectue qu'à partir d'événements discrets. Or, dans les systèmes que nous souhaitons considérer, il est possible pour des dynamiques continues d'entraîner un changement de mode de fonctionnement, changement de mode que nous souhaitons faire apparaître sur la structure discrète. De plus, le modèle obtenu est sous la forme d'un automate hybride, ce qui limite son utilisation future dans notre cas, étant donné notre besoin de parallélisme.

3.2.3.2 Apprentissage direct

Il existe également des méthodes permettant d'apprendre directement le modèle hybride en une seule étape.

Dans LAUER et al. 2008, des modèles hybrides non linéaires et commutés sont appris en utilisant une combinaison des méthodes de machines à vecteurs de support, *Support Vector Machine* (SVM), et de SVR. La méthode proposée est applicable à des systèmes qui présentent des changements de dynamiques non linéaires, mais aussi à des systèmes dont le changement de mode de fonctionnement est régi par des frontières non linéaires. De plus, la méthode fait le lien entre les procédures algébriques habituellement utilisées pour apprendre ce type de systèmes et l'approche d'erreur limite par l'utilisation des SVM et SVR. Enfin, aucune supposition n'est faite quant à la séquence d'événements qui permet de générer les données utilisées, ce qui permet de traiter des simulations comme des cas proches du réel.

Une méthode d'identification de systèmes composés de sous parties affines est proposée dans PILLONETTO 2016. Cette méthode, nommée *Hybrid Stable Spline* (HSS) est basée sur le noyau "stable spline" qui permet de modéliser la réponse à une impulsion appliquée aux différents sous-modèles comme une gaussienne centrée sur zéro, qui contient une information sur la stabilité de la prédiction des sous-modèles. La méthode proposée se découpe en deux étapes. La première étape utilise une interprétation Bayésienne pour

identifier les différents sous-modèles composant le système hybride et y affecter les données correspondantes. Ensuite, l'algorithme est utilisé pour reconstruire chaque sous-système identifié. L'avantage de la méthode proposée est qu'elle ne nécessite pas d'informations sur la complexité des sous-systèmes recherchés, mais aussi qu'elle est capable d'identifier précisément des modèles plus complexes que les autres méthodes de la littérature.

Le travail de Feng et al. 2010 se focalise sur l'apprentissage de systèmes hybrides bruités présentant des évolutions affines en temps discret. L'originalité de la méthode repose sur la prise en compte des bruits de mesure dans l'apprentissage. Le problème est transformé en un problème d'optimisation polynomiale et exploite sa structure particulière pour obtenir des problèmes solvables par calcul. Cette transformation est associée à une approche de type "Hit and Run" aléatoire, qui est une technique d'échantillonnage générant une séquence de points dans un ensemble avec des écarts dans des directions aléatoires (voir Zabinsky et al. 2013 pour plus de détails) ce qui amène une réduction de la complexité du problème et permet de résoudre des problèmes d'une taille similaire à des problèmes réels.

Dans Vidal 2004, des modèles *PieceWise Auto Regressive eXogenous* (PWARX) sont obtenus en utilisant une solution géométrique algébrique qui consiste à identifier le nombre d'états n du système par le degré d'un polynôme p et de voir l'ordre et les paramètres du modèle comme les facteurs de p. Les coefficients de p sont tout d'abord estimés à partir d'une contrainte dérivée à partir d'une relation donnée d'entrées et de sorties. Une fois le polynôme p obtenu, l'ordre et les paramètres de chaque sous-modèle peuvent être estimés en faisant le parallèle entre les dérivées de p et un ensemble de modèles de régression qui minimisent une fonction objectif donnée. La solution proposée ici ne nécessite pas de connaissance préalable sur l'ordre des sous-modèles recherchés (une limite haute est cependant nécessaire). De plus, le mécanisme de transition entre les différents sous-modèles peut être arbitraire : il n'y a pas besoin de prévoir un délai entre deux transitions. Par contre, la méthode proposée ne prend pas en compte le bruit sur les données.

Ces méthodes d'apprentissage direct pourraient s'appliquer à notre cas pour apprendre un système hybride. Cependant, une transformation des équations obtenues dans notre formalisme serait nécessaire pour pouvoir appliquer ensuite notre méthode de gestion de santé. En effet, les systèmes hybrides appris via ces méthodes sont représentés sous forme d'équations qui s'activent ou se désactivent et ne présentent pas de structure discrète comme on souhaiterait en obtenir.

3.2.4 Synthèse

Cet état de l'art a présenté des méthodes d'apprentissage de modèles pour des systèmes à événements discrets, des systèmes continus et des systèmes hybrides.

Suite à l'énoncé des besoins pour l'apprentissage de modèles pour des systèmes hybrides, il apparaît nécessaire de développer une méthode d'apprentissage qui mélange des techniques pour l'apprentissage de modèles à événements discrets et l'apprentissage de modèles continus. Les techniques d'apprentissage de modèles à événements discrets seront utilisées pour apprendre une structure discrète (apprentissage structurel). Les techniques d'apprentissage de systèmes continus seront quant à elles dédiées à l'apprentissage de dynamiques continues.

Pour la structure discrète, DyClee correspond le mieux à nos besoins étant donné qu'il

ne nécessite pas un nombre de clusters précis et est capable de reconnaître des changements de dynamiques. Pour l'apprentissage des dynamiques continues, trois algorithmes de régression, RFR, SVR et la RP vont être mis en place pour leur capacité de généralisation et leur réglage simple.

3.3 Processus d'apprentissage d'un HtPN

3.3.1 Processus d'apprentissage : vue d'ensemble

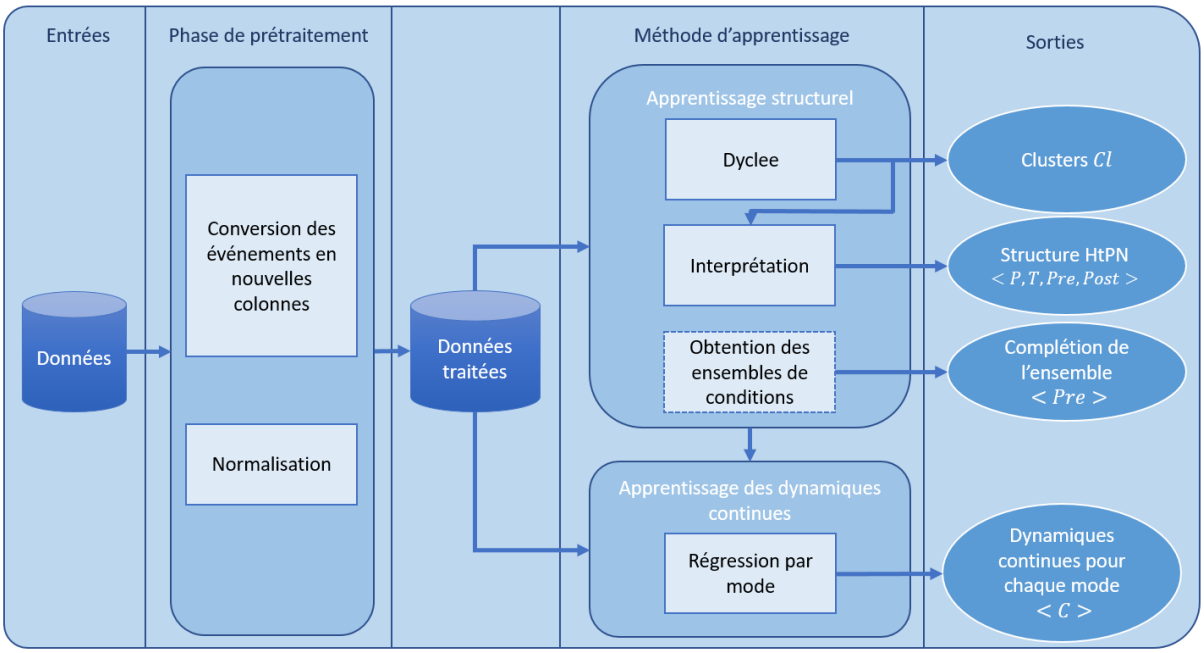


FIGURE 3.7 – Vue globale du processus d'apprentissage

Une vue d'ensemble du processus d'apprentissage d'un HtPN est présentée sur la figure 3.7.

Sur la gauche de la figure se trouvent les entrées, composées de données. Les données d'entrée du processus d'apprentissage représentent les variables observables de notre système, obtenues à travers des capteurs et actionneurs. Si on considère que les capteurs ne sont pas fautifs, les données obtenues représentent parfaitement le comportement du système, aux bruits de mesure près.

Ces données vont passer par une première phase de prétraitement. Cette phase inclut une conversion des événements pour les rendre exploitables et une étape de normalisation. La normalisation nécessite la connaissance des valeurs maximale et minimale pour chaque caractéristique des données. Dans le cas où le processus d'apprentissage est effectué hors-ligne, le nombre d'échantillons considérés est fini et les valeurs limites sont connues. Si les valeurs maximale et minimale sont connues en amont, le processus pourrait être effectué en ligne.

Les données traitées sont ensuite fournies à la méthode d'apprentissage, composée de l'apprentissage structurel et de l'apprentissage des dynamiques continues. L'apprentissage structurel est effectué par l'algorithme de clustering dynamique, DyClee, présenté dans la section 3.2.1.2. L'interprétation des résultats de DyClee permet d'obtenir la structure

du modèle : ses places P, ses transitions T, les matrices Pre et $Post$ avec leurs valeurs par défaut. La fonction d'obtention des ensembles de conditions permet de compléter une partie de la matrice Pre. Cette partie est représentée en pointillés, car elle a été définie théoriquement, mais n'est pas encore implémentée.

L'apprentissage des dynamiques continues associées à chacune des places du modèle est ensuite réalisé. Cette étape donne les dynamiques continues C_p associées à chaque place $p \in P$ de la structure HtPN apprise précédemment.

Les sorties de chaque partie de la méthode d'apprentissage sont visibles sur la droite de la figure.

3.3.2 Données d'entrée

Les données d'entrée utilisées pour apprendre un HtPN sont les variables observables du système. Certaines de ces variables ont des valeurs mesurées en continu, comme le niveau d'eau dans une cuve, une pression ou une température par exemple. D'autres sont des événements discrets, comme le contrôle d'une vanne (ouverture/fermeture) ou l'appui sur un bouton de marche/arrêt. Un événement observable est un événement dont l'occurrence est détectée par des capteurs ou une commande.

Les données d'entrée sont contenues dans une matrice $data$ de dimension $(I \times J)$ (comme on peut le voir dans la figure 3.12). Une ligne i correspond à un échantillon des données. Une colonne j correspond à une caractéristique observable des données. $data[i, :]$ correspond aux caractéristiques de l'échantillon i. $data[:, j]$ correspond à la colonne représentant la caractéristique j de tous les échantillons. $data[i, j]$ correspond donc à la caractéristique observable j de l'échantillon i. Par facilité de notation, on note S l'ensemble des échantillons. On a donc $S[i] = data[i, :]$. Dans le cas des événements discrets, différents événements peuvent être présents sur une même colonne, les commandes d'ouverture et fermeture d'une vanne, par exemple. C'est le cas dans l'exemple de la figure 3.12, où les événements $openv13$ et $closev13$ sont présents sur une même colonne. Les données sont obtenues par observation du système. Ceci implique que l'apprentissage possible du modèle sera dépendant du domaine d'observation.

3.3.3 Prétraitement

Les données brutes ont besoin d'être normalisées avant le processus de clustering afin d'équilibrer l'importance de chaque caractéristique des données dans le processus. En effet, si une des colonnes $data[:, j]$ a des valeurs comprises entre 0 et 2 et une autre des valeurs comprises entre 0 et 10^5, la première colonne va se retrouver négligée par le programme, car ses valeurs sont négligeables comparées aux valeurs de la deuxième colonne.

Par ailleurs, il faut convertir les événements discrets en valeurs booléennes, afin qu'ils puissent être considérés par l'algorithme de clustering. Le pseudo code de prétraitement est donné par l'algorithme 1.

Pour résumer, le prétraitement des données va modifier les données de la manière suivante :

— Créer une nouvelle colonne pour chaque événement discret et convertir l'absence ou la présence de l'événement en une valeur numérique : 0.0 pour l'absence de l'événement et 1.0 pour sa présence. Chaque événement aura ainsi sa propre colonne

Algorithme 1 Prétraitement des données

Entrée: *data*
Sortie: *data*

1: $mins = []$, $maxs = []$, $events = []$, $length_s = 0$
2: $length_s = length(data[0, :])$ ▷ On sauvegarde la taille d'un échantillon
3: ▷ On cherche l'ensemble des événements et les min et max de chaque caractéristique
4: **pour tout** $s \in length(data)$ **faire** ▷ On parcourt les échantillons
5: **pour tout** $j \in length(s.data)$ **faire** ▷ On parcourt les caractéristiques de l'échantillon s
6: **si** $type(s[j]) == string$ **alors** ▷ On regarde si la caractéristique j de s est un événement ou non
7: **si** $s[j] \notin events$ **alors** ▷ Si oui, on regarde si l'événement est connu
8: $events \leftarrow events \cup s[j]$ ▷ Si non connu, on le sauvegarde
9: **fin si**
10: **sinon**
11: **si** $s[j] > maxs[j]$ **alors** ▷ Si $s[j]$ est plus grand que $maxs[j]$ alors on m-à-j $maxs[j]$
12: $maxs[j] = s[j]$
13: **sinon si** $s[j] < mins[j]$ **alors** ▷ Si $s[j]$ est plus petit que $mins[j]$ alors on m-à-j $mins[j]$
14: $mins[j] = s[j]$
15: **fin si**
16: **fin si**
17: **fin pour**
18: **fin pour**
19: ▷ On normalise
20: **pour tout** $s \in length(data)$ **faire** ▷ On parcourt les échantillons
21: **pour tout** $j \in length(s.data)$ **faire** ▷ On parcourt les caractéristiques de l'échantillon s
22: $Append(s.data, zero(length(events)))$ ▷ On rajoute X zéros à la fin des caractéristiques de l'échantillon s. X correspond à la taille de l'ensemble *events*
23: **si** $type(s[j]) == string$ **alors** ▷ On regarde si la caractéristique j de s est un événement ou non
24: **pour tout** $e \in length(events)$ **faire** ▷ On parcourt l'ensemble *events*
25: **si** $s[j] == events[e]$ **alors** ▷ On trouve l'index de l'événement correspondant à $s[j]$ dans *events*
26: $s[length_s + e] = 1$ ▷ On met la nouvelle colonne à 1
27: **fin si**
28: **fin pour**
29: **sinon**
30: $s[j] = \frac{s[j]-mins[j]}{maxs[j]-mins[j]}$ ▷ On normalise
31: **fin si**
32: **fin pour**
33: **fin pour**

(lignes 22 à 26) ;

— Normaliser les valeurs des caractéristiques selon l'équation suivante (ligne 30) :

$$data[i,j] = \frac{data[i,j] - min[j]}{max[j] - min[j]} \tag{3.2}$$

avec $max[j]$ et $min[j]$ qui sont respectivement la valeur maximale et minimale de la colonne j et $data[i,j]$ la valeur de la colonne j à la ligne i.

Un exemple de prétraitement est visible sur la figure 3.13.

3.3.4 Apprentissage structurel

L'apprentissage structurel d'un HtPN consiste à obtenir les ensembles des places P, des transitions T et une partie des ensembles Pre et $Post$.

3.3.4.1 Obtention des clusters par DyClee

Une fois prétraitées, les données sont fournies à l'algorithme de clustering DyClee. Les paramètres utilisateurs pour DyClee, comme la taille des micro-clusters, doivent également être fournis. Fixer la taille des micro-clusters nécessite l'intervention d'un expert connaissant le système considéré. En sortie, DyClee donne un ensemble Cl de clusters, auxquels sont affectés les échantillons S.

3.3.4.2 Interprétation des résultats de DyClee

Les résultats de DyClee sont interprétés automatiquement pour fournir une structure HtPN. Le pseudo code de cette interprétation est présenté dans l'algorithme 2. Pour chaque cluster $Cl_i \in Cl$, une place p_i est créée (lignes 2, 3, 4). Ensuite, les transitions entre ces places sont retrouvées, en se basant sur le changement de cluster d'échantillons successifs. L'obtention des transitions est basée sur l'hypothèse suivante.

Hypothèse 1 (Structure du HtPN appris)**.** *La structure du système apprise n'admet toujours qu'un seul arc entrant dans une transition et un seul arc sortant.*

Appelons $S[0]$ le premier échantillon de l'ensemble S, c.-à-d. $data[0,:]$, la première ligne de l'ensemble de données $data$. Le cluster auquel il est affecté, est noté $Cl^{S[0]}$ et est mémorisé dans la variable act_p (ligne 5). Un compteur cnt, ayant pour valeur initiale 0 et un ensemble vide de transitions créées $created_transitions$ sont définis (lignes 6 et 7). Puis, pour chaque échantillon s de S, le cluster Cl^s est comparé avec le cluster sauvé dans act_p. Tant que ces clusters sont identiques ($Cl^s == act_p$), la valeur de cnt reste à 0 (ligne 12). Quand un échantillon est affecté à un cluster différent de celui dans act_p ($Cl^s! = act_p$), le compteur cnt est incrémenté (ligne 10). Quand la valeur de ce compteur atteint $th_outlier$, le cluster sauvé dans act_p et Cl^s sont considérés comme liés, et une transition $act_p_Cl^s$ est créée (lignes 16). La place associée au cluster sauvé dans act_p est mise en entrée de la transition et la place associée au cluster est mise en sortie. La transition créée est sauvée dans l'ensemble $created_transitions$ (ligne 17) afin d'éviter de multiples créations d'une même transition dans le cas d'une périodicité au niveau des clusters créés. Le compteur cnt a été implémenté afin d'éviter la création de

Algorithme 2 Obtention de la structure HtPN

Entrée: Cl, S, $th_outlier$
Sortie: $HtPN_structure : P, T, Pre, Post$

1: $P = \{\}, T = \{\}, A = \{\}$
2: **pour tout** $Cl_i \in Cl$ **faire**
3: $\quad P \leftarrow P \cup Create(p_i)$ ▷ Création d'une place p_i pour chaque cluster Cl_i
4: **fin pour**
5: $act_p \leftarrow Cl^{S[0]}$
6: $cnt = 0$
7: $created_transitions = \{\}$
8: **pour tout** $s \in S$ **faire**
9: $\quad$ **si** $Cl^s! = act_p$ **alors**
10: $\quad\quad cnt+ = 1$ ▷ Incrémentation du compteur anti-outliers
11: $\quad$ **sinon**
12: $\quad\quad cnt = 0$
13: $\quad$ **fin si**
14: $\quad$ **si** $cnt == th_outlier$ **alors**
15: $\quad\quad$ **si** $act_p_Cl^s \notin created_transitions$ **alors**
16: $\quad\quad\quad [T, Pre, Post] \leftarrow [T, Pre, Post] \cup C_t(act_p_Cl^s, act_p, Cl^s)$ ▷ Si la transition n'existe pas déja, on la créée avec la fonction C_t
17: $\quad\quad\quad created_transitions.add(act_p_Cl^s)$
18: $\quad\quad\quad act_p \leftarrow Cl^s$
19: $\quad\quad$ **fin si**
20: $\quad$ **fin si**
21: **fin pour**

transitions erronées dues à de potentielles erreurs d'observation. L'hypothèse faite ici est que le nombre d'erreurs consécutives au niveau de l'attribution des échantillons dans des clusters est limité à l'hyper-paramètre $th_outlier$. Comme point de départ, ce nombre a été fixé arbitrairement à 3. Une étude plus poussée sur la sélection de cet hyper-paramètre sera nécessaire dans des travaux futurs. Le compteur va donc empêcher la création d'une transition entre le cluster erroné et le reste de la structure, étant donné qu'il n'atteindra pas la valeur de $th_outlier$ pour créer la transition. Il est important de noter qu'étant donné que l'apprentissage de la structure se fait de manière non supervisée, il n'est pas possible d'évaluer la qualité de la structure apprise.

En revenant à la définition formelle des HtPN donnée dans la section 2, après apprentissage de la structure par DyClee, les ensembles suivants sont obtenus entièrement ou partiellement :

— P, l'ensemble des places, obtenu entièrement ;
— T, l'ensemble des transitions, obtenu entièrement ;
— Pre, la matrice des conditions de tir du système, obtenue partiellement. Seule l'existence d'un élément est obtenue, l'élément en question ne l'est pas encore ;
— $Post$, la matrice des assignations de tir du système, obtenue partiellement. Seule l'existence d'un élément est obtenue, l'élément en question ne l'est pas encore.

3.3.4.3 Apprentissage des conditions numériques

Cette partie propose une méthode pour l'apprentissage des conditions continues des HtPN, complétant la matrice Pre. Elle n'a pas encore été implémentée. Les conditions numériques prises en compte à ce stade du travail sont des conditions numériques simples, explicitées dans la définition 22. L'hypothèse suivante est donc formulée.

Hypothèse 2 (Réduction de la classe des systèmes appris)**.** *Le système appris ne contient que des conditions numériques simples.*

Le processus d'apprentissage des conditions continues est basé sur l'hypothèse 2 et est détaillé dans l'algorithme 3 ci-dessous. L'idée générale est de considérer toutes les conditions numériques ayant participé au tir d'une transition t. Ces conditions sont ensuite filtrées de manière à ne conserver que les conditions réellement pertinentes. Ce filtrage se base sur l'étude de la variance de chaque caractéristique. En effet, pour différentes occurrences de tir d'une même transition, si une des variables continues présente toujours des valeurs similaires (c.-à-d. une variance faible), on peut supposer que cette variable continue intervient dans le tir de la transition.

Algorithme 3 Obtention de la condition numérique associée à une transition t

Entrée: $Firing_{Dates}, S, th_{var}$
Sortie: $\Omega^N_{p,t}$

1: $CG = \{\}, Final_CG = \{\}$
2: **pour tout** $DT \in Firing_{Dates}$ **faire**
3: **pour tout** $s \in S$ **faire**
4: **si** $s.time == DT$ **alors**
5: $CG \leftarrow CG \cup s.Data$
6: **fin si**
7: **fin pour**
8: **fin pour**
9: $Final_CG \leftarrow CG$
10: $var_{ft} = 0$
11: **pour tout** $i \in [1..length(S)]$ **faire**
12: $ft = \{\}$ ▷ Les valeurs de chaque caractéristique i sont stockées ici
13: **pour tout** $cg \in CG$ **faire**
14: $ft \leftarrow ft \cup cg[i]$
15: **fin pour**
16: $var_{ft} = Var(ft)$ ▷ Calcul de la variance de ft
17: **si** $var_{ft} > th_{var}$ **alors**
18: $ft_to_rem = ft_to_rem \cup i$
19: **fin si**
20: **fin pour**
21: **pour tout** $cg \in CG$ **faire**
22: $Final_CG[cg] = Final_CG[cg].remove(ft_to_rem)$
23: **fin pour**

Obtention de l'ensemble des conditions numériques CG (lignes 2 à 8 de l'algorithme 3) Supposons qu'une transition t ait été tirée X fois et que les données aient

J caractéristiques. La date d'occurrence de chaque tir de t est supposée connue et est stockée dans l'ensemble $Firing_{Dates}$. Chaque DT dans l'ensemble $Firing_{Dates}$ est donc une date de tir de t. Chaque échantillon s de S est daté. Si la date d'un échantillon s correspond à une date de tir DT de la transition t, alors ses caractéristiques sont stockées dans l'ensemble CG des conditions numériques qui correspondent au tir de t. Pour chaque occurrence i de t, un ensemble CG^i de conditions est ainsi obtenu :

$$CG^i = \{(ft_1^i = a^i) \wedge (ft_2^i = b^i) \wedge (...) \wedge (ft_J^i = J^i)\} \tag{3.3}$$

où ft_j^i est la caractéristique j de l'occurrence i et $a^i, b^i, ..., y^i$ les valeurs continues portées par chaque caractéristique lors de l'occurrence i. Un ensemble de conditions numériques associées à l'arc entrant de la transition CG est ainsi obtenu pour chaque transition t :

$$CG = \{CG^1 \vee CG^2 \vee ... \vee CG^X\} \tag{3.4}$$

Élimination des caractéristiques non pertinentes (lignes 9 à 23 de l'algorithme 3) En se basant sur les X occurrences de t, il est possible de calculer une variance pour chaque caractéristique ft_j. Si la variance d'une caractéristique dépasse un seuil th_{var} donné, la caractéristique est considérée comme n'intervenant pas dans le tir de t. Il est donc possible de supprimer cette caractéristique de l'ensemble final de conditions numériques associées à l'arc entrant de t $Final_CG$ associé à t. Par exemple, supposant que seule la variance de la caractéristique $ft2$ est supérieure au seuil th_{var}. Le nouvel ensemble $Final_CG$ sera donc :

$$Final_CG = \{(ft_1 = a) \wedge (ft_3 = c) \wedge (...) \wedge (ft_Y = y)\} \tag{3.5}$$

pour $i \in [1..X]$. et $Card(Final_CG^i) = Card(CG^i) - 1$. Par exemple, si l'on considère trois tirs différents d'une transition t avec les variables continues suivantes lors des tirs :

- $ft_1^1 = 5 \wedge ft_2^1 = 2 \wedge ft_3^1 = 30$
- $ft_1^2 = 4.9 \wedge ft_2^2 = 15 \wedge ft_3^2 = 31$
- $ft_1^3 = 5.1 \wedge ft_2^1 = -8 \wedge ft_3^3 = 32$

La variance de ft_2 étant importante, cette caractéristique sera considérée comme n'intervenant pas dans le tir de t et on obtiendra donc l'ensemble de condition $Final_CG$ régissant le tir de t suivant :

$$Final_CG = \{(ft_1 = 5) \wedge (ft_3 = 31)\} \tag{3.6}$$

3.3.5 Apprentissage des dynamiques continues

Trois algorithmes de régression ont été utilisés pour apprendre les dynamiques continues du système : le SVR, le RFR et la RP, présentés dans la section 3.2.2. Comme on l'a déjà souligné, ces trois algorithmes sont capables d'apprendre une dynamique continue linéaire ou non-linéaire, même avec des données bruitées. L'avantage principal d'utiliser trois algorithmes différents est que l'utilisateur peut choisir ce qui l'intéresse dans le cadre de l'apprentissage : la précision, la vitesse de calcul nécessaire à l'obtention du modèle de correspondance ou la possibilité d'extrapoler. Afin de représenter la qualité de la correspondance entre les données d'entrée et les prévisions par le modèle appris, un coefficient de détermination R^2 est utilisé :

$$\mathrm{R}^2 = 1 - \frac{(y_{true} - y_{pred})^2}{(y_{true} - y_{true_{mean}})^2} \tag{3.7}$$

où y_{true} est la valeur exacte de la donnée, $y_{true_{mean}}$ la moyenne des y_{true} et y_{pred} la valeur prédite par le modèle appris. Plus le coefficient R^2 est proche de 1, plus la valeur prédite est proche de la valeur réelle. Ce coefficient donne donc une idée de la capacité du modèle appris à répliquer correctement le comportement des données. Dans notre cas, l'algorithme ayant le meilleur coefficient de détermination a été choisi. Ceci signifie que lors de la construction du modèle HtPN final, les dynamiques continues seront représentées par le modèle appris ayant la meilleure correspondance valeurs prédites / valeurs réelles.

L'apprentissage des dynamiques continues est effectué place par place et donne donc l'ensemble des dynamiques continues C. Les échantillons affectés dans le cluster correspondant à la place concernée sont fournis en entrée des algorithmes. Si le système est périodique et que différents jeux d'échantillons sont associés à une même place, le modèle précédemment appris va être appliqué à chaque nouveau jeu. Si le R^2 obtenu par l'application de l'ancien modèle sur les nouvelles données est inférieur à un seuil donné, alors une nouvelle place contenant les nouveaux échantillons va être créée et de nouvelles dynamiques continues lui seront associées. C'est ce qu'on appelle le *mécanisme de ré-apprentissage*.

Pour chaque place $p \in P$, un ensemble d'équations continues C_p est donc obtenu sous la forme d'un modèle SVR, RFR ou RP appris :

$$C_p : y_k = RM(x_k), \tag{3.8}$$

où RM représente le modèle régressif obtenu et $RM(x_k)$ la prédiction donnée par ce modèle avec un vecteur d'état continu x à un temps k.

Bien qu'encore non implémenté, un processus similaire paraît applicable pour apprendre l'ensemble D des dynamiques de dégradation. La difficulté principale dans l'apprentissage des fonctions de dégradation est la fenêtre de temps considérée, étant donné que l'évolution des dynamiques de dégradation est généralement beaucoup plus lente que l'évolution des dynamiques continues qui décrivent le fonctionnement du système.
En se basant sur ces quatre étapes (prétraitement, apprentissage structurel, apprentissage des dynamiques continues), il est bien évidemment possible d'apprendre des modèles dans d'autres formalismes présentés dans la section 2.2. On citera notamment les automates hybrides ou les réseaux de Petri mixtes. En effet, l'application de cette méthode au formalisme des HtPN n'est ici présente que pour l'objectif global de la thèse. La méthode elle-même pourrait tout aussi bien s'appliquer à d'autres formalismes qui valident les hypothèses.

3.3.6 Déclenchement de l'apprentissage pour améliorer un modèle

Dans les travaux proposés dans cette thèse, le processus d'apprentissage est déclenché par l'utilisateur. Néanmoins, des indicateurs de performances, sous forme de scores, ont été définis. Ces indicateurs sont basés sur les propriétés de diagnosticabilité et de pronosticabilité de fautes. Dans des futurs travaux, il sera ainsi possible de se baser sur ces indicateurs pour déclencher l'apprentissage.

La suite de cette section présente les concepts clés du déclenchement de l'apprentissage, puis l'élaboration du score de diagnosticabilité et de pronosticabilité. Elle finit par la présentation de la boucle itérative de validation/mise à jour du modèle basée sur les scores définis auparavant.

3.3.6.1 Diagnosticabilité, pronosticabilité, criticité

Tout d'abord, les concepts de diagnosticabilité, pronosticabilité et criticité d'une faute doivent être définis. Pour cela, une étude des définitions de pronosticabilité et diagnosticabilité présentes dans la littérature a été effectuée dans VIGNOLLES, CHANTHERY et al. 2020. Suite à cette étude, les définitions suivantes de diagnosticabilité et pronosticabilité sont proposées pour le formalisme des HtPN.

Diagnosticabilité

Définition 26 (Faute diagnosticable)**.** *On distingue deux types de fautes :*

- *Les fautes événementielles, représentées par des événements discrets. Considérant $\mathcal{F} = \{f_0 ... f_n\}$ l'ensemble de fautes possibles, avec f_0 le cas nominal du système, une faute f_i est diagnosticable si et seulement si son occurrence produit une séquence de transitions tirées de taille $N \in \mathbb{N}^{+*}$ distincte du cas nominal et de toute autre séquence d'observations provoquée par une autre faute de l'ensemble $\mathcal{F}$.*
- *Les fautes continues, représentées par le dépassement non désiré d'un seuil pour un paramètre. Une faute continue f_c est dite diagnosticable si et seulement si sa signature de faute est unique et est un vecteur non nul (STAROSWIECKI et al. 1991).*

On définit un indicateur de diagnosticabilité booléen δ_{fi} pour chaque faute f_i, $i = 1, \ldots, n$.

Définition 27 (Indicateur de diagnosticabilité)**.** *Si f_i est diagnosticable, alors $\delta_{fi} = 1$, sinon $\delta_{fi} = 0$.*

Pronosticabilité

Définition 28 (Faute pronosticable)**.** *Une faute f est pronosticable si les lois de dégradation sont connues dans toutes les places qui pourraient mener à son occurrence.*

Il est important de noter que cette définition de faute pronosticable ne donne pas d'indices de confiance ou d'exactitude quant aux lois de dégradation représentant la faute.

Il est donc nécessaire d'introduire la notion de faute γ-pronosticable qui utilise la notion de score d'une faute. Le score s_f d'une faute f peut être calculé après son occurrence. Ce score est un réel entre 0 et 1. Il compare la date d'occurrence effective t_f de la faute f (obtenue à partir des données) avec la date d'occurrence estimée $\hat{t_f}$.

Définition 29 (Score d'une faute f)**.** *Le score s_f d'une faute est donné par :*

$$s_f = \frac{min(t_f, \hat{t_f})}{max(t_f, \hat{t_f})} \tag{3.9}$$

avec t_f la date d'occurrence effective de la faute f et $\hat{t_f}$ sa date d'occurrence estimée par l'algorithme de pronostic.

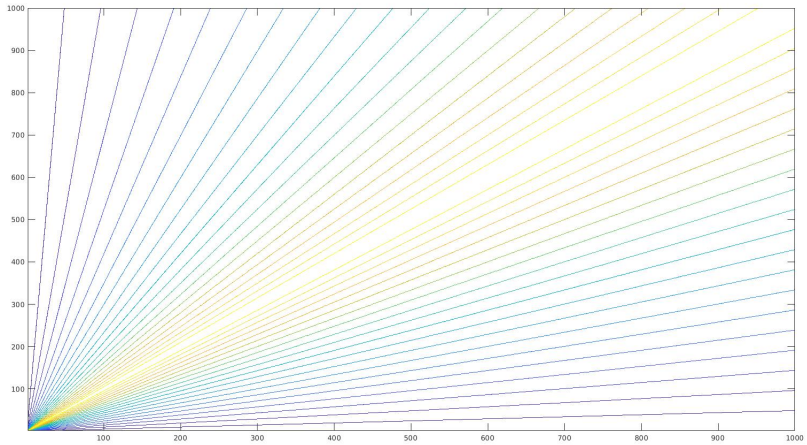


FIGURE 3.8 – s_f calculé pour différents t_f et $\hat{t_f}$

Cette équation traduit le fait que plus les deux dates d'occurrence sont proches, plus le score de la faute sera proche de 1. Ce score traduit la différence relative entre les deux dates d'occurrence : une faible différence à basse échelle temporelle (par exemple, $t_f = 10$, $\hat{t_f} = 5$) donnera un score ayant une valeur faible ($s_f = 0.5$ dans l'exemple donné ; la différence relative étant importante), alors que la même différence à grande échelle temporelle (par exemple, $t_f = 10000$, $\hat{t_f} = 9995$) donnera un score proche de 1 ($s_f = 0.999$ dans l'exemple donné).

Une étude de la valeur du score d'une faute f pour différentes dates t_f et $\hat{t_f}$ allant de 1 à 1000 est illustrée sur la figure 3.8.

Les axes correspondent aux valeurs de t_f et $\hat{t_f}$ et les couleurs représentent des paliers de 0.05. Plus la ligne est sombre, plus la valeur du score s_f est proche de 0. Le premier trait jaune au centre représente la limite de 0.95. Cette figure nous a permis de vérifier que l'équation proposée pour le score correspondait bien à nos attentes.

Remarque 2. *Le score s_f peut être rapproché de la métrique appelée exactitude relative (Relative Accuracy) définie dans* SAXENA *et al. 2009 :*

$$ER = 1 - \frac{|RUL - R\hat{U}L|}{RUL} \tag{3.10}$$

avec RUL la durée de vie réelle et $R\hat{U}L$ la durée de vie estimée. Cette formule n'a pas été choisie, car la valeur de l'exactitude relative n'est pas comprise entre 0 et 1, ce qui rend la métrique moins explicite à notre sens.

Ce score est ensuite comparé à une valeur seuil γ, choisi par l'utilisateur, afin de savoir si elle est γ-pronosticable ou non.

Définition 30 (Faute γ-pronosticable)**.** *Une faute f est γ-pronosticable si les lois de dégradation entraînent un score $s_f > \gamma$, avec γ une valeur seuil choisie par l'utilisateur.*

On définit un indicateur de pronosticabilité γ_{fi} pour chaque faute f_i, $i = 1, \ldots, n$.

Définition 31 (Indicateur de pronosticabilité)**.** *Si f_i est γ-pronosticable, alors $\gamma_{fi} = 1$, sinon $\gamma_{fi} = 0$.*

Criticité d'une faute L'indice de criticité $\alpha_{fi} \in [0, 1]$ d'une faute f_i est un réel compris entre 0 (faute peu critique) et 1 (faute très critique). Il représente le niveau de criticité de la faute et son importance dans le système : cet indice est donc commun au diagnostic et au pronostic. Plus une faute sera critique, plus elle aura un poids dans les indicateurs de performance.

Cet indice doit être fourni dans le modèle HtPN et peut être obtenu à travers des analyses de criticité et/ou des avis d'experts.

3.3.6.2 Indicateurs de performance

Les indicateurs de performance proposés ci-après dépendent des capacités d'observation du système. Dans le cas où de nouvelles données deviennent disponibles au cours du temps, il peut être possible d'améliorer les indicateurs de performance, d'où le déclenchement de l'apprentissage.

La valeur de l'indicateur de performance de diagnosticabilité, *Diagnosability Performance Indicator* (DPI), du système est obtenue en calculant la proportion de fautes diagnosticables (en tenant compte de leur indice de criticité) par rapport à la criticité totale de l'ensemble des fautes. Le DPI est un réel compris entre 0 et 1.

