

Remerciements

Je tiens avant tout à remercier la personne sans laquelle tout ceci n'existerait pas, Monsieur **Fouad BELMAJDOUB**, professeur à la Faculté des Sciences et Techniques de Fès (FSTF), mon directeur de thèse qui m'a accueilli au sein de sa jeune équipe de « Instrumentation, Mesures et commandes des systèmes industriels ». Je souhaite le remercier non seulement pour son encadrement de très haut niveau, ses précieux conseils et orientations que j'ai toujours trouvé pertinents, mais également pour sa disponibilité et son dévouement. Je le remercie aussi de m'avoir toujours poussé vers l'avant, pour toute la confiance qu'il a porté en moi et qui m'a permis d'acquérir une précieuse expérience du métier d'enseignant chercheur.

Je souhaite remercier mes professeurs Monsieur **Mohammed El HAMMOUMI**, directeur de notre laboratoire « Techniques industrielles », et Monsieur **Fouad GADI**, chef de département Génie industriel à la FST de Fès, pour l'ensemble des aides administratif, des conseils précieux et des ressources matérielles qu'ont mis à notre disposition et sans lesquelles aucune thèse ne peut aboutir à sa fin.

Mes remerciements vont aussi aux membres de jury qui ont accepté de sacrifier de leurs précieux temps et de leurs efforts pour évaluer cette modeste thèse. J'adresse mes remerciements aux :

- Professeur ABDALI Abdelmounaïm ;
- Professeur Cherkani Hassani Mohammed ;
- Professeur FRI Mohamed ;
- Professeur MOUHSEN Ahmed ;
- Professeur QJIDAA Hassan ;
- Professeur SOULHI AZIZ ;
- Professeur SRITI Mohammed.

Enfin je remercier tous les membres de laboratoire des techniques industriels et tout le corps professoral de département génie industriel et aux membres de ma famille qui m'ont soutenu et encouragé tout au long de ma thèse.

À ma chère maman Saida

À tous les membres de ma petite famille

À mes collègues et mes amis

Table des matières

Introduction générale.....	12
Chapitre I: Systèmes à évènements discrets	15
1. Introduction	16
2. Systèmes à évènements discrets	16
3. Quelques formalismes des SED	18
3.1 Modélisation des SED	18
3.1.1 Automates finis.....	18
3.1.2 Réseaux de pétri	19
3.1.3 Représentation d'état.....	19
3.1.4 Algèbre de Boole.....	20
3.1.5 Mathématiques tropicales.....	20
3.1.6 Logique temporelle.....	20
3.2 Déterministe non-déterministe	21
3.3 Applications autour des SED.....	21
3.3.1 Contrôle	21
3.3.2 Supervision.....	21
3.3.3 Surveillance	21
3.3.4 Diagnostic.....	22
3.3.5 Robustesse	22
3.3.6 Modélisation	22
3.3.7 Simulation	22
3.4 Structures de prise de décision	22
4. Analyse de la tendance de recherche actuelle autour des SED	23
4.1 Méthodologie.....	23
4.2 Synthèse.....	24
5. État de l'art sur le diagnostic et le pronostic des fautes dans les SED	27
5.1 Définitions	27
5.2 Méthodes de diagnostic à base de modèle.....	29
5.2.1 Diagnostic par les automates finis.....	29
5.2.2 Diagnostic en utilisant les RdP.....	31
5.3 Méthodes de diagnostic sans modèle	33
5.3.1 Méthodes basées sur l'expertise	34
5.3.2 Méthodes basées sur l'analyse de données.....	34
6. Problématique.....	35
7. Conclusion.....	36
Chapitre II: Réseaux de neurones	37
1. Introduction	38
2. Neurone formel.....	38
3. Architectures neuronales	39
3.1 Réseaux de neurones de type « feed-forward »	40
3.1.1 Perceptron simple	40
3.1.2 Perceptron multicouche	41
3.1.3 Réseau "Random Vector Functional Link"	41
3.1.4 RdN à base de fonction radiales	42
3.1.5 Carte de Kohonen.....	42
3.2 Réseaux de neurones récurrents	43
3.2.1 Réseau de neurones DGNN.....	43
3.2.2 Réseau de neurones R2BF.....	44

3.2.3 Réseau de neurones récurrent à base de fonction radiale (RRBF)	44
4. Algorithmes d'apprentissage	44
4.1 Types des algorithmes d'apprentissage	44
4.2 Quelques algorithmes d'apprentissage des RdN	45
4.2.1 Algorithme d'apprentissage par correction d'erreur	45
4.2.2 Algorithme d'apprentissage par la descente de gradient	46
4.2.3 Algorithme de rétro propagation du gradient	47
4.2.4 Algorithme d'apprentissage LBG	47
5. Domaines d'application des RdN	48
5.1 Régression multicritère	48
5.2 Reconnaissance des formes	49
5.3 Clustering	49
6. Diagnostic/ pronostic des fautes dans les SED en utilisant les RdN	50
7. Étude Comparative entre Différents RdN à Propagation Directe	51
7.1 Étude Comparative	51
7.2 Comparaison entre le PMC et le RBFNN	52
7.3 Synthèse	54
8. CONCLUSION	55
Chapitre III: Diagnostic des SED basé sur un modèle statistique et les lois de fiabilité	56
1. Introduction	57
2. Fautes dans les SED	57
3. Modèle statistique pour les SED	58
4. Diagnostic et pronostic des fautes	61
4.1 Diagnostic	61
4.2 Pronostic	63
4.2.1 Prévisibilité	63
4.2.2 Temps restant avant l'occurrence d'une faute prévue	65
5. Comparaison avec les méthodes classiques de diagnostic des SED	66
6. Efficacité maximum du diagnostic et du pronostic	66
6.1 Minimisation des dimensions des fenêtres temporelles	66
6.2 Intégration des lois de fiabilité	68
6.2.1 Lois de fiabilité	69
6.2.2 Logique floue	71
6.2.3 Intégration des lois de fiabilité dans le diagnostic des fautes	73
6.2.4 Intégration des lois de fiabilité dans le pronostic	74
7. Validité des contraintes temporelles dans un SED	74
7.1 Contraintes temporelles	74
7.2 Construction des CT à partir du modèle statistique	76
7.3 Perte d'observabilité	77
8. Conclusion	77
Chapitre IV: Les réseaux de neurones pour le diagnostic et le pronostic des fautes dans les SED	79
1. Introduction	80
2. Réseaux de neurones	80
2.1 Implémentation des RdN dans le diagnostic/pronostic des SED	80
2.2 Choix du RdN	81
2.3 Apprentissage et choix des paramètres des RBFNN	82
3. Diagnostic/pronostic des fautes en utilisant les RBFNN	83
3.1 Diagnostic	83
3.2 Pronostic	84
3.2.1 Pronostic des fautes	84
3.2.2 Prédiction de temps restant avant l'occurrence d'une faute	84

3.3 Cas d'application.....	85
3.4 Fonction de marquage	87
4. Vérification du respect des contraintes temporelles	89
4.1 Vérification en temps réel des CT et fonction de marquage	89
4.2 Apprentissage des RBFNN pour vérifier les CT	90
5. Conclusion.....	91
Chapitre V: Mise en application de notre approche	93
1. Introduction	94
2. Mise en place de l'approche développée dans un SED	94
3. Le langage de programmation : le Python.....	97
4. Programmation de l'application	99
5. Cas d'application.....	100
5.1 Description de système.....	100
5.2 Contraintes temporelles	102
5.3 Collecte des données	103
5.4 Calcul des paramètres de diagnostic/pronostic et formatage des données.....	103
5.5 Apprentissage des RdN	104
5.6 En-ligne diagnostic, pronostic et vérification des CT	106
6. Conclusion.....	107
Conclusion générale et perspectives.....	109

Liste des figures

Figure I.1: SED modélisé par automate fini.....	17
Figure I.2: Comparaison entre l'évolution d'une variable d'état : (a) système continu, (b) système à événements discrets.....	17
Figure I.3 : SED modélisé par AFD.....	19
Figure I.4 : Représentation d'état.....	20
Figure I.5 : Structures de prise de décision, (a) centralisée, (b) décentralisée, (c) distribuée	23
Figure I.6 : Interprétation des courbes	24
Figure I.7 : utilisation des différents outils de modélisation en fonction des années de publication. ...	25
Figure I.8: Nombre de papiers selon l'architecture de prise de décision : centralisé et décentralisé... 25	25
Figure I.9: Nombre de papiers selon le type de système étudié : déterministe et non-déterministe 25	25
Figure I.10 : La variation des pourcentages des articles publiés dans les différents domaines en fonction des années de publication : (a) contrôle, supervision et surveillance. (b) diagnostic, simulation et modélisation.	26
Figure I.11 : Classification des méthodes de diagnostic des SED	27
Figure I.12 : les variations d'une variable lors de fonctionnement normal, La dégradation, la défaillance et la panne causés par une faute <i>ef</i>	28
Figure I.13 : (a) SED modélisé par automate fini, (b) son observateur, (c) son diagnostiqueur.....	30
Figure II.1: Neurone formel.	38
Figure II.2: Fonctions d'activation les plus utilisées, a) fonction linéaire, b) fonction a seuil, c) fonction linéaire par morceaux, d) fonction sigmoïde, e) fonction tangente hyperbolique, f) fonction gaussienne.	39
Figure II.3 : Sens de propagation direct dans un RdN de type « feed-forward ».....	39
Figure II.4: Quelques types des connexions récurrentes dans les RdN.	40
Figure II.5: Eexemple de récurrence interne.	40
Figure II.6: Architecture d'un PMC avec deux couches cachées.	41
Figure II.7: Architecture d'un RVFLNN.	41
Figure II.8 : Architecture d'un RBFNN.....	42
Figure II.9 : Architecture d'une carte de Kohonen.	43
Figure II.10: Architecture d'un DGNN.	43
Figure II.11: Architecture du R2BF.	44
Figure II.12: Aarchitecture d'un RRBF.	44
Figure II.13: Exemple de régression.....	49
Figure II.14: Types de classification : a) linéaire, b) non linéaire.	49
Figure II.15: Diagnostic et pronostic d'un SED en utilisant un réseau de neurones récurrent (Jordan).	50
Figure II.16: Diagnostic et pronostic des SED en utilisant des fenêtres temporelles.	51
Figure II.17: représentation de l'espace, a) global, b) locale.....	52
Figure II.18: Procédure de sélection d'une architecture des réseaux de neurones.....	54
Figure III.1: Le chariot.	59
Figure III.2: Construction des fenêtres temporelles à partir de σ_1	60
Figure III.3: Algorithme pour augmenter l'efficacité de diagnostic/pronostic.....	68
Figure III.4: Synoptique de la méthode développée.....	69
Figure III.5: Profil en baignoire.....	70
Figure III.6: Fonction d'appartenance d'un ensemble flou (loi normale), sa dilatation et sa concentration.	72
Figure III.7 : Fonction d'appartenance d'un ensemble flou, ses dilatations d'ordre $r=0.5, 0.3$ et 0.1 . 73	73
Figure III.8: SED avec trois taches en parallèle.	77
Figure III.9 : SED modélisé par réseau de Pétri.	77

Figure IV.1 : processus d'apprentissage des RdN pour assurer le diagnostic/pronostic	81
Figure IV.2 : Implémentation des RdN dans le diagnostic/pronostic en temps réel.	81
Figure IV.3 : Ensemble des paramètres d'un RBFNN.....	83
Figure IV.4 : les durées des différents déplacements de chariot.....	90
Figure V.1 : Les étapes de la mise en place de l'approche diagnostic/pronostic des SED développée dans cette thèse.	95
Figure V.2: Algorithme de l'application informatique.....	96
Figure V.3 : Capture d'écran de l'interface principale de l'application.	99
Figure V.4 : Capture d'écran de mode de calcul des paramètres de diagnostic/pronostic.	99
Figure V.5 : Capture d'écran de mode d'apprentissage des RBFNN.....	100
Figure V.6 : chariot suspendu.	100
Figure V.7 : Grafcet de fonctionnement de chariot suspendu.	102
Figure V.8 : Réseau Modbus entre l'ordinateur et d'autres équipements industriels.....	103
Figure V.9 :Précision de diagnostic en fonction de nombre des neurones dans la couche cachée.....	105
Figure V.10 : Précision de pronostic en fonction de nombre des neurones dans la couche cachée	106
Figure V.11 :Précision de la vérification des CT en fonction de nombre des neurones dans la couche cachée des trois types de cycle A, B et C.	106
Figure V.12 : Capture d'écran de résultat de mode en-ligne.....	107

Liste des tableaux

Tableau II.1: Avantages et inconvénients de quelques architectures des réseaux de neurones.....	53
Tableau III.1 : Différents évènements générés par le chariot.....	59
Tableau IV.1 : Exemple de construction d'une base de données pour l'apprentissage des trois RdN décrits dans les sections IV.1.4 et IV.1.5.	86
Tableau IV.2: Conversion de problème de l'exemple 1 à un problème de classification.	88
Tableau IV.3: Des informations sur les CT des différents évènements de σ	91
Tableau V.1 : Classement des langages de programmation utilisés pour le développement des applications scientifiques	98
Tableau V.2: Les évènements générés par le chariot suspendu.	101
Tableau V.3: Résultats de l'application de l'algorithme sur le chariot suspend.	104
Tableau V.4 : Caractéristiques des RBFNN et de leurs bases d'apprentissage.....	105

Liste des Notations et des abréviations

AF : Automate Fini

AFD : Automate Fini Déterministe

AFND : Automate Fini Non-Déterministe

AFNOR : Association Française de Normalisation

AMDEC : Analyse des Modes de Défaillances, de leurs Effets et de leur Criticité.

API: Automate Programmable Industriel

CT: Contrainte Temporelle

MTTF: Mean Time To Failure

RBFFNN : Radial Basic Function Neural Nertwork

RdN : Réseau de Neurones artificiel

RdP : Réseau de Pétri

RVFLNN : Random Vector Functional Link Neural Network

SED : Systèmes A Evènements Discrets

PMC: perception multicouche

Notations

Σ : L'ensemble des événements générés par le SED.

Σ_o : L'ensemble des événements observables générés par le SED.

Σ_u : L'ensemble des événements non observables générés par le SED.

Σ_f : L'ensemble des événements de faute générés par le SED.

Σ_{reg} : L'ensemble des événements régulier non observables générés par le SED.

$TW^*(\Sigma)$: L'ensemble des mots temporise sur Σ .

σ_j : Le mot temporisé généré lors de la $j^{ème}$ cycle de fonctionnement de SED.

w_i^k : La $i^{ème}$ fenêtre temporelle de dimension k enregistrée lors de la $j^{ème}$ cycle de fonctionnement

W^k : L'ensemble de toutes les fenêtres temporelles w_i^k , que le SED peut générer.

$P_o(w)$: La projection qui supprime le temps et les évènements inobservables de w .

$P_u(w)$: La projection qui supprime le temps et les évènements observables de w .

$T_\alpha(w)$: La projection qui donne les dates d'occurrence d'un évènement e_α dans w .

F_α : La classe (ou ensemble) des fautes qui ont des caractéristiques communes

S_α^k : L'ensemble des fenêtres temporelles qui contient une faute $e_f \in F_\alpha$

S_0^k : L'ensemble des fenêtres temporelles qui ne contient aucune faute de Σ_f .

$P_o^{-1}(\omega)$: L'ensemble de toutes les fenêtres temporelles de W^k qui ont la même séquence des évènements observables que ω .

$su f_k(\omega)$: L'ensemble de tous les couples de fenêtres temporelles possibles qui contiennent les k évènements observables qui peuvent se produire après l'occurrence de ω .

$D_\alpha(\omega)$: La probabilité d'occurrence d'une faute $e_f \in F_\alpha$ lors de l'enregistrement de ω .

$P_\alpha(\omega)$: La probabilité d'occurrence d'une faute $e_f \in F_\alpha$ dans les k évènements observables postérieurs à ω .

$T^\alpha(\omega)$: L'ensemble des valeurs possibles de temps restant avant l'occurrence d'une faute $e_f \in F_\alpha$ dans les k évènements observables postérieurs à ω .

$con(A)$: Concentration de l'ensemble flou A.

$dil(A)$: Dilatation de l'ensemble flou A.

$dil^r(A)$: Dilatation d'ordre r de l'ensemble flou A.

$E_\alpha(\omega)$: L'état de fonctionnement lors de la génération de ω

$EF_\alpha(\omega)$: L'état de fonctionnement futur après la génération de ω

$R^\alpha(t)$: La fiabilité de composant C^α .

$E_\alpha(\omega, t)$: L'état de fonctionnement lors de la génération de ω en prenant en considération la fiabilité de composant C^α à l'instant t.

$EF_\alpha(\omega, t)$: L'état de fonctionnement future après la génération de ω en prenant en considération la fiabilité de composant C^α à l'instant t.

$CT(e_\alpha)$: La contrainte temporelle associée à l'évènement e_α .

Introduction générale

À la fin du 18^{ème} siècle, les travaux en thermodynamique, notamment ceux de James Watt, ont conduit à la mise au point des premières machines thermiques qui transforment l'énergie thermique extraite de la combustion (bois, charbon) en énergie mécanique. Cette découverte mènera tout naturellement à la mécanisation des processus industriels et ceci est connu sous le nom de « la Révolution industrielle ». Ce progrès est devenu plus important avec l'apparition de l'électricité et des moteurs électriques qui font de nos jours le cœur de quasiment la totalité des systèmes industriels.

Au cours de la 2^{ème} moitié du 20^{ème} siècle, le domaine de l'automatique a connu l'apparition des différentes unités de calcul comme les microcontrôleurs, les microprocesseurs et les automates programmables et qui ont constitué l'ère de l'automatique et de la robotique. Ces deux domaines ont donné un autre sens au développement des systèmes industriels, surtout lorsqu'ils ont permis la naissance d'un autre domaine qui est l'intelligence artificielle.

Tout au long de ce développement, les équations différentielles ont joué un rôle crucial dans la modélisation et l'étude d'une partie importante des systèmes industriels, dit les systèmes continus. Cependant, cet outil a laissé échapper une autre catégorie des systèmes qu'ont des espaces d'état discrets et où le passage d'un état à un autre est fait selon des événements (l'arrivée ou le départ d'une ressource, le début ou fin d'une action ...), d'où l'apparition de la notion des systèmes à événements discrets (SED). Ces SED couvrent une très bonne partie des systèmes, comme :

- Les réseaux de transport ;
- Les réseaux de communications et d'ordinateurs ;
- Les ateliers de production manufacturière, notamment dans leur version moderne "*ateliers flexibles*".

Certes ces développements croissants, que le monde des systèmes industriels a connu, ont garanti l'amélioration des performances et l'augmentation de leurs productivités en diminuant au maximum le taux de non-qualité et le temps de repos tout en augmentant le nombre des unités produites par unité de temps, mais en contrepartie ces systèmes deviennent de plus en plus complexes et plus vulnérables à commettre des fautes plus complexes et plus difficiles à déceler. Ces fautes peuvent mettre en souci la sécurité des employés et des machines. Ceci met l'automaticien dans l'obligation de développer des méthodes de diagnostic/pronostic des fautes plus adaptées à la nature de système, plus rapides et qui peuvent garantir même l'autonomie totale de ces systèmes. Pour satisfaire ces exigences accrues en matière de sûreté de fonctionnement de ces systèmes, il est aussi devenu important d'étudier le pronostic des fautes.

Le diagnostic/pronostic des fautes dans les systèmes industriels peut être vu comme un problème de classification où on essaye de classer chaque comportement de système au mode de fonctionnement

le plus approprié. Ces méthodes de diagnostic/pronostic se sont améliorées rapidement pour pouvoir accompagner les développements technologiques qu'ont connus les systèmes industriels. Au début ces méthodes ont été limitées à l'inspection visuelle et l'expertise, après ces méthodes ont été améliorées avec l'apparition des techniques de modélisation qu'ont rendu la compréhension des systèmes industriels plus simple et ainsi faciliter la détection des fautes. Actuellement ces méthodes sont devenues plus intelligentes en intégrant des outils mathématiques comme la logique floue et des outils de l'intelligence artificielle. Une grande partie de ces dernières sont inspirées des systèmes biologiques comme la colonie des fourmis l'algorithme génétique, les réseaux de neurones...etc. L'intégration des telles méthodes donne aux opérations de diagnostic/pronostic plus d'efficacité en améliorant la précision, la rapidité et en diminuant le coût de ces opérations.