Définition 32 (DPI du système)**.** *Dans le cas où au moins un des indices de criticité des fautes est non nul, le DPI du système est donné par :*

$$DPI = \frac{\sum_{i=1}^{n} \alpha_{fi}.\delta_{fi}}{\sum_{i=1}^{n} \alpha_{fi}} \tag{3.11}$$

Dans le cas où tous les indices de criticité des fautes f_i sont à 0, le DPI est fixé à 1.

Ce choix s'explique par la nature du DPI : si ce dernier est proche de 0, on souhaite déclencher l'apprentissage du modèle pour améliorer la diagnosticabilité. S'il est proche de 1, le déclenchement de l'apprentissage pour améliorer la diagnosticabilité n'est pas jugé nécessaire. Si tous les indices de criticité sont fixés à 0, cela signifie qu'aucune faute n'est vraiment importante pour l'utilisateur, on ne souhaite donc pas déclencher la fonction d'apprentissage.

La valeur de l'indicateur de performance de pronosticabilité, *Prognosability Performance Indicator* (PPI), du système est obtenue après étude du score pour chaque faute en calculant la proportion de fautes γ-pronosticable (en tenant compte de leur indice de criticité). Le PPI est un réel compris entre 0 et 1.

Définition 33 (PPI du système)**.** *Dans le cas où au moins un des indices de criticité des fautes est non nul, le PPI du système est donné par :*

$$PPI = \frac{\sum_{i=1}^{n} \alpha_{fi}.\gamma_{fi}}{\sum_{i=1}^{n} \alpha_{fi}} \tag{3.12}$$

Dans le cas où tous les indices de criticité des fautes f_i sont à 0, le PPI est fixé à 1.

Ce choix s'explique par la nature du PPI : si ce dernier est proche de 0, on souhaite déclencher l'apprentissage du modèle pour améliorer la pronosticabilité. S'il est proche de 1, le déclenchement de l'apprentissage pour améliorer la pronosticabilité n'est pas jugé nécessaire. Si tous les indices de criticité sont fixés à 0, cela signifie qu'aucune faute n'est vraiment importante pour l'utilisateur, on ne souhaite donc pas déclencher la fonction d'apprentissage.

3.3.6.3 Boucle de validation/mise à jour du modèle

La validation/mise à jour du modèle proposée est résumée sur la figure 3.9.

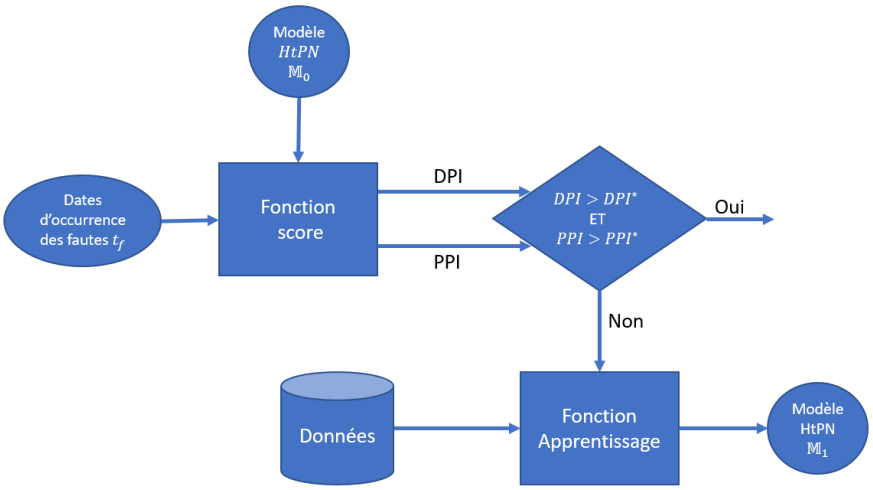


FIGURE 3.9 – Processus de validation du modèle

On dispose à l'instant initial d'un modèle $HtPN_{\Phi,k}$ du système. Ce modèle, ainsi que les dates t_f d'occurrence des diverses fautes, est fourni à la *Fonction score*, qui calcule le DPI, image de la diagnosticabilité du système, et PPI, image de la pronosticabilité du système. Ces deux indicateurs de performance sont comparés à des seuils, choisis par l'utilisateur, que l'on note DPI^* et PPI^*. Si le DPI courant est supérieur à DPI^* et le PPI courant est supérieur à PPI^* alors le système est suffisamment diagnosticable et pronosticable. Sinon, le modèle va suivre un processus d'apprentissage avec des jeux de données. Dans le cas d'un enclenchement de la fonction d'apprentissage, un ensemble de données issues des observations sur le système est utilisé pour l'apprentissage. À l'issue de cet apprentissage, un nouveau modèle $HtPN_{\Phi,k+1}$ est disponible. Cette boucle de validation/mise à jour du modèle est itérative.

3.4 Application de la méthode d'apprentissage

Le système utilisé dans ce chapitre pour vérifier la méthode d'apprentissage est une réduction du système des cuves d'eau illustré sur la figure 2.7 page 40.

3.4.1 Description du système

Les données sur lesquelles est basé l'apprentissage sont obtenues via une simulation du système par le logiciel HeMU. Le modèle est donc *a priori* connu et est utilisé pour vérifier la pertinence et la qualité du processus d'apprentissage.

La partie considérée de la représentation multimode du système de cuves d'eau est composée de 6 modes différents comme illustré par la figure 3.10. La description du système est disponible dans la section 2.4.3.1 page 40. Les 4 modes présents dans le carré rouge sur la figure 3.10 sont les modes qui sont, *a priori*, apprenables par le processus d'apprentissage, que ce soit via l'occurrence d'événements discrets ($close_{v13}$ et $open_{v13}$)

ou par une modification des dynamiques continues (suite à l'occurrence de la faute f_1). L'occurrence de la faute f_0 n'est quant à elle pas apprenable, car elle n'est pas liée à un événement discret et n'entraîne aucune modification des dynamiques continues : elle n'a aucun effet sur les observations émises par le système.

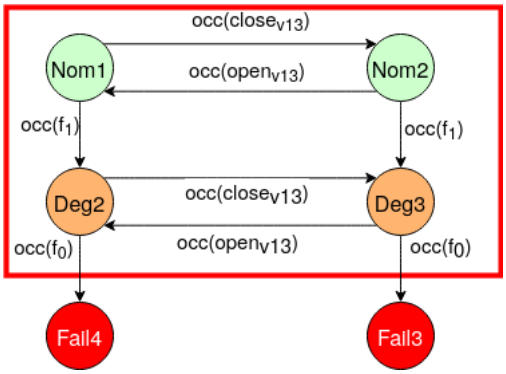


FIGURE 3.10 – Représentation multimode réduite du système des trois cuves d'eau

3.4.2 Données d'entrée

Le scénario simulé est illustré sur la figure 3.11 et est le même que le scénario utilisé dans le chapitre 2. 310 minutes après le début du scénario, la vanne v_{13} est fermée pendant 20 minutes toutes les heures afin d'effectuer un traitement de l'eau présente dans la cuve T_1. La faute f_1 apparaît à 201840s et entraîne la faute f_0 à 206040s. Aucune des fautes, que ce soit f_1 ou f_0 n'est observable directement. Si l'on se réfère seulement à cette simulation, sans considérer le potentiel impact des fautes sur les dynamiques, on peut attendre 5 places après l'apprentissage structurel. Seulement, comme explicité lors de la description du système, l'occurrence de la faute f_0 n'a aucun effet sur les observations émises du système. 4 places seront donc attendues après l'apprentissage structurel. Les données d'entrée considérées sont comprises dans la matrice *data*, de taille 3399×5, ce qui correspond aux 3399 échantillons obtenus après simulation du scénario considéré et au fait que chaque échantillon possède 5 caractéristiques. La figure 3.12 illustre deux échantillons

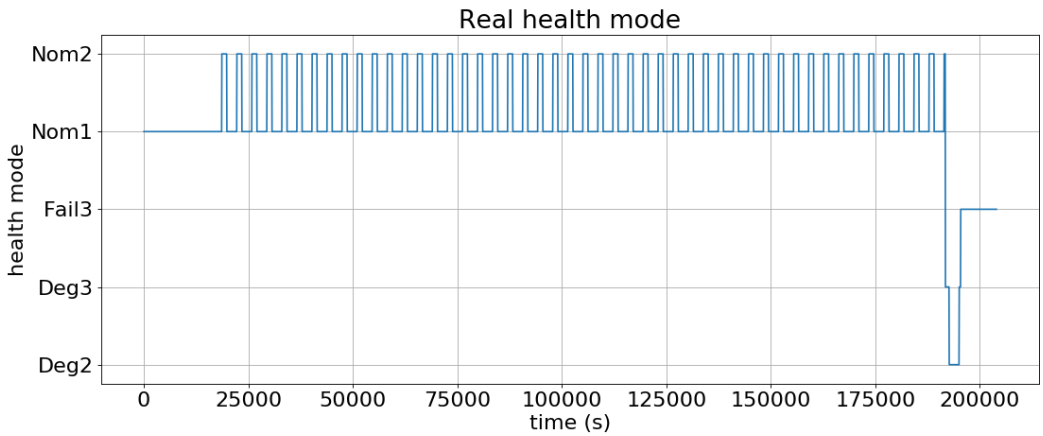


FIGURE 3.11 – Scénario simulé

successifs de données obtenues par la simulation du système. L'échantillonnage est effectué toutes les 60 secondes (c.-à-d. toutes les minutes). La première colonne des données, $data[:, 0]$, est le temps de l'échantillon, qui correspond au temps écoulé depuis le début de la simulation lorsque les données ont été relevées. Les deuxième, troisième et quatrième colonnes, $data[:, 1]$, $data[:, 2]$ et $data[:, 3]$, représentent les niveaux d'eau dans chaque cuve. La dernière colonne, $data[:, 4]$, représente l'occurrence d'événements discrets. Quand un événement est vu par le système, il est mémorisé dans les données jusqu'à l'occurrence d'un nouvel événement. Par exemple, l'événement *openv13* a été vu à l'initialisation du système à $t = 0s$, et a été mémorisé jusqu'à l'occurrence de l'événement *closev13* à $t = 309s$.

```
308|2.0979920734557616|1.8823981584084746|1.98808594503705|openv13
309|2.1323807007763182|1.8827729021376856|1.954482648376446|closev13
```

FIGURE 3.12 – Exemple de données avant normalisation

3.4.3 Prétraitement des données

Les données après prétraitement sont visibles sur la figure 3.13. La matrice *data* est désormais de taille 3399×6. Il y a toujours 3399 échantillons, mais ceux-ci possèdent désormais 6 caractéristiques. En effet, deux événements distincts étaient présents sur la dernière colonne avant normalisation. Ils ont donc été séparés en deux colonnes distinctes. La première colonne, $data[:, 0]$, n'est pas modifiée, car elle correspond au temps de la donnée. Les colonnes $data[:, 1]$, $data[:, 2]$ et $data[:, 3]$, ont été normalisées entre 0 et 1 et correspondent aux niveaux d'eau normalisés de chaque cuve. La dernière colonne a été séparée en deux colonnes : une pour l'événement *openv13*, $data[:, 4]$, et une pour l'événement *closev13*, $data[:, 5]$. Chacune de ces colonnes prend la valeur 1.0 si l'événement est présent sur la ligne correspondante ou 0.0 si ce n'est pas le cas.

```
308.0, 0.3235, 0.8272, 0.8217, 1.0, 0.0
309.0, 0.3314, 0.8274, 0.8078, 0.0, 1.0
```

FIGURE 3.13 – Exemple de données après prétraitement

3.4.4 Apprentissage structurel

3.4.4.1 Obtention des clusters par Dyclee

Les données normalisées sont ensuite fournies à DyClee, configuré avec une taille de micro-clusters valant 0.2. Cette valeur a été obtenue par expérience, en cherchant à obtenir un nombre de clusters proche de celui attendu *a priori*. Les résultats obtenus sont donnés sur la figure 3.14 : 4 clusters sont obtenus, ce qui est cohérent avec les résultats attendus.

3.4.4.2 Interprétation des résultats de DyClee

Les résultats de DyClee sont ensuite interprétés pour obtenir la structure HtPN. Pour chacun des 4 clusters identifiés par DyClee, une place p_i est créée, avec $i \in [1..4]$. Cinq transitions sont ensuite créées en utilisant l'Algorithme 2 : $p_1 \rightarrow p_2$, $p_2 \rightarrow p_1$, $p_1 \rightarrow p_4$, $p_4 \rightarrow p_3$ et $p_3 \rightarrow p_4$. La figure 3.15 montre la structure HtPN obtenue.

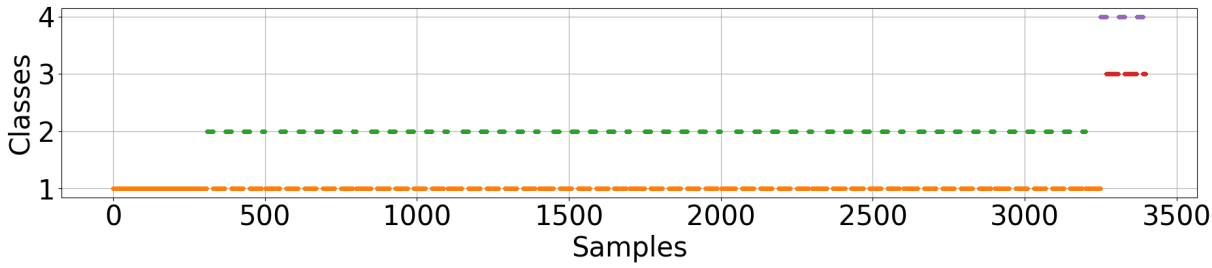


FIGURE 3.14 – Clusters identifiés par DyClee

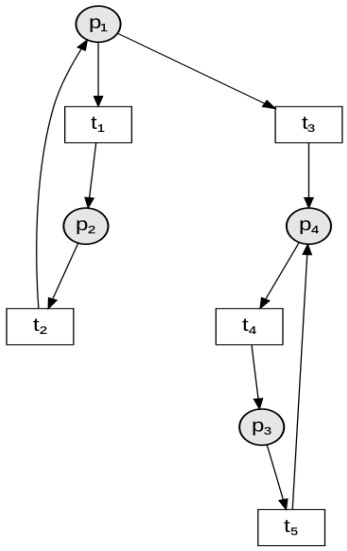


FIGURE 3.15 – Structure HtPN obtenue

Les clusters identifiés par DyClee peuvent être comparés aux modes du système simulé illustré par la figure 3.11. Tout d'abord, en comparant les deux figures, le mode Nom_1 peut être associé à la place p_1 et le mode Nom_2 à la place p_2. L'apparition de la faute f_1 peut être associée au passage de la place p_1 à la place p_4 sur la figure 3.14. Bien que non observable, la faute f_1 impacte le comportement continu du niveau d'eau présent dans la cuve T_1. L'absence d'un cinquième mode assimilable au mode $Fail_X$ est notable. Ceci s'explique par le fait que la faute f_0, lors de son occurrence, n'impacte pas du tout les dynamiques du système et ne peut donc être détectée par DyClee.

La représentation multimode apprise à partir des observations correspond donc effectivement au carré rouge présent sur la figure 3.10, à cause de la non-observabilité de la faute f_0.

En prenant en compte cette limite, l'apprentissage de la structure donne des résultats globalement satisfaisants. Le modèle appris représente le comportement du système de manière précise étant donné que la structure du système multimode apprenable a pu être retrouvée à partir des données disponibles.

3.4.5 Apprentissage des dynamiques continues

L'ensemble des dynamiques continues est obtenu lors du processus d'apprentissage en utilisant trois algorithmes de régression différents : le SVR, le RFR et la RP. Ils ont été implémentés en utilisant la boîte à outils Scikitlearn de Python. Étant donné que le système appris est périodique (alternance entre les places p_1 et p_2, par exemple), le processus va apprendre une dynamique pour chaque sous-ensemble d'échantillons associé à une place. On choisit d'illustrer le processus sur les trois premiers sous-ensembles d'échantillons. Le premier ensemble contient les échantillons datés de 0s à 309s, ce qui correspond à la place p_1, débutant au début de la simulation et jusqu'à l'occurrence de l'événement $close_{v13}$. Le deuxième ensemble contient les échantillons datés de 310s à 329s et est associé à la place p_2. Le dernier ensemble contient les échantillons datés de 330s à 370s et correspond au retour dans la place p_1. Les résultats obtenus pour ces trois ensembles d'échantillons sont donnés sur les figures 3.16, 3.17 et 3.18, respectivement. Les données réelles sont représentées avec les points rouges, le modèle appris par l'algorithme SVR avec la ligne bleue, le modèle appris par l'algorithme RFR avec la ligne jaune et le modèle appris par l'algorithme RP avec la ligne rose. Pour chaque mode (c.-à-d. place), les algorithmes sont notés en utilisant le coefficient de détermination R^2 et le temps de calcul nécessaire. La comparaison des résultats d'apprentissage obtenus par chaque méthode de régression est visible dans le tableau 3.1.

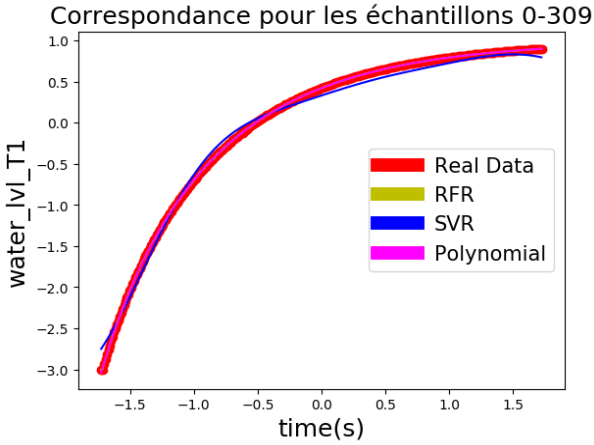


FIGURE 3.16 – Résultats de l'apprentissage pour les échantillons [0 : 309]

TABLEAU 3.1 – Comparaison du score R^2 et du temps de calcul requis

	R^2			Temps de calcul		
Echantillons	SVR	RFR	RP	SVR	RFR	RP
0 to 309	0.99356	0.99997	0.99999	0.7ms	6.7ms	0.5ms
310 to 329	0.98908	0.99604	1.0	0.4ms	4.4ms	0.5ms
330 to 370	0.68164	0.99003	0.99999	0.3ms	4.4ms	0.7ms

Dans notre cas, l'algorithme SVR est généralement plus rapide d'un facteur 10 par rapport au RFR et aussi rapide que la RP, mais a un score R^2 légèrement inférieur à ceux des algorithmes RFR et de RP, sauf pour les échantillons [330 :370] où le coefficient de détermination montre que l'algorithme SVR est moins performant pour transcrire les dynamiques du système.

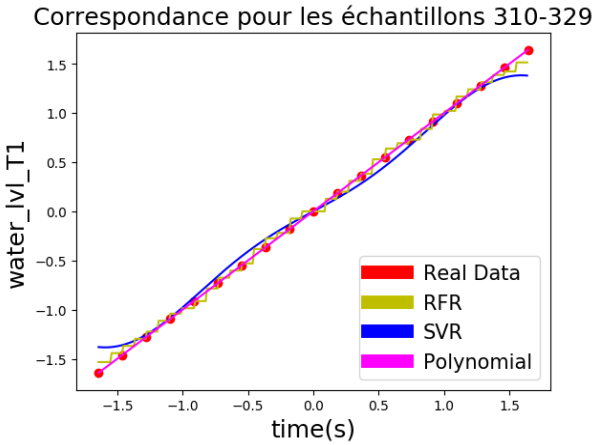


FIGURE 3.17 – Résultats de l'apprentissage pour les échantillons [310 : 329]

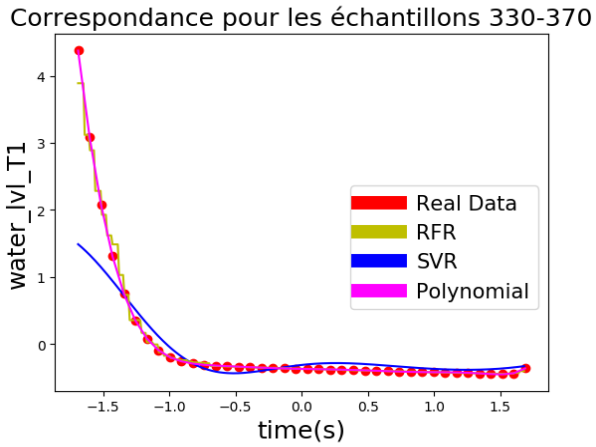


FIGURE 3.18 – Résultats de l'apprentissage pour les échantillons [330 : 370]

Une étude a été réalisée afin d'étudier la correspondance entre le modèle appris et le jeu de données pour différents ordres de régression polynomiale (donc influant sur le degré du polynôme appris). Les résultats sont visibles sur les figures 3.19, 3.20 et 3.21 et sur le tableau 3.2. Comme on peut s'y attendre, le score R^2 augmente avec le degré du polynôme. Le temps de calcul, de son côté, reste constant et n'augmente pas forcément. Le risque d'augmenter le degré est de sur-apprendre les données ou encore d'augmenter la sensibilité du modèle appris aux outliers. Dans notre cas, comme DyClee est capable de détecter les outliers et de les éliminer, il est possible de fixer un ordre plus élevé afin d'essayer de fournir un seul hyper-paramètre, général pour le système, plutôt qu'un hyper-paramètre par place.

Il est aussi intéressant de remarquer que la dynamique continue du niveau d'eau h_1 pour les échantillons [0 :309] est très différente de la dynamique continue apprise à partir des échantillons [330 :370], malgré le fait que les deux ensembles appartiennent à la place p_1. Cette différence s'explique par la présence d'une phase transitoire d'initialisation, où le niveau d'eau, nul au départ, croît très rapidement. Au bout d'un certain temps, le système entre dans un régime permanent. Une place additionnelle p_0 va être ajoutée à l'ensemble P de places du modèle HtPN obtenu, à laquelle la phase d'initialisation de

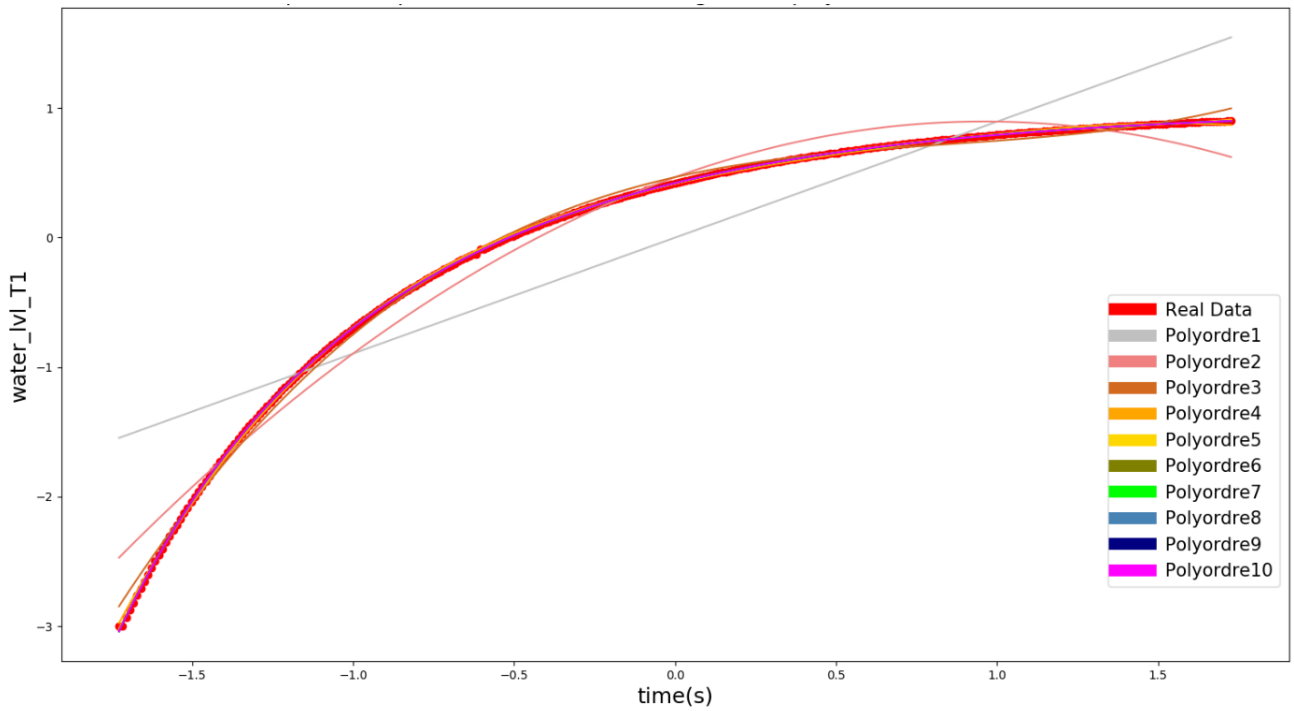


FIGURE 3.19 – Résultats de l'apprentissage pour différents ordres de régression pour les échantillons [0 : 309]

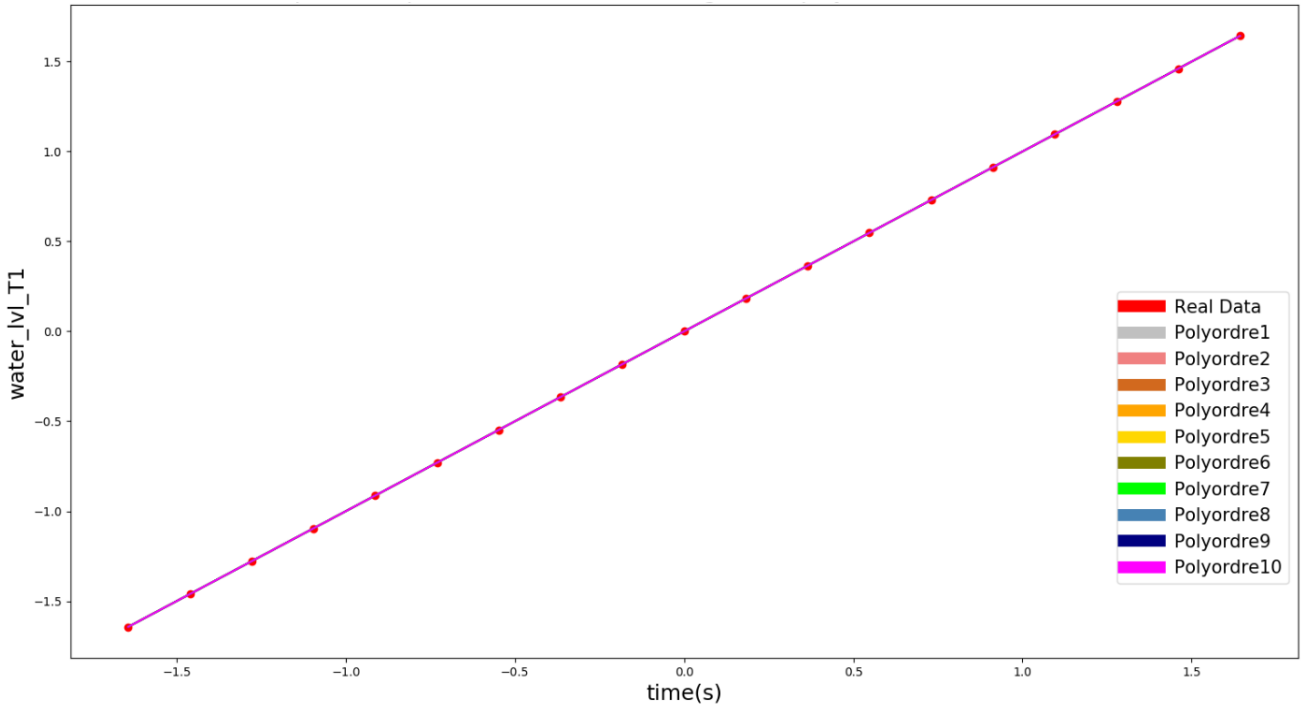


FIGURE 3.20 – Résultats de l'apprentissage pour différents ordres de régression pour les échantillons [310 : 329]

notre système est associée et le mécanisme de ré-apprentissage a été appliqué à l'ensemble des jeux d'échantillons pour le modèle RP, présentant le meilleur score R^2 pour un temps relativement court. La courbe de correspondance pour l'ensemble des échantillons, obtenu en concaténant les modèles appris, est illustrée sur la figure 3.22. Le score R^2 global

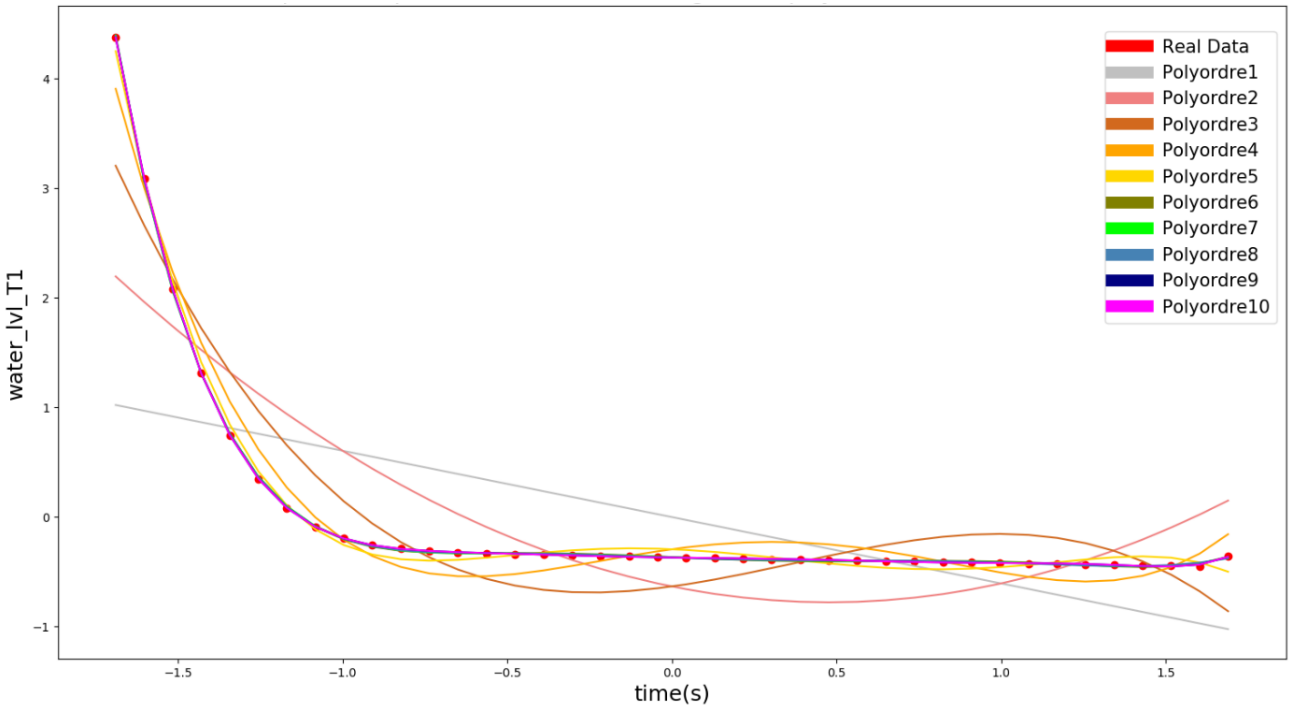


FIGURE 3.21 – Résultats de l'apprentissage pour différents ordres de régression pour les échantillons [330 : 370]

obtenu est de 0.998, ce qui montre une très bonne correspondance entre les modèles appris pour chaque place et les données réelles après ré-apprentissage. Une transition t_0 entre p_0 et p_1 va aussi être créée. Il faut par contre réfléchir aux conditions de tir portées par la transition t_0, conditions qui seront obtenues par le processus d'apprentissage des conditions numériques définis en section 3.3.4.3.

Un processus similaire pourrait être appliqué pour apprendre l'ensemble des dynamiques de dégradation D en se basant sur des jeux de données allant jusqu'à la défaillance du système.

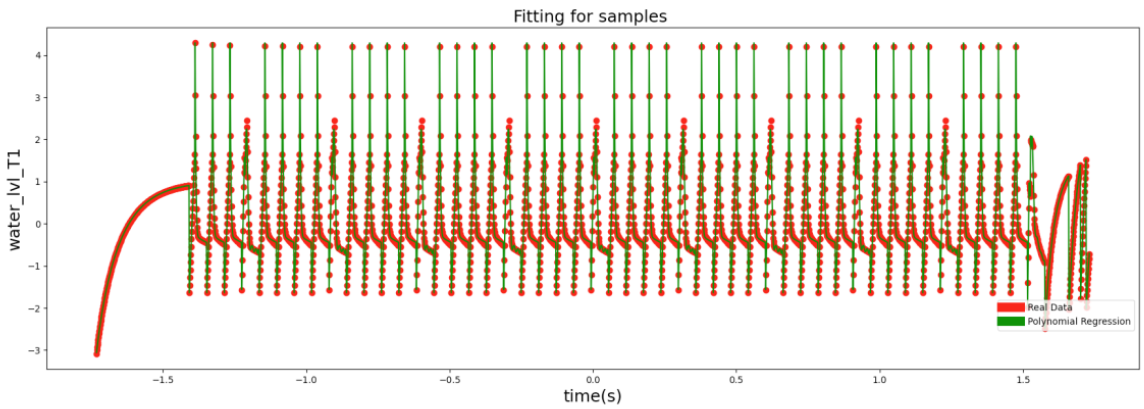


FIGURE 3.22 – Résultats de l'apprentissage du modèle de RP pour l'ensemble des échantillons

TABLEAU 3.2 – Comparaison du score R^2 et du temps de calcul requis pour différents ordre de régression polynomiale

Echantillons	0 : 309		310 : 329		330 : 370	
Ordre RP	R^2	Temps	R^2	Temps	R^2	Temps
1	0.803	0.6ms	1.0	0.6ms	0.366	0.6ms
2	0.977	0.5ms	1.0	0.5ms	0.686	0.6ms
3	0.998	0.5ms	1.0	0.5ms	0.883	0.5ms
4	0.999	0.5ms	1.0	0.6ms	0.973	0.5ms
5	0.999	0.5ms	1.0	0.5ms	0.996	0.5ms
6	0.999	0.5ms	1.0	0.5ms	0.999	0.5ms
7	0.999	0.5ms	1.0	0.5ms	0.999	0.5ms
8	0.999	0.5ms	1.0	0.5ms	0.999	0.5ms
9	0.999	0.5ms	1.0	0.5ms	0.999	0.7ms
10	0.999	0.5ms	1.0	0.5ms	0.999	0.8ms

3.5 Conclusion

3.5.1 Résumé du chapitre

Ce chapitre présente une méthode d'apprentissage de modèles hybrides pour de la gestion de santé. Le formalisme des HtPN a été choisi pour effectuer la gestion de santé sous incertitudes, mais le processus d'apprentissage présenté dans ce chapitre peut s'appliquer à d'autres formalismes qui respectent nos hypothèses, comme les réseaux de Petri binaires ou les automates hybrides. L'état de l'art sur les méthodes d'apprentissage de modèle hybride montre qu'un apprentissage structurel par clustering, associé à un apprentissage des dynamiques continues est le plus pertinent.

Le processus d'apprentissage comporte 4 étapes. La première étape du processus consiste à traiter les données d'entrée. Pour chaque élément observé, une nouvelle colonne correspondante est créée. Si l'événement correspondant à la colonne est présent sur un échantillon, la colonne a la valeur 1, 0 sinon. Ensuite, chaque caractéristique des échantillons est normalisée afin d'équilibrer son importance dans la partie apprentissage. La deuxième étape du processus se concentre sur l'obtention de la structure globale HtPN, composée d'un ensemble de places P, d'un ensemble de transitions T et d'une partie des éléments de Pre et $Post$, représentant les ensembles de conditions et assignations du modèle HtPN. Cette première étape se base sur une méthode de clustering développée dans l'équipe DISCO nommée DyClee. La troisième étape consiste à apprendre les dynamiques continues associées à chaque place obtenue lors de l'apprentissage de la structure. Les dynamiques continues C sont apprises en utilisant les algorithmes SVR, RFR et RP, qui sont tous trois basés sur des techniques de régression. Un mécanisme de ré-apprentissage en cas de non-correspondance entre deux jeux de données associés à une même place a été mis en place. Ce processus entraîne la création de places additionnelles et permet notamment de prendre en compte le régime transitoire du système. Le processus proposé a été appliqué sur l'exemple académique des trois cuves d'eau. En utilisant des données simulées par le logiciel HeMU pour le système des cuves d'eau, un modèle hybride, sous la forme d'un HtPN, a été obtenu via ce processus et comparé au modèle *a priori* connu. Les résultats obtenus sont satisfaisants étant donné que le modèle appris correspond au

modèle apprenable à partir des données disponibles.

3.5.2 Limitations et discussion

Dans cette thèse, une base solide pour l'apprentissage de modèle sous le formalisme des HtPN a été obtenue. Les ensembles P et T ont été appris, et une partie des ensembles Pre et $Post$. De plus, une méthode pour l'obtention des conditions numériques (de manière à compléter les éléments de Pre) a été établie.

Deux des trois algorithmes de régression utilisés, le SVR et le RFR, ne sont pas capables d'extrapoler. Cette incapacité limite l'utilisation de l'apprentissage à une utilisation hors-ligne. L'extrapolation est la capacité pour un modèle entraîné pour des valeurs de x comprises entre [0 ;1000], par exemple, de donner une prédiction pour $x = 1500$. Pour les utiliser, il faudrait donc avoir des jeux de données d'apprentissage qui couvrent l'ensemble du domaine d'évolution du système. Ce n'est pas le cas de l'algorithme de RP, qui est de son côté capable d'extrapoler. Ce dernier nécessite cependant un prétraitement précis des données, car il est très sensible à de potentielles données erronées. L'application de la méthode d'apprentissage présentée dans le chapitre 5 va se focaliser sur l'utilisation de la RP afin de tirer parti de sa capacité d'extrapolation sur cette application réelle.

De plus, la quantité de données nécessaires à l'obtention d'un modèle satisfaisant n'est actuellement pas connue. Il serait intéressant d'effectuer des travaux sur ce domaine.

Enfin, le modèle appris ne correspond qu'à l'évolution d'un seul jeton dans le HtPN. L'identification et l'apprentissage d'un modèle pour un système multicomposants incluant du parallélisme à travers des données nécessite de plus amples travaux.

Une fois un modèle obtenu par apprentissage, il est possible d'appliquer des fonctions de gestion de santé afin de surveiller l'état de notre système. Ces fonctions de santé font l'objet du chapitre suivant.

Chapitre 4

Méthode de suivi de santé

Sommaire

4.1 Introduction 80
4.2 État de l'art sur les méthodes de diagnostic 80
4.2.1 Approches discrètes 81
4.2.2 Approches continues 83
4.2.3 Approches hybrides 84
4.2.4 Synthèse 85
4.3 Méthode de diagnostic avancé 86
4.3.1 Vue globale 86
4.3.2 Concept de métaplace 86
4.3.3 Génération du diagnostiqueur 88
4.3.4 Processus de diagnostic 91
4.4 Application de la méthode de diagnostic à différents systèmes 102
4.4.1 Système à événements discrets 102
4.4.2 Système continu 104
4.4.3 Système hybride 107
4.5 Conclusion 109
4.5.1 Résumé du chapitre 109
4.5.2 Limitations et discussion 109

4.1 Introduction

En se basant sur le formalisme présenté lors du chapitre 2, une méthode de diagnostic avancé a été proposée et implémentée. Le diagnostic avancé consiste à suivre l'état de santé du système, dont la définition est rappelée ici.

Définition 34 (État de santé dans un HtPN)**.** *Dans un modèle sous le formalisme des HtPN, l'état de santé est représenté à travers un jeton présent dans une place et l'information qu'il porte : sa configuration, son état continu et son statut. Ce concept est lié à la fois à l'occurrence de fautes (événements discrets non observables), et à l'état de dégradation, pouvant représenter le vieillissement du système.*

La méthode de diagnostic avancé que nous proposons peut s'appliquer sur différents types de systèmes (discret, continu ou hybride). Cette méthode de diagnostic basée sur le formalisme des HtPN est donc définie pour surveiller l'état de santé du système en prenant en compte des incertitudes. Ces incertitudes, découlant de la complexité du système, peuvent concerner le modèle (erreur de conception ou sur les paramètres des dynamiques) ou les observations (fausse observation, observation manquée ou bruit). Les incertitudes liées au modèle sont prises en compte par le formalisme HtPN et la présence du bruit sur les fonctions d'évolution. Les incertitudes liées aux observations seront quant à elles prises en compte par la méthode de diagnostic définie dans ce chapitre. Le diagnostic obtenu par cette méthode est capable de donner à l'utilisateur des informations détaillées concernant l'hypothèse la plus probable sur l'état de santé du système, représentée par un jeton dans une place du HtPN. Ces informations détaillées sont : les événements vus par le système (à travers la configuration d'un jeton), les valeurs des variables continues (à travers l'état continu du jeton) et les valeurs des variables de dégradation (à travers le statut du jeton). La fiabilité de cette hypothèse est calculée via deux scores différents, qui représentent la similitude entre la configuration du jeton et les événements observés par le système et entre l'état continu du jeton et la valeur des variables continues observées. La méthode de diagnostic avancé proposée dans ce manuscrit se base uniquement sur les informations discrètes et continues. La dégradation, non observable, est estimée tout au long du processus mais n'est actuellement pas utilisée par la méthode de diagnostic. Elle pourra être utilisée dans de futurs travaux pour pondérer les hypothèses de diagnostic ou encore pour implémenter une fonction de pronostic.

Ce chapitre s'organise de la façon suivante. Tout d'abord, la section 4.2 donne un état de l'art de différentes méthodes de diagnostic. La section 4.3 présente la méthode de diagnostic avancé proposée qui permet de gérer les incertitudes sur les observations continues et discrètes du système. Ensuite, la section 4.4 illustre la méthode proposée sur trois systèmes différents : un système à événements discrets, un système continu et un système hybride. Cette illustration montre que la méthode de diagnostic proposée est efficace pour surveiller l'état de santé de différents types de systèmes, tout en tenant compte d'incertitudes sur le modèle et les observations. Enfin, la section 4.5 conclut ce chapitre et propose des perspectives quant à de futurs travaux.

4.2 État de l'art sur les méthodes de diagnostic

Pour effectuer du diagnostic de système, on distingue trois catégories de méthodes. La première catégorie comprend les méthodes basées sur la connaissance. Ces

méthodes utilisent des relations entre les symptômes, les défaillances et les fautes (Venkatasubramanian et al. 2003). Ces relations sont généralement obtenues par des experts lors de la phase de conception du système. Parmi les méthodes basées sur la connaissance, on peut citer les approches à base de règles (rule-based) ou à base de cas (case-based) (Buchanan et al. 1984, Jackson 1986).

La deuxième catégorie comprend les méthodes basées sur les données. Ces méthodes consistent à utiliser uniquement les informations obtenues via différents capteurs présents sur le système ou son environnement (Sampath et al. 1995, Sampath et al. 1996). L'idée de ces méthodes est d'associer des ensembles de mesures à certains modes de fonctionnement du système, via des techniques d'apprentissage. Le modèle du système, obtenu de manière empirique, ne provient pas d'une spécification physique ou structurelle du système.

La troisième catégorie comprend les méthodes basées sur les modèles. Ces méthodes se basent sur une comparaison entre le comportement observé du système et le comportement prédit par un modèle. En cas d'incohérence entre ces deux comportements, une faute est détectée et un raisonnement est effectué pour localiser l'occurrence de la faute. Si le modèle disponible inclut des comportements en présence des fautes du système, il est aussi possible d'identifier l'origine de la faute (isolation). C'est sur cette catégorie, particulièrement adaptée dans le cas où un modèle du système peut être obtenu, que va se focaliser cet état de l'art.

Dans le diagnostic à base de modèles, on distingue également différents types d'approches, selon le type de système : à événements discrets, continus ou hybrides.

4.2.1 Approches discrètes

Concernant le diagnostic de systèmes à événements discrets, l'approche la plus connue est celle du diagnostiqueur, introduite dans Sampath et al. 1995 Sampath et al. 1996, qui se base sur les automates (voir définition 3 page 8).

Un diagnostiqueur est un automate obtenu à partir d'un modèle automate contenant des événements non observables, et dont les événements déclenchant les transitions sont les événements observables du système. L'état du diagnostiqueur donne l'ensemble des hypothèses sur l'état courant du système ainsi qu'une information sur les fautes qui peuvent expliquer le comportement observé. Un diagnostiqueur permet à la fois de réaliser les étapes de détection et d'identification des fautes. Une faute est détectée lorsque qu'il n'existe plus d'ambiguïté entre l'hypothèse du comportement nominal et celle d'un comportement de faute. Dans le diagnostiqueur, cette absence d'ambiguïté correspond à l'obtention d'une seule hypothèse correspondant à l'occurrence de la faute dans le système.

Un modèle automate et son diagnostiqueur associé sont représentés sur les figures 4.1(a) et (b) respectivement. Des événements observables σ_i, $i = 1...6$, non observables σ_{uo} et un événement de faute σ_{f1} sont représentés sur le modèle automate. Chaque état du diagnostiqueur correspond à un diagnostic et donne l'ensemble des hypothèses sur l'état courant du modèle ainsi qu'une information sur l'occurrence de l'événement de faute. L'étiquette N signifie que le mode est nominal et F_1 que la faute f_1 a eu lieu. À l'état initial, le diagnostiqueur est dans l'état 1, labellisé N car aucune faute n'a encore eu lieu. Quand σ_1 est observé, le modèle est soit dans l'état 7, correspondant à un comportement normal ($7N$), soit dans l'état 3, dans lequel la faute f_1 a eu lieu ($3F_1$). Il existe donc une ambiguïté entre le comportement normal et la possibilité de l'occurrence d'une

faute $(7N, 3F_1)$.

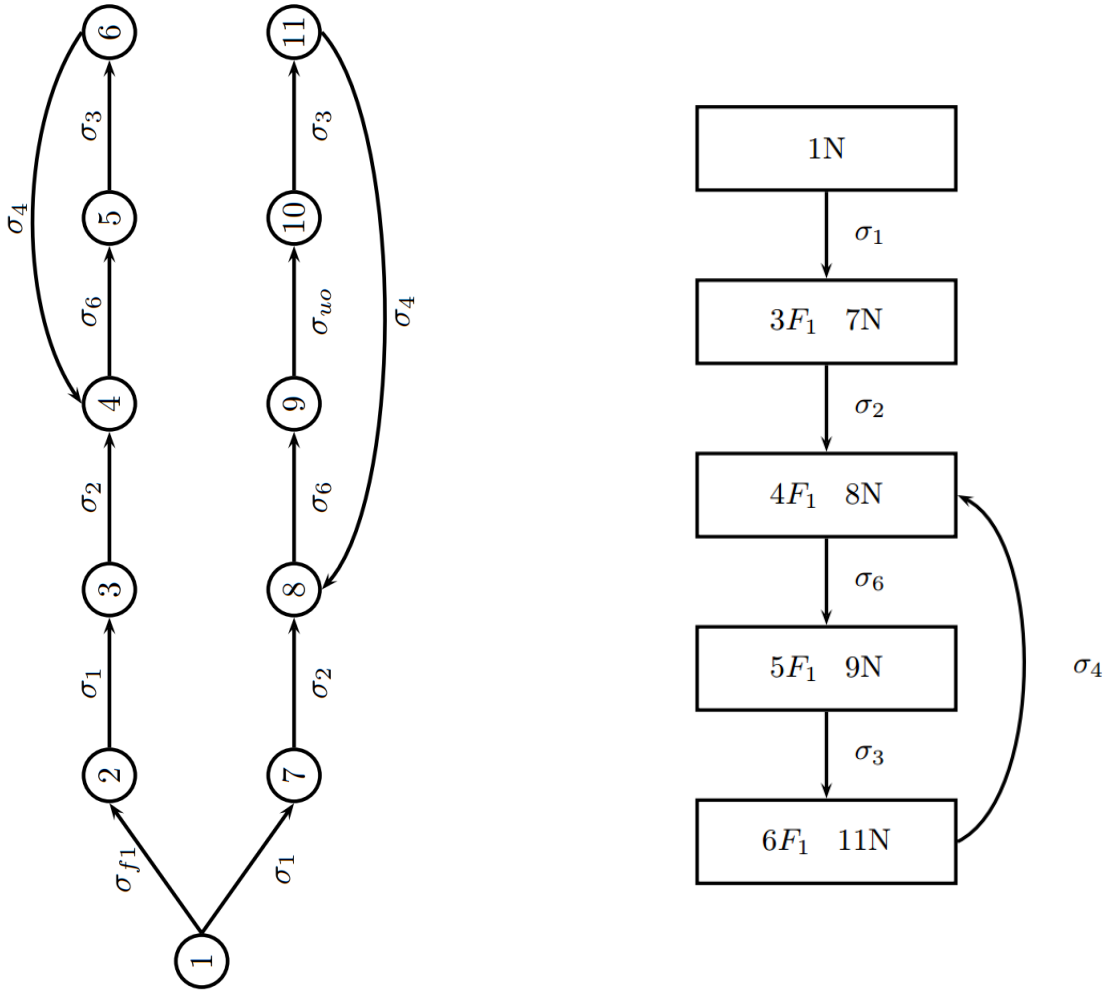


FIGURE 4.1 – Exemples d'un modèle automate (gauche, a) et de son diagnostiqueur (droite, b) (SOLDANI 2008)

Le diagnostiqueur dit "de Sampath" est beaucoup utilisé dans la littérature, mais est limité dans ses utilisations sur des systèmes réels complexes, notamment à cause de l'explosion combinatoire.

Les réseaux de Petri (voir définition 4) étant capables de représenter l'évolution des SED de manière plus compacte et intuitive que les automates, l'extension du diagnostiqueur aux SED modélisés par les réseaux de Petri a fait l'objet de plusieurs travaux (BOUBOUR et al. 1997, SOLDANI et al. 2007, CABASINO et al. 2013). Néanmoins, ces travaux ne gèrent pas les incertitudes liées à la connaissance sur le système ou aux observations discrètes. Dans les travaux de GIUA et al. 2002, une approche à base d'estimateur est utilisée sur des réseaux de Petri dont le marquage est partiellement observable. L'idée est d'estimer le véritable marquage d'un réseau de Petri par un marquage calculé à partir des événements observés. Plusieurs types d'incertitude sont considérés, comme la connaissance partielle ou nulle du marquage initial, mais la structure du réseau est considérée comme connue et tous les tirages des transitions sont observables. Dans RU et al. 2009, les réseaux de Petri partiellement observés, *Partially Observed Petri Nets* (POPN) sont utilisés. Les réseaux de Petri partiellement observés sont transformés en des réseaux de

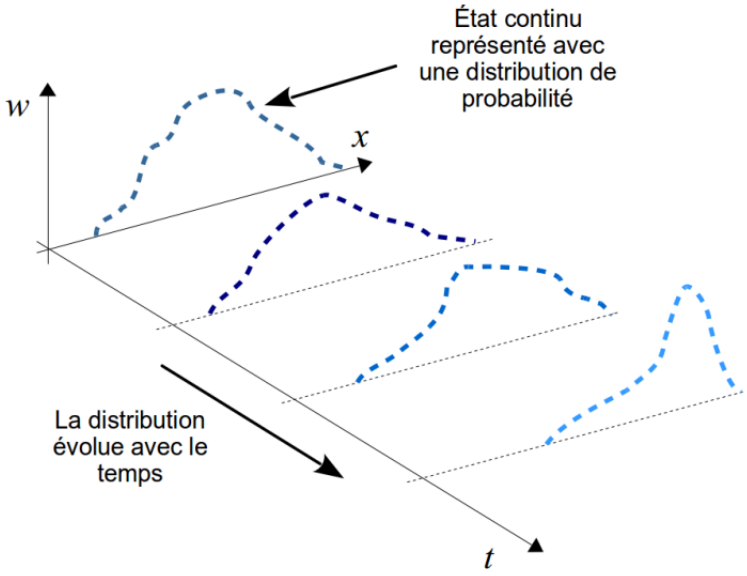


FIGURE 4.2 – Évolution d'une distribution de probabilité représentant l'état continu du système au cours du temps

Petri labellisés équivalents et un moniteur est construit pour diagnostiquer les fautes et délivrer une croyance (degrés de confiance) sur leurs occurrences. Cette approche ne considère que l'incertitude au niveau du résultat du diagnostic, mais pas les incertitudes liées au modèle ou encore aux observations des événements. En outre, ces différents travaux ne considèrent en aucun cas des aspects continus qui pourraient influer sur l'évolution du système.

4.2.2 Approches continues

Pour le diagnostic des systèmes continus, différentes techniques peuvent être employées, notamment les techniques à base d'observateurs et les techniques à base de redondances analytiques. Les techniques à base d'observateurs (ou estimateurs) consistent à estimer les valeurs des variables d'état à partir du modèle et d'un ensemble d'observations sur le système (DING 2014, STAROSWIECKI et al. 1991, HAMMOURI et al. 1999). Dans ces techniques, on utilise la fonction d'évolution du modèle continu (voir équation 1.1 page 9) pour calculer une estimation de l'état $\hat{x}$ qui est ensuite corrigée en fonction de la différence entre la sortie estimée $\hat{y}$, calculée avec la fonction de sortie, et le vecteur des sorties mesurées y. Un certain nombre de ces observateurs, comme les filtres de Kalman (ARULAMPALAM et al. 2002), les filtres de Kalman étendus (JULIER et al. 1997) ou les filtres particulaires (THRUN et al. 2001, VAN DER MERWE et al. 2000), considèrent du bruit sur l'évolution de l'état, mais aussi sur les observations. L'estimation de l'état est alors manipulée comme une distribution continue de probabilité, de croyance, ou de pertinence. L'évolution au cours du temps d'une telle estimation distribuée est illustrée dans la figure 4.2.

Dans les filtres particulaires, par exemple, la distribution de croyance se compose d'un ensemble fini de particules, représentant chacune une valeur possible de l'état du système. Durant une étape de prédiction bayésienne, chaque particule est mise à jour avec la fonction d'évolution du modèle à laquelle est ajouté un bruit aléatoire. Une étape de

correction associe alors un poids à chaque particule en fonction de l'erreur entre sa sortie estimée (calculée avec l'équation de sortie du modèle) et les observations sur le système, puis utilise ce poids pour rééchantillonner l'ensemble des particules. Le rééchantillonnage consiste à sélectionner (une ou plusieurs fois) les particules ayant les poids les plus élevés pour former la nouvelle distribution de particules estimant l'état courant du système.

Les techniques basées sur les relations de redondance analytique, *Analytical Redundancy Relations* (ARR) ont également été largement appliquées dans le cadre du diagnostic de systèmes continus (Cordier et al. 2004). Une relation de redondance analytique est une équation dans laquelle toutes les variables inconnues ont été remplacées par des expressions ne faisant intervenir que des variables observées. Ces méthodes se basent sur la génération de résidus du modèle pour déterminer les valeurs pertinentes à mesurer. Un résidu correspond à une différence entre le comportement simulé par le modèle et le comportement observé du système. Nous utilisons la définition suivante, issue de Gertler 2017.

Définition 35 (Résidu)**.** *Un résidu est associé à une différence résultant de la comparaison de mesures de capteurs à des valeurs calculées analytiquement de la variable considérée dans le système modélisé.*

Dans les approches de détection de faute à base de modèles, un résidu est un signal temporel, fonction des entrées et des sorties du processus, indépendant du point de fonctionnement de celui-ci. En l'absence de faute, ce résidu est statistiquement nul. Lors de l'apparition d'une faute, son amplitude évolue de manière significative. L'approche la plus classique pour calculer / générer ces résidus est celle dite de l'espace de parité (Staroswiecki et al. 1991). Les relations de parité utilisent la redondance directe au moyen de relations algébriques statiques liant les différents signaux ou la redondance temporelle issue de l'utilisation de relations dynamiques. Les méthodes par espace de parité consistent à projeter les estimations des résidus dans un espace binaire afin de reconnaître ce qu'on appelle les signatures des fautes. Les signatures de fautes sont des vecteurs d'indicateurs booléens représentatifs de chaque mode de fonctionnement du système. Ces travaux purement continus prennent en compte le bruit sur les variables observées mais ne tiennent pas compte des événements discrets et des incertitudes qui leur sont associées.

4.2.3 Approches hybrides

La plupart des méthodes de diagnostic appliquées aux systèmes hybrides sont des méthodes combinant des approches appliquées aux systèmes à événements discrets et des approches appliquées aux systèmes continus. Dans Bayoudh et al. 2008, par exemple, les auteurs modélisent le système avec un automate hybride. Le système hybride est décrit comme un système multimode (multi-mode system) dans lequel à chaque mode est associée une dynamique continue particulière. Ces travaux exploitent ensuite les relations de redondance analytique des modèles continus et l'approche par espace de parité pour générer de nouveaux événements observables dans le diagnostiqueur à événements discrets. L'aspect continu permet donc d'enrichir le diagnostic de SED. Une idée similaire se retrouve dans le travail de Narasimhan et al. 2007, où la détection et l'isolation de fautes se fait via l'utilisation de résidus et de signatures de fautes. Cette méthode est appliquée aux graphes de relations hybrides (*Hybrid Bond Graphs*), une technique de modélisation utilisant les échanges d'énergie présents dans le système.

Les auteurs de Lesire et al. 2005 combinent un modèle à événements discrets (réseau de Petri) et un modèle continu (équations dynamiques) dans une extension des réseaux de Petri hybrides appelée les réseaux de Petri particulaires, *Particle Petri Nets* (PPN). Ils proposent de distribuer le marquage rationnel des places continues en un ensemble de particules. Les jetons des places discrètes (les configurations) et ces particules sont ensuite utilisés dans un mécanisme de surveillance combinant le tirage possibiliste (Cardoso et al. 1999) (duplication de jeton ou pseudo-tirage) avec un filtre particulaire pour traiter toutes les incertitudes relatives au système et aux observations discrètes et continues. Ces travaux sont orientés vers la surveillance de mission et non sur la surveillance de santé. Néanmoins, l'idée de pseudo-tirage nous paraît prometteuse vis-à-vis de la gestion de certaines incertitudes, nous allons donc essayer de la conserver. Le formalisme des réseaux de Petri particulaires modifiés, *Modified Particle Petri Nets* (MPPN), est proposé dans Zouaghi et al. 2011 comme une extension des PPN. L'avantage principal des MPPN est qu'ils proposent d'utiliser des transitions associées à des conditions portant à la fois sur les valeurs des configurations et des particules. L'application est encore essentiellement orientée vers la surveillance de mission et non sur la surveillance de santé. Les différents états de santé du système ne sont pas considérés. De plus, il n'y pas de correspondance avec l'objet diagnostiqueur défini dans la littérature et le problème d'ambiguïté n'est pas abordé. Ces problèmes sont abordés dans le travail de Gaudel, Chanthery, Ribot et Daigle 2018, où une méthode de diagnostic est définie sur le formalisme des HPPN. Dans cette méthode, applicable uniquement aux systèmes hybrides, un diagnostiqueur est construit en se basant sur le modèle du système. La méthode utilise les idées du tirage possibiliste et du filtrage particulaire pour effectuer le diagnostic du système.

Un framework générique pour le diagnostic de tout type de système est proposé dans Ribot, Pencolé et al. 2013 et se base sur une combinaison des modèles structurel et fonctionnel du système. Cependant, les incertitudes relatives à la modélisation et aux observations ne sont pas considérées et ce travail ne propose pas de méthode de diagnostic générique.

4.2.4 Synthèse

La plupart des méthodes de diagnostic abordées dans cet état de l'art ne sont pas applicables au formalisme que nous avons. Les méthodes discrètes et continues, comme leurs noms l'indiquent, sont limitées aux systèmes à événements discrets et continus. Les méthodes hybrides étudiées ont des notions qui sont très intéressantes, notamment les notions de tirage possibiliste et du filtrage particulaire que nous allons réutiliser. Nous allons donc, dans ce travail, enrichir la méthode de diagnostic établie pour les HPPN, présentée dans le travail de Gaudel, Chanthery, Ribot et Daigle 2018. Cette méthode est en effet celle qui se rapproche le plus de nos besoins en termes de gestion d'incertitudes. Il faut cependant la faire évoluer afin qu'elle puisse s'appliquer au formalisme nouvellement établi.

4.3 Méthode de diagnostic avancé

4.3.1 Vue globale

Le diagnostic avancé a pour but de suivre l'état de santé courant du système et d'estimer son état de dégradation.

La méthode de diagnostic avancé proposée est présentée sur la figure 4.3.

La génération du diagnostiqueur HtPN ($HtPN_\Delta$) à partir d'un modèle HtPN du système ($HtPN_\Phi$) s'effectue hors ligne. Bien que le diagnostiqueur soit construit à partir d'un modèle HtPN, la notion de parallélisme diffère entre le diagnostiqueur ($HtPN_\Delta$) basé HtPN et le modèle HtPN ($HtPN_\Phi$). Dans le modèle HtPN, le parallélisme est utilisé pour représenter différents composants évoluant simultanément. Dans le diagnostiqueur $HtPN_\Delta$, le parallélisme est utilisé pour représenter différentes *hypothèses* sur l'état de santé du système.

Définition 36 (Hypothèse de diagnostic)**.** *Chaque jeton présent dans le diagnostiqueur représente une hypothèse de diagnostic, c.-à-d. une hypothèse sur l'état de santé du système.*

Le diagnostiqueur HtPN ne peut donc être construit qu'à partir d'un modèle HtPN ne contenant qu'un seul jeton, représentant la situation du système global.

En ligne, le processus de diagnostic au temps k s'effectue en deux étapes. La première étape du processus, la prédiction (section 4.3.4.1), a pour but de déterminer le marquage futur du diagnostiqueur HtPN. Elle est basée sur le marquage courant du diagnostiqueur $\mathbb{M}_{\Delta,k}$ et sur le concept de pseudo-tirage. Cette phase de prédiction va émettre des hypothèses sur l'évolution future du système sous la forme d'un marquage futur estimé du diagnostiqueur $\hat{\mathbb{M}}_{\Delta,k+1}$. Ces hypothèses seront ensuite fournies à la deuxième étape, la correction (section 4.3.4.2), qui, en fonction des observations continues et discrètes O_{k+1} disponibles, va en conserver certaines ou en réfuter. Les hypothèses conservées correspondent au marquage du diagnostiqueur $\mathbb{M}_{\Delta,k+1}$ et représentent le diagnostic du système au temps $k+1$.

La méthode de diagnostic avancé proposée est basée sur l'hypothèse suivante :

Hypothèse 3. *Il est supposé qu'entre un temps k et un temps $k+1$ donnés, un seul événement peut survenir.*

4.3.2 Concept de métaplace

Pour représenter les incertitudes entre deux hypothèses de diagnostic, comme notamment l'incertitude entre $3F_1$ et $7N$ sur le diagnostiqueur de la figure 4.1, le concept de métaplace, une extension du concept de place, a été créé pour les HtPN.

Une métaplace est un objet regroupant toutes les places ayant des dynamiques continues identiques et liées entre elles par des transitions non observables.

Définition 37 (Transition non observable)**.** *Une transition non observable t est une transition dont $Pre(t)$ porte uniquement sur des variables continues ou événements discrets non observables dans un HtPN et dont le tir n'implique pas de modifications de variables continues observables ou l'émission d'événement observable.*

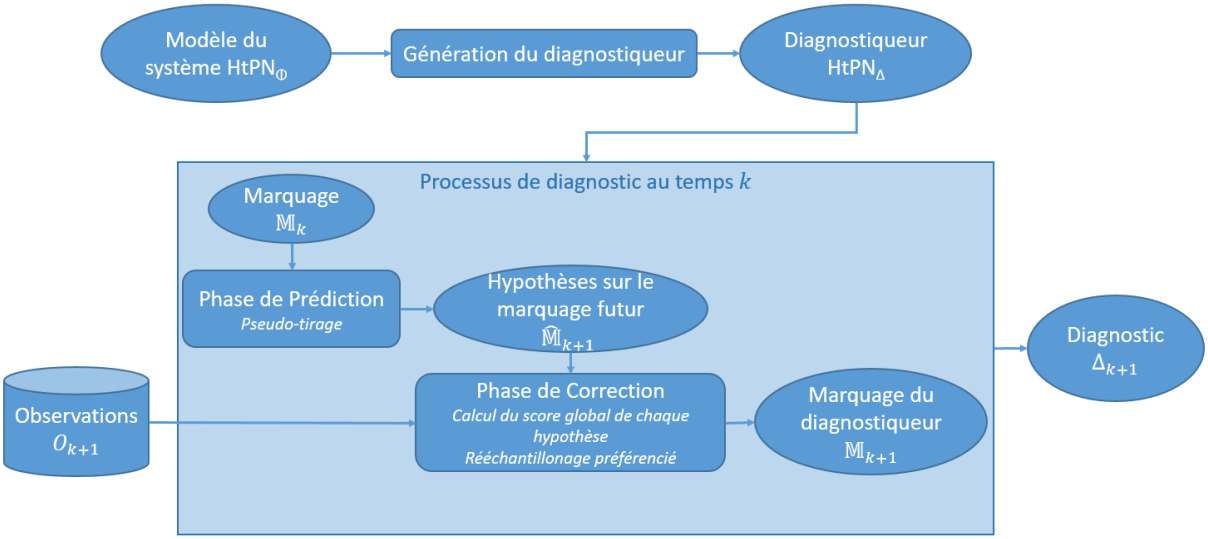


FIGURE 4.3 – Vue globale de la méthode de diagnostic avancé

Une métaplace va donc représenter les incertitudes entre les places ayant les mêmes dynamiques continues C_p, et séparées par une transition non observable dans le diagnostiqueur.

Définition 38 (Métaplace). *Une métaplace est un ensemble* $MP =< mp, Active, Activation_{cond}, Desactivation_{cond} >$ *où :*

- $mp =< p_1, \ldots, p_{mp} >$ *est l'ensemble des places de la métaplace ;*
- *Active est un booléen qui vaut* 1 *si la métaplace est active et* 0 *si la métaplace ne l'est pas ;*
- $Activation_{cond}$ *est l'ensemble des conditions pour activer la métaplace. Cet ensemble est composé des événements observables portés par toutes les transitions dont la place amont est en dehors de la métaplace et la place de sortie appartient à la métaplace ;*
- $Desactivation_{cond}$ *est l'ensemble des conditions pour désactiver la métaplace. Cet ensemble est composé des événements observables portés par toutes les transitions dont la place amont appartient à la métaplace et la place de sortie est hors de la métaplace.*

$$\begin{aligned}
&\forall p_i = \begin{Bmatrix} C_{p_i} \\ D_{p_i} \end{Bmatrix}, p_j = \begin{Bmatrix} C_{p_j} \\ D_{p_j} \end{Bmatrix} s.t\ |p_i, p_j| \in P : \\
&\exists t \in T\ /\ (Pre(p_i, t) \neq \emptyset \cap Post(t, p_j) \neq \emptyset), \Omega^S_{p_i,t} = occ(v), \\
&(C_{p_i} = C_{p_j}) \wedge (v \in E_{uo}) \Rightarrow \exists\ mp \in MP / |p_i, p_j| \in mp
\end{aligned} \tag{4.1}$$

Définition 39 (Métaplace activée). *Une métaplace* MP *est dite activée si une des places de l'ensemble* mp *est marquée.*

Par exemple, sur la figure 4.4, le HtPN comporte 4 places et 5 transitions. Les événements o_1, o_2 et $o3$ sont des événements observables. L'événement f est non observable. Aucune dynamique continue n'est précisée. Pour gérer les incertitudes entre deux hypothèses de diagnostic, une métaplace $MP1$, comprenant les places $Nom2$ et $Deg1$ va

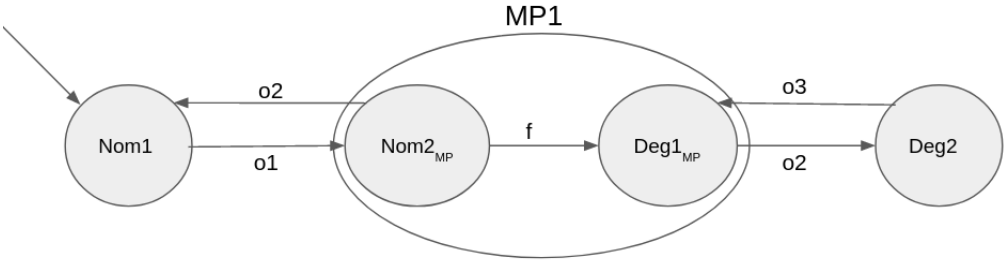


FIGURE 4.4 – Exemple d'un système à événements discrets transformé avec des métaplaces

être créée ($mp =< Nom2, Deg1 >$), étant donné que ces deux places ont la même dynamique continue (aucune) et sont liées par une transition non observable (celle portant l'événement non observable f). La métaplace $MP1$ est activée avec l'occurrence d'un des événements $\{o1, o3\}$. Elle est désactivée avec l'occurrence de l'événement $o2$.

La création des métaplaces va s'effectuer lors de la génération du diagnostiqueur, dont les différentes étapes sont détaillées ci-dessous.

4.3.3 Génération du diagnostiqueur

Soit $HtPN_\Phi < P_\Phi, T_\Phi, Pre_\Phi, Post_\Phi, \mathbb{M}_{0\Phi} >$ le modèle du système. Le diagnostiqueur basé sur ce modèle est défini comme : $HtPN_\Delta =< P_\Delta, T_\Delta, Pre_\Delta, Post_\Delta, \mathbb{M}_{0\Delta} >$. Ce diagnostiqueur est généré en quatre étapes décrites par la suite :

1. construction structurelle et création des dynamiques,
2. gestion des assignations,
3. gestion des conditions,
4. gestion des métaplaces,

La figure 4.5 présente un modèle HtPN (a) et son diagnostiqueur (b). Dans le modèle HtPN présenté, les événements ON et OFF sont observables et l'événement u_o est non observable. L'obtention du diagnostiqueur présenté sur la figure 4.5(b) est détaillée ci-après.

L'*étape 1 "construction structurelle et création des dynamiques"* consiste à faire une copie exacte du modèle HtPN du système, sauf pour les matrices Pre et $Post$, qui sont mises à leur valeur par défaut ($-$ pour les conditions et assignations et 1 pour les poids). L'application de cette étape sur le modèle de la figure 4.5(a) est visible sur la figure 4.6. Les espaces d'états discrets, continus et de dégradation, ainsi que les dynamiques continues et de dégradation du diagnostiqueur sont les mêmes que ceux du modèle :

$$P_\Delta = P_\Phi, T_\Delta = T_\Phi, E_\Delta = E_\Phi, X_\Delta = X_\Phi, \Gamma_\Delta = \Gamma_\Phi, C_\Delta = C_\Phi, D_\Delta = D_\Phi, \mathbb{M}_{0\Delta} = \mathbb{M}_{0\Phi}. \tag{4.2}$$

L'*étape 2 "gestion des assignations"* dépend de l'observabilité de l'événement vérifié par la condition $\Omega^S_{\Phi,p,t}$. Si l'événement considéré est observable, la valeur de $\Omega^S_{\Phi,p,t}$, la condition symbolique du modèle du système associée à $Pre(p,t)$, est reportée sur $\Omega^S_{\Delta,t,p}$, l'assignation symbolique du diagnostiqueur portée par $Post(t,p)$. Sinon, $\Omega^S_{\Delta,t,p}$ prend la valeur neutre $-$. Les assignations numériques et de dégradation du modèle sont quant à

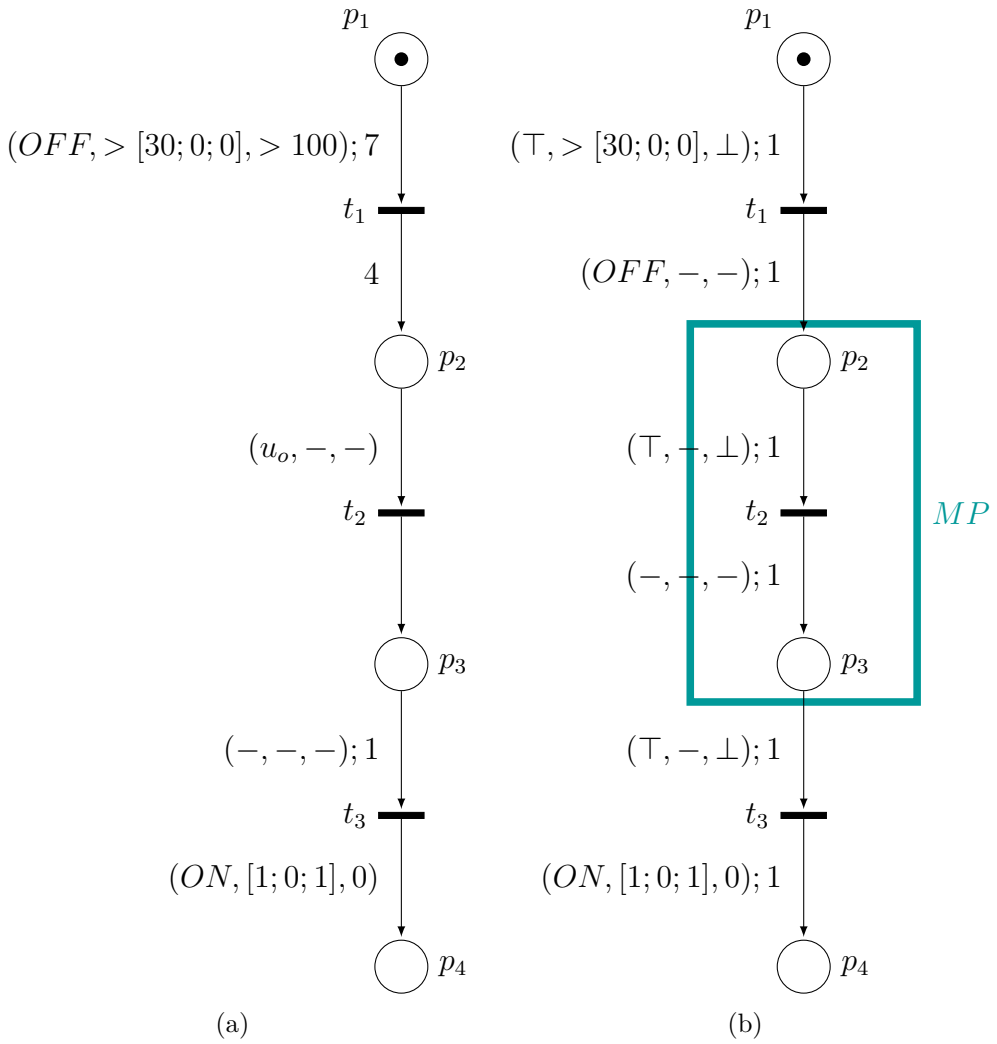


FIGURE 4.5 – Exemple d'HtPN (a) et de son diagnostiqueur (b)

elles copiées. L'application de cette étape sur le modèle de la figure 4.5(a) est visible sur la figure 4.7.

$$\begin{array}{l} Si\ \Omega^S_{\Phi,p,t} = occ(v), v \in E_o\ alors\ \Omega^S_{\Delta,t,p} \leftarrow \Omega^S_{\Phi,p,t};\quad \Omega^N_{\Delta,t,p} \leftarrow \Omega^N_{\Phi,t,p};\quad \Omega^D_{\Delta,t,p} \leftarrow \Omega^D_{\Phi,t,p} \\ Sinon\ \Omega^S_{\Delta,t,p} \leftarrow -;\quad \Omega^N_{\Delta,t,p} \leftarrow \Omega^N_{\Phi,t,p};\quad \Omega^D_{\Delta,t,p} \leftarrow \Omega^D_{\Phi,t,p} \end{array} \tag{4.3}$$

Ceci traduit l'idée que l'événement associé à la condition discrète doit être observé lors du tir de la transition. Ajouter cet événement à l'assignation discrète va, lors du tir de la transition t, ajouter l'événement à la configuration du jeton. Cet ajout sera pris en compte lors de l'étape de correction du processus de diagnostic.