Les réseaux de neurones (RdN) sont des outils qui assimilent le fonctionnement des neurones biologiques. Ces outils ont fait l'objet de plusieurs travaux de recherche qui ont visé leurs améliorations et leurs applications dans plusieurs domaines comme la classification et l'interpolation. Ces outils surtout avec leur capacité d'apprentissage et de généralisation ont prouvé leur efficacité dans la résolution de plusieurs problèmes. Ainsi les RdN peuvent présenter des outils très intéressants pour le diagnostic des systèmes industriels. C'est dans ce contexte que se positionne ce travail de recherche.

Ce rapport de thèse a pour objectif le développement d'une nouvelle approche visant l'exploitation de la capacité de calcul et d'apprentissage des réseaux de neurones dans le diagnostic et le pronostic des fautes dans les systèmes à événements discrets. Dans cet objectif, ce rapport sera organisé en cinq chapitres que nous pouvons les résumer comme suit :

Le premier chapitre donne un aperçu général sur le domaine des systèmes à événements discrets, et présente l'ensemble des notions préliminaires utiles pour la compréhension du reste de ce rapport. Il justifie, à travers une étude de la tendance de recherche en relation avec les systèmes à événements discrets, l'axe de recherche le plus convenable à développer. La dernière partie de ce chapitre résume l'état de l'art en relation avec le sujet à développer.

Le deuxième chapitre constitue un état de l'art concernant les différentes architectures des RdN existantes dans la littérature. Dans un premier temps, il montre comment les RdN peuvent être implémentés dans le processus de diagnostic/pronostic des fautes dans les SED. Par la suite, il présente une comparaison entre les différentes architectures des réseaux de neurones les plus utilisées en diagnostic : à savoir les perceptrons multicouches, le réseau de neurones à base de fonction radiale, la carte de Kohonen et le réseau Random Vector Functional Link.

Le troisième chapitre présente une nouvelle approche de diagnostic et de pronostic des fautes dans les SED. Cette approche utilise un modèle statistique construit à partir des mots temporisés générés par le SED et quelques paramètres qui caractérisent le diagnostic et le pronostic pour décrire le comportement du SED. À la fin de ce chapitre, l'efficacité des opérations de diagnostic et de pronostic a été améliorée

en utilisant un algorithme qui permet de calculer les paramètres les plus adéquats du modèle statistique et en intégrant les lois de fiabilité qui permet de prendre en compte la durée de vie des composants du SED.

Dans le quatrième chapitre, les résultats du troisième chapitre sont utilisés pour construire les bases de données pour l'apprentissage des RdN. Ces derniers vont permettre la mise en place d'un enligne diagnostiqueur, un enligne pronostiqueur et d'un RdN qui permet d'interpréter le temps restant avant l'occurrence d'une faute probable.

Le cinquième chapitre détaille la conception d'une application informatique, qui permet l'implémentation des approches développées dans les systèmes industriels. Un cas d'application a été aussi présenté et qui a permis de tester l'efficacité et la pertinence de l'approche développée dans ce travail de thèse.

Chapitre I:
Systemes à évènements discrets

1. Introduction

Les systèmes à évènement discrets constituent une classe très importante des systèmes dynamiques industriels et ils ont fait l'objet de plusieurs travaux de recherche. Dans ce chapitre, nous présentons un aperçu général sur ces systèmes ainsi que quelques formalismes en relation avec ces systèmes à savoir : leurs outils de modélisation, leurs types (déterministe, non-déterministe) et les applications les plus répandues utilisées dans la littérature. Nous allons présenter aussi une analyse de la tendance actuelle de recherche en relation avec les SED. Ceci a pour objectif de justifier le choix de ce sujet de thèse. À la fin de ce chapitre, nous présenterons un état de l'art sur le diagnostic des systèmes à évènements discrets.

2. Systèmes à évènements discrets

Dans l'industrie il y a trois types des systèmes :

- Les systèmes continus : Ces systèmes sont très répandus dans les procédés chimiques où toutes les variables d'état sont des variables continues telles que le débit, la concentration, l'état d'un réservoir, l'état d'une vanne analogique...
- Les systèmes à évènements discrets : Les systèmes discrets (ou systèmes à évènements discrets) sont des systèmes qui mettent en jeu des informations qui ne sont prises en compte qu'à des moments précis.
- Les systèmes hybrides : Dans ces systèmes les aspects continus et discrets coexistent dans la modélisation. Pour ces systèmes, la modélisation repose sur une représentation par un ensemble d'états et de transitions comme dans un système à évènements discrets dont certains états du système sont modélisés par une approche « continue ».

Le travail de cette thèse se focalise sur les systèmes à évènements discrets (SED) qui forment une classe très importante de systèmes dynamiques. Ce terme a été introduit à la fin des années 1970 pour décrire des systèmes dont nous s'intéressons qu'aux faits importants (appelés évènements discrets) pour son étude. Le comportement de ces systèmes est régi par ces évènements qui se produisent de façon asynchrone dans le temps et qui sont responsables de la génération des transitions d'état. Un SED peut comporter différents natures des évènements, comme :

- L'arrivée ou le départ d'une ressource (objet, personne ...)
- Le début ou la fin d'une action (d'un processus) ...

La figure I.1 donne un exemple d'un simple SED qui est une lampe commandée par un interrupteur à deux positions. Ce système a deux événements qui sont a et b (allumer-lampe et éteindre-lampe) qui permettent la commutation entre deux états qui sont lampe-on et lampe-off.

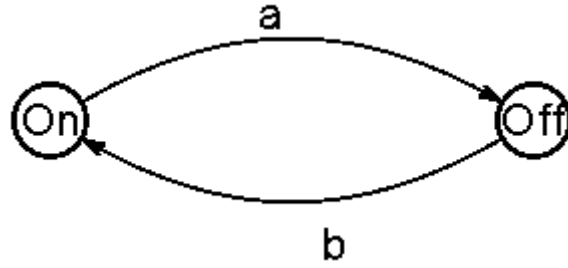


Figure I.1: SED modélisé par automate fini.

Ces systèmes, et contrairement aux systèmes continus, sont des systèmes dynamiques dont les signaux changent de valeur d'une façon non continue. Ils ont un espace d'état discret et leurs évolutions d'un état x_i à un autre état x_{i+1} est conforme à l'occurrence d'un évènement e_i . Cette commutation d'état est liée à la fin de certaines tâches et le commencement d'autres tâches. La figure I.2 montre la différence entre l'évolution d'une variable d'état dans un système continue où elle est en fonction du temps (figure I.2.a) et son évolution dans un SED où elle est conditionnée par l'occurrence des évènements.

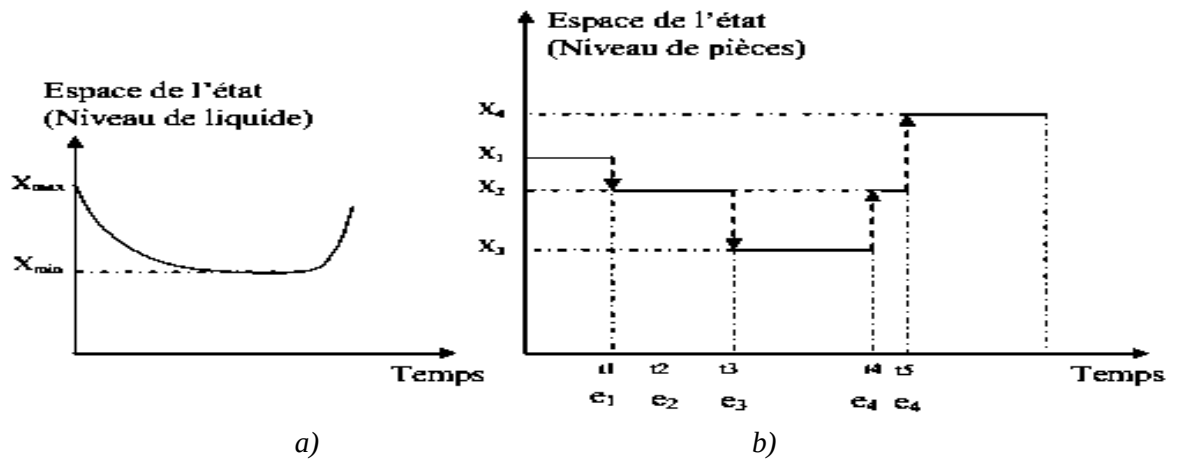


Figure I.2: Comparaison entre l'évolution d'une variable d'état : (a) système continu, (b) système à évènements discrets

En général, un SED, indépendamment de l'outil de modélisation, est un générateur à évènements Σ qui sont responsables du passage d'un état de fonctionnement à un autre. Généralement ces évènements peuvent être observables ou inobservables (I.1).

$$\Sigma = \Sigma_o \cup \Sigma_u \quad (I.1)$$

Avec $\Sigma_o = \{e_1, e_2, \dots, e_n\}$ est l'ensemble des évènements observables qui peuvent être les valeurs obtenues des capteurs ou les commandes issues de contrôleur. $\Sigma_u = \Sigma_{reg} \cup \Sigma_f$ est l'ensemble des évènements inobservables qui peuvent être des fautes commises par le système $\Sigma_f = \{e_{f1}, \dots, e_{fm}\}$ ou des évènements régulières que nous ne pouvons pas l'observer par les capteurs mis en place. $\Sigma_{reg} = \{e_{r1}, \dots, e_{rl}\}$ (Comme les actions manuelles).

3. Quelques formalismes des SED

3.1 Modélisation des SED

L'étude d'un SED exige une représentation par un modèle qui décrit tous ses comportements. Dans la littérature, il existe plusieurs outils qui permettent la modélisation de ce type des systèmes, les plus répandus sont : les automates finis (ou machine à états) [1-4], les réseaux de Pétri[5, 6], la représentation d'état, l'algèbre de Boole, les mathématiques tropicales et la logique temporelle... Dans la suite de cette partie, nous essayerons de donner les définitions formelles de ces outils, mais avant ceci nous rappelons quelques définitions préliminaires en relation avec la théorie des langages formels [2] qui vont représenter la base de certains outils de modélisation que nous définissons par la suite.

Définition 1 : (Alphabet) un alphabet est un ensemble (généralement fini) de lettres ou de symboles. Un alphabet sera en général désigné par une lettre grecque majuscule. Ainsi $\Sigma = \{a,b,c\}$ et $\Delta = \{0,1\}$ sont des alphabets. Dans le contexte des SED les évènements sont considérés comme des alphabets.

Définition 2 : (Mot) un mot est une suite finie et ordonnée des éléments d'un alphabet. Par exemple les mots $\omega = aaba$ et $v = ab$ sont deux mots sur l'alphabet $\Sigma = \{a, b, c\}$. L'ensemble de mots sur un alphabet Σ et noté par Σ^* .

Définition 3 : (Langage) c'est un ensemble de mots sur Σ en d'autres termes c'est une partie de Σ^* . Ainsi, $L = \{a, ab, aa, abc, abb, \dots\}$ est le langage sur l'alphabet $\Sigma = \{a, b, c\}$.

3.1.1 Automates finis

L'automate fini (ou l'automate à état) est un outil formel de modélisation basé sur la théorie des langages et adapté pour la description du comportement événementiel d'un procédé [7]. Un automate fini déterministe (ou AFD) est la donnée d'un quintuple [2] définis par l'équation (I.2).

$$A = (Q, q_0, F, \Sigma, \delta) \quad (I.2)$$

Où :

- Q est un ensemble fini dont les éléments sont les états de A ;
- $q_0 \in Q$ est un état privilégié appelé état initial, marqué par une flèche entrante (l'étape 1 et 2 de l'exemple);
- $F \subseteq Q$ désigne l'ensemble des états finals, marqués par deux cercles (l'étape 1 de l'exemple);
- Σ est l'alphabet de l'automate (ou bien l'ensemble des évènements que le SED peut générer);
- $\delta : Q \times \Sigma \rightarrow Q$ est la fonction de transition de A .

Un AFD prend comme entrée un mot fini $\omega \in \Sigma^*$ et suivant ce mot fait un nombre de transitions à partir de l'état initial. Ce mot fini accepté par l'AFD s'il est l'étiquette d'un chemin menant de l'état initial à un état final. Un automate est considéré comme non déterministe si cet automate comporte plusieurs

états initiaux ou s'il existe un mot qui représente la trace de plusieurs différents chemins. Un langage est dit formel s'il existe un AFD qui accepte tous les mots de ce langage.

Exemple 1 :

La figure I.3 donne l'automate fini déterministe qui accepte les mots : a ; ab ; aba

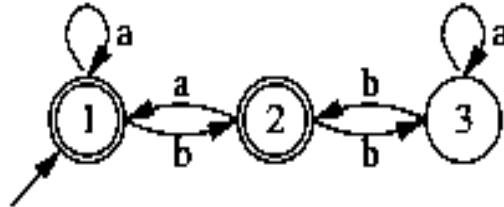


Figure I.3 : SED modélisé par AFD

3.1.2 Réseaux de pétri

Un réseau de Petri est un réseau formé de places et de transitions reliées par des arcs. Il est défini par 6-uplet (S, T, F, M_0, W, K) , où :

- S est l'ensemble des *places*.
- T est l'ensemble des *transitions*.
- F est l'ensemble des *arcs* (flèches).
- $M_0: S \rightarrow \mathbb{N}$ appelé *marquage initial*, où, pour chaque place $s \in S$, il y a $n \in \mathbb{N}$ jetons.

$W: F \rightarrow \mathbb{N}^+$ appelé ensemble d'*arcs primaires*, assignant à chaque arc $f \in F$ un entier positif $n \in \mathbb{N}^+$ qui indique combien de jetons sont *consommés* depuis une place vers une transition, ou sinon, combien de jetons sont produits par une transition et arrivent pour chaque place.

$K: S \rightarrow \mathbb{N}^+$ appelé *limite de capacité*, faisant correspondre à chaque place $s \in S$ un nombre positif $n \in \mathbb{N}^+$ représentant le nombre maximum des jetons qui peuvent occuper une place.

3.1.3 Représentation d'état

En automatique, une représentation d'état (figure I.4) permet de modéliser un système dynamique sous forme matricielle en utilisant des variables d'état. Cette représentation, qui peut être linéaire ou non, continue ou discrète, permet de déterminer l'état du système à n'importe quel instant futur si l'on connaît l'état à l'instant initial et le comportement des variables exogènes qui influent sur le système.

Les variables d'état sont des grandeurs, qui le plus souvent ont une signification physique, et qui sont rassemblées dans un vecteur x .

La représentation d'état des systèmes linéaires invariants, quand ils sont à temps continus, s'écrit sous la forme de système des équations (I.3) :

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (I.3)$$

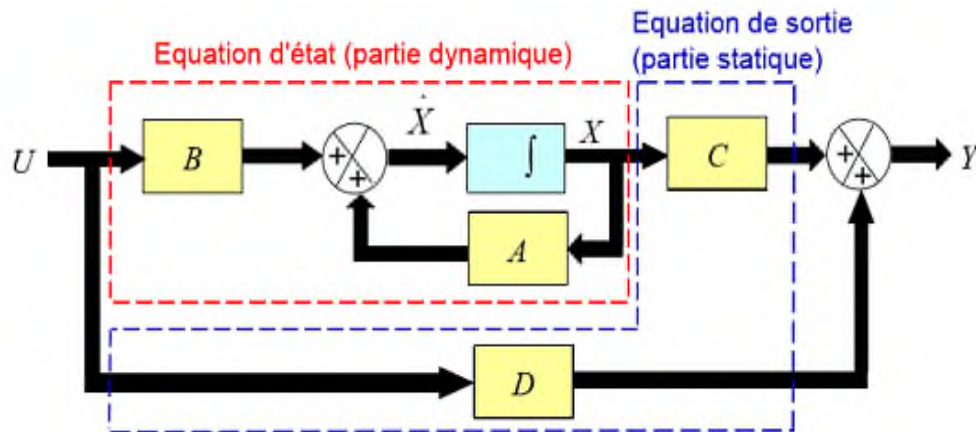


Figure I.4 : Représentation d'état

3.1.4 Algèbre de Boole

Une algèbre permettant de manipuler les propositions logiques au moyen d'équations mathématiques où les énoncés VRAI et FAUX sont représentés par les valeurs 1 et 0, tandis que les opérateurs ET et OU deviennent des opérateurs algébriques de multiplication et d'addition.

Une algèbre de Boole est constituée de :

1. un ensemble E,
2. deux éléments particuliers de E : 0 et 1 (correspondant respectivement à FAUX et VRAI),
3. deux opérations binaires sur E : $+$ et $\cdot$ (correspondant respectivement au OU et ET logiques),
4. une opération unaire sur E : $\bar{}$ (correspondant à la négation logique).

3.1.5 Mathématiques tropicales

Les mathématiques tropicales, ou la géométrie tropicale sont une branche des mathématiques correspondante à l'étude d'un système modifié grâce à la redéfinition de l'addition et de la multiplication

Les mathématiques tropicales sont généralement définies grâce au :

- Minimum et à l'addition : **Algèbre min-plus**,
- Maximum et à l'addition : **Algèbre max-plus**

3.1.6 Logique temporelle

La logique temporelle est une extension de la logique conventionnelle (propositionnelle) qui intègre de nouveaux opérateurs qui expriment la notion du temps.

Il existe 2 types de la logique temporelle :

- ✓ Logique du temps linéaire permettant d'exprimer des propriétés portant sur des chemins individuels (issus de l'état initial) du programme.

- ✓ Logique du temps arborescent permettant d'exprimer des propriétés portant sur les arbres d'exécution (issus de l'état initial) du programme.

3.2 Déterministe non-déterministe

Dans l'industrie les types des SED peuvent être devisés en deux types :

Un système déterministe est un système qui réagit toujours de la même manière à l'occurrence d'une séquence des évènements. L'état du système est décrit par un processus bien défini, c'est-à-dire qu'à partir d'un état donné son évolution sera toujours identique après l'occurrence d'une séquence des évènements [7].

Un système est dit non-déterministe si une ou plusieurs de ses variables de sortie sont des variables aléatoires. Un cadre probabiliste est requis pour l'étude de ces systèmes. [7]

3.3 Applications autour des SED

De la littérature se dégage de multiples applications dans l'industrie en relation avec les SED.

3.3.1 *Contrôle*

C'est la fonction de déclencher l'exécution d'un ensemble d'opérations en donnant des ordres aux actionneurs du processus, qui peuvent être un ensemble d'opérations correspondant à la séquence de fabrication du produit, un ensemble d'opérations exécutées dans l'ordre pour restaurer la fonctionnalité de processus proposé lors de l'exécution normale, des actions avec un niveau de priorité élevé appliqué afin de protéger les utilisateurs et de prévenir l'évolution catastrophique ou certaines opérations de vérification, ou des opérations de réglage ou nettoyage exécutées afin de maintenir le processus dans un état opérationnel [8].

3.3.2 *Supervision*

C'est la fonction de calculer et de définir les paramètres de la séquence de commande à exécuter en fonction de l'état du système de contrôle et de l'état du processus. Ceci inclut les opérations normales et anormales. En fonctionnement normal, la supervision prend les décisions dans le système de commande (ordonnancement temps réel, l'optimisation, ensembles de contrôle et de passage d'une loi de commande à l'autre). En cas de défaillance, la supervision prend toutes les décisions nécessaires pour permettre au système de reprendre le fonctionnement normal (rééchelonnement, les mesures de rétablissement, les procédures d'urgence, ...). Il est à noter que la supervision a lieu dans une structure hiérarchique (d'au moins deux niveaux). Au niveau le plus bas, seules les fonctions de contrôle et de surveillance sont mises en place - pas de véritables décisions doivent être prises [8].