Toutes les conditions discrètes associées aux arcs entrants des transitions vont ensuite être fixées à VRAI. Ceci est une première prise en compte des incertitudes au niveau du modèle. En effet, la structure discrète du modèle peut être en partie fausse ou incomplète et inclure des séquences d'événements soit incomplètes, soit impossibles. Au niveau des observations, un événement peut être observé sans avoir eu lieu (ceci est appelé une fausse observation), ou ne pas être observé mais avoir eu lieu (ceci est appelé une observation manquante). Ceci signifie donc que toutes les configurations des jetons qui sont présents dans le diagnostiqueur HTPN vont satisfaire les conditions discrètes. Le diagnos-

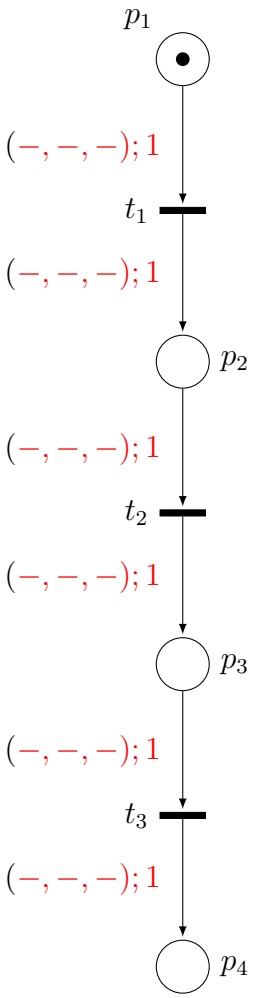


FIGURE 4.6 – Étape 1 de la génération du diagnostiqueur

tiqueur considère l'occurrence probable de tous les événements qui peuvent avoir lieu à partir du mode de fonctionnement courant représenté par le marquage du HtPN. Toutes les conditions de dégradation vont être fixées à FAUX, pour déconnecter l'évolution du marquage du diagnostiqueur de l'état de la dégradation. Le diagnostiqueur ne fait que suivre l'évolution de la dégradation (à travers le statut des jetons et les dynamiques de dégradation), mais ne se base pas dessus pour déterminer l'occurrence de fautes. Cette dégradation pourra être utilisée dans de futurs travaux, pour effectuer du pronostic, notamment. Les conditions continues vont rester, quant à elles, identiques à celles du modèle originel $HtPN_{\Phi}$.

L'*étape 3 "gestion des conditions"* consiste à fixer l'ensemble des conditions symboliques du diagnostiqueur à VRAI, l'ensemble des conditions de dégradation du diagnostiqueur à FAUX et à copier les conditions numériques du modèle. L'application de cette étape sur le modèle de la figure 4.5(a) est visible sur la figure 4.8. $\forall Pre(p,t)$,

$$\Omega^S_{\Delta,p,t} \leftarrow \top; \quad \Omega^N_{\Delta,p,t} \leftarrow \Omega^N_{\Phi,p,t}; \quad \Omega^D_{\Delta,p,t} \leftarrow \bot \tag{4.4}$$

Les assignations continues et de dégradation restent, elles aussi, inchangées lors de la construction du diagnostiqueur.

L'*étape 4 "gestion des métaplaces"* consiste en la création des *métaplaces*. Cette étape commence par l'identification de tous les ensembles de places ayant des dynamiques conti-

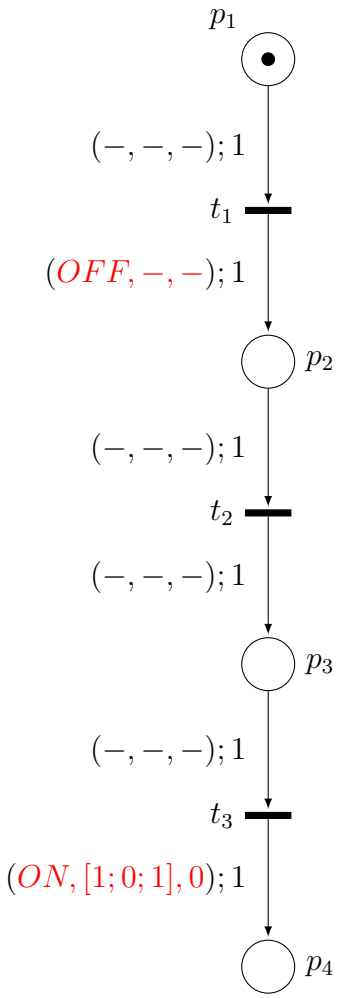


FIGURE 4.7 – Étape 2 de la génération du diagnostiqueur

nues similaires, liées par des transitions non observables. Ces ensembles permettent d'identifier les métaplaces. C'est le cas des places p_2 et p_3 de la figure 4.9. Ces deux places sont en effet liées par l'occurrence d'un événement non observable (u_o) et ont les mêmes dynamiques continues qui sont constantes.

Ensuite, les conditions d'activation et de désactivation pour chaque métaplace trouvée sont recherchées. Dans le cas de la figure 4.5, la métaplace MP a pour conditions d'activation $Activation_{cond} = OFF$ et de désactivation $Desactivation_{cond} = ON$.

4.3.4 Processus de diagnostic

Le marquage initial $\mathbb{M}_0$ du diagnostiqueur HtPN représente l'état de santé initial du système, ce qui équivaut à la distribution initiale des jetons dans le réseau. Chaque jeton h dans la distribution initiale a trois attributs : une configuration avec une valeur b_0^h, un vecteur d'état continu π_0^h et un vecteur d'état de dégradation γ_0^h. N_{H0} est le nombre de jetons présents lors de l'initialisation du modèle.

À partir du marquage initial $\mathbb{M}_0$, le marquage $\mathbb{M}_k$ du diagnostiqueur à un temps k va évoluer selon les observations $O_k = O_k^S \cup O_k^N$, où O^S et O^N représentent respectivement les ensembles d'observations discrètes et continues. Le marquage estimé du système au

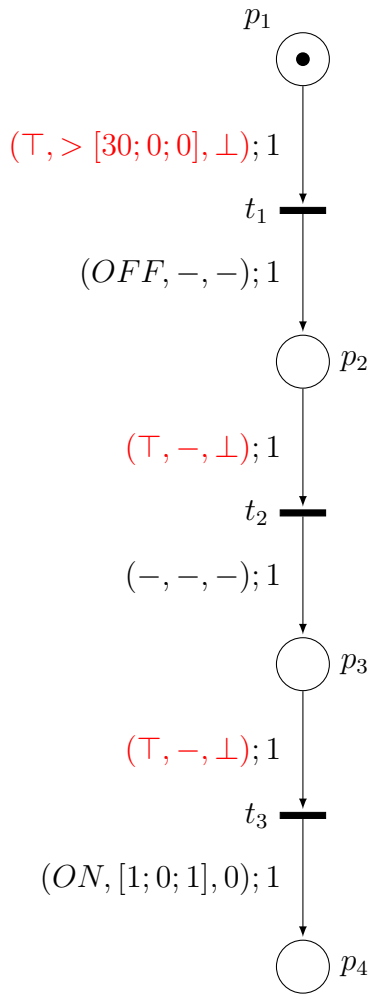


FIGURE 4.8 – Étape 3 de la génération du diagnostiqueur

temps k est noté $\hat{\mathbb{M}}_k$ et représente les différentes hypothèses de diagnostic sur l'état de santé du système au temps k.

L'évolution du marquage du diagnostiqueur HtPN se déroule en deux étapes, une étape de prédiction et une étape de correction, qui combinent du pseudo-tirage, l'évolution des attributs des jetons et un processus de rééchantillonnage de jetons. Dans le filtrage particulaire, le nombre de particules définit la précision du filtre. Dans notre cas, les particules sont représentées par les jetons : chaque jeton pourrait correspondre à une particule. Le but du processus de rééchantillonnage est d'éviter une explosion combinatoire en limitant le nombre de jetons présents à chaque itération de l'algorithme.

4.3.4.1 Prédiction

L'étape de prédiction du processus de diagnostic a pour but d'estimer le marquage futur du diagnostiqueur $\hat{\mathbb{M}}_{k+1|k}$. Cette étape est basée sur le pseudo-tirage des transitions activées (voir la définition 2.3.2.1 pour un rappel sur les transitions activées) et sur la mise à jour des attributs portés par les jetons. Comme on l'a indiqué dans l'état de l'art, le processus de pseudo-tirage a notamment été utilisé dans LESIRE et al. 2005 ou ZOUAGHI et al. 2011. Il crée de nouvelles hypothèses de diagnostic et permet de réduire la sensibilité du système aux fausses observations ; c'est une deuxième manière de gérer des incertitudes

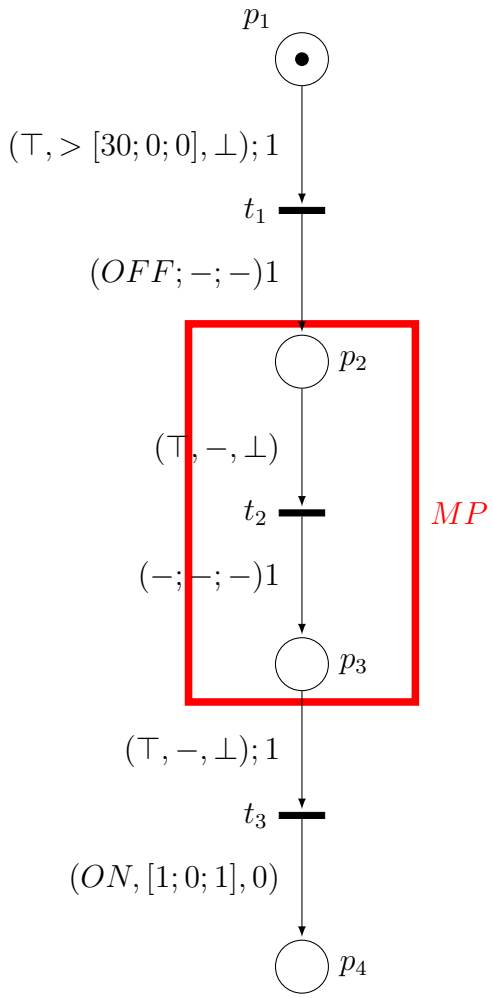


FIGURE 4.9 – Étape 4 de la génération du diagnostiqueur

sur la structure du modèle.

Le pseudo-tirage d'une transition va dupliquer des jetons : les jetons présents dans les places en amont de la transition ne sont pas tirés, mais sont *dupliqués* et leurs copies sont ensuite placées dans les places en aval de la transition. Ceci permet de tirer toutes les transitions activées même si certaines sont concurrentes, étant donné qu'il y aura toujours un jeton dans la place en amont. Ainsi, les différentes évolutions possibles du système peuvent être considérées. De ce fait, il n'est pas nécessaire de définir une priorité sur le tir des transitions dans le diagnostiqueur, étant donné que toutes les transitions activées vont être tirées.

Définition 40 (Pseudo-tirage d'une transition)**.** *Soit $t \in T$ une transition activée. Le pseudo-tirage de t à un temps k est défini comme : $\forall\ p \in P$ t.q. $Pre(p, t) \neq \emptyset$ et $\forall\ p' \in P$ t.q. $Post(t, p') \neq \emptyset$*

$$\begin{aligned} M_{k+1}(p) &= M_k(p), \\ M_{k+1}(p') &= M_k(p') + \rho_{t,p'} \end{aligned} \tag{4.5}$$

Lors du tir d'une transition t, l'ensemble d'événements b_k porté par la configuration δ_k d'un jeton va évoluer via l'assignation discrète portée par $Post(t, p)$. Les valeurs de l'état et du statut d'un jeton vont évoluer selon les dynamiques continues et de dégradation présentes dans la place où se situe le jeton. Il est possible pour certaines de ces équa-

tions d'être bruitées, comme défini dans l'équation 2.2, afin de représenter les incertitudes continues explicitées plus haut comme expliqué dans le chapitre 2. Des jetons présents dans la même place pourront ainsi avoir des vecteurs d'état différents suite à la présence de ces bruits.

4.3.4.2 Correction

Le principe de l'étape de correction est représenté sur la figure 4.10. Cette étape prend en entrée les hypothèses sur le marquage futur émises par l'étape de prédiction, les observations O_{k+1} et des hyper-paramètres choisis par l'utilisateur : α représentant le degré de confiance accordé aux informations discrètes par rapport aux informations continues, utilisé pour le calcul du score global et n_{suff} et n_{max}, utilisés par le processus de rééchantillonage. n_{suff} représente le nombre maximum de duplications pour une hypothèse considérée. n_{max} représente le nombre maximum de duplications effectué par le processus de rééchantillonage. Ce nombre correspond au nombre maximum d'hypothèses pouvant évoluer en parallèle dans le système. La première étape de la phase de correction est le calcul des scores de chaque hypothèse. Chaque hypothèse se voit attribuer un score discret et un score continu, lesquels permettent avec le paramètre α de calculer un score global. Les hypothèses sont ensuite fournies au processus de *rééchantillonage préférencié* explicité en détail dans le paragraphe 4.3.4.2. Une fois ce processus terminé, le marquage futur $\mathbb{M}_{k+1}$ est obtenu et correspond au diagnostic Δ_{k+1}.

L'étape de correction du processus de diagnostic va mettre à jour le marquage du diagnostiqueur $\mathbb{M}_{k+1}$ en fonction du marquage estimé $\hat{\mathbb{M}}_{k+1}$ et des observations O_{k+1}. Tout d'abord, quand une nouvelle observation discrète a lieu, le processus va d'abord vérifier si cette observation appartient aux ensembles d'activation ou de désactivation d'une métaplace. Si cette observation appartient à l'ensemble de conditions d'activation $Activation_{cond}$ d'une métaplace MP, alors *Active* va prendre la valeur 1 : la métaplace est activée. Si cette observation appartient à l'ensemble de conditions de désactivation $Desactivation_{cond}$ d'une métaplace MP alors *Active* va prendre la valeur 0 : la métaplace est alors désactivée. Une fois les métaplaces activées ou désactivées, le calcul des scores pour chaque hypothèse contenue dans le marquage estimé et sur le rééchantillonnage des jetons est effectué. Les scores de chaque hypothèse sont calculés via deux valeurs : Pr^S, qui représente la similarité entre la configuration d'un jeton et les événements discrets observés, et Pr^N, qui représente la similarité entre l'état du jeton et les valeurs continues observées.

Le score discret

Le *score discret* Pr^S d'un jeton est calculé en fonction de la distance entre la configuration du jeton et $O^S_{k+1} = \{O_\kappa | \kappa \leq k+1\}$, l'ensemble des observations discrètes jusqu'au temps k considéré. La distance entre les deux ensembles est le nombre d'éléments présents dans un ensemble, mais pas dans l'autre. Il est possible d'implémenter une fenêtre temporelle pour ne considérer que les événements ayant eu lieu à partir d'une date précise, afin de rajouter une notion d'oubli d'erreurs passées (observation manquée ou fausse observation). Le calcul du score discret Pr^S est décrit dans l'algorithme 4. Par simplicité de compréhension, la notion de fenêtre temporelle n'apparaît pas sur l'algorithme, mais le fonctionnement reste identique. Le score Pr^S obtenu sera compris dans l'intervalle $[0; 1]$, plus le score d'un jeton est proche de 0, plus la distance entre sa configuration et l'ensemble des événements

observés est grande. 1 est le score des jetons dont la configuration correspond parfaitement à l'ensemble des événements observés.

Algorithme 4 Obtention du score discret

Entrée: $O^S_{k+1}, tokens$
Sortie: Pr^S

1: ▷ Recherche de la distance maximale $dist_max$
2: $dist_max \leftarrow 0$
3: $s1 \leftarrow O^S_{k+1}$
4: **pour tout** $tk \in token$ **faire**
5: $\quad s2 \leftarrow configuration_{tk}$
6: $\quad$ **si** $dist(s1, s2) > dist_max$ **alors**
7: $\quad\quad dist_max \leftarrow dist(s1, s2)$
8: $\quad$ **fin si**
9: **fin pour**
10: ▷ Calcul de Pr^S
11: **pour tout** $tk \in tokens$ **faire**
12: $\quad s2 \leftarrow configuration_{tk}$
13: $\quad dist \leftarrow \frac{dist(s1,s2)}{dist_max+1}$
14: $\quad Pr^S_{tk} \leftarrow (1 - dist)$
15: **fin pour**

Considérons par exemple deux jetons $h1$ et $h2$. La configuration de $h1$ est $\delta_{h1} = (ON, 100), (OFF, 150), (ON, 220)$. La configuration de $h2$ est $\delta_{h2} = (ON, 220)$. L'ensemble des observations discrètes est $O^S_{k+1} = (ON, 220)$. $dist_{max}$ est ici la distance entre la configuration δ_{h1} et O^S_{k+1} et vaut 2 (trois événements sont présents dans la configuration contre un seul dans les observations). Il est donc possible de calculer les scores Pr^S pour chaque jeton. $dist_{h1} = (2/3)$ donc $Pr^S_{h1} = 1/3$. $dist_{h2} = (0/3)$ donc $Pr^S_{h2} = 1$. $h2$ aura donc un score discret de 1 et $h1$ de 1/3, ce qui reflète leur correspondance avec les observations O^S_{k+1}.

Le score continu

Le *score continu* Pr^N est calculé en fonction de la distance entre l'état continu du jeton et les observations numériques O^N_{k+1}. La distance entre l'état continu d'un jeton et les observations continues est calculée en faisant la norme 1 de la différence entre les deux. Le calcul du score continu Pr^C est donné dans l'algorithme 5. Le score continu est compris dans l'intervalle $[0; 1]$, 0 étant le score des jetons présentant la plus grande différence entre leur état estimé et les observations continues et 1 le score des jetons dont l'état continu estimé correspond parfaitement aux observations continues.

Score global

Le score d'une hypothèse (c.-à-d. d'un jeton) $h_k =< \delta_k, \pi_k, \gamma_k >$ à un temps k est calculé à partir de son score discret et son score continu :

$$Score(h_k) = \alpha \times Pr^S(\delta_k) + (1 - \alpha) \times Pr^C(\pi_k), \tag{4.6}$$

où $\alpha \in [0, 1]$ est un coefficient représentant l'indice de confiance dans la partie discrète par rapport à la partie continue. Plus α est proche de 1, plus l'information discrète va

Algorithme 5 Obtention du score continu

Entrée: yc, $tokens$
Sortie: Pr^C

1: ▷ Recherche de la différence maximale $diff_max$
2: ▷ Init : création d'un vecteurr $diff_max$ de la taille de yc
3: ▷ yc est le vecteur d'état continu
4: $diff_max \leftarrow zero(size(yc))$
5: **pour tout** $tk \in tokens$ **faire**
6: $\quad eyc \leftarrow Estim_{yc}(state_{tk})$
7: $\quad$ ▷ $eyc,\ yc\ and\ diff_max\ are\ vectors$
8: $\quad$ ▷ $yc[i]\ is\ the\ i-component\ of\ the\ vector\ yc$
9: $\quad$ **si** $(eyc - yc)[i] > diff_max[i]$ **alors**
10: $\quad\quad diff_max[i] \leftarrow (eyc - yc)[i]$
11: $\quad$ **fin si**
12: **fin pour**
13: ▷ Calcul de Pr^C
14: **pour tout** $tk \in tokens$ **faire**
15: $\quad eyc \leftarrow Estim_{yc}(state_{tk})$
16: $\quad diff \leftarrow \frac{eyc-yc}{diff_max}$
17: $\quad$ ▷ Norm2 de $diff$ pour obtenir une valeur dans $[0, 1]$
18: $\quad diff_norm \leftarrow norm2(diff)$
19: $\quad Pr^C_{tk} \leftarrow (1 - diff_norm)$
20: **fin pour**

peser dans le calcul du score global par rapport à l'information continue. Le score global d'une hypothèse est toujours compris entre 0 et 1.

Rééchantillonnage préférencié

L'ensemble des jetons est ensuite rééchantillonné selon le score global de chaque jeton. Pour cette étape, la notion de *clan* est introduite.

Définition 41 (Clan, score de clan)**.** *Tous les jetons ayant un score identique sont regroupés dans un clan. Le score du clan est le score des jetons qui appartiennent à ce clan.*

Tout d'abord, les jetons sont regroupés en *clans* selon leur score. Ceci permet d'éviter de sélectionner de trop nombreuses fois des jetons ayant un même score et ainsi de diversifier les hypothèses sélectionnées. Pour chaque score différent, un clan différent est créé.

Une normalisation des scores de clans est effectuée, de sorte que la somme de tous les scores de clan soit égale à 1. Cette normalisation est explicitée dans l'algorithme 6. Les nouveaux scores des clans seront donc répartis dans l'intervalle $[0; 1]$, ce qui représente la probabilité pour un clan d'être choisi lors du processus de rééchantillonnage. Lorsqu'un clan est choisi pour être rééchantillonné, un jeton aléatoire de ce clan est choisi pour être rééchantillonné.

Pour s'assurer que le clan avec le plus grand score n'est jamais ignoré lors du processus de rééchantillonnage, ce qui peut arriver dans le cas d'un très grand nombre de clans

Algorithme 6 Normalisation des scores des clans

Entrée: *scores_clans_old*, *clans* ▷ Scores non normalisés des clans
Sortie: *scores_clans_new*, *clans* ▷ Scores normalisés des clans

1: ▷ *sum_scores* est la somme des scores des clans
2: ▷ Initialisé à 0
3: $sum_scores \leftarrow 0$
4: **pour tout** $cl \in clans$ **faire**
5: ▷ *score_cl* est le score d'un clan *cl*
6: $sum_scores += score_cl$
7: **fin pour**
8: ▷ Calcul du score normalisé
9: **pour tout** $cl \in clans$ **faire**
10: $score_cl \leftarrow \frac{score_cl}{sum_scores}$
11: **fin pour**

différents, son score est modifié de manière à ce qu'il représente toujours au moins 50% de l'intervalle $[0; 1]$, c'est la partie préférentielle du rééchantillonnage préférencié. Cette modification pour le rééchantillonnage préférencié est explicitée dans l'algorithme 7. Une fois le plus grand score modifié, l'algorithme de normalisation des clans est de nouveau effectué, afin d'assurer que la somme des clans fasse bien 1.

Algorithme 7 Modification du score le plus élevé pour le rééchantillonnage préférencié

Entrée: *clans*
Sortie: *clans* pour le rééchantillonnage préférencié

1: ▷ Soit A le score le plus élevé
2: **si** $A < 0.5$ **alors**
3: $new_A \leftarrow (1 - A)$
4: $A \leftarrow new_A$
5: **fin si**

Une autre manière d'assurer la conservation du clan ayant le score le plus élevé serait par exemple d'utiliser la notion d'élitisme qu'on peut retrouver dans certains algorithmes évolutionnaires, comme dans Beyer et al. 2002.

Le processus de sélection des jetons peut commencer une fois les jetons répartis dans les clans et les scores établis. Une valeur aléatoire dans $[0, 1]$ est générée. Cette valeur va définir quel clan va être choisi par le processus de rééchantillonnage : les scores des clans vont être répartis et se partager l'intervalle [0 ;1]. Une fois qu'un clan a été rééchantillonné un nombre suffisant n_{suff} de fois (choisi par l'utilisateur), ce clan est retiré du processus afin d'assurer la sélection de clans différents. Cette génération aléatoire d'une valeur est répétée n_{max} fois. Le paramètre n_{max} représente le nombre maximum d'hypothèses disponibles pour suivre les différentes hypothèses. Le nombre total de jetons après le rééchantillonnage est toujours inférieur ou égal à n_{max}.

Le processus de rééchantillonnage est illustré sur la figure 4.11. Lors de l'initialisation, un seul jeton est dans le réseau, le jeton bleu. Lors de la phase de prédiction, le jeton bleu est pseudo-tiré, ce qui créé le jeton rose . L'exemple illustré est un système à événements discrets, sans dynamique continue. La valeur du paramètre α est donc mise à 1. Aucun événement n'a encore été observé. La configuration du jeton rose porte l'événement $e1$.

Les scores discret Pr^S et continu Pr^C sont calculés à partir des algorithmes 4 et 5. Les valeurs obtenues sont visibles sur la figure 4.11. Une fois les scores calculés, le processus de rééchantillonnage préférencié a lieu. Les jetons sont regroupés dans des clans selon leur score. Deux clans apparaissent ainsi : cl_1, basé sur le score du jeton bleu et cl_2, basé sur le score du jeton rose . Les scores des clans sont normalisés selon l'algorithme 6. Après la normalisation, le score de cl_1 est de $\frac{1}{1+0.5} = 0.66$ et celui de cl_2 est de $\frac{0.5}{1+0.5} = 0.33$. Le score du clan avec le meilleur score étant supérieur à 0.5, l'algorithme 7 n'est pas appliqué. On attribue donc tout l'intervalle $[0; 0.66]$ au clan cl_1 et l'intervalle $]0.66; 1]$ au clan cl_2. Les paramètres n_{max} et n_{suff} sont respectivement définis à 10 et 7. Comme $n_{max} = 10$, 10 valeurs aléatoires comprises dans l'intervalle $[0; 1]$ seront générées. La première valeur est 0.604 et est localisée dans la partie bleue de l'intervalle $[0; 1]$. Le clan cl_1 est donc sélectionné pour être rééchantillonné. Le processus continue jusqu'à ce qu'un clan soit sélectionné n_{suff} fois, ou que n_{max} valeurs aient été générées. n_{suff} est atteint pour le clan bleu lors de la 8^{ieme} itération du processus. Le clan cl_1 est donc retiré du processus de rééchantillonnage et le clan rose occupe maintenant tout l'intervalle $[0; 1]$. Le clan rose est donc sélectionné lors des deux dernières itérations. Après 10 itérations, lorsque n_{max} est atteint, le clan bleu a donc été rééchantillonné 7 fois et le clan rose 3 fois.

Par conséquent, comme dans un filtrage particulaire classique, certaines hypothèses sont supprimées et d'autres sont dupliquées selon leur score. Durant le processus de rééchantillonnage, si un jeton rééchantillonné n fois appartient à une métaplace active, il sera en réalité équitablement réparti dans chacune des nb places actives de la métaplace. Dans le filtrage particulaire, le nombre de particules définit la précision du filtre et est aussi un facteur de performance calculatoire. Le paramètre n_{max} peut donc être réglé pour répondre à des contraintes de performance.

Remarque 3. *Afin de ne pas perdre une hypothèse intéressante à cause des probabilités induites par le processus de rééchantillonnage, il serait possible dans de futurs travaux d'implémenter une fonction qui sauvegarde le jeton ayant le plus grand score pour chaque place. Si une place est vide après le rééchantillonnage, le jeton sauvé pourrait être ainsi ajouté à la place. Ceci permettrait d'avoir toujours la possibilité de "revenir" après une transition à sens unique et d'explorer de nouveaux chemins si une observation erronée (manquante ou fausse) a entraîné un score faible ou si le jeton n'a pas été sélectionné par le processus aléatoire du rééchantillonnage.*

4.3.4.3 Résultat de diagnostic

Le processus de rééchantillonage termine la phase de correction. On obtient donc le résultat de diagnostic Δ_k au temps k. Le diagnostic Δ_k correspond au marquage du diagnostiqueur basé HtPN à un temps k donné :

$$\Delta_k = \mathbb{M}_k \tag{4.7}$$

Il représente les différentes hypothèses de diagnostic comme une distribution de croyance concernant les états de santé actuels et passés du système. Le marquage $\hat{\mathbb{M}}_k$ indique donc les différentes croyances quant à l'occurrence des fautes et aux états continus et de dégradation. A partir de ce marquage, il est possible de calculer le score d'une place p, comme étant la somme du score global des jetons présents dans cette place. Le calcul du score de chaque place, qui donne lieu aux hypothèses de diagnostic, est illustré dans l'algortihme 8.

Algorithme 8 Calcul du score de chaque place

Entrée: $\mathbb{M}_k$
Sortie: Hypothèses de diagnostic

1: **pour tout** $p \in P$ **faire** ▷ On parcourt toutes les places
2: **pour tout** $tk \in p.tokens$ **faire** ▷ On parcourt les jetons de la place p
3: $Score_{place} = Score_{place} + Score(tk)$
4: **fin pour**
5: **fin pour**

Dans un diagnostic classique, les différentes hypothèses ont généralement la même probabilité. Ce n'est pas le cas dans le diagnostiqueur basé HtPN, où le score des places est basé sur les scores des jetons qui représentent les différentes hypothèses. Le diagnostiqueur basé HtPN est capable de gérer des incertitudes de modélisation et d'observation, et d'évaluer l'ambiguïté entre les différentes hypothèses en se basant sur les informations portées par les jetons.

La section qui suit présente l'application de cette méthode de diagnostic avancé à différents types de système : un système à événements discrets, un système continu et l'exemple académique du système hybride des trois cuves d'eau.

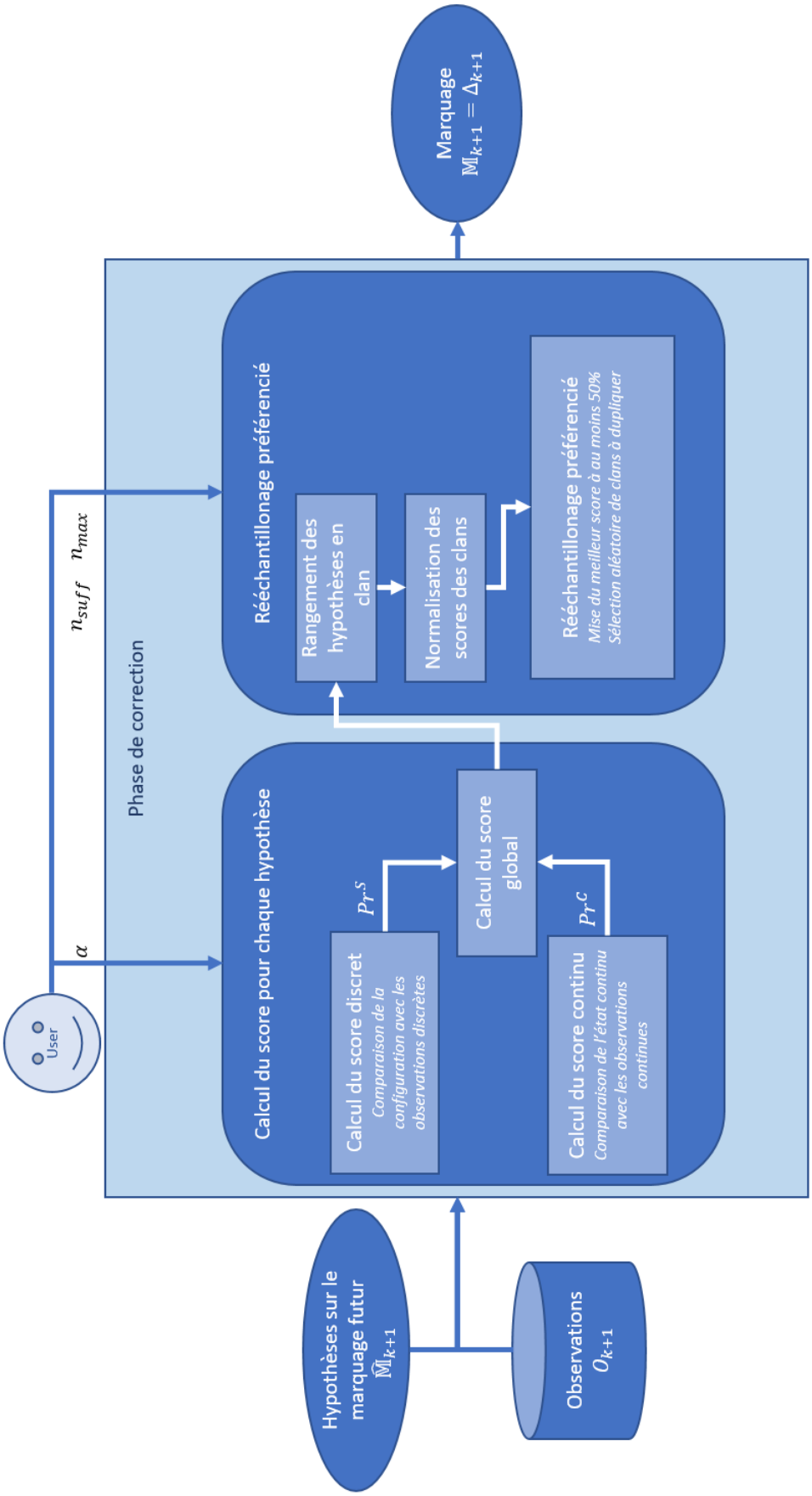


FIGURE 4.10 – Principe de l'étape de correction

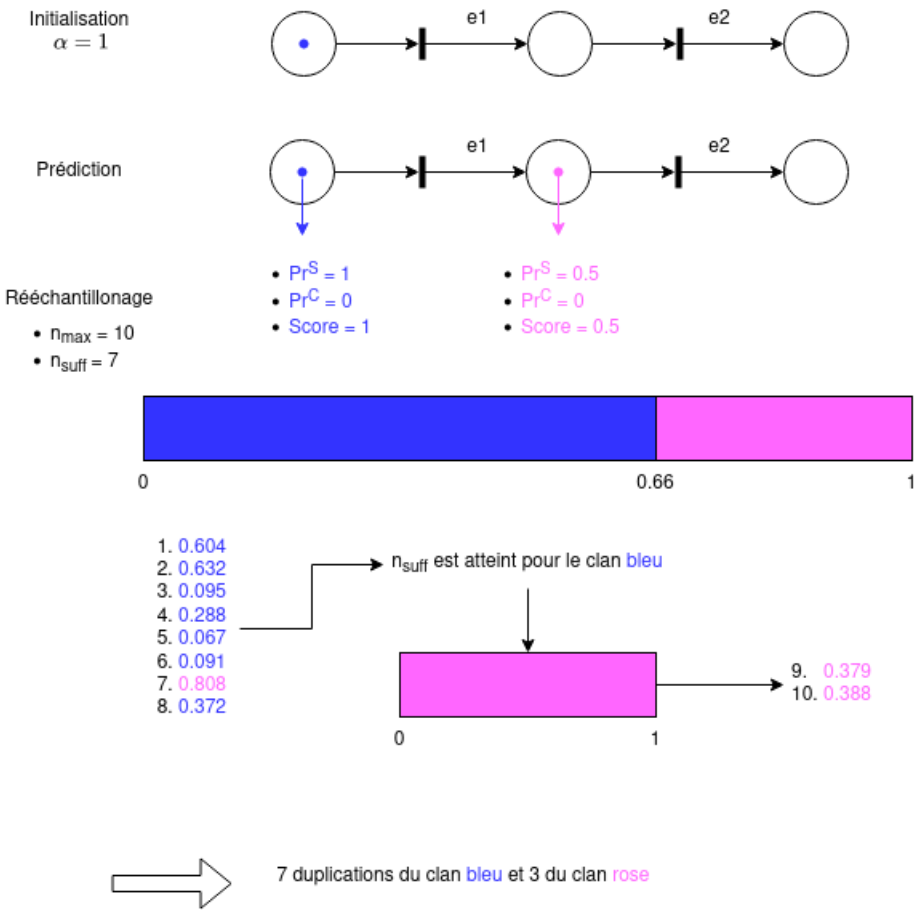


FIGURE 4.11 – Rééchantillonnage

4.4 Application de la méthode de diagnostic à différents systèmes

Afin d'illustrer la méthode de diagnostic avancée, trois systèmes seront étudiés : un système à événements discrets, un système purement continu et un système hybride classique de la littérature. Les trois systèmes ont été modélisés en utilisant le formalisme des HtPN. Les résultats de la simulation et du diagnostic sont expliqués dans cette section. Toutes les sources correspondant à la définition et aux scénarios utilisés, ainsi que les résultats obtenus, sont disponibles sur notre git [1].

4.4.1 Système à événements discrets

Le système à événements discrets considéré est illustré dans la figure 4.12. Il est intéressant, car il met en lumière la notion de diagnostic ambigu et montre la capacité de la méthode à gérer les SED purs.

Ce système peut être représenté par un automate $A =< Q, \Sigma, \delta, x_0 >$ où

- $Q = \{Nom1, Nom2, Deg1, Deg2\}$ est l'ensemble des états discrets ;
- $\Sigma = \{o_1, o_2, o_3, f\}$ est l'ensemble des événements ;
- $\delta \ : Q \times Sigma \rightarrow Q$ représente la fonction de transition ;
- $q_0 = Nom1$ est l'état initial.

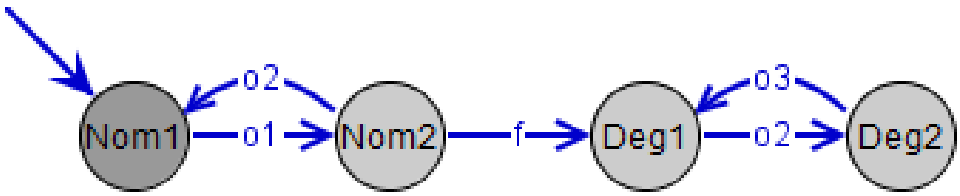


FIGURE 4.12 – Système à événement discret considéré

L'ensemble des événements observables est $\Sigma_o = \{o_1, o_2, o_3\} \subset \Sigma$ et f représente une faute non observable à diagnostiquer : $f \in \Sigma_{uo} \subset \Sigma$.

4.4.1.1 Modèle HtPN et simulation

La représentation HtPN de ce modèle a été créée en utilisant le logiciel HeMU. Dans cette application, aucune dynamique continue ni de dégradation n'ont été définies. Seules les places et les conditions symboliques associées aux arcs entrants des transitions sont spécifiées. Les résultats de diagnostic se concentreront donc uniquement sur les configurations.

Les résultats de la simulation du modèle HtPN développé sont présentés dans la figure 4.13. À partir de l'état initial $Nom1$, la séquence d'événements suivante a lieu : $o_1.f.(o_2.o_3)^*$. Cela signifie que la faute f a été injectée dans le système après l'observation o_1. Après l'occurrence de f, les observations o_2 et o_3 sont répétées jusqu'à la fin de la simulation, à $t = 90000s$.

1. Pour rappel : https ://gitlab.laas.fr/hymu/hemu

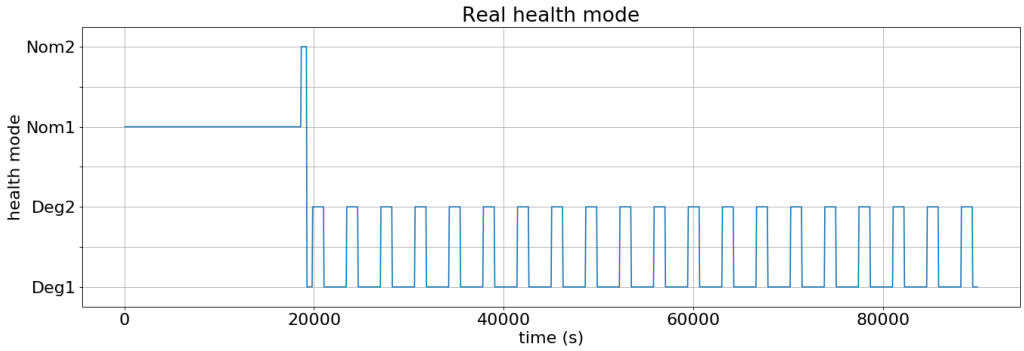


FIGURE 4.13 – Scénario considéré pour le SED

4.4.1.2 Diagnostic

Initialement, le système est en $Nom1$. Les événements o_1, o_2 et o_3 sont observables et disponibles pour la méthode de diagnostic. Dans cette application, la valeur de α, représentant la confiance des informations discrètes par rapport aux informations continues est de 1, la confiance est donnée aux informations discrètes étant donné l'absence d'information continue. Les paramètres n_{suff} et n_{max} sont mis à 120 et 200, respectivement. Il y aura donc au maximum 200 hypothèses présentes dans le système en parallèle, et une hypothèse peut être sélectionnée au maximum 120 fois par le processus de rééchantillonage. Les résultats de diagnostic obtenus après application de la méthode proposée sont illustrés sur la figure 4.14. Les places $Nom2$ et $Deg1$ sont groupées dans une métaplace par le processus. Elles sont en effet liées par l'événement de faute f, non observable, et ont les mêmes dynamiques continues (c.-à-d. aucune).

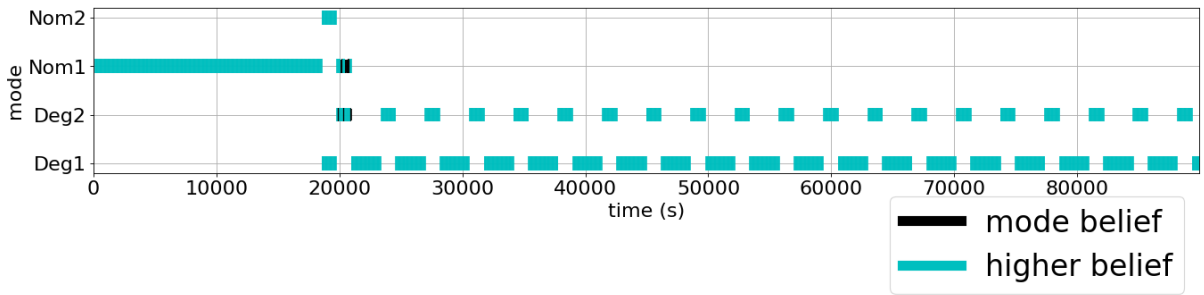


FIGURE 4.14 – Diagnostic du scénario considéré pour le SED

Toutes les hypothèses de diagnostic sont représentées sur la figure 4.14. Les lignes bleues représentent les modes de fonctionnement correspondant aux places ayant le score le plus élevé. Une manière de donner les hypothèses sur l'état de santé serait par exemple de donner la probabilité d'être dans chaque mode de fonctionnement. Par exemple, à $t = 18000s$, il y a 50% de chance d'être dans le mode $Nom2$ et 50% d'être dans le mode $Deg1$. La métaplace contenant $Nom2$ et $Deg1$ est activée par l'occurrence de l'événement o_1, étant donné leur regroupement dans une métaplace $MP1$. Tout jeton rééchantillonné dans une de ces deux places sera placé dans l'autre, ce qui explique la même probabilité de se situer entre $Nom2$ et $Deg1$. Une fois l'événement o_2 apparu, il y a encore une incertitude entre $Nom1$ et $Deg2$. Cependant, une fois l'événement o_3 observé, le diagnostic devient certain et le système oscille entre $Deg1$ et $Deg2$ avec les occurrences des événements observables o_2 et o_3. Le système considéré étant plutôt simple, la durée d'exécution ne

dépasse ici pas une minute. Les résultats de diagnostic obtenus pour ce système discret sont cohérents.

4.4.2 Système continu

La méthode de diagnostic proposée est ensuite appliquée à un système purement continu. Le système continu considéré est un système de deux cuves d'eau connectées, tiré de Bouamama et al. 2003. La représentation HtPN de ce modèle est illustrée sur la figure 4.15. Ce système a pour but de fournir un flot d'eau Q_0 à l'utilisateur. Le vecteur d'état continu de dimension 2 représente le niveau d'eau présent dans chaque cuve. 5 événements non observables sont présents : 4 fautes et un événement de *réparation*, ajouté par nos soins, $\Sigma_{uo} = f_0, f_1, f_2, f_3, \text{réparation} \subset \Sigma$. L'événement de réparation est uniquement présent pour intégrer l'occurrence des 4 fautes différentes dans une seule et même simulation, il n'a pas d'impact sur le vecteur d'état continu du système. Les 4 fautes considérées sont les suivantes : un blocage en position fermée de la pompe d'entrée ($f0$), entrainant un flot d'entrée nul, une fuite dans la cuve T_1 ($f1$), une fuite dans la cuve T_2 ($f2$) et un blocage de la vanne vb ($f3$), située entre les deux cuves, en position fermée. L'occurrence de ces 4 fautes est représentée par un événement non observable afin de facilement contrôler sa date d'occurrence lors d'une simulation du système : aucune dégradation régissant leur occurrence n'est définie pour ce système. La dynamique continue C_1 associée à la place Nom_1 régissant l'évolution du niveau d'eau dans chaque cuve est :

$$C_1 = \begin{cases} h[0]_{k+1} = h[0]_k + \\ \qquad \dfrac{1}{A1} * (u[0]_k - (Cvb * (h[0]_k - h[1]_k) * \sqrt{|h[0]_k - h[1]_k|})) * mtb \\ h[1]_{k+1} = h[1]_k + \\ \qquad \dfrac{1}{A2} * ((Cvb * (h[0]_k - h[1]_k) * \sqrt{|h[0]_k - h[1]_k|})) - Cvo * \sqrt{h[1]_k} \end{cases} \tag{4.8}$$

où $h[0]_k$ est le niveau d'eau au temps k dans la première cuve, $h[1]_k$ le niveau d'eau au temps k dans la deuxième cuve, $u[0]$ le flux d'entrée au temps k, Cvb la quantité d'eau pouvant passer par la vanne vb quand celle ci est ouverte, Cvo la quantité d'eau pouvant passer par la vanne vo quand celle ci est ouverte et $A1$ et $A2$ les tailles des cuves 1 et 2, respectivement.

L'état initial du système est la place Nom_1, avec un niveau d'eau nul dans chaque cuve ($x = [0, 0]$). Sur l'occurrence d'une faute $f_i, i = 1, \ldots, 4$, le système entre dans le mode $Degf_i$.

4.4.2.1 Modèle HtPN et simulation

Le modèle du système a été créé à partir des équations données dans l'article Bouamama et al. 2003. Le modèle multimode représentant le HtPN obtenu est représenté sur la figure 4.15. 5 places ont été définies, une place correspondant au mode nominal et une place pour chaque mode de faute (accessible lorsque la faute a eu lieu). Il est ici considéré qu'un seul événement de faute a lieu en même temps, mais il serait possible d'étendre le modèle et d'avoir une combinaison de fautes. Le scénario considéré est illustré sur la figure 4.16 : la faute f_0 est injectée à 9000s. Une réparation a lieu 9000s après,

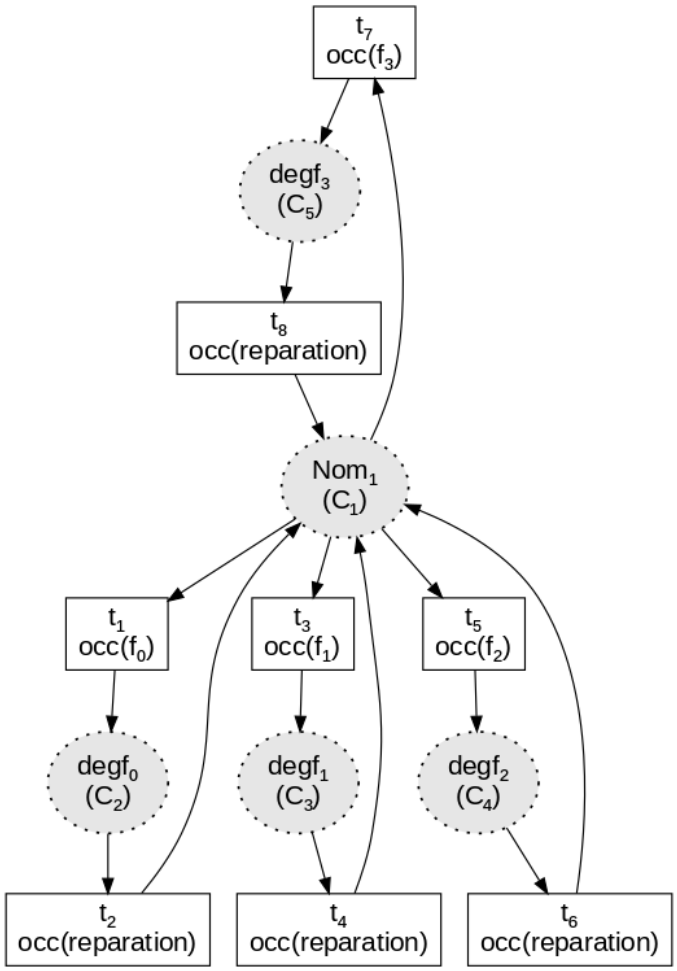


FIGURE 4.15 – Modèle multimode représentant le HtPN du système continu considéré

et le système retourne en mode nominal. La faute f_1 est ensuite injectée à 27000s et est suivie d'une réparation 9000s après. Les fautes f_2 et f_3 sont respectivement injectées à 45000s et 63000s et réparées 9000s après leur occurrence. Les résultats de la simulation sont visibles sur la figure 4.16.

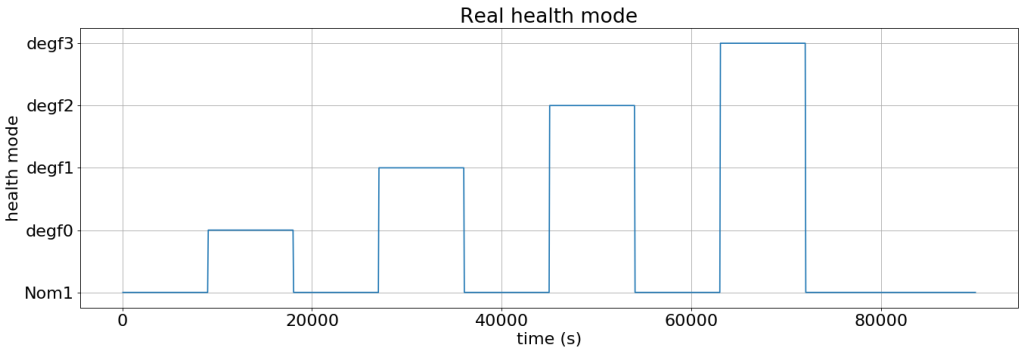


FIGURE 4.16 – Scénario considéré pour le système continu

La figure 4.17 illustre l'évolution des niveaux d'eau présents dans chaque cuve en fonc-

tion du temps (sur les abscisses) et de l'occurrence des divers événements non observables : une faute ou une réparation.

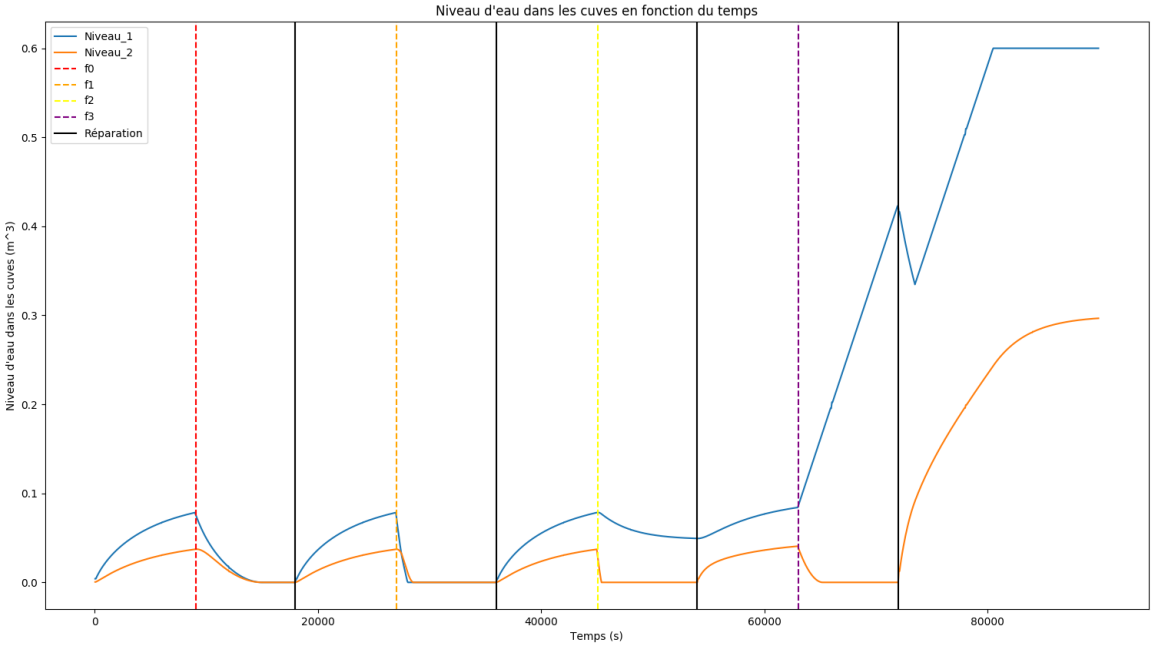


FIGURE 4.17 – Evolution des niveaux d'eau de chaque cuve en fonction du temps et des événements induits

4.4.2.2 Diagnostic

Initialement, le système est en $Nom1$. Le niveau d'eau présent dans chaque cuve est disponible pour le processus de diagnostic. Initialement, chaque cuve est vide. Dans cette application, la valeur de α, représentant la confiance des informations discrètes par rapport aux informations continues est de 0, la confiance est donc donnée aux informations numériques, le système étant purement continu. Les paramètres n_{suff} et n_{max} sont mis à 120 et 200, respectivement. Il y aura donc au maximum 200 hypothèses sur l'état du système dans le diagnostiqueur HtPN, et une hypothèse peut être sélectionnée au maximum 120 fois par le processus de rééchantillonage du diagnostic. Les résultats de diagnostic obtenus sont visibles sur la figure 4.18. Les événements de fautes et de réparation étant non observables, le diagnostic se base uniquement sur les dynamiques continues et le score continu (étant donné qu'il n'y a pas d'observations discrètes, toutes les hypothèses ont le même score discret ; le score continu est donc le seul moyen de les différencier) des jetons représentant les hypothèses. Les lignes bleues représentent les modes de fonctionnement correspondant aux places ayant le score le plus élevé. Une confusion sur la fin de l'occurrence de $f0$ est présente, entre $16000s$ et $18000s$ environ : elle s'explique par la durée écoulée après l'occurrence de la faute. En effet, au bout d'un certain temps, la cuve T_1 est vide, le système n'est donc plus capable d'identifier si cela est dû à une fuite ou à une arrivée d'eau nulle induite par un blocage de la vanne d'arrivée d'eau. Néanmoins, les résultats de diagnostic obtenus pour un système continu sont jugés satisfaisants, étant donné

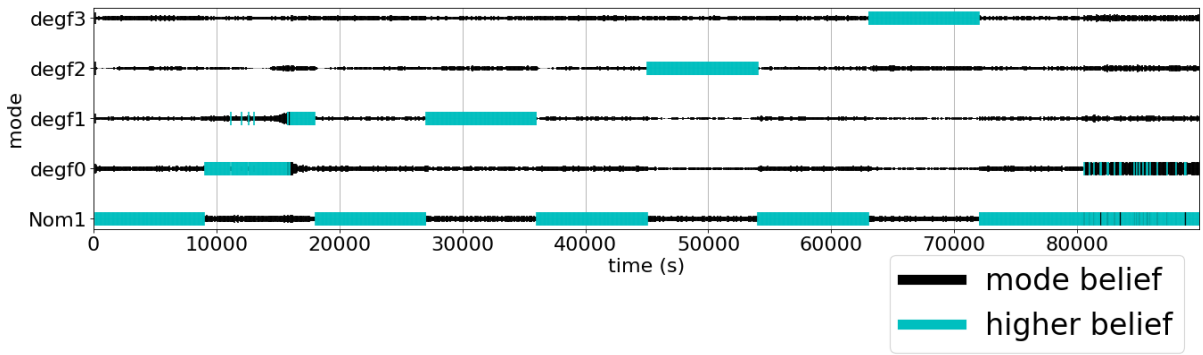


FIGURE 4.18 – Diagnostic du scénario considéré pour le système continu

qu'aucune autre méthode de diagnostic disposant des mêmes informations ne donnera de meilleurs résultats.

4.4.3 Système hybride

La méthode est maintenant appliquée à un système hybride plus complexe. Le système hybride considéré est composé de trois cuves d'eau, connectées en série par deux vannes. Le comportement du système et sa description multimode sont explicitées en détail dans RIBOT, CHANTHERY et al. 2017 et dans la section 2.4.3.1. Pour résumer, une pompe délivre un flot constant d'eau q_1 dans T_1, et la cuve T_2 (placée en troisième position) se vide avec un flot q_{20}. Le but du système est de maintenir le niveau d'eau présent dans la cuve T_2 supérieur à une valeur seuil h_{2min}. Une description multimode du système des cuves d'eau est présentée dans la figure 2.8 page 43.

4.4.3.1 Modèle HtPN et simulation

Le modèle HtPN du système hybride des cuves d'eau a été créé en se basant sur la représentation multimodale de la figure 2.8. La simulation du modèle HtPN du système des cuves d'eau est présentée dans la figure 4.19. Le scénario utilisé est le même que celui dans les chapitres 2 et 3.

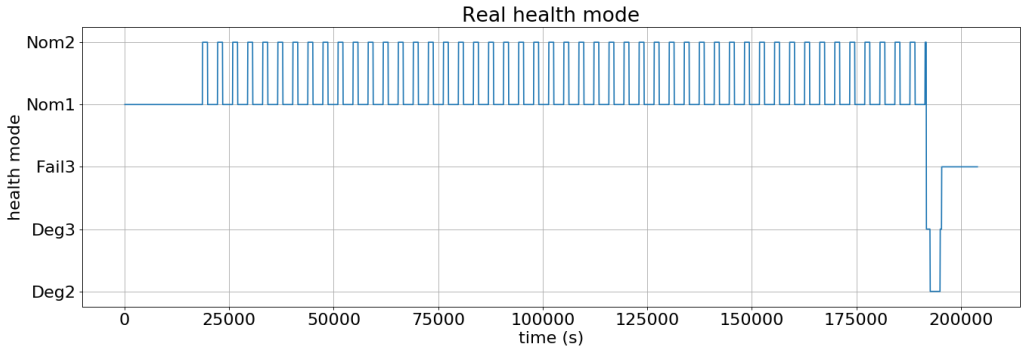


FIGURE 4.19 – Scénario considéré pour le système des cuves d'eau

Dans le mode de fonctionnement initial Nom_1, les vannes v_{13} et v_{32} sont ouvertes, le vecteur d'état continu représentant les niveaux d'eau dans les cuves est $x_0 = [l_{10}, l_{20}, l_{30}] = [0.60, 0.55, 0.58]$ et le vecteur d'état de dégradation est $d_0 = [\mathrm{p}_{10}, \mathrm{p}_{20}, \mathrm{p}_{30}, \mathrm{p}_{40}, \mathrm{p}_{50}] =$

$[0, 0, 0, 0, 0]$. Le flot d'eau entrant q_1, délivré par la pompe, est considéré constant. Après 310 minutes d'opération, la vanne v_{13} est fermée pendant 20 minutes chaque heure, afin d'effectuer un traitement de l'eau dans T_1. La faute f_1 est injectée à 201840s et entraînera la défaillance du système (représentée par l'occurrence de f_0) à 206040s.

4.4.3.2 Diagnostic

Initialement, le système est dans le mode Nom_1. Le niveau d'eau présent dans chaque cuve d'eau est disponible pour le processus de diagnostic, ainsi que les événements observables $close_{v_{13}}$ et $open_{v_{13}}$. Initialement, chaque cuve contient 0.6, 0.55 et $0.58m^3$ d'eau. Dans cette application, la valeur de α, représentant la confiance des informations discrètes par rapport aux informations continues est de 0.4, une confiance légèrement plus importante est donnée aux informations continues. Les paramètres n_{suff} et n_{max} sont mis à 20 et 200, respectivement. Il y aura donc au maximum 200 hypothèses sur l'état du système dans le diagnostiqueur HtPN, et une hypothèse peut être sélectionnée au maximum 20 fois par le processus de rééchantillonage du diagnostic. Les résultats de diagnostic obtenus pour le système des cuves d'eau sont présentés dans la figure 4.20.

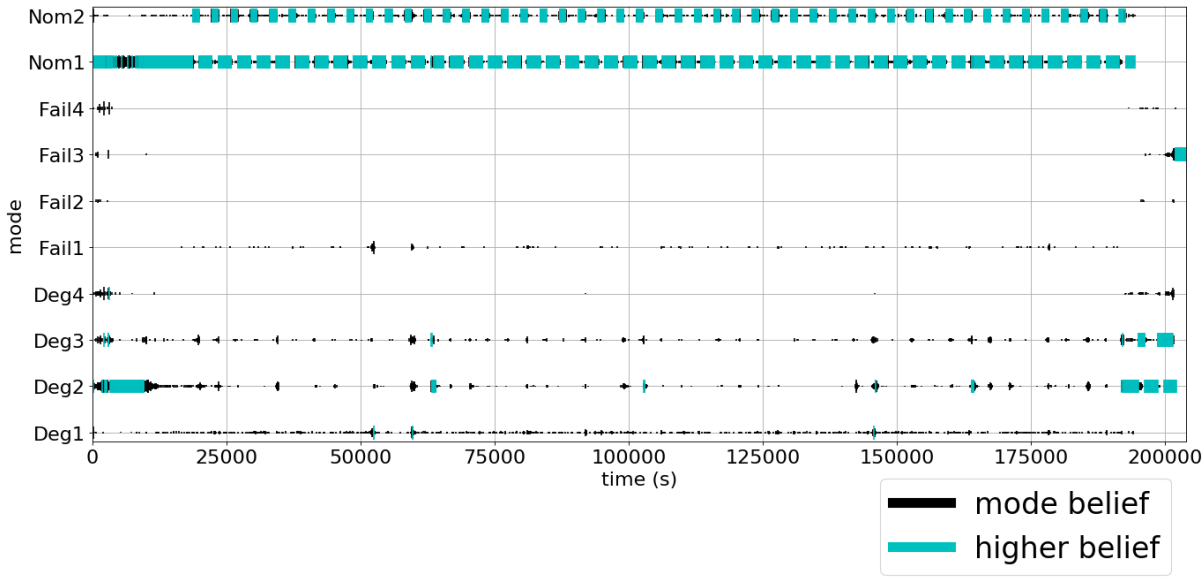


FIGURE 4.20 – Diagnostic du scénario considéré pour le système hybride avec 200 jetons maximum

Les différentes hypothèses de diagnostic sont visibles sur la figure 4.20. Les lignes bleues représentent les modes de fonctionnement correspondant aux places ayant le score le plus élevé. L'incertitude présente entre $Nom1$ et $Deg2$ s'explique par des dynamiques continues proches dans ces deux modes de fonctionnement. Bien que non représentée ici, et non utilisée par le processus de diagnostic, la valeur du vecteur d'état de dégradation de chaque hypothèse (c.-à-d. de chaque jeton) est disponible à travers son statut. Les résultats obtenus correspondent de manière satisfaisante aux résultats attendus vis-à-vis de la simulation. On retrouve le passage vers $Deg2$ puis l'alternance entre les modes $Deg3$ et $Deg2$ autour de $t = 185000$ et finalement le passage dans le mode de défaillance $Fail3$ à $t = 210000$ environ.

Si le nombre maximal de jetons présents dans le diagnostiqueur HtPN est fixé à 200, le

temps d'exécution de la méthode proposée est de 4 minutes et 22 secondes. Si le nombre maximal est fixé à 400, le temps d'exécution est de 7 minutes et 47 secondes. Ces informations sont liées à la complexité de la méthode proposée et ouvrent la porte à de futurs travaux sur l'étude de la complexité. En effet, cette dernière n'est pas juste fonction du nombre de jetons considérés, mais a aussi une part d'aléatoire liée à la complexité des dynamiques continues présentes dans les places qui peut varier conséquemment.

Cette section montre, à travers les trois exemples étudiés, que la méthode de diagnostic proposée est bel et bien applicable à tout type de système et offre des résultats satisfaisants.

4.5 Conclusion

4.5.1 Résumé du chapitre

Ce chapitre présente une méthode de diagnostic avancé pour des systèmes en tenant compte des incertitudes de modélisation et d'observation. Cette approche se base sur le formalisme des HtPN, défini dans le chapitre 2. L'approche proposée permet de gérer de l'incertitude de différentes manières, à travers du pseudo-tirage lié à l'étape de prédiction du processus de diagnostic ou du rééchantillonnage, par exemple. La méthode de diagnostic avancé proposée est découpée en deux principales étapes. La première étape est une étape de prédiction, dont le but est d'estimer le marquage futur du diagnostiqueur. Ensuite, l'étape de correction va calculer un score pour les différentes hypothèses estimées par l'étape de prédiction en fonction des observations discrètes et continues disponibles. Une fois que les scores des hypothèses ont été calculés, les hypothèses ayant des scores similaires vont être regroupées en clan et rééchantillonnées. Des applications sur un système à événements discrets, un système continu et un système hybride ont montré l'efficacité de la méthode de diagnostic pour différents types de système et la possibilité de prendre en compte les incertitudes.

4.5.2 Limitations et discussion

Dans cette thèse, une méthode de diagnostic avancée a été apprise. Cette méthode suit la valeur du vecteur de dégradation mais ne l'utilise pas à ce jour. Une utilisation possible, outre le fait d'implémenter une fonction de pronostic, pourrait être de pondérer les hypothèses de diagnostic, étant donné que la valeur du vecteur de dégradation représente la probabilité d'occurrence de différentes fautes. La méthode a été appliquée sur différents cas d'étude avec des paramètres n_{suff} et n_{max} donnés. L'impact de ces paramètres sur les résultats de diagnostic et sur la complexité de l'algorithme ne sont pas étudiés. Enfin, il existe encore quelques cas pour lesquels la gestion de l'incertitude présente des problèmes (un tel cas se présentera dans le chapitre 5). En effet, un effet "boule de neige" peut arriver, suite à une combinaison du processus de rééchantillonnage, du pseudo-tirage et de l'impossibilité de revenir en arrière suite à une faute dans certains modèles considérés. Dans ce problème, une hypothèse erronée proche de la vérité va prendre de plus en plus de poids dans le processus de diagnostic et finir par évincer l'hypothèse correcte.

Le prochain chapitre présente une application physique réelle des trois chapitres précédents. Cette application physique est composée de deux panneaux photovoltaïques et de charges. Tout d'abord, une modélisation théorique du système va être proposée. La méthode de diagnostic va ensuite être appliquée au modèle théorique proposé pour divers scénarios. Enfin, la méthode d'apprentissage à partir de données va être appliquée afin de vérifier son applicabilité à des systèmes réels.

Chapitre 5

Application

Sommaire

5.1 Introduction 112
5.2 Description du système 112
5.3 Modélisation d'un système hybride vieillissant pour la gestion de santé 115
5.3.1 Modélisation du système des panneaux en HtPN 115
5.3.2 Simulation du système 118
5.4 Application de la méthode de diagnostic avancé 122
5.4.1 Scénario 1 : cas nominal 122
5.4.2 Scénario 2 : insertion d'une faute menant à la défaillance . . . 123
5.4.3 Scénario 3 : vieillissement des panneaux 123
5.4.4 Scénario 4 : fusion des trois scénarios précédents 124
5.5 Application de la méthode d'apprentissage 125
5.5.1 Données d'entrées 126
5.5.2 Prétraitement des données 127
5.5.3 Apprentissage structurel du modèle HtPN 127
5.5.4 Apprentissage des dynamiques continues 128
5.5.5 Application de la méthode de diagnostic au modèle appris . . . 130
5.6 Conclusion 134

5.1 Introduction

Ce chapitre porte sur l'application des différentes contributions proposées par cette thèse à une maquette réelle composée de deux panneaux photovoltaïques reliés à différentes charges et pouvant être éclairés par un soleil artificiel ou par la lumière naturelle. La maquette a été conçue lors d'un stage de master que nous avons encadré.

Cette maquette a pour but d'appréhender l'applicabilité du formalisme et des méthodes proposés dans un cadre réel et s'inscrit dans un contexte écologique et sociétal important.

Ce chapitre s'organise de la manière suivante. La section 5.2 présente le système et la section 5.3 propose une modélisation des deux panneaux photovoltaïques avec le formalisme des HtPN. La section présente également les résultats de la simulation de ce modèle. Elle illustre le chapitre 2 de la thèse. La section 5.4 consiste en l'application de la méthode de diagnostic avancé proposée dans le chapitre 4 sur divers scénarios. La méthode de diagnostic utilisera ici le modèle HtPN des deux panneaux proposé dans la section 5.3. Enfin, la section 5.5 consiste en l'application de la méthode d'apprentissage proposée dans le chapitre 3 à des jeux de données réels obtenus à partir de la maquette. Deux modèles seront obtenus après apprentissage, un modèle obtenu avec application du mécanisme de réapprentissage et un modèle obtenu sans application du mécanisme de réapprentissage. La méthode de diagnostic avancé sera ensuite de nouveau appliquée sur les modèles obtenus par apprentissage afin de comparer les deux modèles obtenus. Cette comparaison permettra de mettre en évidence l'intêret du choix du seuil dans le mécanisme de réapprentissage. Dans ce chapitre, seule la méthode de régression polynomiale sera utilisée, pour sa capacité à extrapoler étant donné que les jeux de données ne couvrent pas forcément tout le domaine d'évolution du système.

5.2 Description du système

Un système hybride vieillissant a été imaginé et mis en place durant ce travail de thèse. Il est composé de deux panneaux photovoltaïques connectés à une batterie lithium-ion. Les deux panneaux photovoltaïques [1] que nous utilisons sont théoriquement identiques et possèdent donc les mêmes caractéristiques. Plusieurs grandeurs physiques définissent une cellule photovoltaïque : la tension à vide (V_{co}), le courant de court-circuit (I_{cc}) et le point de puissance maximal (MPP). Le MPP est la caractéristique la plus importante, car elle nous permet de connaître la puissance maximale que l'on peut espérer obtenir avec la cellule considérée. À ce MPP sont associés une tension (V_{MPP}) et un courant (I_{MPP}). Nos cellules possèdent les caractéristiques suivantes :

— V_{MPP}= 5.5 V
— I_{MPP} = 540 mA
— MPP = 3 W

La batterie que nous utilisons est un accu Li-Ion [2] de 3.7 V d'une capacité de 1050 mAh.

1. https://www.gotronic.fr/art-cellule-solaire-sol3w-18996.htm#complte_desc
2. https://www.gotronic.fr/art-accu-li-ion-3-7-v-1050-mah-pr474446-5811.htm

Le système contient aussi une carte électronique [3]. C'est une carte d'interface compatible avec la carte Arduino Leonardo que nous utilisons. Elle a initialement pour objectif d'alimenter la carte Arduino avec une batterie rechargeable. La batterie peut elle-même être rechargée par micro-USB ou à l'aide de cellules photovoltaïques. Cependant, nous ne nous intéressons pas à la possibilité d'alimenter la carte Arduino de cette façon. Le véritable intérêt de la carte dans notre cas est sa capacité d'empêcher une décharge de la batterie dans les panneaux lorsqu'ils ne sont pas éclairés les protégeant ainsi de la surchauffe : c'est le shield sur la figure 5.1

Des charges sont aussi présentes et activables sur demande, deux lampes et un moteur, permettant de gérer la vitesse à laquelle la batterie se décharge. Le système comprend aussi des capteurs connectés à une carte Arduino Leonardo qui permet de collecter des données. Les deux panneaux photovoltaïques sont illuminés par un soleil artificiel. Ils sont théoriquement identiques et ont donc les mêmes paramètres caractéristiques.