3.3.3 *Surveillance*

C'est la fonction de collecter les données des processus et du contrôleur, déterminer l'état actuel du système contrôlé et rendre les inférences nécessaires pour produire des données supplémentaires

(historique, diagnostic, etc.). La surveillance est limitée au traitement des données et n'a pas d'action directe sur les modèles ou sur le processus [8].

3.3.4 Diagnostic

C'est la fonction qui établit les liens de causalité, entre les symptômes observés, et les défauts qui sont survenus, leurs causes et leurs conséquences. Il s'intègre plus généralement dans le cadre de la supervision et la surveillance, et considéré comme un système d'aide à la décision. Cette fonction est généralement divisée en trois sous fonctions :

- La localisation : qui sert à déterminer le sous-système qui est l'origine du défaut.
- L'identification : qui sert à caractériser les causes d'une défaillance.
- L'explication : élabore les conclusions du diagnostic [8].

3.3.5 Robustesse

La robustesse d'un système peut être définie comme sa stabilité de ses performances. Autrement dit c'est sa capacité à conserver les propriétés spécifiées en présence de variations ou de perturbation prévues ou imprévues [9].

3.3.6 Modélisation

La modélisation d'un système physique est une description de sa structure à l'aide des équations mathématiques, des règles ou des modèles graphiques. Cette description peut être :

- Une représentation comportementale qui consiste à déterminer les relations de causes à effets entre les différentes variables de système.
- Une représentation fonctionnelle qui se focalise sur les objectifs que le système doit accomplir.

3.3.7 Simulation

C'est la démarche qui consiste à reproduire d'une façon artificielle le fonctionnement d'un phénomène ou d'un système donné. Ceci en construisant un modèle et en observant le comportement de ce modèle lors de la variation de ses paramètres et lors des perturbations [10].

3.4 Structures de prise de décision

Ces applications peuvent être implémentées à un SED selon plusieurs différentes structures (figure I.5), nous citons (pour le cas de diagnostic) :

- La structure centralisée : un seul module de diagnostic (appelé diagnostiqueur) est associé avec le modèle global du procédé. Ce module rassemble les informations nécessaires du procédé avant de prendre la décision finale sur l'état de fonctionnement du procédé.
- La structure décentralisée : structure basée sur un modèle global du système, à qui nous associons plusieurs diagnostiqueurs locaux indépendants, en mesure de gérer les pannes

et leur propagation. Chacun de ces diagnostiqueurs reçoit des observations spécifiques afin de prendre une décision locale basée sur ses observations locales.

- La structure distribuée : structure composée de plusieurs sous modèles chacun a son diagnostiqueur local. Ce type de structure nécessite un protocole de communication entre les différents diagnostiqueurs locaux pour affiner le diagnostic.

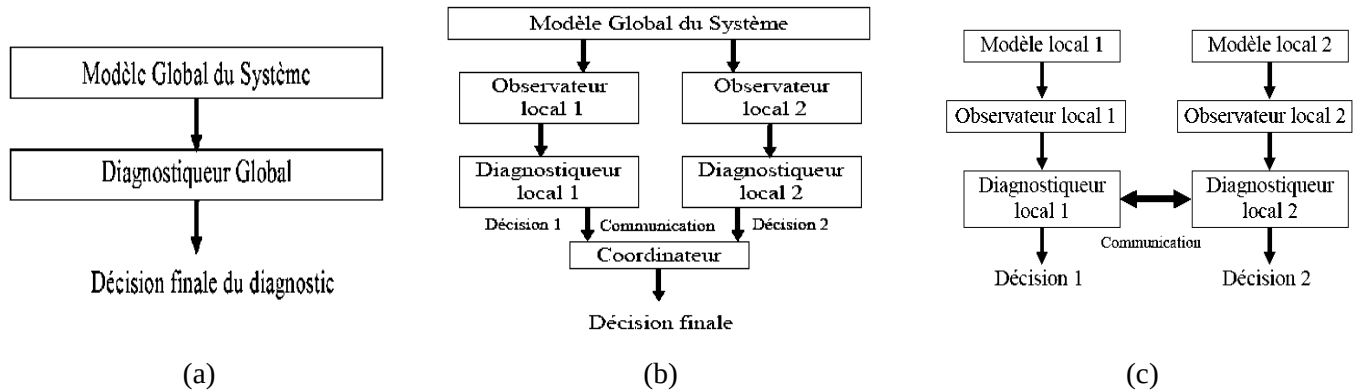


Figure I.5 : Structures de prise de décision, (a) centralisée, (b) décentralisée, (c) distribuée

4. Analyse de la tendance de recherche actuelle autour des SED

4.1 Méthodologie

La théorie des systèmes à événements discrets a été largement appliquée dans plusieurs domaines en industrie surtout l'industrie automobile, l'industrie aéronautique et les systèmes manufacturiers. Ainsi les SED ont fait l'objet de plusieurs recherches scientifiques et ont constitué, depuis bientôt 40 ans, un domaine de recherche très actif (vu son intérêt théorique et économique) qui a donné naissance à de nombreuses publications.

L'ensemble des états d'art [11, 12], proposés par les chercheurs, en relation avec les SED se focalise sur un axe de recherche précis et ignore les autres. Ceci ne permet pas de faire une comparaison entre les différents axes de recherche en relation avec les SED. Nous visons, à travers cette partie, d'extraire l'axe de recherche le plus actif, en relation avec les SED, cet axe de recherche fait l'objet de développement par notre équipe de recherche. Cette comparaison va aussi permettre aux nouveaux chercheurs d'être informés sur l'axe de recherche le plus convenable à développer.

Les travaux de recherche sur les SED sont définis par quatre éléments :

- L'outil de modélisation (voir § I.2.1) ;
- Le type du système : déterministe/non déterministe (voir § I.2.2) ;
- L'application : contrôle, diagnostic la supervision, la modélisation, la surveillance et la simulation (voir § I.2.3) ;
- Le type de la structure de prise de décision : centralisée, décentralisée, distribuée (voir § I.2.4).

Afin de mener cette comparaison, nous avons analysé une base de données contenant plus que 2000 articles publiés dans des revues indexées (Scopus ou Thomson Reuters) pendant les trois dernières décennies, et qui couvrent la majorité des axes de recherche en relation avec les SED.

Dans un premier temps, nous comparons le degré d'utilisation des différents outils de modélisation en fonction des années de publication. Nous construisons dans la figure I.7, les histogrammes qui traduisent le nombre des papiers qui utilisent les différents outils de modélisation (AF, les RdP et les autres outils) en fonction de leurs années de publication. Ensuite nous traçons, dans la figure I.10, les courbes représentant la variation des pourcentages des articles publiés dans chaque domaine en fonction de leurs années de publication. Et à la fin nous traçons, dans la même figure, la courbe de tendance linéaire de chacune de ces courbes afin de comparer leurs progressions. Selon la pente de cette courbe de tendance, et comme le montre la figure I.6, un axe de recherche peut avoir trois cas possibles :

- Axe de recherche actif si la courbe est toujours en croissance ;
- Axe de recherche saturé si la courbe est décroissante dans les dernières années ;
- Axe de recherche presque saturé si la courbe devient constante dans les dernières années.

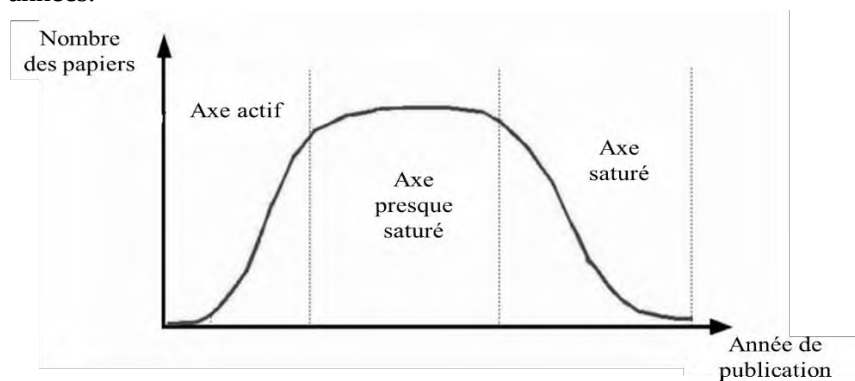


Figure I.6 : Interprétation des courbes

4.2 Synthèse

La figure I.7 compare entre l'utilisation des différents outils de modélisation en fonction des années de publication. Il est clair que le nombre des papiers qui ont utilisé l'automate fini surpasse ceux des papiers qui ont utilisé les autres outils. Ceci est dû au fait que l'automate fini a constitué la base des premiers travaux dans le domaine des SED.

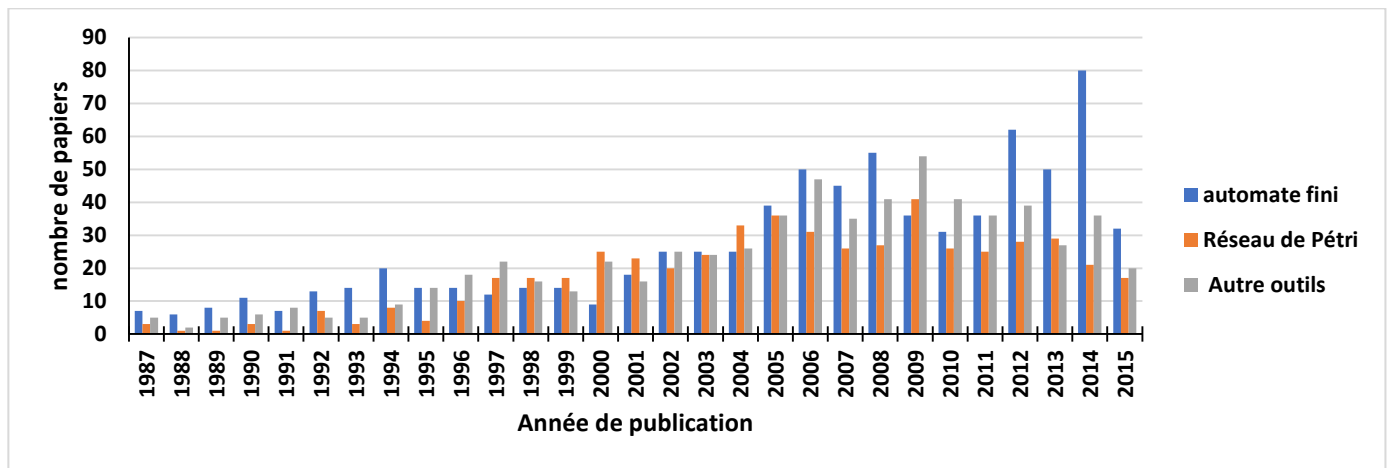


Figure I.7 : utilisation des différents outils de modélisation en fonction des années de publication.

Les figures I.8 et I.9 montrent que les architectures centralisées déterministes font l'objet de la majorité des papiers scientifiques en relation avec les SED.

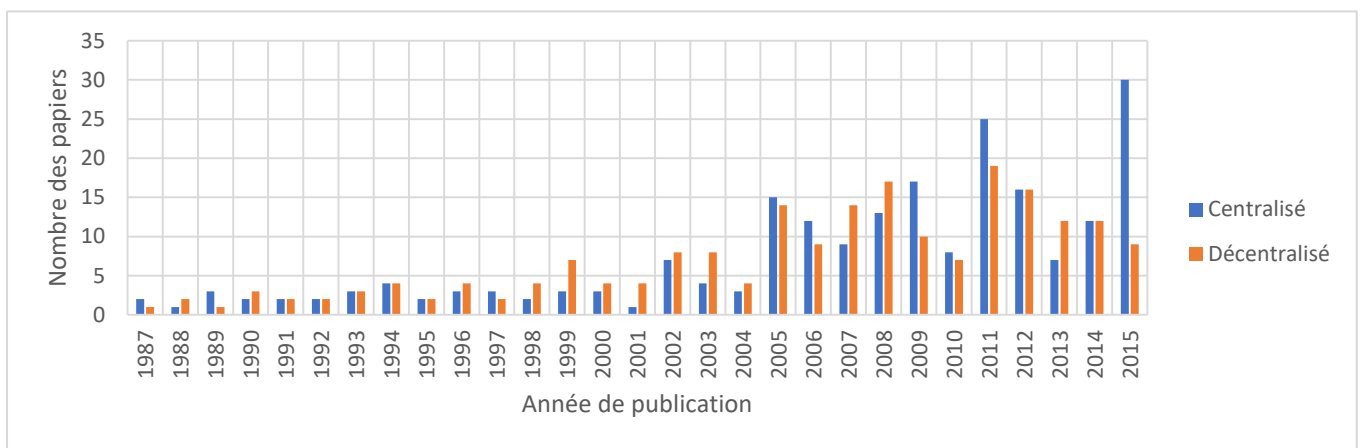


Figure I.8: Nombre de papiers selon l'architecture de prise de décision : centralisé et décentralisé.

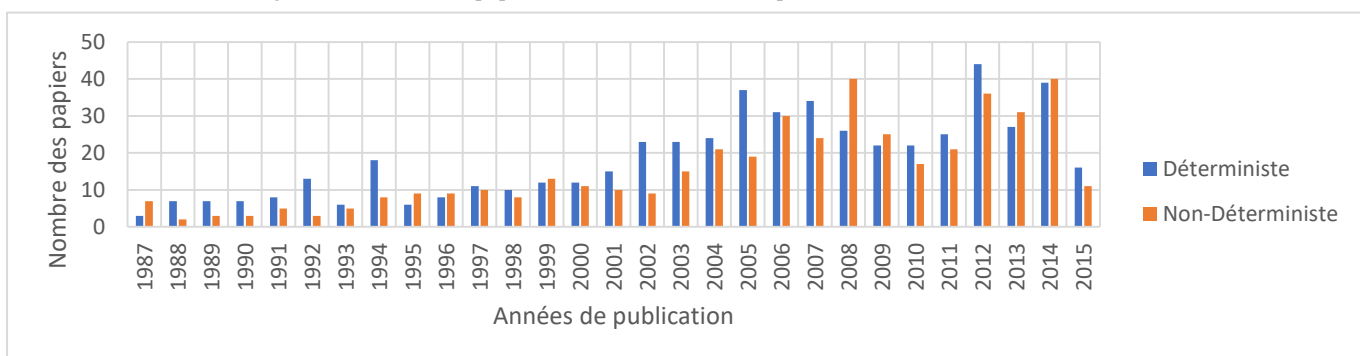


Figure I.9: Nombre de papiers selon le type de système étudié : déterministe et non-déterministe

La figure I.10 illustre la variation des pourcentages des articles publiés dans les différents domaines en fonction de leurs années de publication. Cette figure donne aussi les courbes de tendance linéaires de chaque domaine. Après une comparaison entre les pentes de ces derniers, nous constatons que :

- La courbe de tendance du contrôle est en décroissance

- Les courbes de tendance de la supervision et de la surveillance ont des allures quasiment constantes
- Les courbes de tendance du diagnostic, de la modélisation et de la simulation ont des allures croissantes. Cependant, la courbe de diagnostic à une pente fortement croissante par rapport à celles de la courbe de modélisation et de simulation.

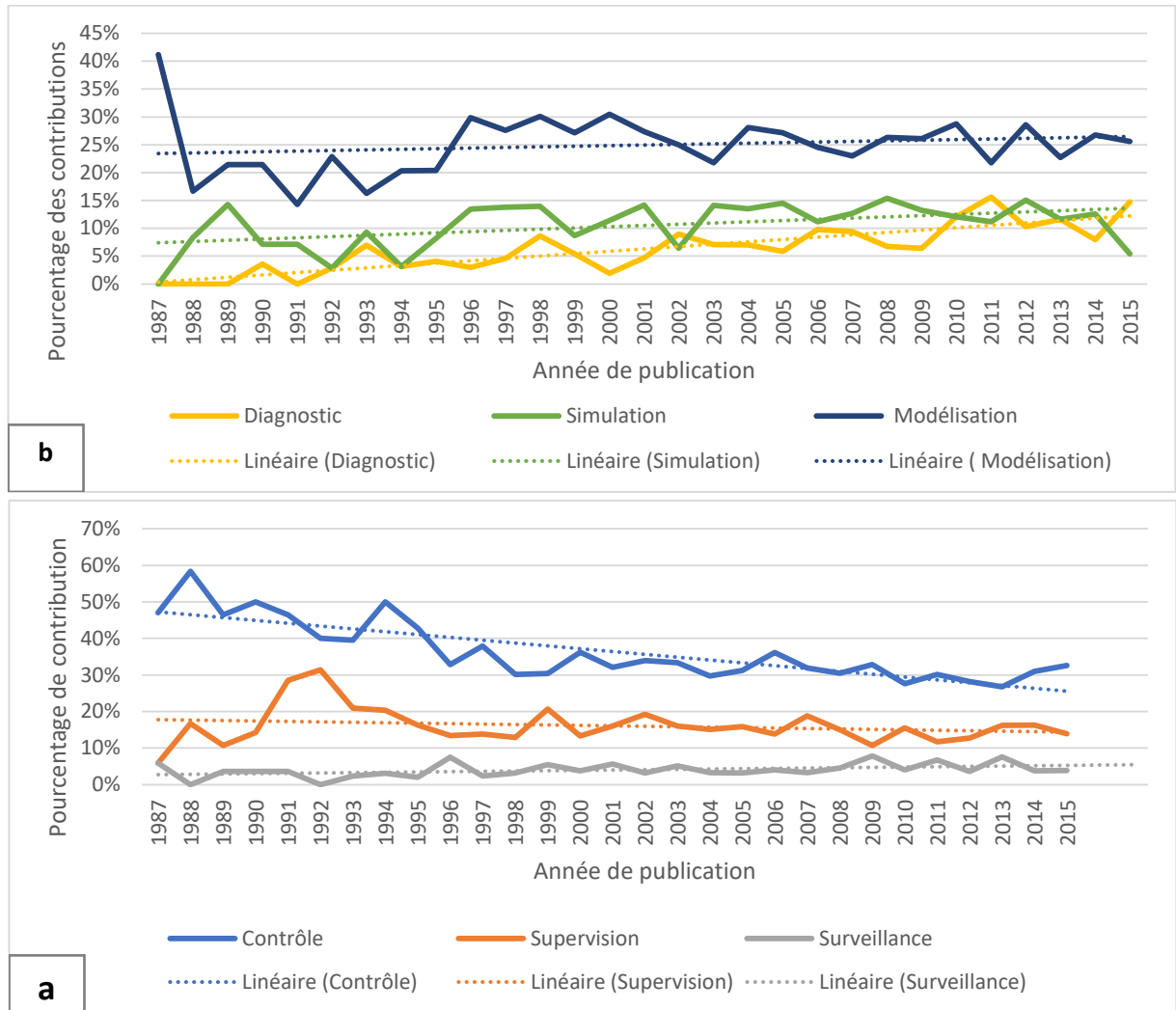


Figure I.10 : La variation des pourcentages des articles publiés dans les différents domaines en fonction des années de publication : (a) contrôle, supervision et surveillance. (b) diagnostic, simulation et modélisation.

À l'issue de cette analyse de la tendance de recherche actuelle en relation avec les SED, nous avons conclu que l'utilisation de l'**automate** pour le **diagnostic centralisé** des SED **déterministes** constitue l'axe de recherche le plus actif actuellement. Ceci a mené notre équipe de recherche à se focaliser sur le développement de trois axes de recherche où différents outils sont utilisés tels que les automates finis, les RdP, leurs extensions et quelques outils de l'intelligence artificielle. Ces axes sont intitulés comme suit :

- L'élaboration d'un superviseur à base des réseaux de neurones (RdN) pour le diagnostic et le pronostic des SED ;

- Synthèse de contrôleurs et diagnostic décentralisé des SED en milieu industriel ;
- Développement d'une méthode de diagnostic des systèmes industriels hybrides.

Cette thèse a été consacrée à « l'élaboration d'un superviseur à base des RdN pour le diagnostic et la maintenance des SED », mais dans un premier temps, nous allons présenter l'état de l'art concernant le diagnostic des fautes dans les SED.

5. État de l'art sur le diagnostic et le pronostic des fautes dans les SED

Les méthodes de diagnostic des SED peuvent être divisées en deux grandes catégories selon le degré de maîtrise de fonctionnement de SED (figure I.11) :

- **Méthodes à base de modèle (ou méthodes internes)** qui sont basées sur des modèles quantitatifs et/ou qualitatifs (comme les automates finis et les réseaux de Pétri). Ces méthodes sont utilisées lorsqu'on a une bonne connaissance de fonctionnement de SED ;
- **Méthodes sans modèle (ou méthodes externes)** qui sont des méthodes à base de connaissances (systèmes experts, arbres de défaillances, etc.) ou à base d'apprentissage automatique (reconnaissance des formes, réseaux de neurones, etc.). Ces méthodes ont été utilisées lorsqu'on n'avait pas des informations suffisantes sur le fonctionnement de SED pour construire un modèle qui décrit tous les comportements potentiels de ce SED.

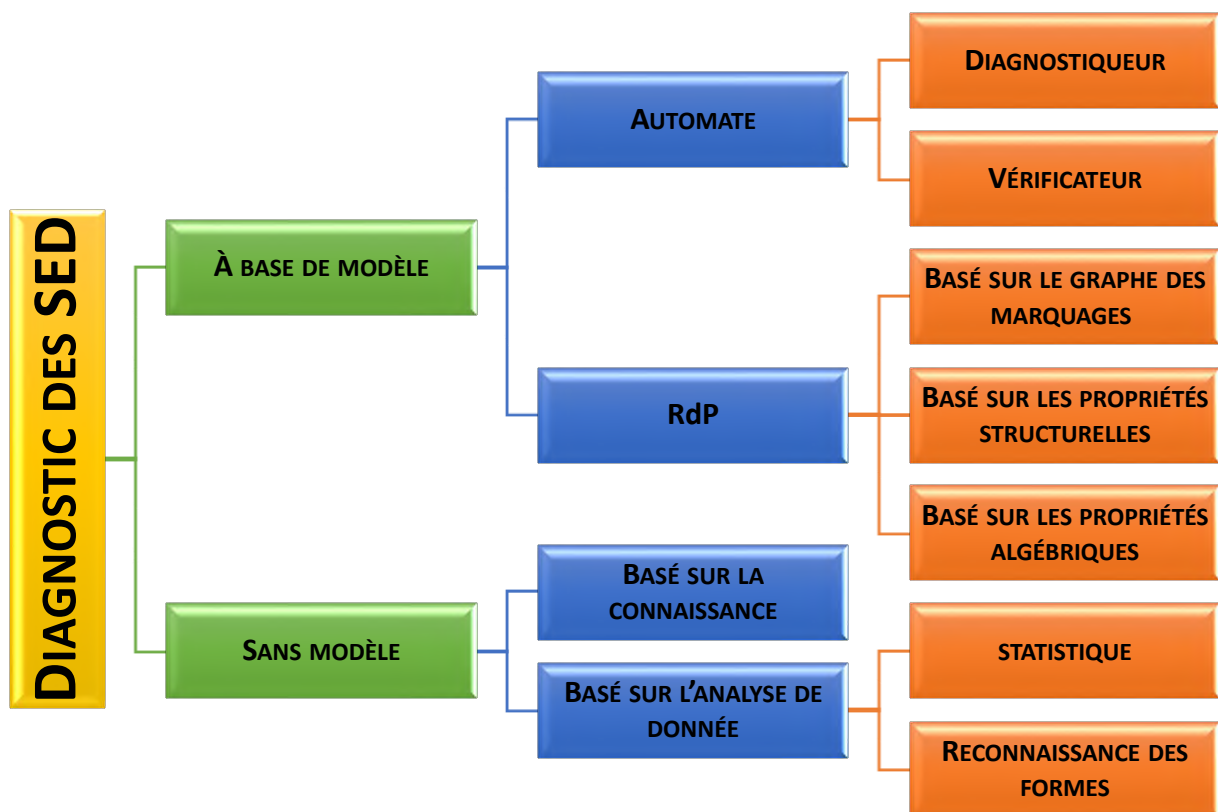


Figure I.11 : Classification des méthodes de diagnostic des SED

5.1 Définitions

La diversité des définitions, en relation avec les différentes notions du diagnostic des SED, existantes dans la littérature nous oblige à choisir le lexique le plus convenable pour la compréhension des

différentes méthodes de diagnostic que nous allons résumer par la suite et la compréhension aussi de l'approche qui sera développée dans la suite de ce travail.

Faute : C'est un événement qui perturbe le comportement de système et qui cause la dégradation d'une de ses fonctions ou la non-prise en compte correcte d'une directive, d'une contrainte exprimée par le cahier des charges.

Dégradation : C'est une perte progressive d'une fonction jusqu'à la disparition totale de cette fonction. La dégradation peut être décrite par la croissance ou la décroissance d'une variable d'état. Si cette variable dépasse un seuil limite défini dans les spécifications fonctionnelles de cet équipement, le système passe à un état de défaillance.

Défaillance : C'est un événement qui cause l'altération ou la cessation de l'aptitude d'un ensemble à accomplir sa ou ses fonctions requises avec les performances définies dans les spécifications techniques.

Panne : C'est un état de fonctionnement où une entité (composant ou système) est incapable d'assurer une fonction requise. Cet état peut être causé par une défaillance ou une erreur de conception.

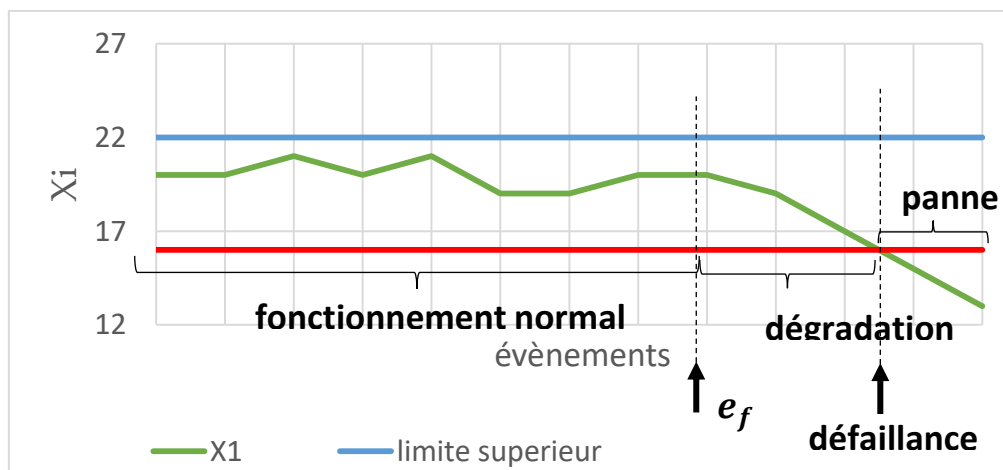


Figure I.12 : les variations d'une variable lors de fonctionnement normal, La dégradation, la défaillance et la panne causés par une faute e_f .

Mode de fonctionnement : Un mode de fonctionnement décrit la façon dont le système se comporte afin d'atteindre un objectif spécifique. Un système peut avoir plusieurs modes de fonctionnement normaux, d'autres anormaux et d'autres de dégradation.

Système complexe : C'est un ensemble structuré de plusieurs équipements hétérogènes indépendants qui sont en interaction, connectés et communiquent entre eux dans le but d'assurer une fonction bien définie.

Diagnostic : Selon l'AFNOR, 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 provenant d'une inspection, d'un contrôle ou d'un test.

Hors-ligne diagnostic : Dans ce type de diagnostic aucune limitation sur le temps de réponse n'est exigée.

En-ligne diagnostic : Consiste à donner les résultats de diagnostic en temps réel pendant que le système est en fonctionnement.

5.2 Méthodes de diagnostic à base de modèle

Dans la littérature en relation avec le diagnostic des fautes dans les SED, la majorité des travaux de recherche proposés ont été basés sur des modèles graphiques (automate fini ou réseau de Petri). Ces travaux ont été initiés par ceux de Lin [13] qui a proposé une approche basée sur les états de SED pour le diagnostic de ces fautes. Dans ces travaux le diagnostic en-ligne et hors-ligne et la notion de diagnosticabilité dans les deux cas ont été discutés. Il a aussi présenté un algorithme qui permet le diagnostic.

5.2.1 Diagnostic par les automates finis

L'automate fini a fait la base des premiers travaux dans ce domaine et il représente le cœur de nombreuses approches :

Approche Diagnostiqueur

Les travaux de Sampath *et al.* (1995[1], 1996[14]) font, de nos jours, la base de la plupart des travaux qui concernent le diagnostic des fautes dans les SED modélisés par les automates finis et ils ont même présenté une source d'inspiration pour les autres travaux basés sur les autres outils de modélisation. Ils ont considéré les fautes comme des événements non observables et ils ont présenté une approche applicable pour une large classe des SED. Ces approches permettent de vérifier le mode de fonctionnement du SED après chaque changement de l'état du système. Ceci est élaboré en utilisant la notion d'observateur introduit dans [15] pour construire un diagnostiqueur pour le SED considéré. Ils ont aussi présenté des conditions nécessaires et suffisantes pour la diagnosticabilité du système en utilisant la notion de langage formel. Nous rappelons, par la suite, les notions d'observateur et de diagnostiqueur.

Définition 4 : l'observateur est un automate fini construit à partir de l'automate fini du SED considéré qui donne l'ensemble des états du système possibles après l'occurrence de chaque séquence des événements observables.

Définition 5 : le diagnostiqueur est un observateur augmenté par des étiquettes associées aux états du système. Le diagnostiqueur donne, en plus de l'état de système, une estimation de son comportement (normal ou anormal) le long de chemin suivi pour arriver à cet état.

Le diagnostiqueur pour un SED modélisé par un automate G (I.2) est l'automate fini $G_d = (Q_d, \Sigma_o, \delta_d, q_0)$ tel que Q_d, Σ_o, δ_d et q_0 ont les mêmes interprétations usuelles. q_0 est l'état initial de diagnostiqueur défini par $\{(x_0, \{N\})\}$ ce qui signifie que le SED est à son état initial avec un fonctionnement normal. Chaque état de diagnostiqueur $q_i \in Q_d$ est un sous-ensemble de $X_o \times \{N, F_1, \dots, F_m\}$, où X_o est l'ensemble des états de l'automate G qui ont une transition entrante étiquetée par un évènement observable, N et F_i sont des étiquettes. N indique que le système a atteint l'état actuel en suivant un chemin sans aucune faute et F_i indique que le système a atteint l'état actuel suivant un chemin contenant une faute appartenant à la classe des fautes F_i .

Un état est dit normal si toutes ses étiquettes sont des N (par exemple l'état $\{2N, 4N\}$ de la figure I.13), de fonctionnement anormal si toutes ces étiquettes sont des F_i (par exemple l'état $\{3F\}$ de la figure I.13) et il est incertain s'il contient des étiquettes N et F_i (par exemple l'état $\{1N, 3F\}$ de figure I.13). Un langage généré par un automate est considéré comme diagnosticable si et seulement si son diagnostiqueur ne comporte aucun cycle des états incertains.

Cassandras [7] a présenté une version améliorée du diagnostiqueur développé par Sampath *et al.* [1, 14] où les états atteints par une transition entrante étiquetée par un évènement non observable sont introduits dans le diagnostiqueur.

Exemple 2 :

Nous considérons un SED défini par les ensembles $\Sigma_o = \{a, b, c\}$, $\Sigma_f = \{f\}$, $\Sigma_{reg} = \{e_r\}$ et modélisé par l'automate fini de la figure I.13.a. Cette figure donne aussi son observateur et son diagnostiqueur dans sa version améliorée. L'évènement a s'est produit, le diagnostiqueur montre qu'on est à l'état $\{1N, 3F\}$. Ce qui signifie que ce SED peut être dans l'une des deux états : l'état 1 avec un fonctionnement normal ou l'état 3 avec un fonctionnement anormal.

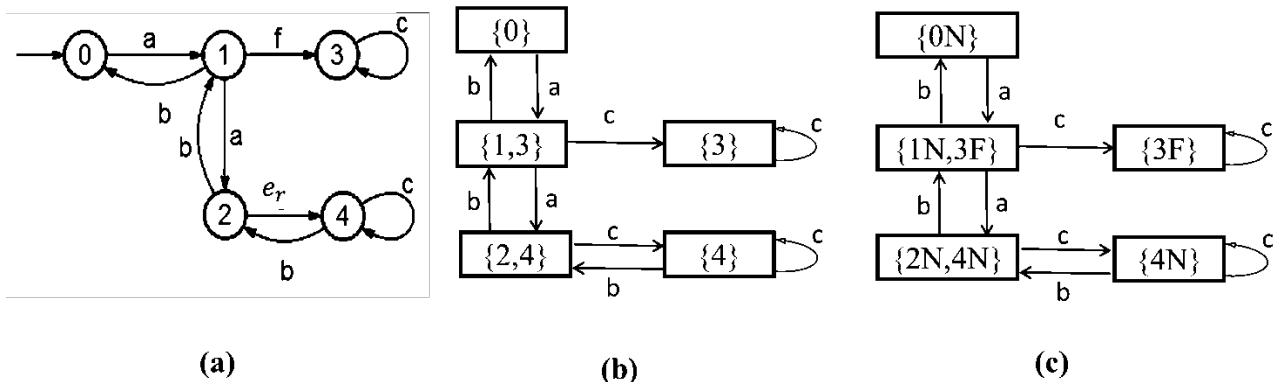


Figure I.13 : (a) SED modélisé par automate fini, (b) son observateur, (c) son diagnostiqueur

Ces méthodes nécessitent la construction de diagnostiqueur complet ce qui peut engendrer une explosion combinatoire. Cette complexité est traduite par le nombre des états de diagnostiqueur qui augmente de façon exponentielle au nombre des états de l'automate et le nombre des types des fautes. Si $|X|$ est le nombre des états de l'automate et $|\Sigma_f|$ est le nombre des fautes, le diagnostiqueur peut avoir jusqu'à $2^{|X| \times 2^{|\Sigma_f|}}$ états.

Approche vérificateur ou modèles jumeaux « Twin-plant »

Afin de réduire la complexité rencontrée lors de la construction des diagnostiqueurs proposés par Sampath [1, 14] et Cassandras [7], l'approche des modèles jumeaux (twin-plant) a été proposée par les auteurs de [16, 17]. L'idée principale de cette approche est de construire une composition parallèle de l'automate, qui décrit le système considéré, avec lui-même après l'élimination de toutes les transitions non observables. Le résultat de cette composition est appelé modèle jumelle ou vérificateur. L'analyse de la structure de ce vérificateur permet le diagnostic des fautes. Le SED considéré est diagnosticable si et seulement s'il n'existe aucune séquence des événements observables qui mène à deux cycles différents des états, dont une contient une faute et l'autre non. La complexité de cette approche est polynomiale. Cependant, dans l'approche de diagnostiqueur la complexité est exponentielle.

Cette approche a été améliorée par Moreira [18], qui a proposé une composition parallèle de l'automate qui modélise le comportement total du SED considéré (fonctionnement normal et anormal) ainsi que la réduction de la taille de vérificateur.

5.2.2 Diagnostic en utilisant les RdP

Certains auteurs, et pour échapper au problème de l'explosion combinatoire rencontrée lors de l'utilisation des AF afin de résoudre les problèmes de diagnostic des fautes dans les SED, ont adopté les RdP. Surtout que cet outil modélise facilement le parallélisme et la synchronisation et offre un riche cadre mathématique, ce qui le rend plus adapté à un grand nombre d'applications pratiques. Ainsi, les RdP sont devenus incontournables pour la modélisation des différents types des systèmes (manufacturiers, informatiques, logistiques...), et par la suite, ils ont constitué dernièrement la base de la plupart des contributions autour de diagnostic des SED. Les méthodes proposées dans la littérature se divisent en trois catégories :

- **Méthodes basées sur le graphe des marquages ;**
- **Méthodes basées sur les propriétés structurelles ;**
- **Méthodes basées sur des techniques algébriques.**

Par la suite, nous détaillerons chacune de ces méthodes.

Méthodes basées sur le graphe des marquages

La première contribution a été proposée par Ushio et al. [19], où il a étendu le travail de Sampath [1, 14] pour qu'il soit applicable au cas des RdP non bornés. Dans ce travail, les auteurs considèrent que l'ensemble des transitions sont non-observables et une partie des places sont observables. Ils ont construit un diagnostiqueur à base des RdP, appelé « simple diagnostiqueur ». Cette méthode demande un effort considérable de calcul, problème qui a évoqué plusieurs autres contributions proposent des solutions, notamment l'utilisation d'une architecture de diagnostic distribuée avec des modèles de RdP couplés par des places communes [20, 21]. Les auteurs de [22] ont proposé une méthode qui permet un diagnostic en-ligne des RdP labélisés et ils ont montré que l'analyse seulement d'une partie du graphe des marquages est suffisante pour l'analyse de la diagnosticabilité et pour exécuter un diagnostic en-ligne.

Benveniste *et al.* [23] ont utilisé le dépliage des RdP (unfolding Petri net) qu'a comme objectif la représentation explicite de l'ordre partiel des événements du système étudié par un RdP particulier construit en-ligne. Ceci a été appliqué à une architecture des capteurs distribués et a permis d'effectuer un diagnostic en-ligne et d'éviter l'énumération totale de l'espace d'état. Ce travail a été étendu plus tard dans [24].

Une autre technique dite graphe d'accessibilité de base est proposée dans [25-27] et qui s'appuie sur les RdP étiquetés pour construire une partie réduite de graphe de marquage. Ce graphe est construit en utilisant la notion des explications minimales inspirée de [28] et qui correspondent aux séquences d'événements non mesurés de longueur minimale qui doit être produit avant le franchissement d'un événement mesuré. Dans [29] un modèle réduit appelé automate-ROF pour tester la diagnosticabilité des RdP bornés en assumant que la partie non mesurée est acyclique (ROF pour automate, qui ne contient que les événements observables et les événements des fautes).

Boussif a proposé dans [30] la construction d'un diagnostiqueur semi-symbolique et il a comparé par la suite cette approche dans [31] avec deux autres approches du diagnostiqueur [22, 32].

Méthodes basées sur les propriétés structurelles

L'une des premières contributions qui ont utilisé les propriétés structurelles des RdP pour le diagnostic des SED est de Prock [33] où le système est représenté par un RdP et les fautes sont détectées par une variation des jetons dans le P-invariants (le P-invariants est la solution de l'équation $C^T x = 0$).

Quant à Wen et Jeng [34], le test de la diagnosticabilité est fait en utilisant la propriété de T-invariant de RdP (le T-invariants est la solution de l'équation $Cx = 0$). Ils ont utilisé le diagnostiqueur de Ushio [Ushio *et al.* (1998)] pour vérifier la validité de leur approche. Ils ont même présenté un algorithme,

basé sur la programmation linéaire, pour le calcul d'une condition suffisante pour la diagnosticabilité des SED modélisé par des RdP.

Dans [35], la partie nominale du RdP interprété est utilisée pour la construction de diagnostic des SED. Un algorithme, qui compare la sortie du système avec celle du diagnostiqueur pour la détection d'une déviation du comportement synonyme de l'occurrence d'une faute, est présenté.

Genc et Lafortune (2007 [21]) ont proposé un diagnostic basé sur une approche modulaire. Ils ont construit pour chaque module un diagnostiqueur. Ils ont établi une connexion entre les différents modules afin de préserver le comportement réel du système et de donner une décision finale de diagnostic. Ceci ne permet pas d'éviter l'explosion combinatoire des états, cependant ce problème peut être surmonté en modélisant le SED sous forme d'une collection des RdP.

L'approche « vérificateur » [16, 17] basé sur les AF a inspiré plusieurs autres travaux de recherche telles que [27, 36-38] et plus récemment [39, 40] pour effectuer un diagnostic des SED modélisé par des RdP bornés. Ceci a été effectué en créant une copie de RdP modélisant le SED, renommant les transitions silencieuses, éliminant les transitions de faute, effectuant le produit asynchrone entre cette copie et le modèle de SED et en fin vérifie la présence, sur ce nouveau modèle, de cycles ambigus.