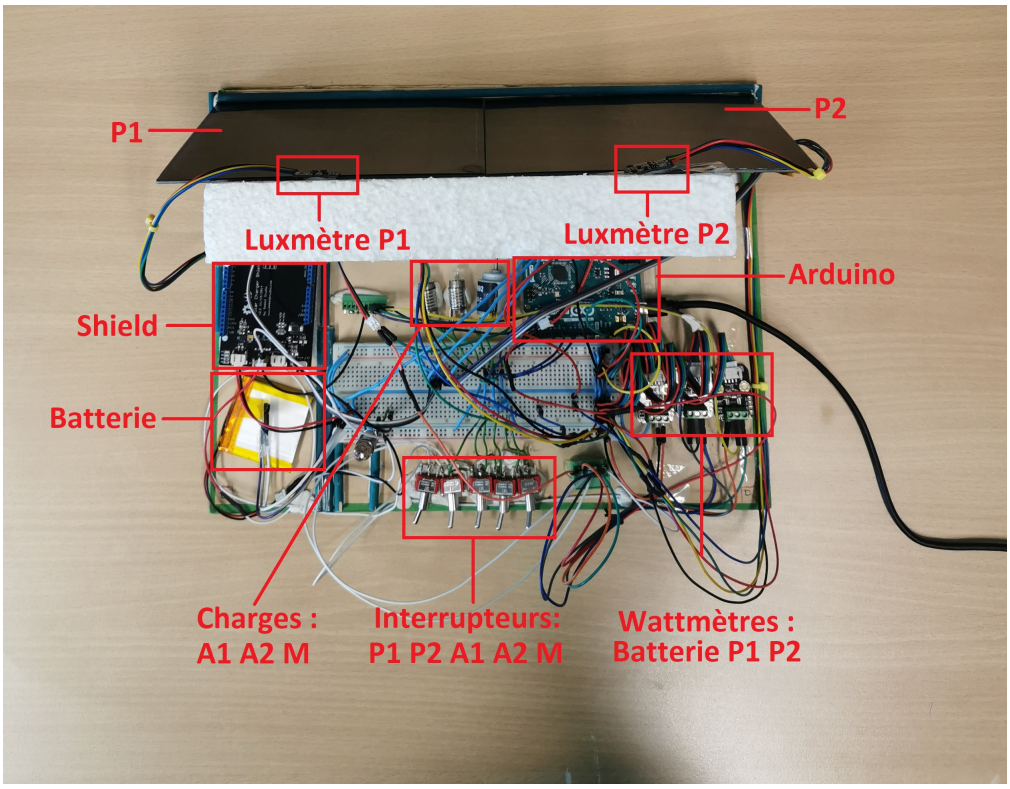


FIGURE 5.1 – Présentation du système

3. https://www.gotronic.fr/art-shield-chargeur-solaire-v2-2-106990020-23077.htm#complte_desc

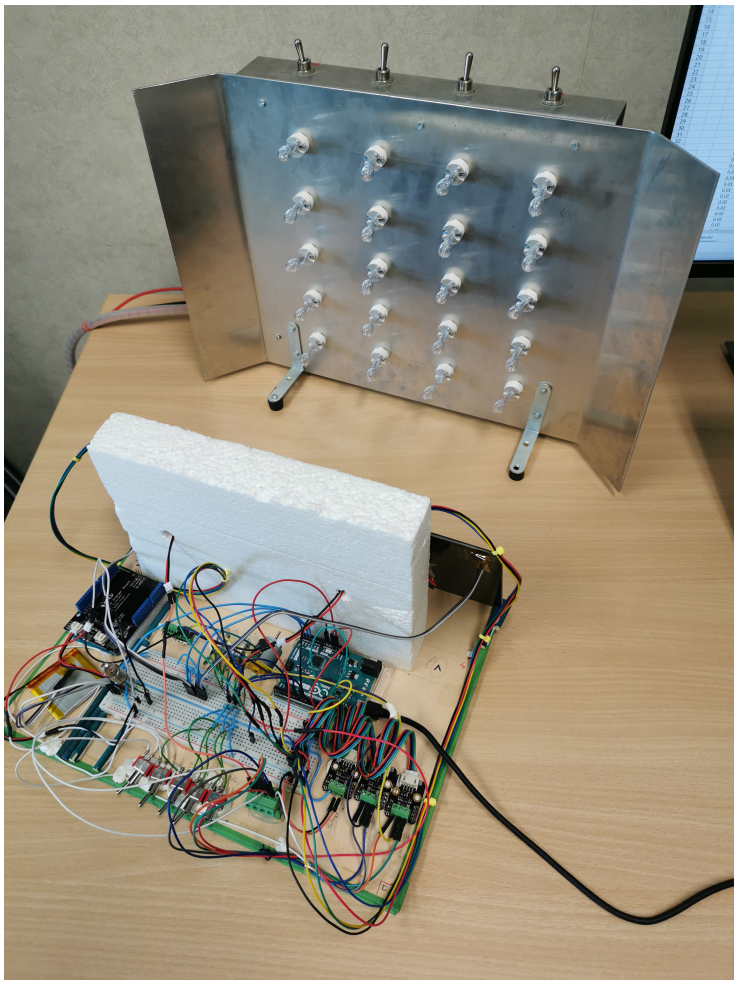


FIGURE 5.2 – Système des panneaux devant le soleil artificiel

Le courant et la tension fournis par un panneau solaire dépendent de l'éclairement capté. Afin de charger la batterie plus rapidement, les deux panneaux sont branchés en parallèle. La batterie Lithium-ion (3.7V et d'une capacité de 1050mAh) est connectée de manière à pouvoir être chargée par les panneaux solaires et déchargée par les charges. La carte de protection permet d'éviter une décharge de la batterie dans les panneaux quand ces derniers sont éteints et ainsi d'éviter une surchauffe. Pour chaque panneau, les données obtenues via la carte Arduino Leonardo sont les suivantes : la tension, le courant et la puissance fournis, la température et la luminosité captée par le panneau. Le courant, la tension et la puissance de chaque panneau et de la batterie sont mesurés par trois wattmètres [4] (un par panneau et un pour la batterie). La température est quant à elle mesurée par des capteurs de températures [5], dont la plage de mesure s'étend de -40°C à +125°C, placés derrière chaque panneau et sur la batterie. Enfin, la luminosité captée est mesurée par des luxmètres [6], dont la plage de mesure va de 1 à 65000 lux, placés au-dessus de chaque panneau.

La figure 5.3 est un schéma récapitulatif du modèle de la maquette et des capteurs utilisés. On y retrouve également les noms des variables utilisées dans les codes Arduino et Python utilisés pour récupérer les données, ainsi que les ports associés aux capteurs.

4. https://www.dfrobot.com/product-1827.html
5. https://fr.rs-online.com/web/p/capteurs-de-temperature-et-d-humidite/0403620
6. https://www.gotronic.fr/art-capteur-de-lumiere-sen0097-19372.htm

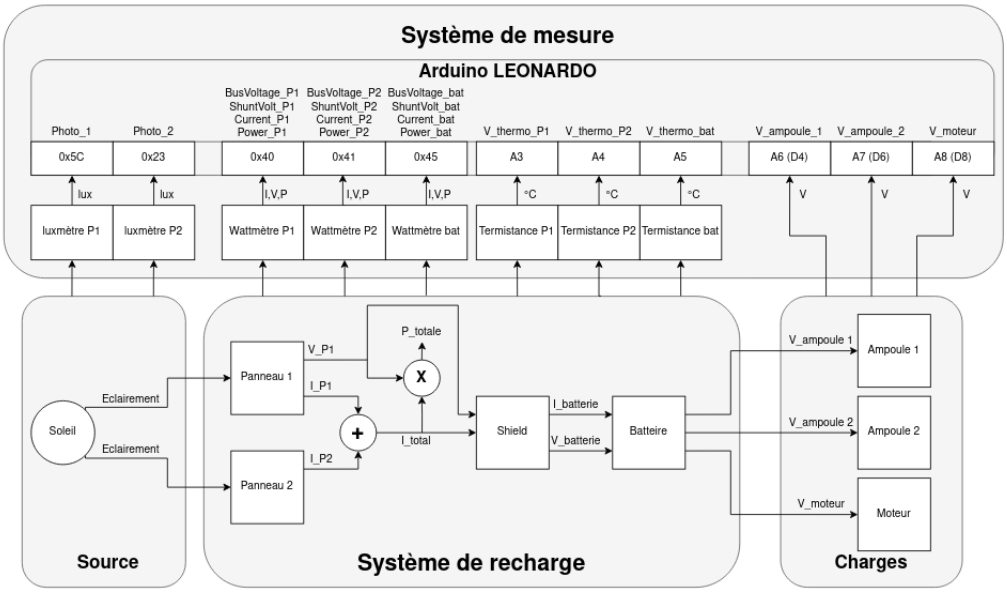


FIGURE 5.3 – Schéma récapitulatif de la maquette

L'objectif de la section suivante est de modéliser ce système de panneaux photovoltaïques à partir des connaissances dont on dispose sous le formalisme des HtPN afin de pouvoir surveiller leur état de santé.

5.3 Modélisation d'un système hybride vieillissant pour la gestion de santé

5.3.1 Modélisation du système des panneaux en HtPN

Les hypothèses simplificatrices suivantes ont été faites concernant le comportement physique du système :

— On suppose que la luminosité reçue s'exprime par un sinus positif dont le pic se trouve à midi, $(E^i_{th}(k+1) = E_{max}.|sin(\pi\frac{k}{T_{days}})|)$

— Le courant observé par les capteurs est obtenu en se basant sur un courant théorique, qui dépend de la luminosité et de la dégradation du panneau, $(I^i_{sensor}(k+1) = (a.E^i_{sensor}(k)+b).(1-\frac{Degradation_i(k)}{I^{max}_{th}})$

— Selon des mesures réalisées sur le système, les courants dépendent de la luminosité. En première approximation, un modèle affine est adopté :$I^i_{th}(k+1) = a.E^i_{th}(k)+b$

— Afin d'avoir une durée de simulation acceptable, le vieillissement des panneaux a été accéléré afin que les panneaux entrent dans un mode de défaillance au bout de quelques semaines.

Afin d'obtenir le modèle HtPN du système, la démarche de modélisation proposée dans la section 2.4.1 va être suivie.

Étape 1 : identification des modes de fonctionnement nominaux et des dynamiques d'évolution Pour chaque panneau, deux modes nominaux, un dans lequel le panneau est allumé (noté $Nom1_i$ ou N_i, i=1 ou 2) et un dans lequel le panneau est éteint (noté $Nom2_i$ ou E_i) ont été identifiés. Quatre places (deux par panneaux) vont donc être créées et ajoutées dans l'ensemble P. Quatre événements discrets et observables ont été identifiés et représentent l'allumage ou l'extinction d'un panneau : $E_o = \{ON1, OFF1, ON2, OFF2\}$. Ces événements permettent le passage d'un mode nominal à l'autre. On en déduit donc, d'une part l'existence de deux transitions $t1_i$ et $t2_i$ par panneau qui vont être ajoutées à l'ensemble T et d'autre part les ensembles Pre et $Post$ suivants :

- $Pre(Nom1_i, t1_i) = \{(OFFi, -, -); 1\}$
- $Post(t1_i, Nom2_i) = \{(-, -, -); 1\}$
- $Pre(Nom2_i, t2_i) = \{(ONi, -, -); 1\}$
- $Post(t2_i, Nom1_i) = \{(-, -, -); 1\}$

Le vecteur d'état continu est un vecteur de taille 8 composé de l'éclairement théorique, de l'éclairement mesuré simulé, du courant théorique et du courant mesuré simulé, et ce pour chaque panneau. Les dynamiques continues C_p sont considérées identiques pour toutes les places $p \in P$ pour lesquelles un panneau est allumé et sont décrites par les équations suivantes :

$$\begin{cases} E^i_{th}(k+1) = E_{max}.|sin(\pi \frac{k}{T_{days}})| \\ E^i_{sensor}(k+1) = E^i_{th}(k) + noise(k) \\ I^i_{th}(k+1) = a.E^i_{th}(k) + b \\ I^i_{sensor}(k+1) = (a.E^i_{sensor}(k) + b).(1 - \frac{Degradation_i(k)}{I^{max}_{th}}) \end{cases} \tag{5.1}$$

avec $E_{max} = 120000$lux étant la luminosité typique obtenue au zénith (à midi), $T_{days} = 86400s$ la durée d'un jour en seconde et I^{max}_{th} le courant théorique maximum obtenu quand la luminosité maximale est atteinte. Un bruit blanc de 1% est considéré sur la luminosité lue par le luxmètre. Par identification à partir de données on a $a = 1.9e-2$ et $b = 7.3$. Ces équations permettent d'estimer au temps $k+1$ la luminosité théorique, la luminosité vue par le luxmètre, le courant théorique que les panneaux devraient fournir et le courant mesuré. $Degradation_i$ est un indicateur de dégradation du panneau i qui correspond à une perte du courant observé et qui est utilisé pour représenter une perte de production du panneau. Cet indicateur de dégradation est déterminé via une fonction de dégradation D_p qui est identique pour toutes les places du modèle.

La fonction modélisant la dégradation d'un panneau D_p est considérée continue et suit une distribution gaussienne qui modélise des phénomènes comme la corrosion, l'usure du verre, la décoloration, l'apparition de fissures ou la délamination (Vázquez et al. 2008). Cette approche permet de se rapprocher de la réalité en implémentant une part d'aléatoire dans la dégradation d'un panneau. Une gaussienne centrée sur 0 est considérée pour représenter la probabilité d'occurrence d'une faute sur le panneau. Plus le temps passe, plus la déviation de cette gaussienne est augmentée, ce qui permet de plus larges valeurs de dégradation, pour représenter le vieillissement du panneau et l'augmentation de sa fragilité. Ainsi, plus le temps passe, plus la probabilité d'occurrence d'une faute sur le panneau augmente. Cette dégradation est cumulative et impacte le rendement du panneau.

Étape 2 : identification des modes de fonctionnement dégradés Pour chaque panneau, deux modes dégradés ont été identifiés, dans lesquels le panneau a perdu du rendement, un dans lequel le panneau est allumé ($Deg1_i$) et un dans lequel le panneau est éteint ($Deg2_i$). Un mode de défaillance ($Fail_i$) a également été identifié dans lequel le panneau doit être réparé ou remplacé. Ces trois modes vont être représentés par six places $Deg1_i$, $Deg2_i$, $Fail_i$, i=1, 2, ajoutées à l'ensemble P. Un panneau peut aussi être éteint dans un mode dégradé Deg_i. On déduit également d'une part l'existence de transitions entre les places $Deg1_i$ et $Deg2_i$ et les ensembles Pre et $Post$ suivants :

— $Pre(Deg1_i, t5_i) = \{(ONi, -, -); 1\}$
— $Post(t5_i, Deg2_i) = \{(-, -, -); 1\}$
— $Pre(Deg2_i, t6_i) = \{(OFFi, -, -); 1\}$
— $Post(t6_i, Deg1_i) = \{(-, -, -); 1\}$

Une faute f représentant un court-circuit ou un circuit ouvert est considérée et entraîne une défaillance immédiate du panneau. Cette faute est représentée par un événement discret non observable : $f \in E_{uo}$. Cette faute intervient aussi dans les conditions symboliques Ω^S associées aux arcs entrants dans une transition. La faute peut survenir à n'importe quel moment au cours du fonctionnement du système, qu'il soit en mode nominal ou en mode dégradé. Des transitions dont la condition symbolique est associée à l'occurrence de f sont rajoutées pour chaque place $p \in P$. L'occurrence de la faute f est représentée par les ensemble Pre et $Post$ suivants :

— $Pre(Nom1_i, t_j) = \{(f, -, -); 1\}$
— $Post(t_j, Fail_i) = \{(-, -, -); 1\}$
— $Pre(Nom2_i, t_i) = \{(f, -, -); 1\}$
— $Post(t_i, Fail_i) = \{(-, -, -); 1\}$
— $Pre(Deg1_i, t_j) = \{(f, -, -); 1\}$
— $Post(t_j, Fail_i) = \{(-, -, -); 1\}$
— $Pre(Deg2_i, t_i) = \{(f, -, -); 1\}$
— $Post(t_i, Fail_i) = \{(-, -, -); 1\}$

L'évolution de la dégradation va déclencher le passage du système d'un mode nominal vers un mode dégradé ou d'un mode dégradé vers un mode de défaillance. Ceci implique la création de transitions entre les modes nominaux et dégradés et entre les modes dégradés et de défaillance, ainsi que les ensembles de conditions et assignations définis ci-après. Les valeurs limites choisies sont extraites de Vázquez et al. 2008 :

— $Pre(p_i, t_j) = \{(-, -, 0.2 < \frac{Degradation_i}{I_{th}^{max}}); 1\}$ avec p_i un mode où le panneau i est nominal ;
— $Post(t_j, p_g) = \{(-, -, -); 1\}$ avec p_g une place où le panneau i est dégradé ;
— $Pre(p_i, t_j) = \{(-, -, 0.95 < \frac{Degradation_i}{I_{th}^{max}}); 1\}$ avec p_i un mode où le panneau i est dégradé ;
— $Post(t_j, p_g) = \{(-, -, -); 1\}$ avec p_g une place où le panneau i est défaillant.

La notation $\frac{Degradation_i}{I_{th}^{max}}$ représente une perte d'efficacité du panneau entre le courant maximal fourni et le courant produit. Sa valeur est comprise entre 0 et 1. Avec une perte de rendement de 95%, le panneau est considéré comme défaillant.

Étape 3 : compléments de modélisation Aucun complément de modélisation n'a été identifié dans la modélisation de ce système.

Le modèle HtPN d'un panneau simple est illustré dans la figure 5.4. On y retrouve les deux modes nominaux, Nom_1 et Nom_2, en vert, les deux modes dégradés, Deg_1 et Deg_2, en orange, et le mode de défaillance $Fail$ en rouge.

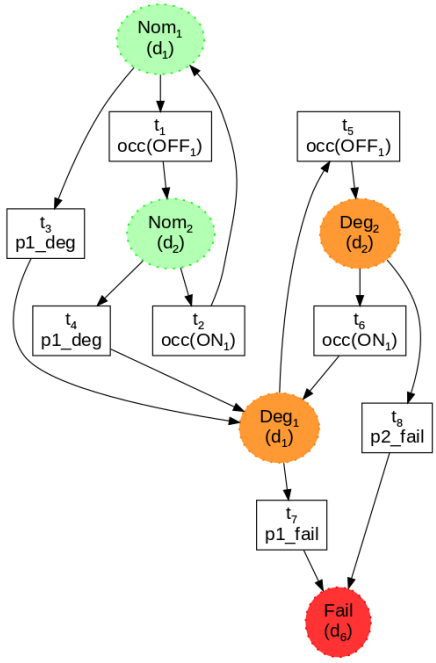


FIGURE 5.4 – Structure HtPN pour un seul panneau

Le modèle HtPN des deux panneaux est illustré sur la figure 5.5 et correspond à une composition des deux modèles correspondant à chaque panneau comprenant 25 places et 74 transitions. Les places vertes correspondent aux modes nominaux des deux panneaux. Les places jaunes correspondent à la combinaison d'un mode dégradé pour un panneau et d'un mode nominal pour l'autre. Les places oranges correspondent soit à la combinaison d'un mode de défaillance et d'un mode nominal, soit de deux modes dégradés. Les places rouges correspondent à la combinaison d'un mode de défaillance et d'un mode dégradé. Enfin, la place noire correspond aux deux panneaux en mode de défaillance. Dans un souci de taille du nom des places, les places de chaque panneau ont été renommées lors de la concaténation. Le tableau 5.1 donne une liste non exhaustive des nouveaux noms obtenus afin de fournir une idée de la façon de renommer.

5.3.2 Simulation du système

Le modèle HtPN obtenu a été simulé avec le logiciel HeMU afin de valider le modèle en simulation. Le scénario de simulation est le suivant : les panneaux sont éteints et rallumés de manière périodique chaque jour. Les panneaux sont éteints entre 22h00 et 5h00 chaque nuit. Une faute est injectée à $t = 43200s$ sur le panneau $P1$, le panneau $P2$ continue de fonctionner jusqu'à sa défaillance due à la dégradation.

Les résultats de simulation sont illustrés sur les figures 5.6 et 5.7. Les courbes re-

TABLEAU 5.1 – Tableau offrant un aperçu des noms composés dans le modèle dans deux panneaux

Deux places composées	Nom dans la composition
$Nom1_1 + Nom1_2$	$N1N2_nom$
$Nom1_1 + Nom2_2$	$N1E2_nom$
$Nom1_1 + Deg1_2$	$N1D2_deg2$
$Nom1_1 + Deg2_2$	$N1E2_deg2$
$Deg1_1 + Deg1_2$	$D1D2_deg12$
$Deg1_1 + Deg2_2$	$D1E2_deg12$
$Nom1_1 + Fail_2$	$N1M2_nom$
$Deg1_1 + Fail_2$	$D1M2_deg$
$Fail_1 + Fail_2$	$M1M2_mort$

présentent les courants fournis simulés et la valeur de la dégradation simulée pour les panneaux $P1$ et $P2$ qui varient entre les deux modes nominaux $Nom1$ et $Nom2$. La faute injectée à $t = 43200s$ sur le panneau $P1$ est représentée par la ligne bleue verticale. Le panneau $P1$ entre alors dans un mode de défaillance et le courant qu'il fournit diminue brusquement jusqu'à 0. Lorsqu'un panneau entre dans un mode de défaillance, la valeur de sa dégradation n'évolue plus. C'est ce qu'on observe sur la figure 5.7. Le courant dans le panneau $P2$, représenté par la courbe orange, continue d'évoluer et sa dégradation d'augmenter. Cette augmentation de la dégradation résulte en une perte progressive de courant visible sur la figure 5.6, et, lorsque la dégradation simulée atteint le seuil des 0.95, le panneau $P2$ entre alors dans son mode de défaillance : le courant simulé dans le panneau devient nul, la dégradation reste constante.

La simulation réalisée montre que le modèle HtPN proposé arrive à représenter le comportement continu et la dégradation de notre système étudié. Ce modèle HtPN arrive à englober les informations nécessaires à l'implémentation d'une fonction de diagnostic avancé et de pronostic afin d'effectuer de la surveillance de santé.

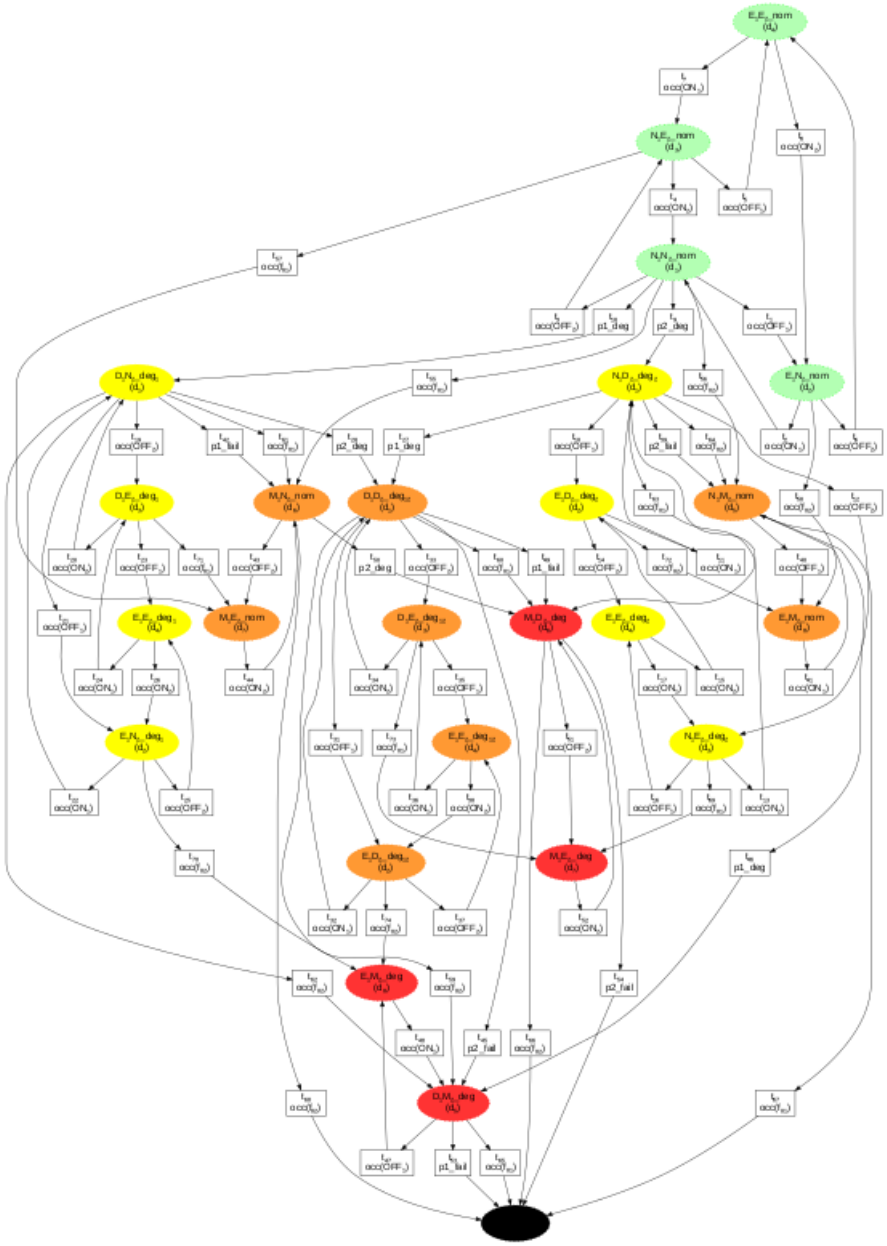


FIGURE 5.5 – Représentation multimode du modèle HtPN pour les deux panneaux

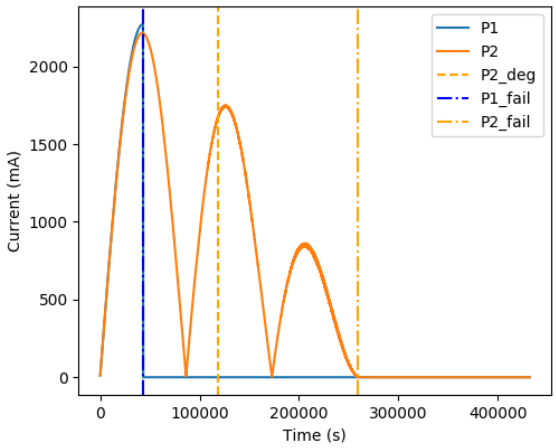


FIGURE 5.6 – Courant mesuré par les capteurs pour chaque panneau

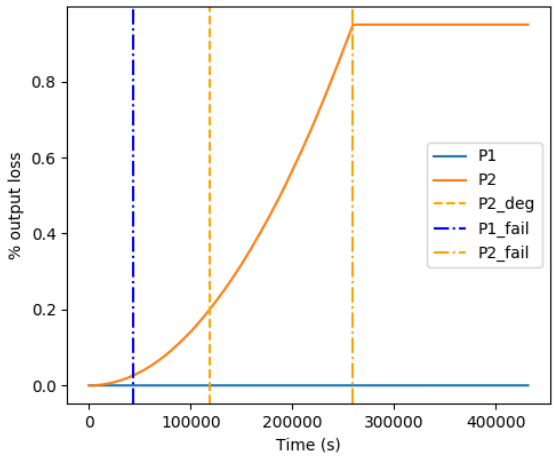


FIGURE 5.7 – Évolution de la dégradation pour chaque panneau jusqu'à la défaillance

5.4 Application de la méthode de diagnostic avancé

La méthode de diagnostic proposée dans le chapitre 4 a été appliquée sur différents scénarios simulés pour le modèle HtPN des deux panneaux défini précédemment. Pour l'ensemble de ces scénarios, les paramètres n_{suff} et n_{max} ont été mis à 100 et 200, respectivement. Cela signifie que 200 hypothèses maximum peuvent évoluer dans le diagnostiqueur, et qu'une hypothèse peut être choisie au maximum 100 fois par le processus de rééchantillonnage de la méthode de diagnostic. Le paramètre α, symbolisant la confiance dans les observations discrètes (c.-à-d. les événements discrets observés) par rapport aux observations continues (c.-à-d. la valeur des variables continues observée) est quant à lui mis à 0.5, signifiant qu'une confiance équivalente est accordée aux deux types d'observations.

5.4.1 Scénario 1 : cas nominal

Ce scénario est composé d'une alternance entre le cas où les deux panneaux sont éteints et le cas où les deux panneaux sont allumés. La figure 5.8 compare les résultats de simulation (en haut) et de diagnostic (en bas) obtenus après application du scénario 1 sur le modèle HtPN créé.

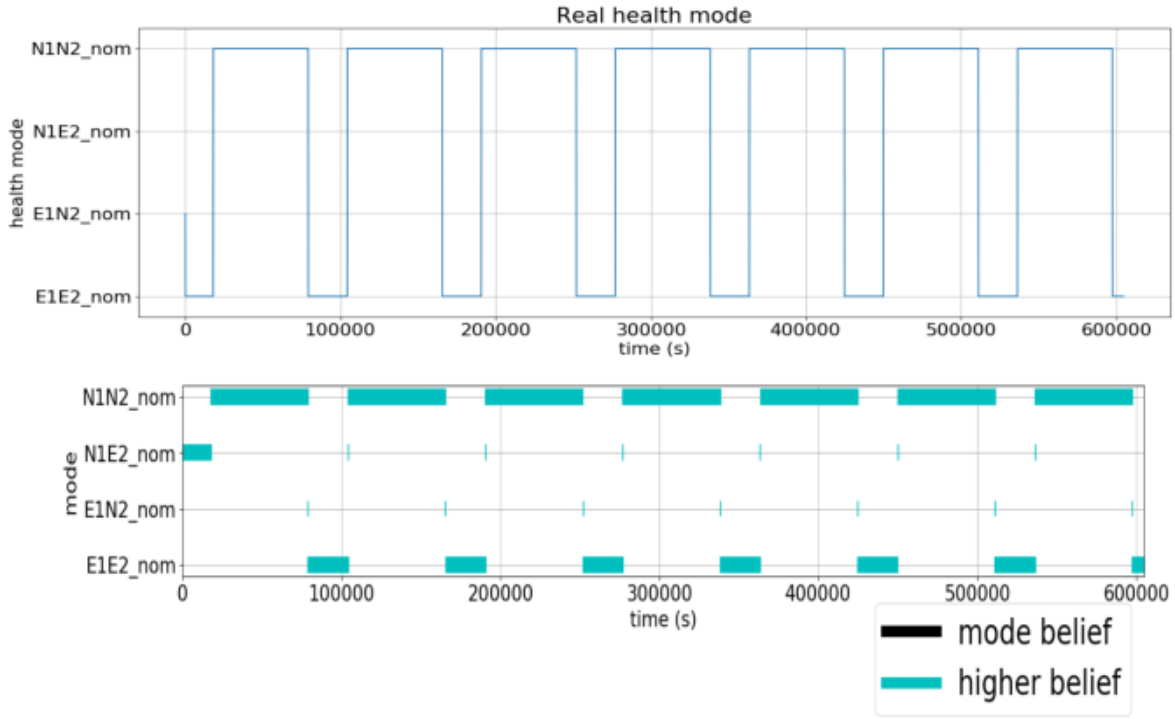


FIGURE 5.8 – Simulation et diagnostic pour le scénario 1 (nominal)

Sur cette figure, on peut distinguer le fait que pour le diagnostic, deux événements ne peuvent avoir lieu en même temps (cf hypothèse 3), par le bref passage dans les états $N1E2_{nom}$ et $E1N2_{nom}$ qui représente le fait que les événements $ON1$ et $ON2$ (ou $OFF1$ et $OFF2$) sont considérés l'un après l'autre et non pas simultanément. Dans ce scénario, les occurrences des événements discrets, associées aux variables continues évoluant ou mises à 0 (courant / tension : en cas de faute ou d'extinction d'un panneau, la variable

continue représentant le courant passe à 0), permettent à la méthode de diagnostic de suivre l'état de santé simulé du système.

5.4.2 Scénario 2 : insertion d'une faute menant à la défaillance

Dans ce scénario, des fautes fatales, entraînant la défaillance immédiate d'un panneau, sont introduites sur les deux panneaux. La première faute est introduite sur le panneau 1 à $t = 42500s$ environ, donnant lieu au passage du système vers un mode de défaillance appelé ici $M1N2_nom$. La deuxième faute fatale porte sur le deuxième panneau et est introduite à $t = 58000s$ environ, donnant lieu au passage du système vers un mode de défaillance appelé ici $M1M2_mort$. La figure 5.9 compare les résultats de simulation (en haut) et de diagnostic (en bas) obtenus après application du scénario 2 sur le modèle HtPN créé.

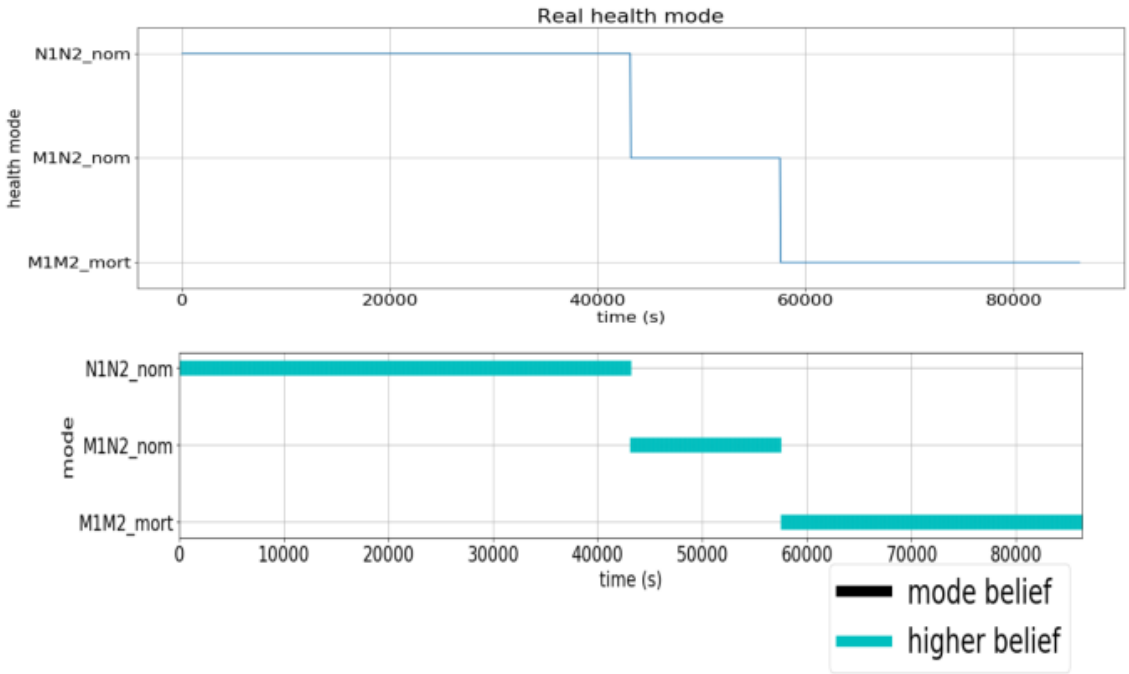


FIGURE 5.9 – Simulation et diagnostic pour le scénario 2 (fautes fatales)

Dans ce scénario, les variables continues nulles et l'absence d'événements discrets ont permis à la méthode de diagnostic de suivre l'état de santé simulé du système. L'absence d'événements discrets permet la distinction entre l'occurrence d'une faute et l'extinction nominale du panneau.

5.4.3 Scénario 3 : vieillissement des panneaux

Dans ce scénario, les panneaux restent allumés et le changement de mode de fonctionnement est dicté uniquement par le vieillissement et l'usure continus des panneaux photovoltaïques. Le premier panneau s'use et passe dans un mode dégradé autour de $t = 28000s$. Le second passe quant à lui dans un mode dégradé autour de $t = 38000s$. Les panneaux continuent de vieillir et passent dans des modes de défaillance à $t = 52000$ pour

le premier et $t = 75000s$ pour le second. La figure 5.10 compare les résultats de simulation (en haut) et de diagnostic (en bas) obtenus après application du scénario 3 sur le modèle HtPN créé.

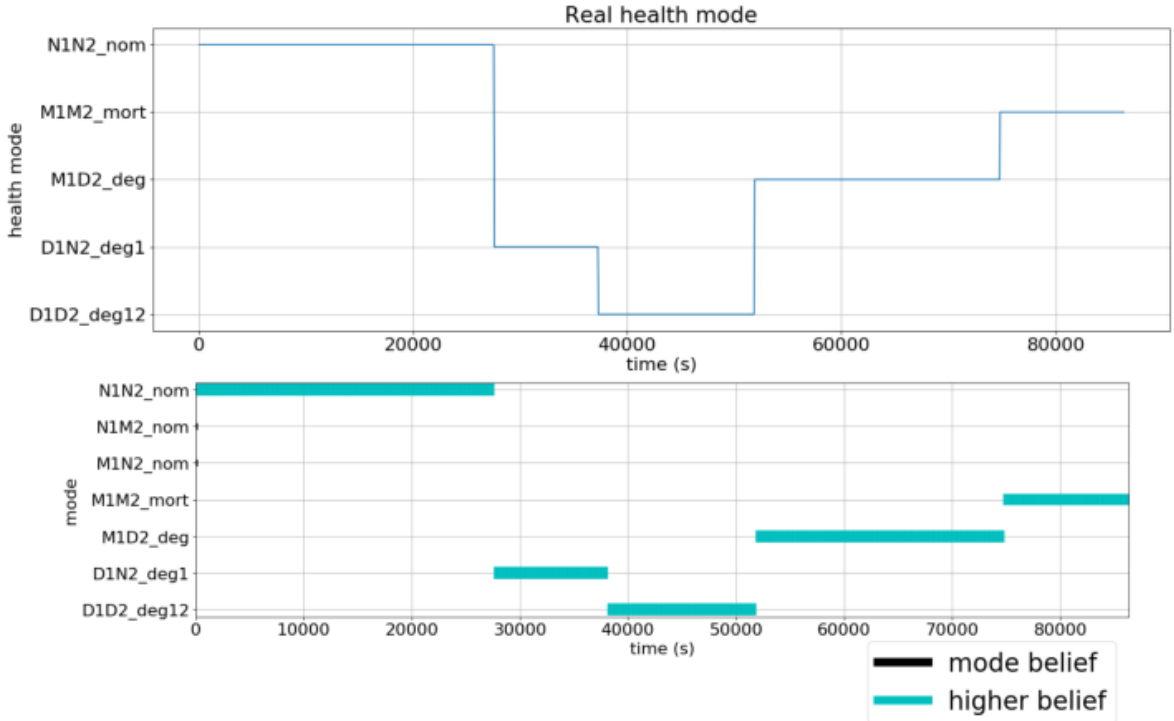


FIGURE 5.10 – Simulation et diagnostic pour le scénario 3 (usure des panneaux)

Dans ce scénario, la diminution progressive des valeurs des variables continues associées au courant produit par chaque panneau a permis à la méthode de diagnostic de suivre l'état de santé simulé du système.