Méthodes basées sur les techniques algébriques

La plupart des méthodes basées sur les techniques algébriques utilisent les propriétés matricielles des RdP. Nous citons :

Wu and Hadjicostis [41] ont utilisé la redondance dans un RdP donné en utilisant une technique de décodage algébrique. Dans ce papier l'auteur a considéré deux types des fautes : ceux qui causent un marquage incorrect et ceux qui interrompent la mise à jour de marquage après l'occurrence d'un événement.

Lefebvre and Delherm [42] ont utilisé un modèle qui inclut même les fautes associées à quelques transitions et ils ont présenté une approche pour la détermination de l'ensemble des places qu'ils doivent être observable pour la bonne estimation des fautes.

Il existe d'autres approches basées sur les modèles, telle que [43-45] qu'ont généralisé le travail du M. Sampath [1] pour qu'il soit adopté aux structures distribuées.

5.3 Méthodes de diagnostic sans modèle

Par opposition aux méthodes nécessitant un modèle décrivant l'ensemble des modes de fonctionnement du système, les méthodes sans modèle s'appuient uniquement sur les données collectées sur le système [46] ou sur le degré de connaissance du système. Ces méthodes ont une grande efficacité pour les systèmes larges où la modélisation par les méthodes citées dans la section 5.2 est difficile ou parfois impossible.

5.3.1 Méthodes basées sur l'expertise

L'idée principale de ces méthodes est d'établir les relations de causalité potentielle entre les symptômes, les défaillances et les fautes, en se basant sur la connaissance observée du système. Cette connaissance peut être collectée à partir d'une analyse structurelle ou fonctionnelle du système. Il existe plusieurs approches qui s'appuient sur la connaissance, nous détaillerons par la suite quelques-uns.

La méthode AMDEC (Analyse des Modes de Défaillance et de leurs Criticités) [47] consiste à décomposer le système, lister la fonction et l'ensemble des modes de défaillance que peut avoir chaque composant de système et ainsi relier chaque mode de défaillance avec son élément correspondant. Cette approche est très répandue dans les systèmes industriels. Cependant, elle est lourde à mettre en place pour des systèmes complexes et elle est souvent insuffisante à cause de la nécessité d'identifier tous les composants du système ainsi que tous les dysfonctionnements relatifs à chaque composant.

Les systèmes experts [48, 49] se basent essentiellement sur l'expertise humaine et l'analyse des données historique pour mettre en place des règles de la forme : si P est vrai (*symptôme*) et si on sait que P implique Q (*une défaillance*), alors, Q est vrai. Ceci permet de relier chaque comportement anormal du système avec une défaillance potentielle. Cependant, il est à noter que ce type d'approches dépend fortement des connaissances des experts, et il engendre souvent des grandes bases de données que son traitement reste un problème délicat. Souvent, nous faisons recours aux méthodes de l'intelligence artificielle pour résoudre ce genre de problème.

Les arbres de défaillances [50-52] est une méthode déductive qui permet d'identifier les causes probables d'un évènement qui est généralement, un comportement anormal de système et ainsi défini des relations de causalité entre les différents évènements redoutés, et enfin déduit leurs causes probables et leurs conséquences dérivées. Cette méthode permet, et par opposition à la méthode AMDEC et les systèmes experts, de définir l'ensemble des scénarios possibles que peut avoir le système après l'occurrence d'un évènement redouté. L'élaboration d'un diagnostic correct à partir des arbres des défaillances nécessite que ces arbres expliquent tous les dysfonctionnements que peut avoir le système.

5.3.2 Méthodes basées sur l'analyse de données

Ces méthodes utilisent les informations provenant des capteurs et des contrôleurs, et regroupés sous forme d'un vecteur. À partir de ces vecteurs une base de données est conçue et qui associe chaque vecteur à un mode de fonctionnement. En général, ces approches utilisent des méthodes mathématiques et de l'intelligence artificielle qui permettent l'apprentissage et la classification de l'ensemble de cette base de données afin d'établir un modèle de référence. La logique floue et les réseaux de neurones consistent les méthodes les plus utilisées pour ce genre de problème.

La logique floue est fondée sur la théorie mathématique des ensembles flous qui, contrairement à la théorie des ensembles classique, considère des fonctions d'appartenance avec des valeurs réelles

comprises entre 0 et 1. Ces fonctions assignent à chaque vecteur les degrés d'appartenances aux différents modes de fonctionnement [53]. Ainsi il permet de diagnostiquer l'état de système en fonction de ces vecteurs.

Les réseaux de neurones [54, 55] permettent d'établir et d'apprendre les différentes relations entre les différents vecteurs contenant les informations provenant des capteurs et les différents modes de fonctionnement. Ces vecteurs sont extraits de l'historique des pannes et ils constituent une base d'apprentissage pour le réseau de neurones. Après l'apprentissage des relations entre ces vecteurs et les modes de fonctionnement de système, le réseau de neurones permet une classification en temps réel des différents comportements du système.

Plus de détails sur l'état de l'art peut être trouvé dans [56, 57]

6. Problématique

Quelques systèmes industriels actuels, comme les systèmes aéronautiques, les unités pétrochimiques, les centrales de production d'énergie..., ont intégré des technologies très avancées. Ceci a rendu ces systèmes plus complexes, plus larges et ainsi leurs analyses nécessitent beaucoup de connaissances techniques. Cette complexité qui ne cesse d'augmenter dans les SED industriels est étroitement liée aux exigences de bon fonctionnement et de sécurité qui sont de plus en plus demandées. La taille énorme et l'architecture complexe de ces systèmes les rendent plus vulnérables à commettre des fautes plus dangereuses qui peuvent menacer la sécurité du personnel et des équipements. Ce qui demande un grand développement et une adaptation des méthodes de diagnostic pour qu'elles poursuivent les développements technologiques des systèmes industriels.

Dans les SED complexes, les outils de modélisation, comme les automates finis ou les réseaux de pétri, peuvent donner des modèles énormes, difficiles à construire, demande une bonne maîtrise de ces outils et nécessite des calculateurs très performants afin d'assurer un diagnostic rapide des fautes les plus complexes dans un délai acceptable. Ainsi les méthodes de diagnostic qui se base sur des modèles restent insuffisantes devant la complexité de ces systèmes. Ce type des systèmes peut comporter un grand nombre des composants et une grande diversité des modes de fonctionnement ce qui rend l'application des méthodes qui se base sur l'expertise, comme l'AMDEC, l'arbre des défaillances ou les systèmes experts, est quasiment impossible. Cependant, les méthodes qui se basent sur l'analyse des données apparaissent plus convenables pour assurer le diagnostic de ces systèmes. En effet, ces méthodes peuvent extraire, à partir d'une grande base de données, les informations utiles qui peuvent servir aux opérations de diagnostic et ainsi elles peuvent garantir la rapidité et l'efficacité de ces opérations.

Les méthodes de diagnostic peuvent être renforcées en développant aussi des méthodes de pronostic qui peuvent prédire l'occurrence d'une faute et ainsi éviter ses fautes dérivées et ses conséquences

dangereuses [13] en reconfigurant le système ou en réparant les composants responsables de l'occurrence de cette faute.

7. Conclusion

Ce premier chapitre nous a permis de présenter les principales notions en relation avec les SED et de faire une analyse de la littérature en relation avec ce type des systèmes afin de pouvoir détecter les axes de recherche le plus convenable à développer. Ceci a donné naissance à trois axes de recherche et le travail présenté dans cette thèse porte sur un de ces axes et qui est « **l'élaboration d'un superviseur pour le diagnostic et le pronostic des SED en utilisant les réseaux de neurones** ». À la fin de ce chapitre, un résumé de l'état de l'art concernant ce thème a été présenté.

Dans le chapitre suivant, nous allons essayer de mettre en évidence les différentes notions en relation avec les réseaux de neurones, ainsi qu'une étude comparative entre les différentes architectures neuronales existantes dans la littérature.

Chapitre II:
Réseaux de neurones

1. Introduction

Les réseaux de neurones représentent des outils de l'intelligence artificielle utilisée pour résoudre une variété des problèmes, notamment ceux de la reconnaissance des formes. Vu leur capacité d'apprentissage, ils ont été largement appliqués dans divers domaines d'ingénierie et ils ont été réservés pour résoudre les problèmes les plus complexes. Dans ce chapitre, nous allons présenter une vue générale sur ces outils, leurs différentes architectures existence dans la littérature et aussi leurs domaines d'application. À la fin de ce chapitre, nous allons présenter une étude comparative entre les différentes architectures des réseaux de neurones ainsi qu'une méthode pour choisir l'architecture la plus convenable pour le diagnostic/pronostic des fautes dans les systèmes à événements discrets.

2. Neurone formel

Le neurone formel ou le neurone simple (figure II.1) est le premier réseau de neurones développé, il constitue aujourd'hui la base de tous les réseaux de neurones artificiels (RdN). Il a été développé initialement par Warren McCulloch et Walter Pitts (1943 [58]) et il a été considéré comme une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique, en particulier la sommation pondérée des entrées. Cette sommation est appelée activation et se calcule comme suit :

$$y = \vec{w} \cdot \vec{x} = \sum_{i=1}^n w_i x_i \quad (\text{II.1})$$

Telle que x_i est la $i^{\text{ème}}$ entrée de neurone et w_i est son poids associé.

La figure II.1 représente un neurone formel qui possède généralement plusieurs entrées et une sortie. La fonction d'activation est la fonction de transfert qui relie la sommation pondérée au signal de sortie. Cette fonction peut avoir différentes formes. La figure II.2 présente les fonctions d'activation les plus utilisées dans la littérature. Le choix d'une de ces formes dépend de l'architecture neuronale choisie et l'application visée.

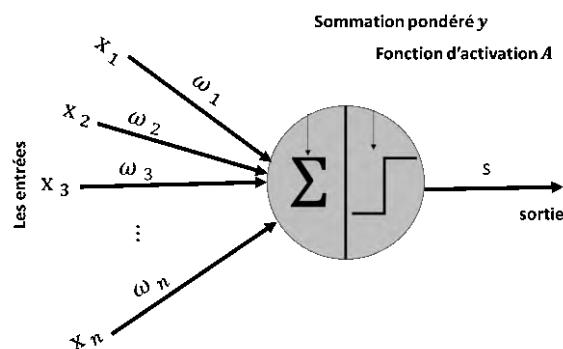


Figure II.1: Neurone formel.

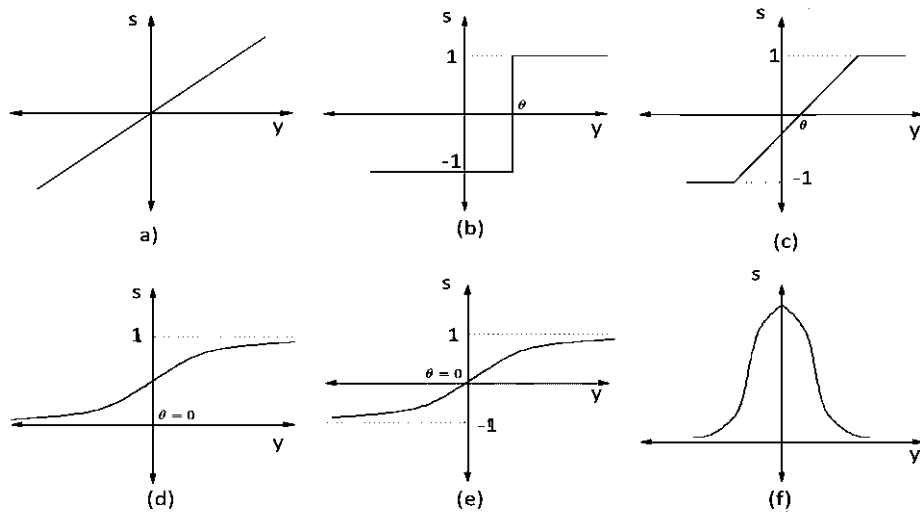


Figure II.2: Fonctions d'activation les plus utilisées, **a)** fonction linéaire, **b)** fonction à seuil, **c)** fonction linéaire par morceaux, **d)** fonction sigmoïde, **e)** fonction tangente hyperbolique, **f)** fonction gaussienne.

Le neurone formel tout seul ne possède pas une grande force calculatoire, mais cette force apparaît lorsqu'il est connecté à d'autres neurones formels en formant un réseau des neurones (RdN), où chaque connexion entre deux neurones est associée à un poids ω . Nous présenterons dans la section suivante les architectures des RdN les plus répandues dans la littérature.

3. Architectures neuronales

Les architectures des RdN se divisent en deux types :

- Les RdN de type « feed-forward » ou à propagation directe, sont les RdN où l'information passe d'un neurone à un autre dans le sens direct (figure II.3), à partir des neurones de la couche d'entrée jusqu'aux neurones de la couche de sortie. Autrement dit, dans ce type des RdN, un neurone de la couche i ne peut envoyer l'information qu'à un autre neurone d'une autre couche j , telle que $i < j$.

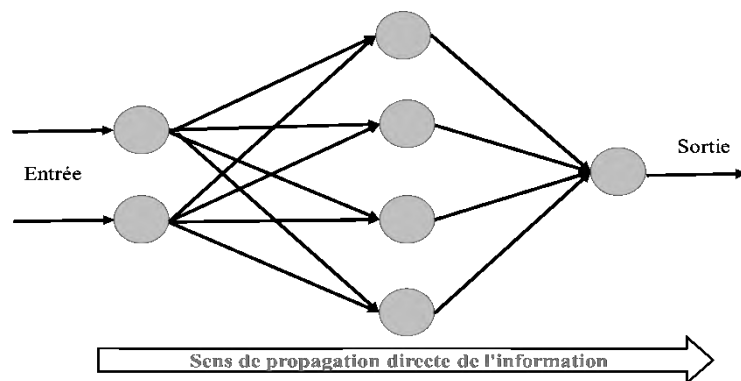


Figure II.3 : Sens de propagation direct dans un RdN de type « feed-forward ».

- Les RdN récurrents (ou de type « back-propagation ») sont des RdN qui ont au moins un cycle de connexion où l'information, comme le montre la figure II.4, peut passer dans le sens indirect ou entre les neurones de la même couche. On peut même trouver des RdN qui ont des connexions récurrentes à l'intérieur de même neurone. La figure II.5 illustre un neurone avec une récurrence interne où la sommation pondérée y de ce neurone est remise à son entrée avec un retard temporel.

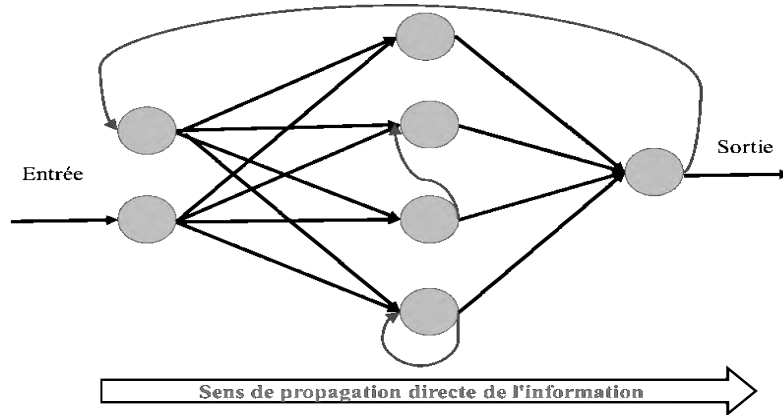


Figure II.4: Quelques types des connexions récurrentes dans les RdN.

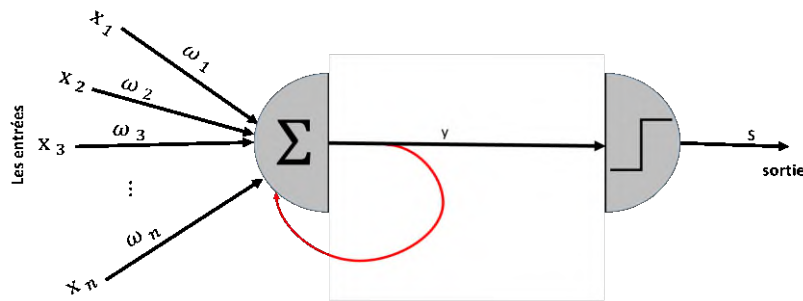


Figure II.5: Exemple de récurrence interne.

3.1 Réseaux de neurones de type « feed-forward »

Dans cette section nous détaillerons les cinq types des RdN les plus répandus dans la littérature, à savoir : Le perceptron simple, le perceptron multicouche, le réseau «Random Vector Functional Link, les RdN à base des fonctions radiales et les cartes de Kohonen.

3.1.1 Perceptron simple

Le perceptron simple ou perceptron monocouche est un RdN développé par Rosenblatt [59]. Ce réseau constitue la base de quasiment tous les RdN existants dans la littérature. Il est constitué d'un seul neurone simple qui agit par la fonction d'activation (II.2) (figure II.1) :

$$A = \begin{cases} 1 & \text{si } y \geq 0 \\ 0 \text{ (ou } -1) & \text{si } y < 0 \end{cases} \quad (\text{II.2})$$

Avec y est la sommation pondérée définie en (II.1).

3.1.2 Perceptron multicouche

Le perceptron multicouche (PMC) [60] est un ensemble des perceptrons simples répartis dans des couches successives (figure II.6), à savoir : une couche d'entrée, un nombre des couches cachées et une couche de sortie. La fonction d'activation O de chaque neurone est la fonction sigmoïde (figure II.2.d) représentée par l'équation (II.3) :

$$o(\vec{x}) = \frac{1}{1 + e^{-y}} \quad (\text{II.3})$$

Avec y est la sommation pondérée définie par l'équation (II.1).

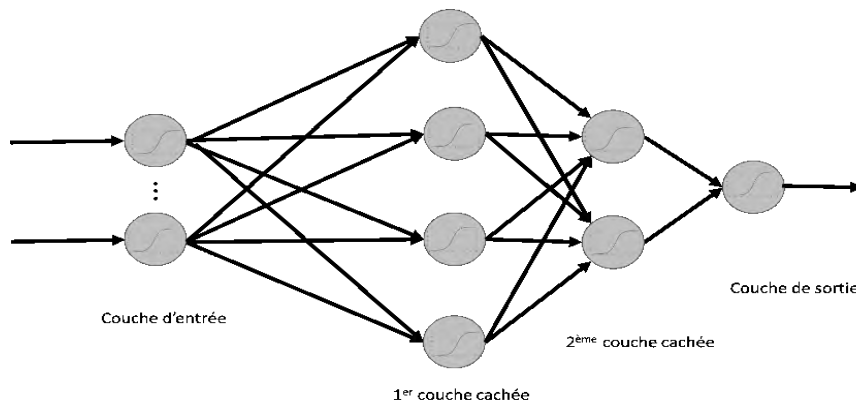


Figure II.6: Architecture d'un PMC avec deux couches cachées.

3.1.3 Réseau "Random Vector Functional Link"

Le réseau RVFLNN (Random Vector Functional Link Neural Network) est un modèle composé d'un ensemble des perceptrons simples, développé par Pao et al [61]. Ce réseau a une architecture similaire à celle du PMC, mais avec des connexions supplémentaires entre la couche d'entrée et la couche de sortie (figure II.7). Les connexions qui relient la couche cachée et la couche de sortie, selon Pao [61], représentent des entrées optimisées et ne font pas l'objet de l'apprentissage. La fonction d'activation utilisée pour les neurones cachés et de sortie est une sigmoïde (figure II.2.d).

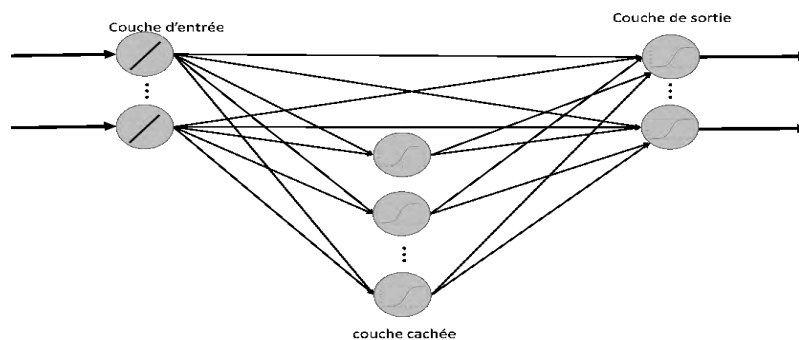


Figure II.7: Architecture d'un RVFLNN.

3.1.4 RdN à base de fonction radiales

Le RdN à base de fonctions radiales (RBFNN : radial basic function neural network) [62, 63] est une architecture, comme le montre la figure II.8, composée de trois couches (d'entrée, cachée et de sortie) où chaque couche est totalement connectée à la couche suivante. La particularité de ce réseau réside dans sa couche cachée (deuxième couche) qui réagit par des fonctions d'activation H_i à base radiale. L'utilisation de ce type de fonction permet au RdN de créer lors de l'apprentissage de régions localisées dans l'espace des entrées ce qui engendre une représentation locale de l'espace (figure II.17). Il y'a différentes formes des fonctions à base radiales, la plus utilisée est la fonction gaussienne (II.4).

$$H_i = \exp\left(\frac{-\|xi - \mu_i\|^2}{2\sigma_i}\right) \quad (II.4)$$

Où $\| \cdot \|$ représente la norme euclidienne, x_i , μ_i et σ_i sont respectivement, le vecteur d'entrée, le centre et le rayon d'influence de neurone i de la couche cachée.

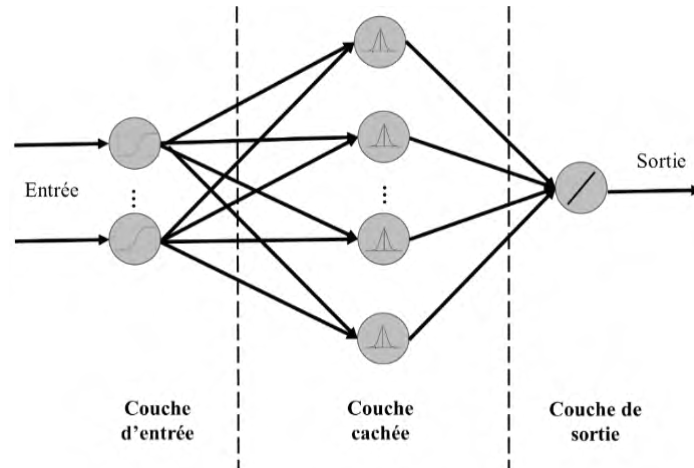


Figure II.8 : Architecture d'un RBFNN.

La couche cachée représente une mémoire statique, elle mémorise les informations (μ_i et σ_i) des différentes classes du problème.

3.1.5 Carte de Kohonen

La carte de Kohonen [64], appelée aussi carte auto-organisatrice, est un ensemble des neurones organisés en deux couches (figure II.9), une couche d'entrée et une autre de sortie. La couche de sortie est appelée aussi couche compétitive où les neurones sont souvent organisés sous forme d'une carte à deux dimensions. Chaque neurone d'entrée est connecté à l'ensemble des neurones de sortie par des poids w_{ij} . La carte de Kohonen est une carte à apprentissage non supervisé, qui, à partir d'une base d'apprentissage constitué seulement des éléments d'entrées, permet de regrouper ces entrées en forme de classes.

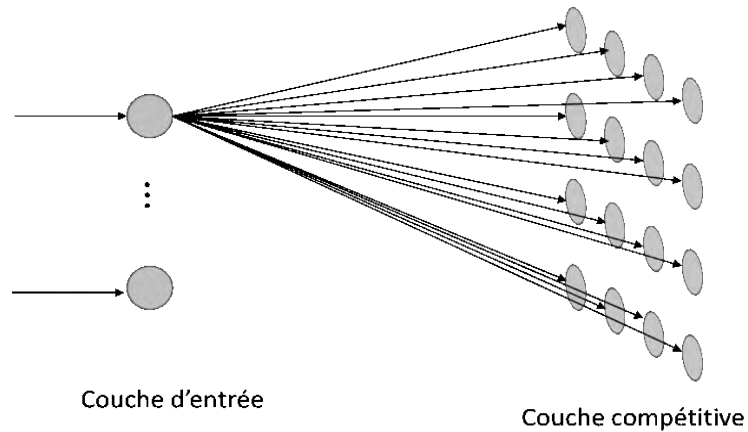


Figure II.9 : Architecture d'une carte de Kohonen.

3.2 Réseaux de neurones récurrents

3.2.1 Réseau de neurones DGNN

Le réseau DGNN (*Dynamic General Neural Network*) est un réseau de neurones qui combine les avantages du PMC et de RBFNN : la capacité de généralisation des PMC et l'efficacité calculatoire des RBFNN. Son architecture, comme le montre la figure II.10, est une sorte de combinaison entre les architectures de ces deux RdN. En effet, il est composé de trois couches : une couche d'entrée, une couche de sortie et une couche cachée. Cette dernière comporte deux types de neurones, quelques-uns avec des fonctions d'activation sigmoïde et d'autres avec des fonctions d'activations gaussiennes.

Ce réseau comporte des connexions récurrentes internes dans les neurones de la couche cachée et de la couche de sortie. Cependant, ce RdN contient plusieurs paramètres que son utilisation nécessite l'intégration d'autres algorithmes de l'intelligence artificielle, comme les algorithmes génétiques, pour déterminer ses nombreux paramètres.

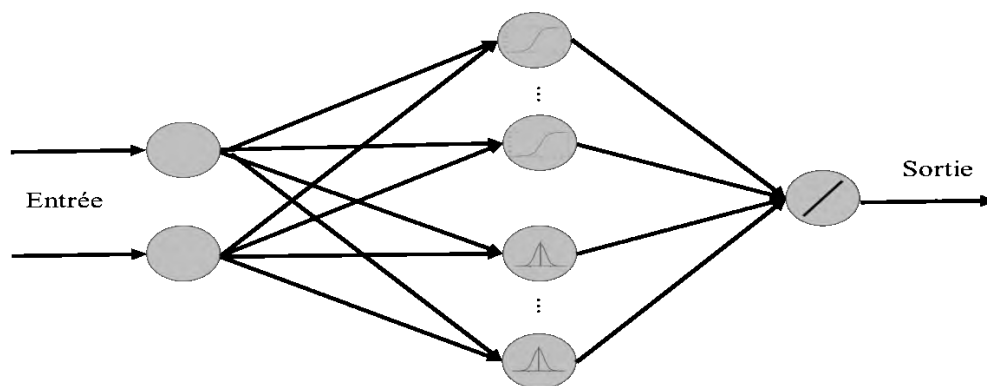


Figure II.10: Architecture d'un DGNN.

3.2.2 Réseau de neurones R2BF

Le réseau de neurones R2BF est un RdN composé de quatre couches : une couche d'entrée, une couche de sortie et deux couches cachées. La première couche cachée a une fonction gaussienne comme fonction d'activation et la deuxième a une fonction sigmoïde. Ce réseau comporte des connexions récurrentes entre les sorties des neurones de la deuxième couche cachée et les entrées des neurones de la première couche cachée.

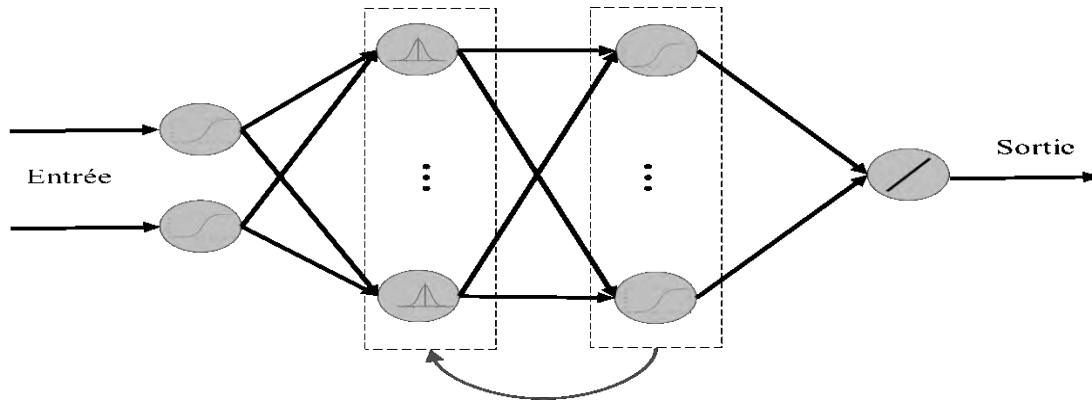


Figure II.11: Architecture du R2BF.

3.2.3 Réseau de neurones récurrent à base de fonction radiale (RRBF)

Le réseau de neurones RRBF (**Recurrent Radial Basis Function**) est un RdN composé de trois couches : une couche d'entrée, une couche de sortie et une couche cachée. Cette dernière comporte des neurones avec des fonctions d'activation gaussiennes. Les neurones de la couche d'entrée de ce RdN contiennent des connexions récurrentes qui relient leurs sorties avec leurs entrées.

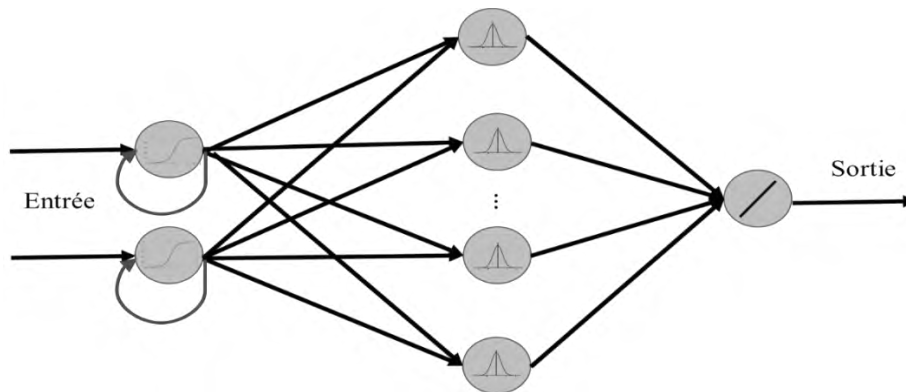


Figure II.12: Architecture d'un RRBF.

4. Algorithmes d'apprentissage

4.1 Types des algorithmes d'apprentissage

Les réseaux de neurones sont des outils de calcul non linéaire qui établissent une relation entre un ensemble d'entrées et un ensemble de sorties. L'apprentissage de cette relation par un RdN est fait en utilisant un ensemble de données

appelé base d'apprentissage. Cet ensemble est souvent divisé en trois parties : une pour l'apprentissage, une autre pour la généralisation et une dernière pour tester les connaissances acquises à un nouvel ensemble de données [65].

La capacité calculatoire des RdN est le résultat d'un processus d'apprentissage similaire à celui des neurones biologiques chez les humains. Lors de cet apprentissage, un ensemble de données, qui constitue la base d'apprentissage, est présenté au RdN. En fonction de ces données, le RdN ajuste les poids des connexions entre ses différents neurones et ses autres paramètres. Ainsi, il adapte sa fonction de transfert pour pouvoir délivrer les sorties convenables. Selon le type de RdN et l'application envisagée, l'apprentissage peut être de différents types [66]:

- **Apprentissage supervisé** consiste à présenter au réseau les vecteurs d'entrées avec leurs sorties associées. À chaque itération le réseau reconfigure sa fonction de transfert en fonction de l'écart entre la sortie de réseau et la sortie désirée. Cette reconfiguration se poursuit tant que les sorties de réseau ne sont pas conformes aux sorties désirées. Il existe plusieurs algorithmes qui permettent de réaliser ce genre d'apprentissage nous citons : l'apprentissage par correction d'erreur, la descente de gradient et l'algorithme de rétropropagation du gradient ;
- **Apprentissage non supervisé** consiste à présenter au réseau les entrées seulement et le laisser reconfigurer ses paramètres sans prendre en compte les sorties. Lors de cette configuration, le réseau essaie d'extraire les vecteurs d'entrée qui ont des caractéristiques communes et les regrouper dans la même classe. Ce type d'apprentissage a comme condition d'arrêt la stabilisation de réseau, et il est utilisé lorsque les classes auxquelles doivent appartenir les vecteurs d'entrées ne sont pas connues a priori ;
- **Apprentissage mixte** consiste à reconfigurer les paramètres de réseau en deux phases une supervisée et une autre non supervisée.

Ces algorithmes peuvent être classés en deux catégories

- **Hors-ligne apprentissage** : Consiste à reconfigurer le RdN après la présentation de toute la base d'apprentissage.
- **En-ligne apprentissage** : Consiste à reconfigurer le RdN après la présentation de chaque élément de la base d'apprentissage.

4.2 Quelques algorithmes d'apprentissage des RdN

4.2.1 Algorithme d'apprentissage par correction d'erreur

Cet algorithme présente la base de la majorité des algorithmes d'apprentissage supervisé. C'est un off-ligne algorithme utilisé dans le cas des RdN monocouche. Il modifie les poids des connexions entre les neurones en fonction de l'écart entre la sortie de réseau et la sortie désirée. Cet algorithme peut être présenté comme suit :

Entrée : un ensemble des couples $(\vec{x}, c)$
 Initialisation aléatoire de tous les poids
Répéter
Pour chaque $(\vec{x}, c)$
 Calculer la sortie s de perceptron
 # Mise à jour des poids
 Pour i de 1 à n
 $w_i \leftarrow w_i + (c - s)x_i$
 FinPour
FinPour
FinRépéter
Sortie : un perceptron définit par ses poids $(w_1, w_2, \dots, w_n)$

Il y'a une version en-ligne de cet algorithme appelé algorithme d'apprentissage de Widrow-Hoff, où un pas d'apprentissage ε est impliqué dans le processus d'apprentissage de tel sorte que la mise à jour des poids est faite après chaque itération selon la formule : $w_i \leftarrow w_i + \varepsilon (c - s)x_i$. L'ajout d'un ε bien choisi assure la convergence rapide vers le minimum de la fonction erreur.

4.2.2 Algorithme d'apprentissage par la descente de gradient

Cet algorithme vise à modifier les poids des connexions en fonction de l'erreur quadratique entre la sortie de réseau et la sortie désirée. C'est un off-ligne algorithme utilisé dans le cas des RdN monocouche. Il modifie les poids après la présentation de toute la base d'apprentissage. Cet algorithme peut être présenté comme suit :

Entrée : un ensemble des couples $(\vec{x}, c), \varepsilon$
 Initialisation aléatoire de tous les poids
Répéter
 Pour tous i
 $\Delta w_i \leftarrow 0$
 FinPour
 Pour chaque $(\vec{x}, c)$
 Calculer la sortie s de perceptron
 Pour tout i :
 $\Delta w_i = \Delta w_i + \varepsilon * (c - s) * x_i * \sigma'(x.w)$
 FinPour
 FinPour
 # Mise à jour des poids
 Pour i de 1 à n
 $w_i \leftarrow w_i + \Delta w_i$
 FinPour
FinRépéter
Sortie : un perceptron définit par ses poids $(w_1, w_2, \dots, w_n)$

où $\sigma'(\vec{w} \cdot \vec{x}) = \sigma'(y)$ est la dérivée de la fonction d'activation. Dans le cas de la fonction sigmoïde :

$$\sigma'(y) = \sigma(y) * (1 - \sigma(y)) \quad (\text{II.5})$$

4.2.3 Algorithme de rétro propagation du gradient

Cet algorithme est l'algorithme le plus utilisé dans le cas des RdN à propagation directe [67]. Il représente une version améliorée de l'algorithme de la descente de gradient que nous pouvons appliquer aux RdN multicouches, il utilise le même principe de minimisation de l'erreur quadratique. L'utilisation de cet algorithme n'est applicable que sur les RdN avec des fonctions d'activation continues et dérivables. Cependant, cet algorithme ne peut assurer la convergence que vers un minimum local. Ceci a suscité l'intérêt de plusieurs travaux de recherche [66, 68] qui ont visé son amélioration et l'élimination de ses inconvénients. Cet algorithme peut être présenté comme suit :

Entrée : un ensemble des couples $(\vec{x}, c)$, un pas d'apprentissage ε
Initialiser aléatoirement les coefficients w_{ij} dans $[-0.5 ; 0.5]$

Répéter

Pour chaque $(\vec{x}, c)$

Calculer la sortie s

calcule des δ_i par rétropropagation

Pour toute cellule de sortie i

$\delta_i = \sigma'(y_i) * (c_i - s_i) = s_i * (1 - s_i) * (c_i - s_i)$

Fin Pour

Pour chaque couche de $q - 1$ à 1

Pour chaque cellule i de la couche courante

$\delta_i = \sigma'(y_i) * [\sum_{k \in Succ(i)} (\delta_k * w_{ki})]$

Fin Pour

Fin Pour

mise à jour des poids

Pour tout poids w_{ij}

$w_{ij} = w_{ij} + \varepsilon * \delta_i * x_{ij}$

FinPour

FinRépéter

Sortie : un PMC définit par ces poids w_{ij}

où $Succ(i)$ est l'ensemble des neurones qui prennent comme entrée la sortie du neurone i et $\sigma'(y)$ est défini en (II.5)

4.2.4 Algorithme d'apprentissage LBG

Cet algorithme dont le nom porte les initiales des auteurs qui l'ont développé [69] (Linde, Buso et Gray 1980) peut être utilisé dans le cas de l'apprentissage non supervisé. C'est un algorithme est de type « winner takes all » (ou bien le gagnant prend tout) où chaque entrée de la base d'apprentissage modifie les poids d'un seul neurone de la couche de sortie, ce neurone est appelé neurone gagnant. Cet algorithme peut être présenté comme suit :

Entrée : un ensemble des vecteurs $\vec{x}$
 Initialiser les w_i par des $\vec{x}_i$ tirés aléatoirement de l'ensemble d'apprentissage
Répéter
 Pour chaque vecteur $\vec{x}$
 chercher le neurone de sortie gagnant
 FinPour
 Pour chaque neurone de sortie
 Modifier le vecteur de référence avec la moyenne arithmétique de tous les $\vec{x}_i$ pour lesquels ce neurone est considéré gagnant.
 FinPour
 FinPour
FinRépéter
Sortie : un RdN défini par ses paramètres

5. Domaines d'application des RdN

De nos jours, et vu l'intersection des différents domaines scientifiques (mathématiques, électronique, neurobiologie...), nous pouvons constater que les RdN ont été utilisés dans plusieurs domaines comme :

- L'industrie : la commande des systèmes, le diagnostic des pannes, le Contrôle qualité...
- Finance : la prédiction des mouvements de bourse, la sélection de portefeuilles d'actions...
- Télécommunication : La détection des pannes de réseaux, élimination des bruits...

L'intégration d'un tel outil de l'intelligence artificielle dans ces domaines peut être sous différentes formes :

5.1 Régression multicritère

La régression est l'application la plus classique des RdN. Il permet la mise en relation d'un ensemble de variables d'entrée avec une variable de sortie et ainsi il permet, après l'apprentissage, d'approximer à travers une sortie continue la valeur d'une variable en fonction des autres variables regroupées sous forme d'un vecteur d'entrée. Cette technique peut être utilisée, par exemple, pour déterminer la valeur du signal d'un capteur en fonction des signaux des autres capteurs ou bien la prédiction de sa valeur future en fonction de sa valeur actuelle.

La figure II.13 illustre un exemple de régression non linéaire. Cette figure donne l'ensemble des points (en rouge), qui constitue la base d'apprentissage de RdN, et la fonction $f(x)$ (en bleu) approximée par le RdN après l'apprentissage.

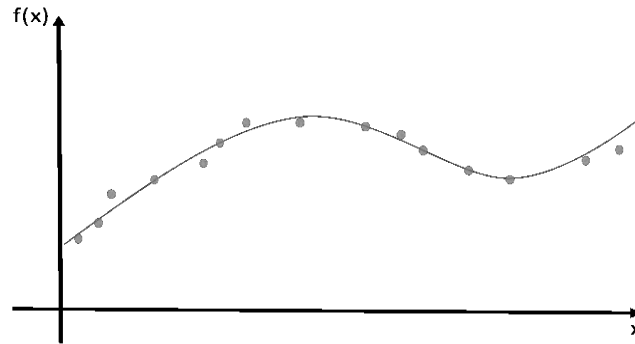


Figure II.13: Exemple de régression.