5.4.4 Scénario 4 : fusion des trois scénarios précédents

Le diagnostiqueur donne des résultats qui sont satisfaisants, car correspondant aux scénarios simulés, sur les trois scénarios simples précédemment étudiés. Ces trois scénarios sont combinés dans un 4ième scénario afin de simuler et diagnostiquer un comportement réel des panneaux. Dans ce scénario, l'événement $OFF1$ est émis à $t = 11000s$ pour éteindre le premier panneau. Ce dernier est rallumé par l'événement $ON1$ à $t = 13000s$. L'événement $OFF2$ est ensuite émis à $t = 19000s$ pour éteindre le second panneau. Ce dernier est rallumé par l'événement $ON2$ à $t = 21000s$. L'usure des panneaux se fait ressentir par le passage du premier panneau dans un mode dégradé à $t = 24000s$. Le deuxième panneau va passer en mode dégradé à $t = 39000s$. Le premier panneau entre dans un état de défaillance de manière naturelle à $t = 52000s$. Le second panneau va lui entrer dans un état de défaillance suite à l'injection d'une faute fatale à $t = 74000s$. La figure 5.11 compare les résultats de simulation (en haut) et de diagnostic (en bas) obtenus après application du scénario 4 sur le modèle HtPN créé.

Le fait que l'ensemble de ces scénarios ne présente qu'une hypothèse s'explique d'une part par les valeurs attribuées aux paramètres (n_{suff} et n_{max}) et d'autre part par le fait

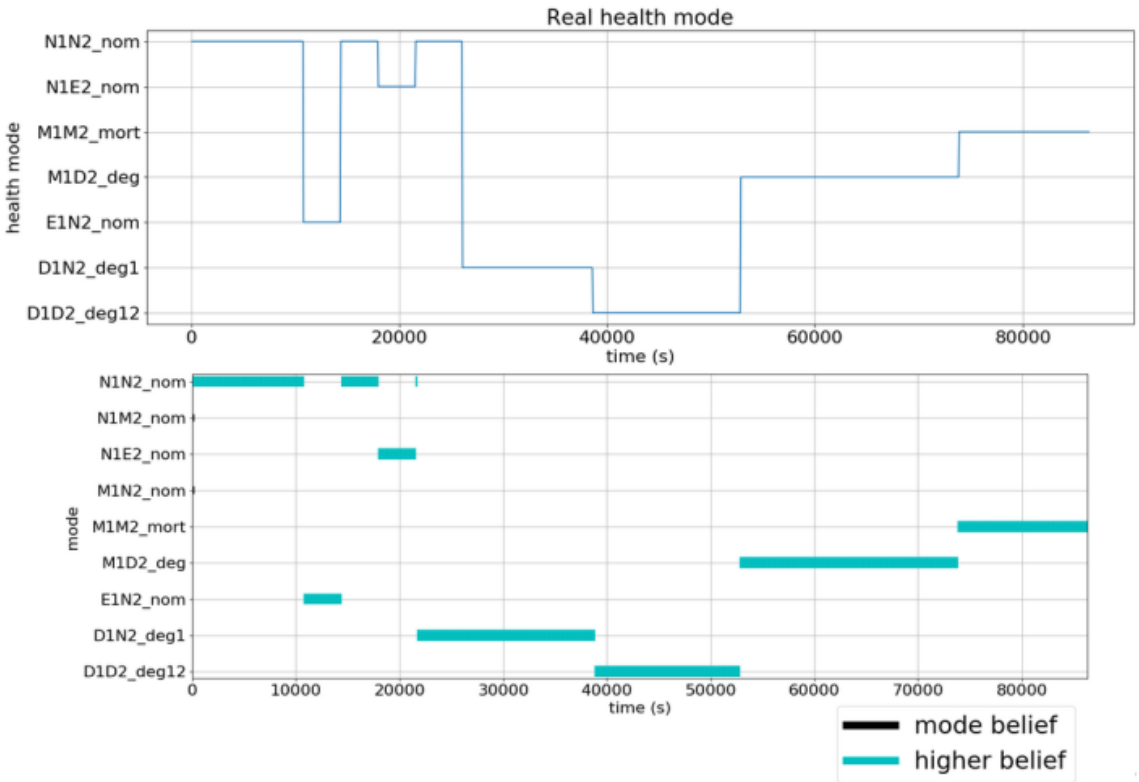


FIGURE 5.11 – Simulation et diagnostic pour le scénario 4 (nominal, fautes fatales et usure des panneaux)

que les dynamiques continues associées à chaque place sont très différentes. Sur une application avec des données réelles, il est peu probable que les résultats de diagnostic soient aussi précis. Néanmoins, ces résultats, globalement satisfaisants, ont permis de mettre en évidence des problèmes liés à la méthode de diagnostic. Le premier est un problème lié à la façon de gérer les incertitudes et à l'absence de "retour" possible après l'occurrence d'une faute (par une réparation, par exemple). En effet, la combinaison du pseudo-tirage, du rééchantillonnage et de l'impossibilité de revenir en arrière peut donner lieu à un "effet boule de neige" lors de la sélection des jetons et conduire à une perte d'hypothèses intéressantes (visible sur le diagnostic à $t = 21000s$ environ). Les concepts de clans, de métaplaces et de rééchantillonage préférencié ont été mis en place pour limiter au maximum l'occurrence de cet effet, mais un travail plus poussé serait nécessaire pour essayer de le faire complétement disparaître. Un deuxième problème porte sur la complexité de la méthode. Pour ce scénario, une heure était nécessaire pour diagnostiquer l'évolution du système sur un horizon d'un jour. Le diagnostic a été appliqué sur un scénario d'une durée d'un an et les résultats ont été obtenus au bout de 5 jours, ce qui est déjà conséquent pour le modèle considéré, relativement simple. Cette durée pose la question de la complexité de la méthode de diagnostic.

5.5 Application de la méthode d'apprentissage

L'objectif de cette section est d'obtenir un modèle représentatif des panneaux par apprentissage, en appliquant les concepts énoncés dans le chapitre 3. Étant donné que les

jeux de données disponibles ne couvrent pas forcément l'ensemble du domaine d'évolution du système, la régression polynomiale va être utilisée pour apprendre les dynamiques continues, grâce à sa capacité à extrapoler.

5.5.1 Données d'entrées

Différents jeux de données ont été obtenus à partir de la maquette. Ces jeux de données comportent les informations de tension, de courant, de puissance fournie, de température et de luminosité captée par le panneau. En tout, ce sont 21 caractéristiques qui ont été enregistrées dans la base de donnée. Parmi ces jeux de données, certains ont été obtenus en plaçant la maquette en extérieur et d'autres en utilisant le soleil artificiel en intérieur. Le jeu de données qui va servir à illustrer le processus d'apprentissage dans ce manuscrit a été obtenu en extérieur et présente une succession de fautes ayant lieu sur les panneaux 1 et 2. Ces fautes ont été injectées manuellement, via des interrupteurs qui déconnectent ou reconnectent un panneau donné. Il est obtenu sous la forme d'une matrice $data = [180; 21]$. Cette matrice symbolise le fait que le jeu de données est composé de 180 échantillons (180 lignes) possédant 21 caractéristiques (21 colonnes). Parmi ces caractéristiques, le courant mesuré est situé en 15ième colonne pour le panneau 1 et en 19ième colonne pour le panneau 2. La tension mesurée est quant à elle située sur les 13ième (panneau 1) et 17ième (panneau 2) colonnes. Le système est échantillonné environ chaque seconde, le temps est représenté sur la 21ième colonne. Les modes de fonctionnement sont représentés sur la figure 5.12. Cette figure est une représentation graphique des modes dans lesquels ont été obtenues les données. Le mode labellisé "2.00" correspond à un mode nominal où les deux panneaux fonctionnent normalement. Le mode labellisé "1.00" correspond à un mode dégradé dans lequel une faute est introduite sur le panneau $P1$ en déconnectant le panneau du système via un interrupteur. Enfin, le mode "0.00" correspond à un mode dégradé dans lequel une faute est introduite sur le panneau $P2$.

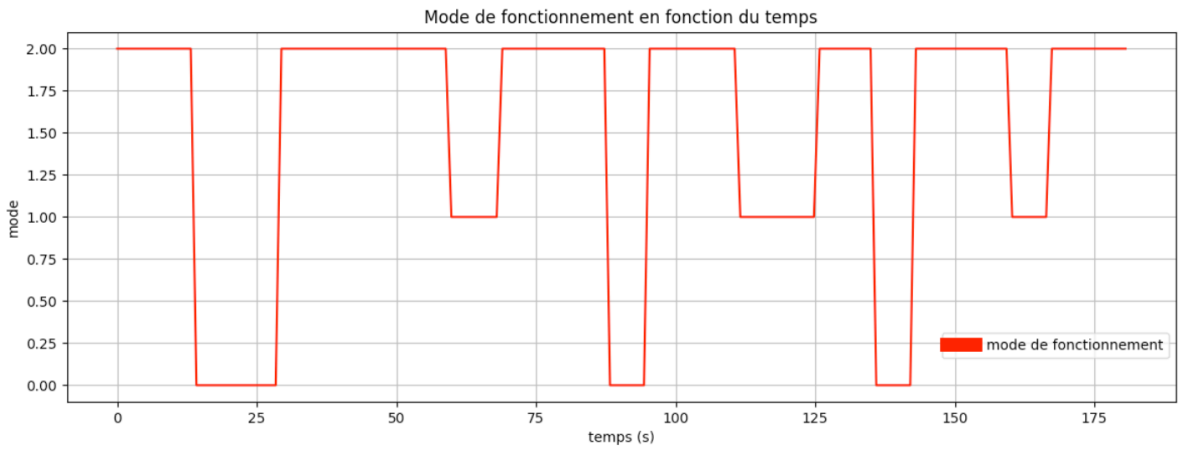


FIGURE 5.12 – Mode de fonctionnement du jeu de données considéré

Un aperçu des données obtenues est donné en figure 5.13. Le temps est représenté dans la dernière colonne. La tension et le courant du panneau 1 sont dans la 13ième et 15ième colonnes, respectivement. La tension et le courant du panneau 2 sont dans la 17ième et 19ième colonnes, respectivement.

```
0.00|0.00|0.00|37.49|35.04|25.76|54612.50|54612.50|0.02|-0.02|-2.00|0.00|4.93|3.84|384.00|1900.00|5.14|1.84|184.00|960.00|3.043
0.00|0.00|0.00|37.49|35.04|25.76|54612.50|54612.50|0.02|-0.02|-2.00|0.00|4.93|3.84|384.00|1900.00|5.13|1.86|186.00|960.00|4.058
0.00|0.00|0.00|37.49|35.04|25.76|54612.50|54612.50|0.02|-0.03|-3.00|0.00|4.93|3.84|385.00|1900.00|5.13|1.86|186.00|960.00|5.072
0.00|0.00|0.00|37.49|35.04|25.76|54612.50|54612.50|0.02|-0.03|-3.00|0.00|4.93|3.84|384.00|1900.00|5.14|1.86|186.00|960.00|6.087
```

FIGURE 5.13 – Exemple de données disponibles dans le jeu de données

5.5.2 Prétraitement des données

Les données obtenues sont prétraitées suivant la méthode exposée en section 3.3.3 page 59 avant d'être fournies à la méthode d'apprentissage. Aucun événement discret n'est présent dans les données, il n'y aura donc pas d'ajout de nouvelles colonnes. De plus, la dernière colonne, le temps, n'est pas normalisée. La figure 5.14 montre les données présentées dans la figure 5.13 après leur prétraitement.

```
1.0, 1.0, 1.0, 0.08362, 0.2851, 0.5013, 1.0, 1.0, 1.0, 0.6667, 0.6667, 1.0, 0.3587, 0.9974, 0.9974, 1.0, 0.113, 0.9641, 0.9641, 0.9796, 3.043
1.0, 1.0, 1.0, 0.08362, 0.2851, 0.5013, 1.0, 1.0, 1.0, 0.6667, 0.6667, 1.0, 0.3587, 0.9974, 0.9974, 1.0, 0.1043, 0.9744, 0.9744, 0.9796, 4.058
1.0, 1.0, 1.0, 0.08362, 0.2851, 0.5013, 1.0, 1.0, 1.0, 0.3333, 0.3333, 1.0, 0.3587, 0.9974, 1.0, 1.0, 0.1043, 0.9744, 0.9744, 0.9796, 5.072
1.0, 1.0, 1.0, 0.08362, 0.2851, 0.5013, 1.0, 1.0, 1.0, 0.3333, 0.3333, 1.0, 0.3587, 0.9974, 0.9974, 1.0, 0.113, 0.9744, 0.9744, 0.9796, 6.087
```

FIGURE 5.14 – Exemple de données prétraitées

Une fois les données prétraitées, la phase d'apprentissage structurel peut commencer.

5.5.3 Apprentissage structurel du modèle HtPN

5.5.3.1 Obtention des clusters par DyClee

Les données prétraitées sont fournies à DyClee afin d'obtenir la structure du modèle. La taille des micro-clusters de DyClee a été fixée expérimentalement à 0.8. Les clusters obtenus par DyClee sont visibles sur la figure 5.15.

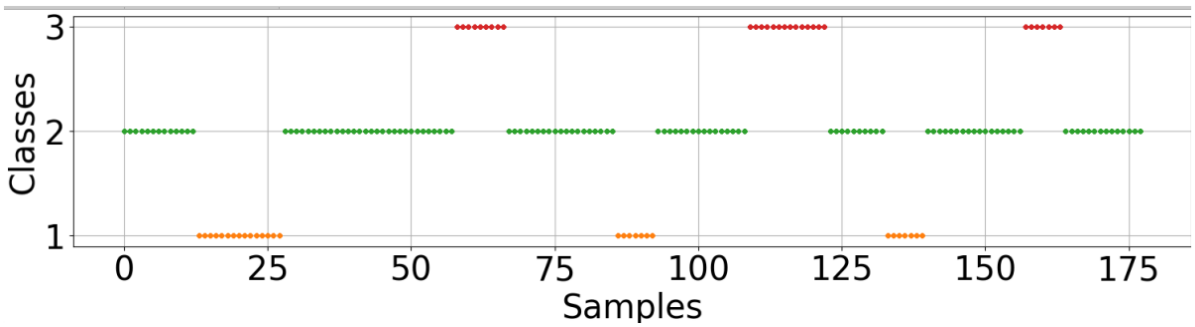


FIGURE 5.15 – Clusters obtenus par DyClee avec une taille de micro-clusters = 0.8

En comparant avec la figure 5.12, on constate qu'avec la taille de micro-clusters choisie, DyClee est capable de reconnaître les différents modes de fonctionnement. Le cluster 2 peut être assimilé au mode de fonctionnement nominal, le cluster 1 au mode dégradé dans lequel le panneau $P2$ est fautif et le cluster 3 au mode dégradé dans lequel le panneau $P1$ est fautif.

5.5.3.2 Interprétation des résultats de DyClee

Les résultats fournis par DyClee sont ensuite interprétés et convertis en une structure HtPN, composée de 4 places (une par cluster et une d'initialisation créée pour gérer le régime transitoire du système (cf section 3.3.5)) et 4 transitions pour représenter les différentes évolutions entre ces modes. La structure obtenue est visible sur la figure 5.16.

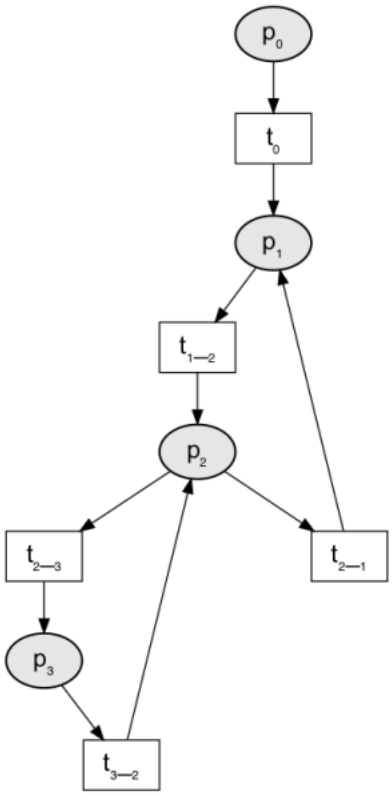


FIGURE 5.16 – Structure HtPN obtenue après interprétation des résultats de DyClee

À l'issue de cette étape, le modèle HtPN obtenu comporte les ensembles suivants :

— $P = \{P_0, P_1, P_2, P_3\}$
— $T = \{t_0, t_{1-2}, t_{2-1}, t_{2-3}, t_{3-2}\}$
— Pre et $Post$, de dimension $(4, 5)$, sont obtenues partiellement (l'existence des éléments non vides est connue).

Une fois la structure discrète obtenue, l'apprentissage des dynamiques continues va pouvoir commencer.

5.5.4 Apprentissage des dynamiques continues

Comme explicité plus haut, l'algorithme de RP a été choisi dans ce cas pour sa capacité à extrapoler. Un degré 5 a été choisi dans un premier temps pour avoir une très bonne correspondance entre les données et le modèle appris (au risque d'un sur-apprentissage), mais une étude sur l'équilibre complexité/qualité serait intéressante à effectuer. L'apprentissage des dynamiques d'évolution du courant et de la tension de chaque panneau a été

effectué. Seuls les résultats du panneau 2 sont montrés ici, étant identiques à ceux du panneau 1. La figure 5.17 montre les résultats de l'apprentissage pour l'évolution du courant sur l'ensemble du jeu de données considéré en concaténant les dynamiques apprises pour chaque place comparé au jeu de données réel. On constate que le courant modélisé correspond au courant réel dans chaque mode.

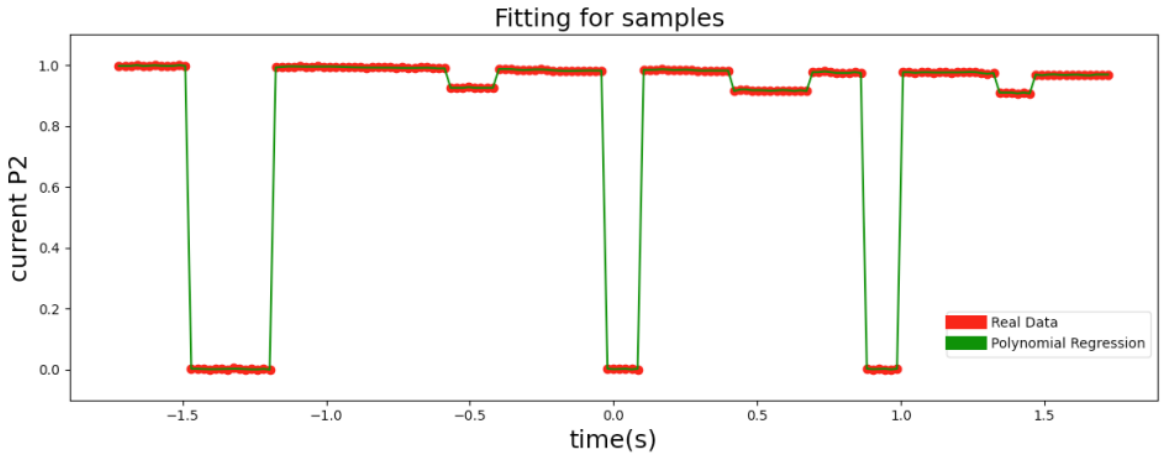


FIGURE 5.17 – Courant appris pour l'ensemble du jeu de données

Le coefficient de détermination R^2 global obtenu avec l'algorithme de régression polynomiale de degré 5 pour la dynamique d'évolution du courant est de 0.999, ce qui est très satisfaisant. Le R^2 global est obtenu par concaténation des différents modèles appris pour obtenir un modèle appris sur la globalité du système.

La figure 5.18 montre les résultats de l'apprentissage pour l'évolution de la tension sur l'ensemble du jeu de données considéré. Sur cette figure, le mécanisme de ré-apprentissage en cas de non-correspondance (cf section 3.3.5) est désactivé.

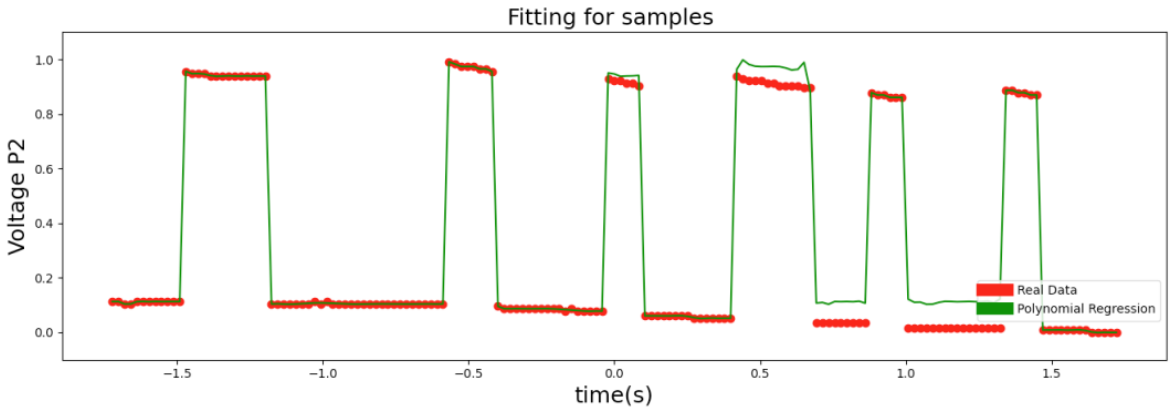


FIGURE 5.18 – Tension apprise pour l'ensemble du jeu de données, sans intervention du mécanisme de ré-apprentissage

Le coefficient de détermination R^2 global obtenu avec l'algorithme de régression polynomiale de degré 5 pour la dynamique d'évolution de la tension est de 0.980, ce qui est satisfaisant. Néanmoins, on peut constater qu'il existe des endroits où la correspondance n'est pas tout à fait correcte. Le mécanisme de ré-apprentissage est donc appliqué aux sous-parties du jeu de données présentant un coefficient de détermination inférieur

à 0.8. La figure 5.19 montre les résultats obtenus après application du mécanisme de ré-apprentissage.

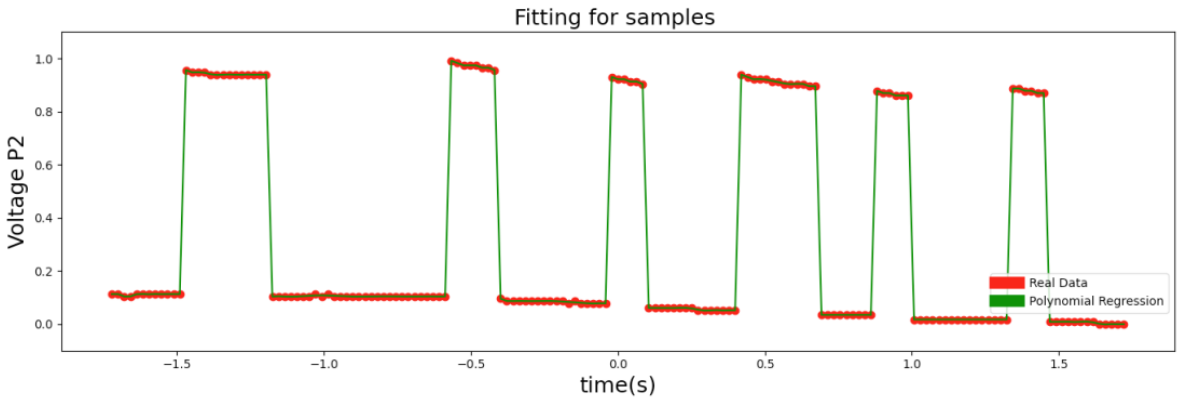


FIGURE 5.19 – Tension apprise pour l'ensemble du jeu de données, avec intervention du mécanisme de ré-apprentissage

Le coefficient de correspondance R^2 nouvellement obtenu est de 0.999.

L'application de la méthode d'apprentissage à ce cas réel donne des résultats satisfaisants, qui témoignent de l'applicabilité de la méthode conçue à des applications réelles.

Afin de vérifier la validité du modèle appris, la méthode de diagnostic avancé a été appliquée sur ce dernier.

5.5.5 Application de la méthode de diagnostic au modèle appris

La méthode de diagnostic a tout d'abord été appliquée au modèle obtenu par le processus d'apprentissage incluant le mécanisme de réapprentissage.
Le modèle ainsi appris est composé de 10 places ($P1$, ..., $P10$) et de 12 transitions (t_1, ..., t_{12}). Chacune des places Pi, $i = 1...10$ est associée à des dynamiques continues C_i régissant l'évolution du courant émis par chaque panneau en fonction du temps passé dans la place. Le vecteur d'état est un vecteur de taille 3 : $x = [x_0, x_1, x_2]$ où $x[0]$ correspond au temps écoulé depuis l'arrivée dans la place p et $x[1]$ et $x[2]$ les courants produits par les panneaux 1 et 2, respectivement. Ces dynamiques continues sont obtenues par l'algorithme RP. Par exemple, la dynamique C_1 associée à la place $P1$ est :

$$C_1 = \begin{cases} x_{0,k+1} = x_{0,k+1} \\ x_{1,k+1} = 380 + 0.6833x_{0,k} - 0.6445x_{0,k}^2 - 1.04x_{0,k}^3 \\ \qquad + 0.3168x_{0,k}^4 + 0.2536x_{0,k}^5 \\ x_{2,k+1} = 180 + 0.0718x_{0,k} + 0.1588x_{0,k}^2 - 0.01716x_{0,k}^3 \\ \qquad - 0.0452 * x_{0,k}^4 - 0.0127 * x_{0,k}^5 \end{cases} \tag{5.2}$$

Le seuil de réapprentissage est fixé à $R^2 = -20$. Cela signifie que si le score R^2 d'un modèle appris précédemment pour un ensemble d'échantillons correspondant au cluster i est inférieur à -20 lors de l'application sur un autre ensemble d'échantillons affectés au cluster, alors le mécanisme de réapprentissage sera utilisé pour apprendre une nouvelle dynamique sur un autre jeu de données affecté au cluster i. Dans un premier temps, une

dynamique est apprise par cluster, on obtient donc $P1$, $P2$ et $P3$. Ensuite, 7 nouvelles places ont été créées par le mécanisme de réapprentissage, avec les transitions correspondantes. $P5$, $P4$, $P8$ et $P10$ sont ainsi une variation de $P2$ (elles sont toutes issues du cluster "2" de Dyclee). De même, $P7$ et $P9$ sont une variation de $P3$, et $P6$ est une variation de $P1$. La représentation structurelle du modèle appris est visible sur la figure 5.20.

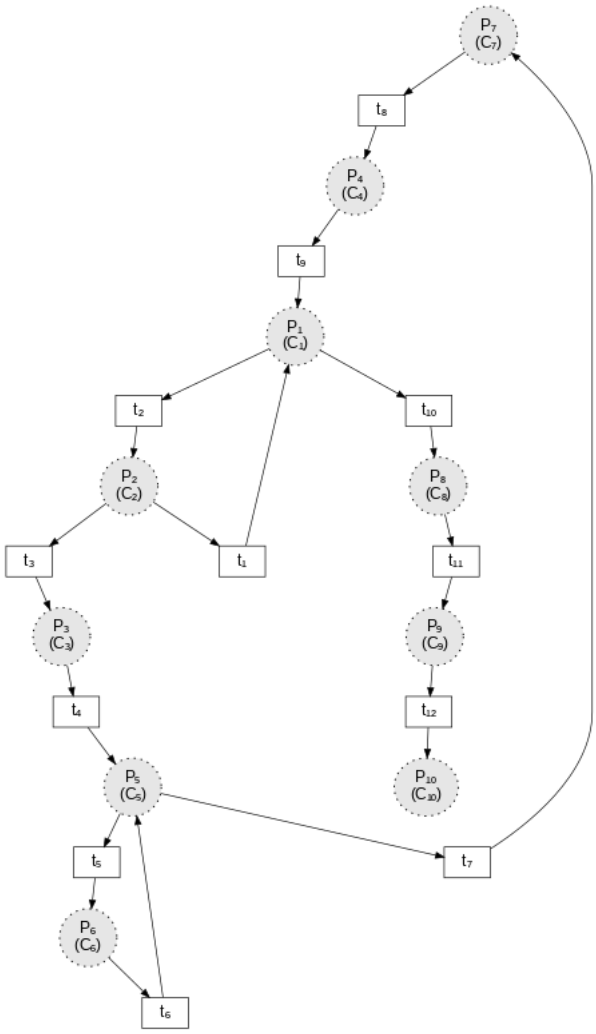


FIGURE 5.20 – Représentation structurelle du HtPN appris avec réapprentissage

Les données d'entrées utilisées pour l'apprentissage ont été fournies au diagnostiqueur. Les paramètres suivants ont été fournis : $n_{suff} = 20$, une hypothèse sera rééchantillonnée au maximum 20 fois, $n_{max} = 100$, 100 hypothèses maximum évolueront en même temps et $\alpha = 0$ pour accorder une confiance totale aux informations continues, aucune observation discrète n'étant présente. En comparant les places aux échantillons sur lesquels les apprentissages sont effectués, les résultats attendus sont présentés dans le tableau 5.2.

Les résultats du diagnostiqueur sont visibles sur la figure 5.21. En comparant les résultats du diagnostic aux résultats attendus, on se rend compte que ces derniers ne correspondent pas. Cependant, si on pousse l'analyse plus loin et qu'on croise avec les clusters dont sont extraites les places ajoutées (tableau 5.2), on se rend compte que les

TABLEAU 5.2 – Correspondance entre échantillons et place attendue

Échantillons	Place attendue
1-13	P2
14-28	P1
29-58	P2
59-67	P3
68-86	P5
87-93	P6
94-109	P5
110-123	P7
124-133	P4
134-140	P1
141-156	P8
157-164	P9
165-180	P5

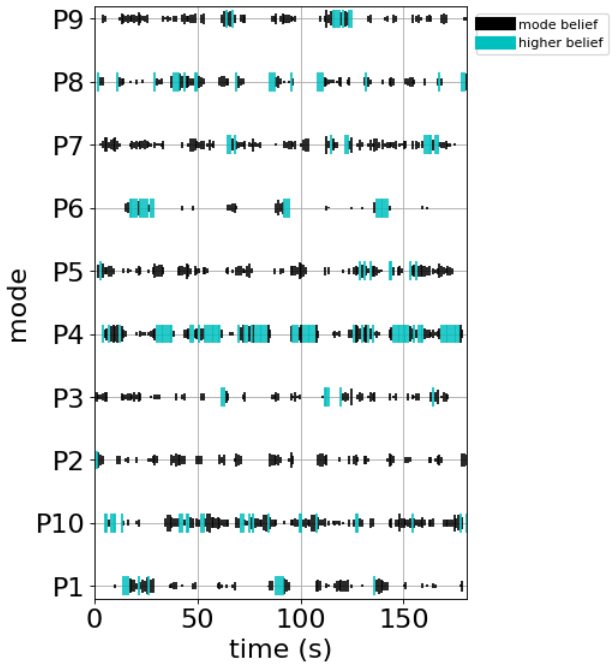


FIGURE 5.21 – Diagnostic du scénario pour le modèle avec réapprentissage

confusions présentes le sont entre des places originellement issues du même cluster (par exemple, pour les échantillons 0-13, les places ayant le plus grand score sont $P2$, $P5$ et $P4$, qui correspondent toutes initialement à $P2$). Ce diagnostic erroné est issu du mécanisme de réapprentissage. Cela peut s'expliquer par plusieurs raisons. Tout d'abord, il y a peu d'échantillons par cluster, ce qui entraîne facilement une erreur plus importante au niveau du score R^2. L'ordre choisi pour la régression polynomiale peut aussi être une explication pour ce réapprentissage. Comme dit dans le chapitre 3, l'algorithme de régression polynomiale est prompt au sur-apprentissage. Le choix du seuil de réapprentissage doit donc tenir compte de ces paramètres.

Afin de valider la cause de ce diagnostic erroné, le modèle a été appris sans utiliser le mécanisme de réapprentissage. On obtient ainsi un modèle composé de 3 places uniquement, $P1$, $P2$ et $P3$, correspondant aux trois clusters identifiés par DyClee et associées aux dynamiques apprises C_i. 4 transitions sont présentes dans le modèle appris, il n'y a pas de transition directe entre $P1$ et $P3$. La représentation structurelle du modèle HtPN "simple" appris est visible sur la figure 5.22.

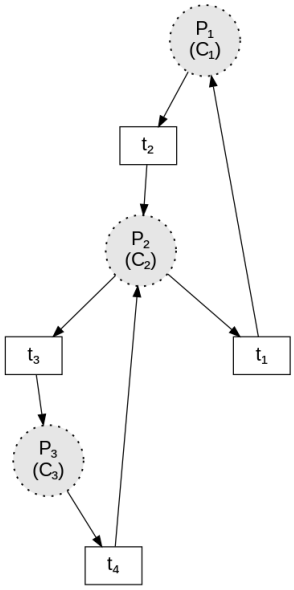


FIGURE 5.22 – Représentation structurelle du HtPN appris sans réapprentissage

Le même scénario a été fourni en entrée de la méthode de diagnostic, avec les mêmes paramètres ($n_{suff} = 20$, $n_{max} = 100$ et $\alpha = 0$). Les résultats obtenus sont visibles sur la figure 5.23.

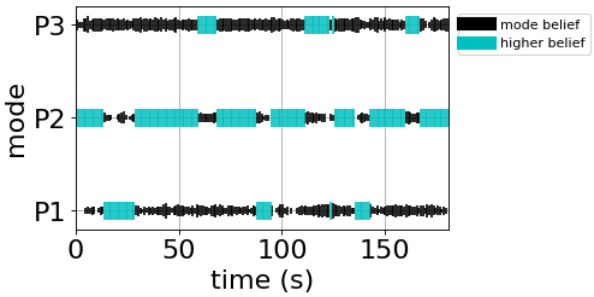


FIGURE 5.23 – Diagnostic du scénario pour le modèle sans réapprentissage

En comparant de nouveau avec la figure 5.12, on constate que la place ayant le plus grand score correspond au mode de fonctionnement identifié lors de l'acquisition des données, ce qui donne un résultat satisfaisant quant à, d'une part la correspondance modèle appris / données et d'autre part l'application de la méthode de diagnostic sur un modèle appris.

5.6 Conclusion

Ce chapitre illustre les différentes contributions proposées par cette thèse sur une maquette réelle composée de deux panneaux photovoltaïques reliés à différentes charges et pouvant être éclairés par un soleil artificiel ou par lumière naturelle. Il a pour but de montrer l'applicabilité des méthodes proposées à un système réel. Le système a tout d'abord été modélisé sous le formalisme des HtPN. Cette formalisation suit la méthode proposée dans la section 2.4.1 page 33. Un modèle représentant un seul panneau a été réalisé dans un premier temps. Le modèle global des deux panneaux a ensuite été obtenu par composition des deux modèles de chaque panneau. La méthode de diagnostic avancé a été appliquée sur le modèle réalisé dans un premier temps, à partir de divers scénarios simulés. Cette application a pu exacerber l'hypothèse 3 page 86 comme quoi un seul événement pouvait survenir en même temps pour la méthode de diagnostic. Enfin, à partir de données acquises sur la maquette, la méthode d'apprentissage a été appliquée et un modèle a été obtenu. Le modèle obtenu contient plus de places qu'attendues par rapport aux modes de fonctionnement identifiés lors de l'acquisition des données, dû au mécanisme de ré-apprentissage. La méthode de diagnostic avancé a d'abord été appliquée au modèle appris afin de vérifier la validité de ce dernier, puis au modèle appris sans considérer le mécanisme de réapprentissage, qui correspond plus aux modes de fonctionnement identifiés. En comparant les échantillons affectés à chaque place, les résultats de diagnostic obtenus ont pu être comparés à l'attendu. Cette application de la méthode de diagnostic sur le modèle obtenu par apprentissage et la comparaison des résultats obtenus ont pu mettre en lumière la nécessité d'un seuil de réapprentissage précis, afin de ne pas obtenir un nombre de places élevé ayant des dynamiques proches, ainsi que certains paramètres intervenant dans le choix de ce seuil, comme la quantité de données disponibles ou l'ordre du polynôme recherché par la régression polynomiale. Globalement, les résultats obtenus sont intéressants et prometteurs quant à l'utilité des méthodes proposées. L'application des méthodes sur un cas réel a pu mettre en évidence leur intérêt, mais aussi certaines limites, comme la complexité liée à la méthode de diagnostic.

La maquette choisie est d'autant plus intéressante qu'elle s'inscrit dans une thématique écologique importante.

Chapitre 6

Conclusion et perspectives

Conclusion

L'objectif de cette thèse était l'obtention d'un modèle par apprentissage, permettant d'effectuer du suivi de santé pour des systèmes hybrides sous incertitude.

La première partie du travail s'est focalisée sur la notion de modèle. Les systèmes auxquels on s'intéresse sont des systèmes hybrides, définis comme des systèmes pouvant être divisés en différents sous-systèmes pouvant communiquer entre eux et de différents types (SED, continu, hybrides classiques). Ces systèmes sont soumis à des incertitudes. Pour les représenter sous forme mathématique, **le formalisme des *Heterogeneous Petri Nets* (HtPN) a été développé**. Ce formalisme est une extension des réseaux de Petri classiques. Il permet de représenter des systèmes complexes, purement discrets, purement continus ou hybrides. Les incertitudes de modélisation, ainsi que celles liées aux observations, peuvent être prises en compte par le formalisme pour la modélisation et la simulation.