5.2 Reconnaissance des formes

La reconnaissance des formes (ou la classification) permet à travers des sorties booléennes d'identifier la classe à laquelle appartient le vecteur d'entrée. Les formes à reconnaître peuvent être de différente forme traduite en vecteurs d'entrées numériques. Il peut s'agir d'un ensemble des écritures manuscrites, de contenu visuel, sonore ou bien d'autres.

La figure II.14 donne la résolution d'un problème de classification à deux dimensions. Ceci montre la différence entre les deux types de classification existants dans la littérature :

- La classification linéaire (figure II.14.a) : le RdN crée des segments des droits pour séparer les différentes classes.
- La classification non linéaire (figure II.14.b) : le RdN crée une courbe non linéaire pour séparer les différentes classes.

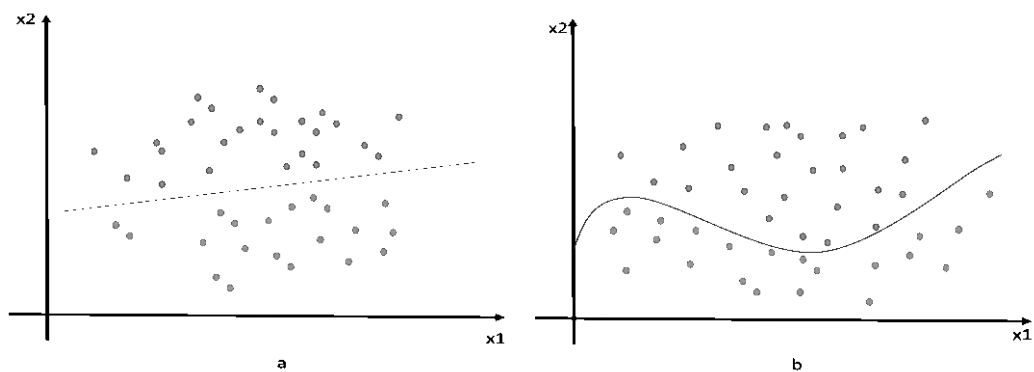


Figure II.14: Types de classification : a) linéaire, b) non linéaire.

5.3 Clustering

Le clustering est la méthode de partitionner un ensemble des points de données en sous-ensembles. Ceci peut être vu comme une sorte de reconnaissance des formes non supervisée dans lequel un ensemble des points est présenté au RdN qui va essayer d'extraire les données qui ont des caractéristiques similaires et de les regrouper sous forme des classes sans connaissance préalable de ces classes.

L'architecture des RdN les plus utilisés dans le clustering est la carte de kohonen (appelée aussi les cartes auto adaptatives). Ceci est un RdP composé de deux couches une couche d'entrée et une couche de sortie. Chaque neurone de cette dernière est relié à un vecteur de référence auquel les entrées sont comparées pour déterminer leurs classes d'appartenance.

6. Diagnostic/ pronostic des fautes dans les SED en utilisant les RdN

Les RdN, selon leurs types, ont plusieurs applications en industrie comme le off-ligne diagnostic [54, 70], le enligne diagnostic [71], le pronostic [72, 73], le contrôle [74, 75] et la surveillance des systèmes industriels.

L'application des RdN dans le diagnostic/pronostic vise, à partir d'un vecteur d'entrée contenant les paramètres de système à diagnostiquer et les signaux provenant de ses capteurs, donner une sortie qui indique l'état de système. Cette intégration des RdN dans les processus de diagnostic/pronostic peut être vue comme une sorte de reconnaissance de forme, où on essaie d'associer à chaque faute la cause la plus probable ou comme une sorte de régression qui permet de calculer des paramètres en relation avec le diagnostic/pronostic des fautes.

Les évènements générés par le SED sont des séquences que nous pouvons les considérer comme des données temporelles. Dans la suite de ce travail, nous exploitons les réseaux de neurones pour analyser ces données afin de donner l'état de fonctionnement du SED (diagnostic) ou bien prédire son état de fonctionnement futur (pronostic). La littérature propose plusieurs techniques pour l'analyse des données temporelles par RdN. Les techniques les plus populaires sont :

- Les RdN récurrents : Les connexions récurrentes de ces réseaux contiennent des retards temporels et permettent au réseau d'avoir un comportement dynamique qui peut mettre en relation le dernier évènement généré et les évènements générés précédemment. Cependant, cette technique donne naissance à des architectures neuronales complexes [76]. Cette technique a été utilisée dans plusieurs travaux nous citons l'architecture de Jordan [77], l'architecture d'Elman [78], l'architecture R2BF [79] et l'architecture RRBF [80]. La figure II.15 présente l'architecture de Jordan utilisé pour le diagnostic et le pronostic d'un SED, le vecteur d'entrée est construit par le dernier évènement généré par le SED et la sortie retardée du réseau.

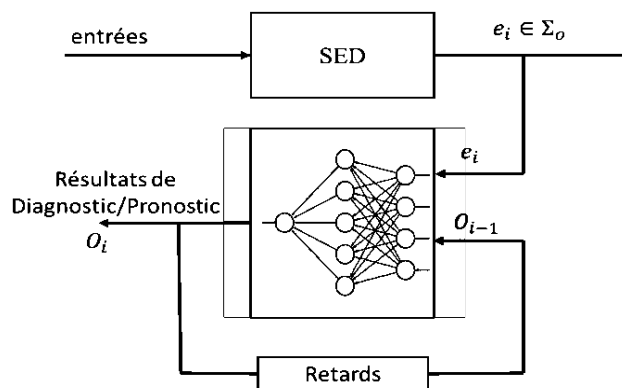


Figure II.15: Diagnostic et pronostic d'un SED en utilisant un réseau de neurones récurrent (Jordan).

- La représentation spatiale de temps (fenêtre temporelle) : consiste à transformer les données temporelles à des données spatiales. Au lieu de présenter au réseau chaque évènement dès son occurrence, nous attendons jusqu'à l'occurrence de k évènements. Le vecteur d'entrée est composé par un ensemble des retards temporels entre les évènements successifs. Ce vecteur est traité par la suite à l'aide du RdN à propagation directe. Dans la littérature, il existe plusieurs architectures basées sur la notion des fenêtres temporelle, nous citons le NETtalk (Sejnowski et al., 1986[81]), le TDNN (Time Delay Neural Networks) (Waibel et al., 1989[82]) et le TDRBF (Time Delay Radial Basis Function) (Berthold, 1994 –a–[83]).

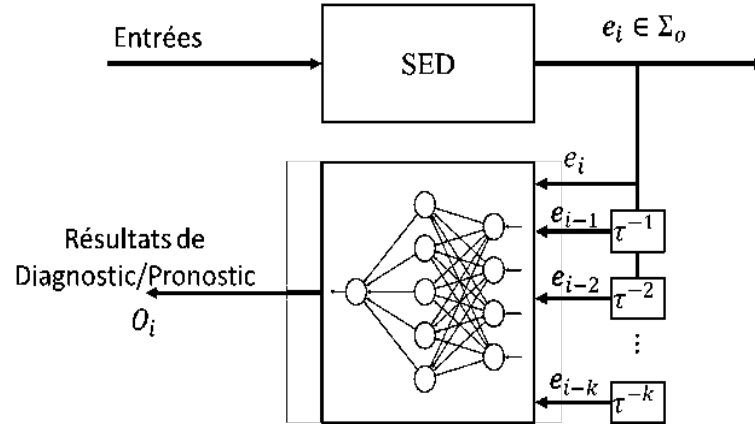


Figure II.16: Diagnostic et pronostic des SED en utilisant des fenêtres temporelles.

Les RdN récurrents sont un peu plus performants que les RdN à propagation directe dans le traitement des données temporelles [84], mais la complexité de leurs architectures et de leurs apprentissages peuvent influencer le temps de réponse de diagnostiqueur/pronostiqueur basé sur ces RdN. Ces RdN nécessitent des calculateurs très performants et ils prennent plus de temps lors de l'apprentissage que les RdN non récurrentes. Dans le reste de ce travail, nous utilisons la notion des fenêtres temporelles pour la description de fonctionnement des SED et nous utilisons un RdN à propagation directe pour le traitement de ces fenêtres temporelles. Nous consacrons la suite de ce chapitre à une comparaison entre les différentes architectures des RdN non récurrents. La notion des fenêtres temporelles sera traitée dans le chapitre III.

7. Étude Comparative entre Différents RdN à Propagation Directe

7.1 Étude Comparative

Les réseaux de neurones, avec leurs différentes architectures, peuvent offrir des solutions très performantes pour le diagnostic/pronostic des fautes dans les SED industriels [54, 70-72]. L'application de chacune de ces architectures présente des points forts et des points faibles qui peuvent influencer les opérations de diagnostic/pronostic. Plusieurs travaux ont été élaborés par des auteurs qui ont pour objectif de comparer les différents types des RdN [54, 85, 86]. En se basant sur ces travaux, nous résumons, dans le tableau II.1, quelques

informations sur les RdN à propagation directe qui sont déjà citées dans le paragraphe 3.1 ainsi que l'ensemble de leurs avantages et leurs inconvénients.

7.2 Comparaison entre le PMC et le RBFNN

Les RdN de type PMC et RBFNN sont les RdN les plus appropriés pour résoudre ce genre des problèmes (classification et régression). Cependant, les capacités du RBFNN surpassent ceux de PMC. Ceci réside dans la représentation locale de l'espace que le RBFNN adopte (figure II.17). Ainsi, le RBFNN présente plus de précision et il peut détecter s'il y a une donnée complètement nouvelle. Ceci peut être utilisé pour améliorer l'architecture de RdN en reconstruisant des nouvelles bases d'apprentissages améliorées par ces nouvelles données.

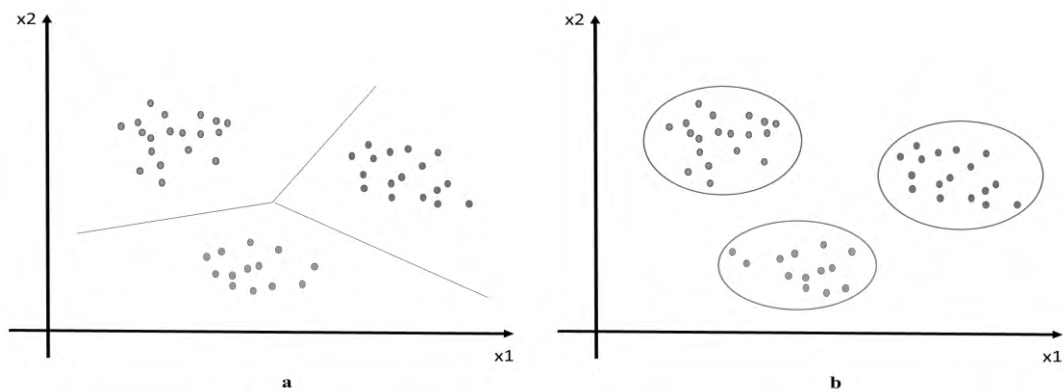


Figure II.17: représentation de l'espace, a) global, b) locale

RdN	Description	Apprentissage	Avantages	Inconvénients
Perceptron simple	Un seul neurone	Supervisé	- Architecture simple	N'accepte pas les données bruitées et la classification non linéaire
PMC	Plusieurs neurones organisés en couches Sorties des neurones : sigmoïde	Supervisé	- Accepte les données bruitées et la classification non linéaire - Représentation globale de l'espace - Architecture simple	- Classe les éléments qui n'appartiennent à aucune classe à la classe la plus proche - Le nombre des couches cachées et des neurones dans les couches cachées est indéfini.
RVFLNN	Réseau en trois couches : Couche d'entrée reliée à la couche de sortie	Supervisé (seulement les poids reliant la couche d'entrée avec celle de sortie)	- Accepte les données bruitées et la classification non linéaire - Représentation globale de l'espace - Des entrées optimisées	-Architecture plus complexe que le PMC Moins de précision que le PMC
RBFNN	Plusieurs neurones organisés en 3 couches Couche cachée : gaussienne	Mixte : Non supervisé (μ, σ) + Supervisé	- Accepte les données bruitées et la classification non linéaire - Capable de dire « je ne sais pas » - Représentation locale de l'espace - Une grande précision. - Apprentissage rapide. - Donne directement les probabilités d'appartenance	- Apprentissage complexe - Nécessite une grande capacité de calcul
Carte de Kohonen	Couche de sortie sous forme d'une carte (2D)	Non supervisé	- Classifier les données sans connaissance préalable de procédé. - regroupement automatique des données	- Parfois les résultats ne correspondent pas à celle souhaitée

Tableau II.1: Avantages et inconvénients de quelques architectures des réseaux de neurones

7.3 Synthèse

À la base de la comparaison présentée dans le tableau II.1 et celle présentée dans la section 7.2 nous constatons que le choix de l'architecture de RdN la plus convenable à une application donnée dépend principalement de quatre critères qui sont:

- La fonction à remplir : régression ou classification ou clustering
- La performance des ressources sur lesquelles on va implémenter les réseaux de neurones (ordinateur, carte électronique...)
- La précision voulue : Ceci s'exprime par le taux d'erreur commis lors de la phase de test.
- Le degré de maîtrise de système traité : l'existence d'une base de données contenant les variables d'entrées et toutes les classes correspondantes.

La figure II.18 résume la procédure à suivre pour choisir une architecture des RdN selon les quatre critères déjà cités ci-dessus.

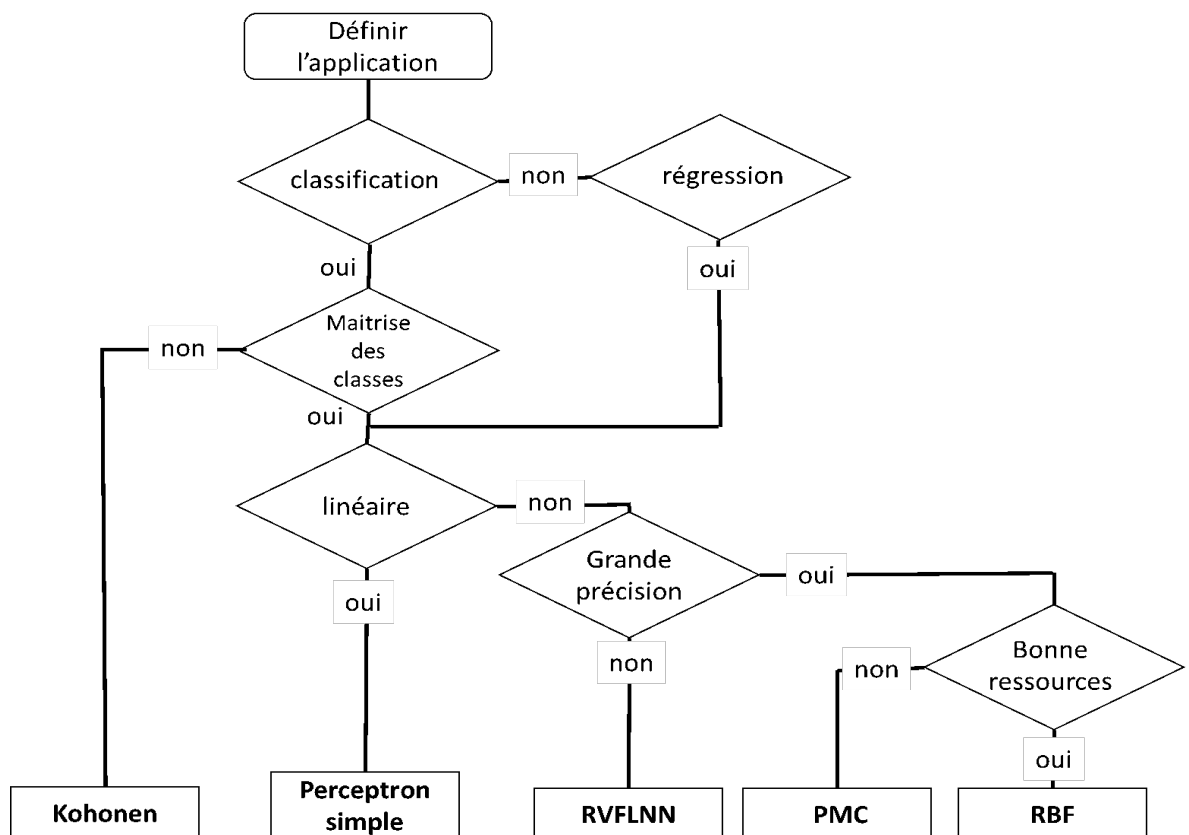


Figure II.18: Procédure de sélection d'une architecture des réseaux de neurones.

Cette étude a montré que chaque architecture des RdN peut être utilisée pour résoudre des problèmes bien spécifiques : la carte de Kohonen est le RdN le plus approprié lorsqu'on n'a pas suffisamment d'information sur le système traité. Cette carte peut regrouper les données sous forme de classes et donc la préparation de ces données pour l'apprentissage des autres types des réseaux de neurones. Les réseaux RBFNN donnent la meilleure précision par rapport à la précision donnée par les autres architectures,

donc ce RdN est réservé aux problèmes non linéaires qui demandent beaucoup de précision et où les bases d'apprentissages peuvent contenir des données bruitées. Cependant son implémentation nécessite une grande capacité calculatoire.

8. CONCLUSION

Dans ce chapitre, nous avons présenté un aperçu général sur les réseaux de neurones, leurs domaines d'application et comment ces outils peuvent être impliqués dans le processus de diagnostic/pronostic des fautes dans un SED. À la fin de ce chapitre une comparaison entre les cinq architectures des RdN à propagation directe les plus répandues dans la littérature a été présentée. À l'issue de cette étude comparative, nous pouvons conclure que les RdN de type RBFNN sont les RdN qui ont les capacités les plus adéquates avec les problèmes de diagnostic/pronostic des SED. Ainsi dans le reste de ce travail, ce type des RdN va être utilisé pour l'analyse des événements générés par les SED qui seront regroupés sous forme des fenêtres temporelles. Dans le chapitre suivant, nous essaierons de développer une approche de diagnostic/pronostic fondée sur la notion des fenêtres temporelles.

Chapitre III:
Diagnostic des SED basé sur un modèle statistique et les lois de
fiabilité

1. Introduction

Ce chapitre se focalise sur le diagnostic et le pronostic des fautes dans les systèmes à événements discrets décrits par des séquences des événements. Nous nous intéressons aux systèmes à événements discrets larges, où la modélisation par les outils usuels des systèmes à événements discrets (automate fini, réseau de Pétri...) est une tâche complexe. Nous décrivons le comportement du SED en utilisant un modèle statistique contenant des fenêtres temporelles construites à partir des séquences des événements générés par le SED. En se basant sur ce modèle, une approche de diagnostic/pronostic des fautes dans les SED a été présentée. Ensuite, l'efficacité de cet approche a été améliorée en intégrant la notion de la fiabilité des composants de SED et en développant un algorithme, basé sur des procédures récursives et des approches statistiques. La vérification des contraintes temporelle dans les SED a été aussi abordée. Différents exemples seront présentés pour illuminer les approches présentées dans ce chapitre.

2. Fautes dans les SED

Les fautes sont toute sorte de déviation d'au moins une caractéristique, une propriété ou un paramètre du système de leurs états de fonctionnement normal [87]. Ces fautes sont souvent classifiées en trois catégories [56], à savoir :

- Les fautes processus sont ceux qui impliquent le processus et son environnement (la chute d'un colis ou un paramètre erroné). Ces fautes englobent aussi l'endommagement des équipements qui constituent le châssis de la machine.
- Les fautes capteurs sont les fautes qui impliquent un capteur, une connexion de capteur au contrôleur ou la carte d'entrée du contrôleur. Ces fautes englobent aussi celles provenant de la destruction des connexions qui relient les capteurs avec le contrôleur.
- Les fautes actionneur regroupent toutes les fautes en relation avec la défaillance d'un composant matériel : actionneur ou préactionneur. Ces fautes englobent aussi la destruction des équipements qui assurent des mouvements spécifiques ou qui assurent la transmission de mouvement (roulement, les engrenages, les tapis roulants...).