Un HtPN est composé de places, qui représentent l'information discrète, auxquelles sont associées des dynamiques continues et de dégradation, et de transitions. Les places contiennent un ou plusieurs jetons, qui portent une information discrète, continue et de dégradation. Le tir de ces transitions est conditionné par un ensemble de conditions situé en amont de la transition et peuvent avoir un effet sur les informations portées par les jetons à travers un ensemble d'assignations situé en aval de la transition. Le formalisme a l'avantage d'être intuitif de par sa structure similaire à un réseau de Petri classique et puissant par sa capacité à représenter des mécanismes d'évolutions complexes liés aux systèmes vieillissants. Une méthodologie permettant de modéliser un système sous la forme d'un HtPN a été proposée. Cette méthodologie passe par l'identification des différents modes de fonctionnements du système, nominaux ou dégradés, et par l'ajout de complément d'information. Le formalisme a été utilisé pour modéliser et simuler le comportement de deux systèmes distincts, un système de production issu de la littérature, qui montre la possibilité pour le formalisme de faire du contrôle et de représenter les mêmes informations qu'un réseau de Petri classique, et un système hybride axé gestion de santé, représentant des cuves d'eau connectées. Ce dernier système montre la capacité du formalisme à représenter non seulement un système hybride, mais également des fonctions de dégradation, qui peuvent représenter le vieillissement d'un système et à en tenir compte pour simuler l'occurrence de fautes.

Une fois le formalisme défini et pour pallier les difficultés de modélisation par voie directe des systèmes complexes, cette thèse propose **une méthode d'apprentissage à base de données pour obtenir un modèle sous le formalisme des HtPN**.

Le processus d'apprentissage se découpe en trois étapes. La première étape consiste en un prétraitement des données. Elle inclut une conversion des événements et une étape de normalisation. La deuxième étape est la méthode d'apprentissage, composé de l'apprentissage structurel et de l'apprentissage des dynamiques continues. L'apprentissage structurel consiste en l'obtention de la structure discrète du modèle (les places et les transitions) et se base sur l'interprétation des résultats d'un algorithme de clustering. La seconde partie se focalise sur l'obtention des dynamiques continues associées aux places, à travers trois algorithmes de régression différents : la régression basée sur les machines à vecteurs supports, la régression basée sur des forêts d'arbres aléatoires et la régression polynomiale. Les performances de ces algorithmes sont évaluées via une mesure de correspondance et le temps de calcul nécessaires, afin que l'utilisateur puisse choisir celui qui correspond le mieux à ses besoins. La troisième et dernière étape se concentre sur l'obtention des conditions continues simples de tir des différentes transitions. Elle se base sur une étude de la variance des variables observées lors des différents tirs de la transition afin d'isoler la ou les variables intervenant dans le processus de tir. La méthode d'apprentissage a été appliquée au modèle théorique des trois cuves d'eau, afin de vérifier sa capacité à obtenir des modèles corrects. Cette application a montré que la méthode était robuste dans le cas où le système était dans un régime permanent mais baissait cependant en précision dans le cas où des données correspondaient à des phases transitoires, comme des phases d'initialisation.Le mécanisme de réapprentissage a été ensuite ajouté à la méthode pour pallier cette baisse de précision.
Enfin, **une méthode de diagnostic avancé appliquée au formalisme des HtPN a été développée**. À partir du modèle HtPN, un diagnostiqueur basé HtPN est construit. Les différentes hypothèses sur l'état de santé du système au cours du temps sont représentées par les différents jetons présents dans le diagnostiqueur, utilisant ainsi la capacité du formalisme des HtPN à représenter le parallélisme.

La méthode de diagnostic se découpe en deux phases distinctes. La première phase, la prédiction, utilise le modèle HtPN pour émettre des hypothèses quant aux états futurs possibles du système, qui sont représentées par différents jetons évoluant en parallèle dans le modèle. La notion de pseudo-tirage, centrale dans la sémantique des HtPN, est utilisée dans la phase de prédiction afin de prendre en compte l'incertitude sur les observations. La seconde phase est une phase de correction. Elle se base sur les observations discrètes et continues disponibles sur le système, ainsi que sur les futures commandes, pour valider ou réfuter les hypothèses. Ce choix s'effectue par le calcul d'un score global, qui représente la correspondance entre les informations portées par un jeton et les observations. Le score global est calculé à partir d'une somme pondérée entre deux scores. Le premier est le score discret, compris entre 0 et 1, représentant la correspondance entre la configuration du jeton (l'information discrète) et les observations discrètes. Le second est le score continu, compris entre 0 et 1, représentant la correspondance entre l'état continu du jeton (l'information continue) et les observations continues. Pour ces deux scores, plus le score est proche de 1, plus les informations portées par le jeton correspondent aux observations. Les jetons sont regroupés en clans en fonction de leur score ce qui limite le nombre d'entités à traiter par la méthode de rééchantillonnage. Le concept orignal de métaplace, rassemblant les places présentant des observables identiques, intervient aussi dans le processus de rééchantillonnage pour forcer une répartition équitable entre les places indiscernables.

La méthode de diagnostic est appliquée sur trois systèmes, un SED, un système continu et un système hybride, afin de montrer l'applicabilité de la méthode à différents types de système. Cette application a montré que la méthode de diagnostic était capable de donner des résultats satisfaisants pour les trois types de système.
L'ensemble de ces contributions ont été implémentées dans un logiciel codé en python, nommé HeMU. Ce logiciel permet donc de modéliser, simuler, apprendre et diagnostiquer des systèmes sous le formalisme des HtPN.
Le travail effectué a été appliqué sur un cas réel, élaboré par nos soins. Le système considéré est un système composé de deux panneaux photovoltaïques et de diverses charges (deux lampes et un moteur). Dans un premier temps, un modèle des deux panneaux photovoltaïques a été identifié à la main et représenté sous le formalisme des HtPN. Dans un deuxième temps, la méthode de diagnostic avancé a été appliquée sur ce modèle et donne des résultats correspondant aux scénarios simulés. Enfin, des jeux de données ont été obtenus et utilisés pour apprendre un modèle du système. Le modèle appris avec le jeu de données considéré et le mécanisme de réapprentissage possède plus de modes de fonctionnement que les différents modes de fonctionnement identifiés lors de l'acquisition. Sans le mécanisme de réapprentissage, le modèle correspond aux modes de fonctionnement identifiés. La méthode de diagnostic a été appliquée aux deux modèles appris, avec et sans mécanisme de réapprentissage. Cette application a montré que le seuil de réapprentissage choisi devait être étudié plus en profondeur, afin de ne pas complexifier le modèle appris avec de nouvelles places ayant des dynamiques similaires aux places déja existantes. Cette application a permis de montrer l'applicabilité des travaux effectués à un cas réel.

Perspectives

Le travail effectué ouvre la piste à de nombreuses améliorations et potentiels futurs travaux à court et long termes.

Perspectives à court terme

Bien que le modèle appris en l'état actuel puisse servir directement comme modèle d'entrée à la méthode de diagnostic avancé, une première perspective à court terme serait de compléter l'apprentissage du formalisme. L'application de la théorie pour l'obtention des ensembles de conditions, l'obtention de l'ensemble des événements E (récupérable lors de la phase de traitement des données), la théorisation et l'obtention des ensembles d'assignations sont notamment des choses à faire dans un travail futur.
L'obtention de l'ensemble des événements E peut s'effectuer lors de la phase de traitement des données, plus précisément lors de la conversion des événements en colonnes, où une association colonne / événement peut être faite.
Une idée pour obtenir l'ensemble d'assignations serait d'étudier les variations des variables continues lors du tir d'une transition et de les comparer avec les variations induites par les dynamiques continues apprises. Si la variation induite par le tir est différente des dynamiques continues auxquelles étaient soumises les variables, on peut supposer que la variation est due à l'ensemble d'assignations.

Une troisième perspective à court terme porte sur l'implémentation de la méthode

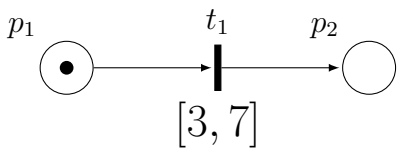


FIGURE 6.1 – Exemple d'un Réseau de Petri temporel

de pronostic, qui peut être basée sur la méthode de pronostic appliquée au formalisme des HPPN définie dans la thèse de GAUDEL 2016. Cette dernière pourrait se baser sur les dynamiques de dégradation, présentes, mais non utilisées par le diagnostiqueur, pour prévoir de futures fautes. En effet, en connaissant la distribution des probabilités d'occurrence de la faute et les dynamiques de dégradation, il est possible de prévoir l'évolution de la dégradation de l'hypothèse la plus probable du diagnostiqueur et d'en déduire une durée de vie résiduelle avant l'occurrence d'une faute.

Une quatrième perspective porte sur la méthode de diagnostic et la possibilité de faire du diagnostic en ligne, c.-à-d. pendant que le système est en cours de fonctionnement. Actuellement, la façon dont les fichiers d'entrées sont fournis ne permet qu'une implémentation hors-ligne de la méthode de diagnostic, ce qui en limite l'aspect pratique. Une utilisation en-ligne augmenterait aussi grandement l'intérêt de la création de la méthode de pronostic.
De plus, une étude de l'impact des paramètres n_{suff} et n_{max} sur le temps nécessaire pour diagnostiquer et la qualité du diagnostic obtenu pourrait s'avérer intéressante.

Perspectives à moyen ou long terme

Un premier axe de perspective à long terme porte sur l'amélioration du formalisme. En effet, bien que le formalisme englobe de nombreux autres formalismes, il est limité dans certains cas. L'aspect stochastique ou l'incertitude temporelle qu'on peut retrouver dans les réseaux de Petri temporels de BERTHOMIEU et al. 1983 ne sont par exemple pas représentables pour le moment, car une transition est tirée dès qu'elle est validée, et nécessiteraient des travaux plus approfondis. Par exemple, dans le réseau de Petri temporel présenté dans la figure 6.1, la transition t_1 peut être tirée entre 3 et 7 unités de temps écoulées depuis le marquage de p_1. Cette incertitude sur la date de tir de t_1 n'est à ce jour pas représentable avec le formalisme créé. La représentation de ces incertitudes, peut-être via des conditions numériques spécifiques, est une piste pour de futurs travaux sur le formalisme. Une autre piste d'amélioration porte aussi sur l'étude des propriétés propres aux réseaux de Petri sur les HtPN.

Un deuxième axe de perspective à long terme concerne l'amélioration de la méthode d'apprentissage. Tout d'abord, il serait intéressant d'étudier des moyens d'apprentissage de conditions plus complexes que celles étudiées actuellement.
De plus, l'obtention de l'ensemble des dynamiques de dégradation D nécessite un travail plus profond. La difficulté de l'apprentissage de ces dynamiques est principalement située au niveau des fenêtres de temps considérées, ces dynamiques étant beaucoup plus lentes. De plus, au niveau de l'obtention des données, il est difficile, et peut être coûteux, d'effectuer des *runs-to-failure*, c.-à-d. de laisser vieillir le système jusqu'à l'occurrence d'une

faute. Il serait intéressant d'étudier la possibilité d'apprendre des dynamiques de dégradation sans avoir la nécessité d'avoir des données contenant une faute.
La possibilité d'identifier la présence de différents jetons évoluant en parallèle dans les données est une autre idée qui mériterait d'être creusée. En effet, afin d'avoir un apprentissage correct, un seul jeton doit être représenté dans les données, afin de ne pas fausser les places apprises.
Par ailleurs, on parle ici de modèle appris "from scratch". Une autre perspective est de se placer dans le cadre de l'amélioration de modèles, et d'étudier comment des observations en ligne peuvent permettre de modifier le modèle courant d'un système sous surveillance. L'étude de la diagnosticabilité et de la pronosticabilité du modèle, et le déclenchement de l'apprentissage en fonction de ces propriétés est aussi une source de travaux intéressante. L'étude de l'apprenabilité, c.-à-d. la possibilité d'obtention d'un modèle précis par apprentissage, est un concept qui pourrait être étudié plus précisément. Il en est de même pour la relation entre quantité / qualité de données nécessaires pour obtenir un modèle intéressant en donnant des métriques pertinentes.
Enfin, une étude sur le choix du seuil de réapprentissage désiré, lié à la quantité de données disponibles et la complexité du modèle obtenu, entre autres, est une perspective intéressante pour trouver un équilibre entre amélioration du modèle pour un meilleur suivi et une complexification du modèle.

De plus, l'augmentation des possibilités de gérer les incertitudes et la réduction de la sensibilité à l'aléatoire induite par le processus de rééchantillonnage mériteraient une étude plus poussée. En effet, dans certains cas, comme celui où un retour entre deux places n'est pas possible, l'aléatoire induit par le processus de rééchantillonnage peut entraîner la disparition de certaines hypothèses.

Enfin, un dernier axe porte sur l'étude de la complexité et des temps de calcul nécessaires aux différentes méthodes élaborées. Une éventuelle étude de méthodes permettant leur réduction serait intéressante. Cela pourrait permettre, par exemple, l'ouverture des méthodes proposées à des systèmes embarqués nécessitant un faible coût de calcul. L'explicabilité des méthodes créées est aussi une potentielle piste d'amélioration.

Bibliographie

Vignolles, Amaury, Elodie Chanthery et Pauline Ribot (2020). "An overview on diagnosability and prognosability for system monitoring". In : *European conference of the Prognostics and Health Management Society (PHM Europe)* (cf. p. 15, 66).

Vignolles, Amaury, Pauline Ribot et Elodie Chanthery (2021). "A Holistic Advanced Diagnosis Approach For Systems Under Uncertainty". In : *32nd International Workshop on Principle of Diagnosis–DX 2021* (cf. p. 15).

Chanthery, Elodie, Pauline Ribot et Amaury Vignolles (2019). "HyMU : a software for Hybrid Systems Health Monitoring under Uncertainty". In : *30th International Workshop on Principles of Diagnosis DX'19* (cf. p. 15).

Cassandras, Christos G et Stéphane Lafortune (2009). *Introduction to discrete event systems*. Springer Science & Business Media (cf. p. 8, 9).

Sampath, Meera, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen et Demosthenis C Teneketzis (1995). "Diagnosability of discrete-event systems". In : *IEEE Transactions on automatic control* 40.9, p. 1555-1575 (cf. p. 8, 81).

David, René et Hassane Alla (2010). *Discrete, continuous, and hybrid Petri nets*. T. 1. Springer (cf. p. 8).

Dousson, Christophe, Paul Gaborit et Malik Ghallab (1993). "Situation recognition : Representation and algorithms". In : *IJCAI*. T. 93, p. 166-172 (cf. p. 8).

Verwer, Sicco E, Mathijs M De Weerdt et Cees Witteveen (2006). "Identifying an automaton model for timed data". In : *Benelearn 2006 : Proceedings of the 15th Annual Machine Learning Conference of Belgium and the Netherlands, Ghent, Belgium, 11-12 May 2006* (cf. p. 8).

Henzinger, Thomas A (2000). "The theory of hybrid automata". In : *Verification of digital and hybrid systems*. Springer, p. 265-292 (cf. p. 9, 19, 24).

Isermann, Rolf (1997). "Supervision, fault-detection and fault-diagnosis methods—an introduction". In : *Control engineering practice* 5.5, p. 639-652 (cf. p. 10).

Zwingelstein, Gilles (1995). *Diagnostic des défaillances : théorie et pratique pour les systèmes industriels*. Hermès (cf. p. 11).

Brotherton, Tom, Gary Jahns, Jerry Jacobs et Dariusz Wroblewski (2000). "Prognosis of faults in gas turbine engines". In : *2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484)*. T. 6. IEEE, p. 163-171 (cf. p. 11).

Goh, KM, Benny Tjahjono, Tim Baines et S Subramaniam (2006). "A review of research in manufacturing prognostics". In : *2006 4th IEEE International Conference on Industrial Informatics*. IEEE, p. 417-422 (cf. p. 12).

Engel, Stephen J, Barbara J Gilmartin, Kenneth Bongort et Andrew Hess (2000). "Prognostics, the real issues involved with predicting life remaining". In : *2000 ieee aerospace conference. proceedings (cat. no. 00th8484)*. T. 6. IEEE, p. 457-469 (cf. p. 12).

Chanthery, Elodie et Pauline Ribot (2013). “An integrated framework for diagnosis and prognosis of hybrid systems”. In : *arXiv preprint arXiv :1308.5332* (cf. p. 12).

Wilkinson, Chris, Dave Humphrey, Bert Vermeire et Jim Houston (2004). “Prognostic and health management for avionics”. In : *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*. T. 5. IEEE, p. 3435-3447 (cf. p. 12).

Kirkland, Larry V, T Pombo, K Nelson et F Berghout (2004). “Avionics health management : Searching for the prognostics grail”. In : *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*. T. 5. IEEE, p. 3448-3454 (cf. p. 12).

Gaudel, Quentin, Elodie Chanthery et Pauline Ribot (juin 2015). “Hybrid Particle Petri Nets for Systems Health Monitoring under Uncertainty”. In : *International Journal of Prognostics and Health Management*. Special Issue Uncertainty in PHM 6. url : `https://hal.archives-ouvertes.fr/hal-01229083` (cf. p. 12, 22, 24).

Vignolles, Amaury, Elodie Chanthery et Pauline Ribot (2022). “Hybrid Model Learning for System Health Monitoring”. In : *Safeprocess 2022 : 11th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes* (cf. p. 15).

Vignolles, Amaury, Pauline Ribot et Elodie Chanthery (2022). “Modeling complex systems with Heterogeneous Petri Nets (HtPN)”. In : *33rd International Workshop on Principle of Diagnosis–DX 2022* (cf. p. 15).

Napoleone, Alessia, Marco Macchi et Alessandro Pozzetti (2020). “A review on the characteristics of cyber-physical systems for the future smart factories”. In : *Journal of Manufacturing Systems* 54, p. 305-335. issn : 0278-6125. doi : `https://doi.org/10.1016/j.jmsy.2020.01.007` (cf. p. 18).

Gaudel, Quentin, Elodie Chanthery et Pauline Ribot (sept. 2014). “Health Monitoring of Hybrid Systems Using Hybrid Particle Petri Nets”. In : *Annual Conference of the Prognostics and Health Management Society 2014*. Proceedings of The Annual Conference of the Prognostics and Health Management Society 2014, p. 51 (cf. p. 18).

Alla, Hassane et René David (1998). “Continuous and hybrid Petri nets”. In : *Journal of Circuits, Systems, and Computers* 8.01, p. 159-188 (cf. p. 18-20, 24, 36).

Alur, Rajeev et al. (1995). “The algorithmic analysis of hybrid systems”. In : *Theoretical computer science* 138.1, p. 3-34 (cf. p. 19, 24).

Zhou, MengChu, Frank DiCesare et Alan A Desrochers (1992). “A hybrid methodology for synthesis of Petri net models for manufacturing systems”. In : *IEEE transactions on robotics and automation* 8.3, p. 350-361 (cf. p. 19, 24).

DiCesare, Frank, George Harhalakis, Jean-Marie Proth, M Silva et FB Vernadat (1993). *Practice of Petri nets in manufacturing*. Springer (cf. p. 19, 24).

Balduzzi, Fabio, Alessandro Giua et Carla Seatzu (2001). “Modelling and simulation of manufacturing systems with first-order hybrid Petri nets”. In : *International Journal of Production Research* 39.2, p. 255-282 (cf. p. 19, 24).

Valentin-Roubinet, C (1999). “Hybrid Systems modelling : mixed Petri nets”. In : *IEEE Conference CSCC*. T. 99 (cf. p. 20, 24).

Cortés, Luis Alejandro, Petru Eles et Zebo Peng (1999). “A petri net based model for heterogeneous embedded systems”. In : *Proc. NORCHIP Conference*, p. 248-255 (cf. p. 21, 24).

Gaudel, Quentin, Elodie Chanthery, Pauline Ribot et Matthew J. Daigle (2018). “Diagnosis of hybrid systems using Hybrid Particle Petri nets : theory and application on a planetary rover”. In : *Fault Diagnosis of Hybrid Dynamic and Complex Systems*. Sous la dir. de Moamar Sayed-Mouchaweh. Springer Verlag, p. 209-241 (cf. p. 22-24, 85).

Gaudel, Quentin (sept. 2016). "Approche intégrée de diagnostic et de pronostic pour la gestion de santé des systèmes hybrides sous incertitude". Theses. INSA de Toulouse. URL : `https://hal.laas.fr/tel-01417300` (cf. p. 40, 138).

Berthomieu, Bernard et Miguel Menasche (1983). "An enumerative approach for analyzing time Petri nets". In : *Proceedings IFIP*. Citeseer (cf. p. 45, 138).

Dotoli, Mariagrazia, Maria Pia Fanti et Agostino Marcello Mangini (2008). "Real time identification of discrete event systems using Petri nets". In : *Automatica* 44.5, p. 1209-1219 (cf. p. 49).

Moreira, Marcos V et Jean-Jacques Lesage (2019). "Discrete event system identification with the aim of fault detection". In : *Discrete Event Dynamic Systems* 29.2, p. 191-209 (cf. p. 49).

Leclercq, Edouard, Souleiman Ould el Medhi et Dimitri Lefebvre (2008). "Petri nets design based on neural networks." In : *ESANN*. T. 16. Citeseer, p. 529 (cf. p. 50).

Krishna, K et M Narasimha Murty (1999). "Genetic K-means algorithm". In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 29.3, p. 433-439 (cf. p. 51).

Zhang, Tian, Raghu Ramakrishnan et Miron Livny (1996). "BIRCH : an efficient data clustering method for very large databases". In : *ACM sigmod record* 25.2, p. 103-114 (cf. p. 51).

Cao, Feng, Martin Estert, Weining Qian et Aoying Zhou (2006). "Density-based clustering over an evolving data stream with noise". In : *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, p. 328-339 (cf. p. 51).

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In : *kdd*. T. 96. 34, p. 226-231 (cf. p. 51).

Ankerst, Mihael, Markus M Breunig, Hans-Peter Kriegel et Jörg Sander (1999). "OPTICS : Ordering points to identify the clustering structure". In : *ACM Sigmod record* 28.2, p. 49-60 (cf. p. 51).

Roa, Nathalie Barbosa, Louise Travé-Massuyès et Victor H Grisales-Palacio (2019). "DyClee : Dynamic clustering for tracking evolving environments". In : *Pattern Recognition* 94, p. 162-186 (cf. p. 51, 52).

Barbosa, Nathalie A, Louise Travé-Massuyès et Victor H Grisales (2017). "Diagnosability improvement of dynamic clustering through automatic learning of discrete event models". In : *IFAC-PapersOnLine* 50.1, p. 1037-1042 (cf. p. 52).

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In : *arXiv preprint arXiv :1609.04747* (cf. p. 53).

Tamssaouet, Ferhat, Khanh TP Nguyen, Kamal Medjaher et Marcos Orchard (2020). "A Contribution to Online System-level Prognostics based on Adaptive Degradation Models". In : *ePHM Conf.* T. 5. 1 (cf. p. 53).

Djedidi, Oussama et Mohand A Djeziri (2020). "Power profiling and monitoring in embedded systems : A comparative study and a novel methodology based on NARX neural networks". In : *Journal of Systems Architecture* 111, p. 101805 (cf. p. 53).

Awad, Mariette et Rahul Khanna (2015). "Support vector regression". In : *Efficient learning machines*. Springer, p. 67-80 (cf. p. 53).

Drezet, PML et RF Harrison (1998). "Support vector machines for system identification". In : (cf. p. 53).

ZHANG, Li et Yugeng XI (2004). "Nonlinear system identification based on an improved support vector regression estimator". In : *International Symposium on Neural Networks*. Springer, p. 586-591 (cf. p. 53).

LIAW, Andy, Matthew WIENER et al. (2002). "Classification and regression by randomForest". In : *R news* 2.3, p. 18-22 (cf. p. 54).

OSTERTAGOVÁ, Eva (2012). "Modelling using polynomial regression". In : *Procedia Engineering* 48, p. 500-506 (cf. p. 55).

NIGGEMANN, Oliver, Benno STEIN, Asmir VODENCAREVIC, Alexander MAIER et Hans Kleine BÜNING (2012). "Learning behavior models for hybrid timed systems". In : *Twenty-Sixth AAAI Conference on Artificial Intelligence* (cf. p. 55).

LAUER, Fabien et Gérard BLOCH (2008). "Switched and piecewise nonlinear hybrid system identification". In : *HSCC 2008*. Springer, p. 330-343 (cf. p. 56).

PILLONETTO, Gianluigi (2016). "A new kernel-based approach to hybrid system identification". In : *Automatica* 70, p. 21-31 (cf. p. 56).

FENG, Chao, Constantino M LAGOA et Mario SZNAIER (2010). "Hybrid system identification via sparse polynomial optimization". In : *Proceedings of the 2010 American Control Conference*. IEEE, p. 160-165 (cf. p. 57).

ZABINSKY, Zelda B. et Robert L. SMITH (2013). "Hit-and-Run Methods". In : *Encyclopedia of Operations Research and Management Science*. Sous la dir. de Saul I. GASS et Michael C. FU. Boston, MA : Springer US, p. 721-729. ISBN : 978-1-4419-1153-7. DOI : `10.1007/978-1-4419-1153-7_1145`. URL : `https://doi.org/10.1007/978-1-4419-1153-7_1145` (cf. p. 57).

VIDAL, René (2004). "Identification of PWARX hybrid models with unknown and possibly different orders". In : *Proceedings of the 2004 American Control Conference*. T. 1. IEEE, p. 547-552 (cf. p. 57).

STAROSWIECKI, M, V COCQUEMPOT et J Ph CASSAR (1991). "Observer based and parity space approaches for failure detection and identification". In : *IMACS-IFAC international symposium, Lille, France*. T. 25, p. 536-541 (cf. p. 66, 83, 84).

SAXENA, Abhinav, Jose CELAYA, Bhaskar SAHA, Sankalita SAHA et Kai GOEBEL (2009). "On applying the prognostic performance metrics". In : *Annual Conference of the PHM Society*. T. 1. 1 (cf. p. 67).

VENKATASUBRAMANIAN, Venkat, Raghunathan RENGASWAMY, Kewen YIN et Surya N KAVURI (2003). "A review of process fault detection and diagnosis : Part I : Quantitative model-based methods". In : *Computers & chemical engineering* 27.3, p. 293-311 (cf. p. 81).

BUCHANAN, Bruce G et Edward H SHORTLIFFE (1984). "Rule-based expert systems : the MYCIN experiments of the Stanford Heuristic Programming Project". In : (cf. p. 81).

JACKSON, Peter (1986). "Introduction to expert systems". In : (cf. p. 81).

SAMPATH, Meera, Raja SENGUPTA, Stéphane LAFORTUNE, Kasim SINNAMOHIDEEN et Demosthenis C TENEKETZIS (1996). "Failure diagnosis using discrete-event models". In : *IEEE transactions on control systems technology* 4.2, p. 105-124 (cf. p. 81).

SOLDANI, Siegfried (2008). "Vers le diagnostic embarqué de défaillances dans les systèmes à événements discrets : application au domaine automobile". Thèse de doct. Institut National des Sciences Appliquées de Toulouse (INSA Toulouse) (cf. p. 82).

BOUBOUR, Renée, Claude JARD, Armen AGHASARYAN, Eric FABRE et Albert BENVENISTE (1997). "A Petri net approach to fault detection and diagnosis in distributed systems. I. Application to telecommunication networks, motivations, and mo-

delling". In : *Proceedings of the 36th IEEE Conference on Decision and Control*. T. 1. IEEE, p. 720-725 (cf. p. 82).

Soldani, Siegfried, Michel Combacau, Audine Subias et Jérome Thomas (2007). "On-board diagnosis system for intermittent fault : Application in automotive industry". In : *IFAC Proceedings Volumes* 40.22, p. 151-158 (cf. p. 82).

Cabasino, Maria Paola, Alessandro Giua et Carla Seatzu (2013). "Diagnosability of discrete-event systems using labeled Petri nets". In : *IEEE Transactions on Automation Science and Engineering* 11.1, p. 144-153 (cf. p. 82).

Giua, Alessandro et Carla Seatzu (2002). "Observability of place/transition nets". In : *IEEE Transactions on Automatic Control* 47.9, p. 1424-1437 (cf. p. 82).

Ru, Yu et Christoforos N Hadjicostis (2009). "Fault diagnosis in discrete event systems modeled by partially observed Petri nets". In : *Discrete Event Dynamic Systems* 19.4, p. 551-575 (cf. p. 82).

Ding, Steven X (2014). *Data-driven design of fault diagnosis and fault-tolerant control systems*. Springer (cf. p. 83).

Hammouri, Hassan, Michel Kinnaert et EH El Yaagoubi (1999). "Observer-based approach to fault detection and isolation for nonlinear systems". In : *IEEE transactions on automatic control* 44.10, p. 1879-1884 (cf. p. 83).

Arulampalam, M Sanjeev, Simon Maskell, Neil Gordon et Tim Clapp (2002). "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In : *IEEE Transactions on signal processing* 50.2, p. 174-188 (cf. p. 83).

Julier, Simon J et Jeffrey K Uhlmann (1997). "New extension of the Kalman filter to nonlinear systems". In : *Signal processing, sensor fusion, and target recognition VI*. T. 3068. Spie, p. 182-193 (cf. p. 83).

Thrun, Sebastian, Dieter Fox, Wolfram Burgard et Frank Dellaert (2001). "Robust Monte Carlo localization for mobile robots". In : *Artificial intelligence* 128.1-2, p. 99-141 (cf. p. 83).

Van Der Merwe, Rudolph, Arnaud Doucet, Nando De Freitas et Eric Wan (2000). "The unscented particle filter". In : *Advances in neural information processing systems* 13 (cf. p. 83).

Cordier, M-O, Philippe Dague, François Lévy, Jacky Montmain, Marcel Staroswiecki et Louise Travé-Massuyès (2004). "Conflicts versus analytical redundancy relations : a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives". In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.5, p. 2163-2177 (cf. p. 84).

Gertler, Janos J (2017). *Fault detection and diagnosis in engineering systems*. CRC press (cf. p. 84).

Bayoudh, Mehdi, Louise Travé-Massuyes et Xavier Olive (2008). "Hybrid systems diagnosis by coupling continuous and discrete event techniques". In : *IFAC Proceedings Volumes* 41.2, p. 7265-7270 (cf. p. 84).

Narasimhan, S. et L. Browston (2007). "HyDE - a general framework for stochastic and hybrid modelbased diagnosis". In : 18th *Int. Workshop on Principles of Diagnosis*, p. 162-169 (cf. p. 84).

Lesire, Charles et Catherine Tessier (2005). "Particle Petri nets for aircraft procedure monitoring under uncertainty". In : *International Conference on Application and Theory of Petri Nets*. Springer, p. 329-348 (cf. p. 85, 92).

Cardoso, Janette, Robert Valette et Didier Dubois (1999). “Possibilistic petri nets”. In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 29.5, p. 573-582 (cf. p. 85).

Zouaghi, L, A Alexopoulos, A Wagner et E Badreddin (2011). “Modified particle Petri nets for hybrid dynamical systems monitoring under environmental uncertainties”. In : *2011 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, p. 497-502 (cf. p. 85, 92).

Ribot, Pauline, Yannick Pencolé et Michel Combacau (2013). “Generic characterization of diagnosis and prognosis for complex heterogeneous systems”. In : *International Journal of Prognostics and Health Management* 4 (cf. p. 85).

Beyer, Hans-Georg et Hans-Paul Schwefel (2002). “Evolution strategies–a comprehensive introduction”. In : *Natural computing* 1.1, p. 3-52 (cf. p. 97).

Bouamama, B Ould, AK Samantaray, M Staroswiecki et G Dauphin-Tanguy (2003). “Derivation of constraint relations from bond graph models for fault detection and isolation”. In : *Simulation Series* 35.2, p. 104-109 (cf. p. 104).

Ribot, Pauline, Elodie Chanthery et Quentin Gaudel (2017). “HPPN-based prognosis for hybrid systems”. In : *Annual Conference of the Prognostics and Health Management Society 2017* (cf. p. 107).

Vázquez, M. et I. Rey-Stolle (2008). “Photovoltaic module reliability model based on field degradation studies”. In : *Progress in photovoltaics : Research and Applications* 16.5, p. 419-433 (cf. p. 116, 117).

Résumé

Cette thèse concerne la modélisation et la gestion de santé de systèmes hybrides sous incertitude.
La modélisation consiste à représenter un système dans un formalisme mathématique. Cette thèse propose un formalisme, les réseaux de Petri Hétérogènes (HtPN), permettant de modéliser et surveiller des systèmes hybrides sous incertitudes. Ce formalisme est une extension des réseaux de Petri classiques et permet de représenter tous les types de systèmes. Il permet donc de représenter des systèmes purement discrets, des systèmes continus ou des systèmes hybrides, dont une définition est proposée, ainsi qu'une communication entre ces systèmes. De plus, il permet de prendre en compte des incertitudes, que ce soit au niveau du modèle ou des observations, afin de représenter au mieux le comportement du système. Le vieillissement et l'usure des systèmes peuvent être estimés et inclus dans le modèle du système sous forme de lois de dégradation. Pour ces systèmes, les modèles issus de connaissances physiques seules sont difficiles à obtenir. L'intelligence artificielle est donc utilisée pour apprendre des modèles à partir de données. Cet apprentissage de modèle est découpé en deux parties : l'apprentissage de la structure à événements discrets du modèle, basé sur un algorithme de clustering, et l'apprentissage des dynamiques d'évolution à temps continu du système, basé sur deux algorithmes de régression, dont le choix est laissé à l'utilisateur. Par gestion de santé, on entend généralement du diagnostic, qui est de savoir identifier et isoler un problème, appelé une faute, et du pronostic, qui consiste en une estimation de la durée de vie résiduelle du système. Une méthode de diagnostic avancée, basée sur le formalisme des HtPN, est présentée et mise à l'épreuve sur trois systèmes : un système à événements discrets, un système continu et un système hybride. Ce dernier consiste en des réservoirs d'eau connectés entre eux. Une faute peut par exemple correspondre à l'apparition d'une fuite dans un des réservoirs. La méthode de diagnostic se déroule en deux étapes : une étape de prédiction, calculant les différentes hypothèses sur l'état futur du système ; et une étape de correction, dans laquelle les différentes hypothèses sont validées ou réfutées en fonction des observations reçues. Les résultats de diagnostic obtenus permettent de prendre en compte les incertitudes sur la modélisation et sur les observations et de donner des hypothèses de diagnostic pondérées. L'évaluation de la dégradation du système peut également être obtenue, menant à des informations pour le pronostic. Enfin, un cas d'étude réel, composé de deux panneaux photovoltaïques et de diverses charges est présenté. Les deux panneaux ont d'abord été modélisés avec le formalisme des HtPN. Ensuite, un modèle global a été obtenu par apprentissage. Enfin, la méthode de diagnostic avancée est appliquée sur divers cas d'utilisation, montrant la pertinence de la méthode.

Abstract

The main topics of this thesis are modeling and monitoring the health of hybrid systems under uncertainties. Modeling a system is representing the system through a mathematical formalism. A new formalism, the Heteregeneous Petri Nets is introduced in this thesis. This formalism, an extension of the Petri Nets, allows to model and monitor hybrid systems under undertainties. With this formalism, it is possible to represent discrete systems, continuous systems or hybrid systems, for which a new definition is proposed. Contrary to most formalisms, the HtPN formalism allows for a communication between different types of system. Uncertainties, on the model or the observations, can be taken into account to better represent the behavior of the system. Finally, dynamics representing the aging and the wear of the system can be estimated and implemented on the model. For hybrid systems, models based on physical knowledge are often hard to obtain. Artificial intelligence is thus used to learn model from data. This learning is comprised of two steps. The first one is based on a clustering algorithm and outputs the discrete events structure of the model. The second one is based on two different regressions algorithms, support vector and random forest, and learns the continuous dynamics of the system. The choice of which regression algorithm to choose is left to the user, according to his needs. Health monitoring is usually associated to diagnosis, identifying and isolating a problem, usually referred to as a fault, or prognosis, estimating the remaining useful life of a system. An advanced diagnosis methodology, based on the HtPN formalism, is introduced and applied on three different systems. The first system considered is a discrete system. The second one is continuous and the third one is a hybrid system. The hybrid system is a system composed of connected water tanks. In this system, a fault example could be a leakage in one of the tanks. The diagnosis method is done in two steps. The first step is the prediction, which will give hypotheses on the future state of the model. The second step is the correction, in which the different hypotheses given by the first step will either be kept or forgotten, according to the observations received. The diagnosis results obtained are able to take into account uncertainties on the modeling or the observation and gives a belief on different health state hypotheses. Evaluating the degradation of the system can also be done and could be used to apply prognosis methods. Finally, a real case study is introduced. It consists in two communicating photovoltaic panels with some loads : two lamps and a motor. The two panels are modeled using the HtPN formalism. A global model is obtained using the learning method with some collected datasets. Finally, the diagnosis method is applied on data obtained by simulation of the two panels model.