Dans la suite de ce travail, nous considérons un SED avec m classes des fautes. Ces classes des fautes sont construites en respectant deux critères :

- Le type de la faute : faute processus, faute capteur ou faute actionneur ;
- L'impact de la faute : les dégâts que peut engendrer l'occurrence de la faute sur l'être humain et les équipements. La gravité des dégâts est de différents niveaux qui peuvent mettre en menace la sécurité du personnel et des équipements.

Ainsi, les fautes ayant même type et même impact seront regroupées dans la même classe, à l'égard de laquelle des actions compatibles peuvent être programmées. Les classes des fautes seront notées F_α ($\alpha = 1, 2, 3, \dots, m$).

3. Modèle statistique pour les SED

Les automates finis et les réseaux de Pétri font de nos jours, les outils de modélisation les plus utilisés dans le domaine des SED. Cependant, ces modèles sont insuffisants face à des problèmes d'explosion combinatoire et du manque de maîtrise de fonctionnement qu'on peut trouver dans les SED larges et complexes (comme les systèmes aéronautiques, les systèmes militaires, les centrales nucléaires...). Face à ces problèmes, et à l'opposition des outils de modélisation usuelle, les modèles statistiques représentent l'alternatif le plus convenable à ce type des systèmes, car son implémentation dans ces systèmes est simple et ne demande pas une grande maîtrise des outils de modélisation.

Un modèle statistique d'un SED est une base de données qui contient des descriptions de tous les comportements possibles. Ce modèle est construit à partir de l'historique du système et il est souvent considéré comme la description idéale du fonctionnement réel du système. La construction d'un tel modèle pour un SED nécessite des données historiques contenant les séquences des événements enregistrés.

Les SED fonctionnent généralement selon des cycles répétitifs. Chacun des cycles, le SED effectue plusieurs tâches pour atteindre un objectif bien précis (chaque tâche est généralement délimitée par deux événements). Chaque cycle peut être décrit par un mot temporisé fini contenant tous les événements (observable et inobservable) qui se sont produits où chaque événement est couplé avec sa date d'occurrence. Nous rappelons par la suite la notion des mots temporisés finis.

Définition 6 : Un mot temporisé fini, sur un ensemble des événements Σ , est une séquence d'événements où chaque événement est couplé ou bien indexé avec sa date d'occurrence. Donc $\sigma = e(1)^{d_1}e(2)^{d_2} \dots e(h)^{d_h}$ est un mot temporisé fini où $e(i) \in \Sigma$ et $d_i \in \mathbb{R}^+$ ($0 \leq d_1 \leq \dots \leq d_h$) représente la date d'occurrence de l'événement $e(i)$. Par exemple $a^{1.0}b^{2.7}c^{4.1}$ est un mot temporisé où la valeur 4.1 représente la date d'occurrence de l'événement c . L'ensemble des mots finis temporisés sur l'ensemble des événements Σ est noté par $TW^*(\Sigma)$ [88].

Remarque 1 : quelques auteurs [89, 90] considèrent différentes notations pour les mots temporisés tels que : $\sigma_j = e(1)^{t_1}e(2)^{t_2} \dots e(h)^{t_h}$ où t_i est le délai entre les événements $e(i-1)$ et $e(i)$. Les deux notations sont équivalentes, il suffit de considérer le changement de variable $t_i = d_i - d_{i-1}$.

Remarque 2 : Si un SED ne fonctionne pas selon des cycles, nous prenons un seul mot temporisé σ contenant l'ensemble des événements générés pendant une période de temps significative, où l'ensemble de tous les modes de fonctionnement ont été au moins survenu une seule fois.

Exemple 3 : Considérons le chariot de la figure III.1 qui peut être considéré comme un SED. Dans chaque cycle, le chariot fait un aller-retour en passant par les trois positions déterminées par les capteurs de position pA, pB et pC. Lors de son déplacement, entre deux positions, le chariot subit des actions manuelles qui représentent des événements non observables. L'ensemble des événements générés par le

SED est décrit dans le tableau 1. Le cycle de fonctionnement peut être décrit par le mot temporisé suivant :

$$\sigma = e_{p0}^{t1} e_{pA}^{t2} e_{AB}^{t3} e_{pB}^{t4} e_{BC}^{t5} e_{pC}^{t6} e_{pB}^{t7} e_{pA}^{t8} e_{p0}^{t9}$$

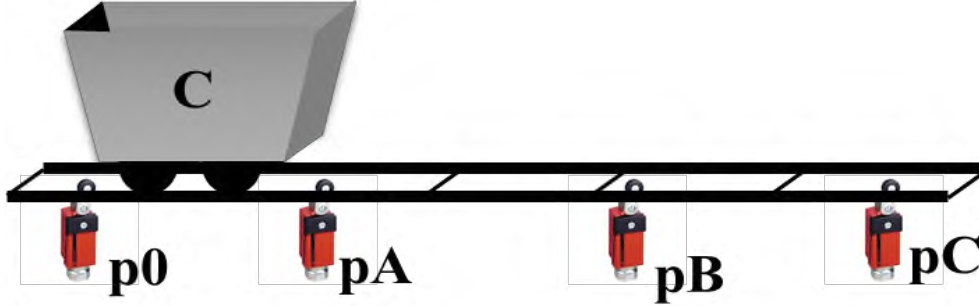


Figure III.1: Le chariot.

	Évènement	Description
Observables	e_{p0}	Chariot à la position initiale
	e_{pA}	Chariot à la position pA
	e_{pB}	Chariot à la position pB
	e_{pC}	Chariot à la position pC
Non observables	e_{AB}	Fin de l'action manuelle entre pA et pB
	e_{BC}	Fin de l'action manuelle entre pB et pC

Tableau III.1 : Différents évènements générés par le chariot.

Afin de pouvoir traiter les mots temporisés générés par le SED, ces mots doivent contenir le même nombre des évènements observables. Cependant, lors de chaque cycle le SED peut effectuer des tâches différentes, par conséquent on aura des mots temporisés différents avec différents évènements et différentes dimensions. Face à ceci, nous proposons de construire, à partir de ces mots enregistrés, des fenêtres temporelles qui auront la même dimension.

Avant de définir comment les fenêtres temporelles peuvent être extraites d'un mot temporisé w décrivant le fonctionnement lors d'un cycle donné, nous adoptons les projections suivantes :

- $P_o(w): TW^*(\Sigma) \rightarrow \Sigma_o$: la projection qui supprime le temps et les évènements inobservables.
- $P_u(w): TW^*(\Sigma) \rightarrow \Sigma_u$: la projection qui supprime le temps et les évènements observables.
- $T_\alpha(w): TW^*(\Sigma) \rightarrow \mathbb{R}$: la projection qui donne les dates d'occurrence d'un évènement e_α dans w .

Définition 7 : soit $\sigma_j = e(1)^{d_1} e(2)^{d_2} \dots e(s)^{d_s} \in TW^*(\Sigma)$ est le mot temporisé enregistré pendant le $j^{ème}$ cycle avec $s = |\sigma_j|$ et $h_j = |P_o(\sigma_j)|$ ($|\cdot|$ représente le nombre des évènements). $w_{j,i}^k = w_i^k(\sigma_j) = e(i)^{t_i} e(i+1)^{t_{i+1}} \dots e(i+p)^{t_{i+p}} \in TW^*(\Sigma)$ représente la $i^{ème}$ fenêtre temporelle de σ_j , autrement dit

c'est la $i^{\text{ème}}$ séquence d'évènement qui contient k -évènements observables dans σ_j , tel que $p = |w_{j,i}^k|$ et $k = |P_o(w_i^k(\sigma_j))|$ ($k < p$) [91].

Remarque 3 : Dans la suite de ce travail, nous parlerons du diagnostic d'une fenêtre temporelle qui signifie le diagnostic de l'état de fonctionnement du SED lorsque cette fenêtre temporelle est générée.

Soit $\sigma_j \in TW^*(\Sigma)$ le mot temporisé enregistré pendant la $j^{\text{ième}}$ cycle de fonctionnement. Il est décomposé en fenêtres temporelles $w_{i,j}^k$ ($|P_o(w_{i,j}^k)| = k$) (définition 7). Pour réduire le nombre des paramètres, nous notons $w_{i,j}^k$ par w_l^k . Ainsi, nous remplaçons le couple des paramètres (i, j) par un seul paramètre l , tel que : $l = \sum_{m=1}^j h_m - (j-1)(k-1) + 1$ où $h_m = |P_o(\sigma_m)|$. Nous notons aussi, par W^k l'ensemble de toutes les fenêtres temporelles w_l^k , cet ensemble représente un modèle statistique de SED. L'équation (III.1) définit l'ensemble de toutes les fenêtres temporelles qui contiennent une faute $e_f \in F_\alpha$ par :

$$S_\alpha^k = \{w_l^k \in W^k | P_o(w_l^k) \cap F_\alpha \neq \emptyset\} \quad (\text{III.1})$$

Exemple 4 : soit $\sigma_1 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5} e_3^{9.0}$ et $\sigma_2 = e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}$ les mots temporisés générés pendant les deux premiers cycles du fonctionnement d'un SED défini par les ensembles des évènements suivants : $\Sigma_o = \{e_1, e_2, e_3\}$, $\Sigma_f = \{e_{f1}, e_{f2}\}$, $\Sigma_{reg} = \{e_{r1}, e_{r2}\}$. La figure III.2 montre l'extraction des fenêtres temporelles, pour $k = 3$, à partir du mot temporisé σ_1 , et ainsi pour σ_1 et σ_2 nous obtenons :

$$\begin{aligned} W^3 = \{ & w_1^3 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, w_2^3 = e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}, w_3^3 = e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}, \\ & w_4^3 = e_2^{8.0} e_1^{8.5} e_3^{9.0}, w_5^3 = e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0}, w_6^3 = e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0}, \\ & w_7^3 = e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5}, w_8^3 = e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0} \} \end{aligned}$$

$$S_1^3 = \{w_2^3, w_3^3\} \text{ et } S_2^3 = \{w_1^3\}.$$

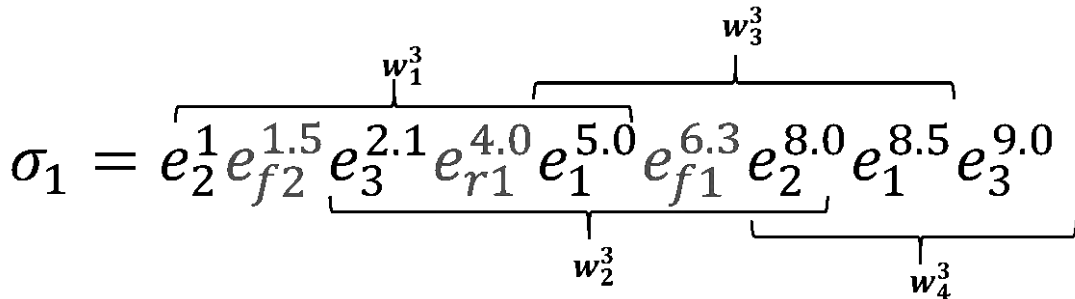


Figure III.2: Construction des fenêtres temporelles à partir de σ_1 .

La construction du modèle statistique d'un SED nécessite une phase d'étude. Les événements générés (observable et inobservable) sont enregistrés avec leurs dates d'occurrences sous forme des mots temporisés σ_j .

4. Diagnostic et pronostic des fautes

Considérons un SED qui fonctionne selon $m + 1$ états de fonctionnement, dont m états sont des états de fonctionnement anormal et un état de fonctionnement normal et décrit par un modèle statistique, contenant des enregistrements des événements générés pendant une durée significative. Ce modèle est construit comme ce qui est montré dans la section III.2.

Lors de fonctionnement réel de SED, nous ne pouvons enregistrer que les événements observables, les mots temporisés constitués de ces événements seront utilisés pour le diagnostic/pronostic des fautes. Dans le reste de ce travail, nous désignons par ω , la fenêtre temporelle contenant les k derniers événements observables générés par le SED et par w_i^k la $i^{\text{ème}}$ fenêtre temporelle du modèle statistique associé au SED. Nous considérons aussi la projection inverse $P_o^{-1}(\omega)$ qui donne, à partir d'une séquence des événements observables l'ensemble des fenêtres temporelles existantes dans le modèle statistique qui ont la même projection P_o que ω . Plus formellement :

$$P_o^{-1}(\omega) = \{v \in W^k : P_o(v) = \omega\} \quad (\text{III.2})$$

4.1 Diagnostic

Le diagnostic d'une fenêtre temporelle ω par rapport à une classe des fautes F_α consiste à déterminer la probabilité d'occurrence d'une faute $e_f \in F_\alpha$ sachant que ω a été généré par le SED.

Définition 8 : une fenêtre temporelle ω est diagnosticable par rapport à une classe des fautes F_α , si nous pouvons détecter sans ambiguïté l'occurrence ou non de toutes les fautes $e_f \in F_\alpha$ lorsque ω est généré par le SED.

Lorsqu'une fenêtre temporelle est diagnosticable par rapport à une classe des fautes, ces fautes sont détectables avant l'occurrence de au plus k événements observables postérieurs à ces fautes, donc ces fautes sont k - diagnosticable [1].

À partir du modèle statistique, nous pouvons définir le paramètre de diagnosticabilité $D_\alpha^k(\omega)$ (III.3). Ce paramètre est calculé en fonction des fenêtres temporelles existantes dans le modèle statistique qui ont la même séquence des événements observables. Il caractérise le diagnostic d'une fenêtre temporelle par rapport à une classe des fautes F_α . Autrement dit, c'est la probabilité d'occurrence d'un événement $e_f \in F_\alpha$ dans ω .

$$D_\alpha(\omega) = \frac{\text{nombre des } w_i \text{ similaire à } \omega \text{ contenant } e_f \in F_\alpha}{\text{nombre des } w_i \text{ similaire à } \omega}$$

Formellement :

$$D_{\alpha}(\omega) = \frac{\text{card}(P_o^{-1}(\omega) \cap S_{\alpha}^k)}{\text{card } P_o^{-1}(\omega)} \quad (\text{III.3})$$

Où $P_o^{-1}(\omega) \cap S_{\alpha}^k$ est l'ensemble de toutes les fenêtres temporelles du modèle statistique qui contiennent la même séquence des événements observables que ω (en genre, en nombre et en ordre) ainsi qu'une faute appartenant à la classe F_{α} . $P_o^{-1}(\omega)$ et S_{α}^k sont définis respectivement dans les équations (III.2) et (III.1).

Proposition 1 : Soit une fenêtre temporelle ω de dimension k , ω est diagnosticable par rapport à F_{α} si et seulement si : $D_{\alpha}^k(\omega) = 1$ ou $D_{\alpha}^k(\omega) = 0$.

Preuve 1 : si ω est diagnosticable par rapport à une classe de fautes F_{α} , une des deux propositions suivantes est vrai.

- $\forall v \in W^k$, telle que $P_o(v) = \omega$ et $F_{\alpha} \cap P_u(v) \neq \emptyset$, donc $\text{card}(P_o^{-1}(\omega) \cap S_{\alpha}^k) = \text{card } P_o^{-1}(\omega)$ d'où $D_{\alpha}(\omega) = 1$. Donc ω est diagnosticable et contient une faute appartenant à la classe F_{α} .
- $\forall v \in W^k$, telle que $P_o(v) = \omega$ et $F_{\alpha} \cap P_u(v) = \emptyset$, donc $\text{card}(P_o^{-1}(\omega) \cap S_{\alpha}^k) = 0$ d'où $D_{\alpha}(\omega) = 0$. Donc ω est diagnosticable et ne contient aucune faute appartenant à la classe F_{α} .

Contrairement, si ω n'est pas diagnosticable $\exists \vartheta \in W^k$, $\exists v \in W^k$ tel que $P_o(\vartheta) = P_o(v) = \omega$, $P_u(\vartheta) \cap F_{\alpha} = \emptyset$ et $P_u(v) \cap F_{\alpha} \neq \emptyset$ alors $0 < \text{card}(P_o^{-1}(\omega) \cap S_{\alpha}^k) < \text{card } P_o^{-1}(\omega)$ donc $0 < D_{\alpha}(\omega) < 1$.

Exemple 5 : considérons une autre fois σ_1 et σ_2 de l'exemple 4 et soit $\omega = e_2^1 e_1^{2.1} e_3^{4.0}$, la dernière fenêtre temporelle générée par le SED ($k = 3$), on a :

$$P_o^{-1}(\omega) = \{w_4^3 = e_2^{8.0} e_1^{8.5} e_3^{9.0}, \quad w_5^3 = e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0}, \quad w_8^3 = e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}\}$$

- $P_o^{-1}(\omega) \cap S_1^3 = \emptyset$, donc $D_1(\omega) = 0$ et
- $P_o^{-1}(\omega) \cap S_2^3 = \emptyset$, donc $D_2(\omega) = 0$
- $P_o^{-1}(\omega) \cap S_2^3 = P_o^{-1}(\omega)$, donc $D_0(\omega) = 1$

Ainsi, la fenêtre temporelle ω est diagnosticable et elle représente un fonctionnement normal de SED.

Exemple 6 : Maintenant considérerons, en plus de σ_1 et σ_2 de l'exemple 4, un autre mot temporel enregistré lors du troisième cycle de fonctionnement de ce SED $\sigma_3 = e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5} e_3^{9.0}$ et considérons que la dernière séquence des événements observables générée par le SED est $\omega = e_2^1 e_3^{2.1} e_1^{5.0}$, on a (pour $k = 3$) :

$$\begin{aligned}
 W^3 = \{ & w_1^3 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, w_2^3 = e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}, w_3^3 = e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}, \\
 & w_4^3 = e_2^{8.0} e_1^{8.5} e_3^{9.0}, w_5^3 = e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0}, w_6^3 = e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0}, w_7^3 \\
 & = e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5}, w_8^3 = e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}, w_9^3 = e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, \\
 & w_{10}^3 = e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}, w_{11}^3 = e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}, w_{12}^3 = e_2^{8.0} e_1^{8.5} e_3^{9.0} \}
 \end{aligned}$$

$$S_1^3 = \{w_2^3, w_3^3, w_{10}^3, w_{11}^3\}, S_2^3 = \{w_1^3\}$$

et

$$P_o^{-1}(\omega) = \{w_1^3 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, \quad w_9^3 = e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}\}$$

Donc, on a :

- $P_o^{-1}(\omega) \cap S_2^3 = \{w_1^3\}$, donc $D_0(\omega) = \frac{1}{2}$;
- $P_o^{-1}(\omega) \cap S_1^3 = \emptyset$, donc $D_1(\omega) = 0$;
- $P_o^{-1}(\omega) \cap S_2^3 = \{w_1^3\}$, donc $D_2(\omega) = \frac{1}{2}$.

La fenêtre temporelle ω n'est pas diagnosticable avec précision. Dans ce cas, la probabilité d'être dans un état de fonctionnement normal est de 50% ($D_0(\omega) = \frac{1}{2}$), la probabilité d'être en état de faute F_2 est de 50% ($D_2(\omega) = \frac{1}{2}$) et l'occurrence des fautes de la classe F_1 n'est pas probables ($D_1(\omega) = 0$).

4.2 Pronostic

Dans les SED, le pronostic des fautes $e_f \in F_\alpha$, lorsque ω est généré, il consiste à prédire la probabilité d'occurrence d'une des fautes de $e_f \in F_\alpha$ dans les k fenêtres temporelles postérieures à ω .

4.2.1 Prévisibilité

Définition 9 : une classe des fautes F_α est prévisible après l'occurrence de ω , si nous pouvons prédire sans ambiguïté l'occurrence ou non de toutes les fautes $e_f \in F_\alpha$ dans les k fenêtres temporelles postérieures à ω .

La définition 9 est équivalente à la définition originale de la k- prévisibilité proposée dans [92]: Si une faute est prévisible dans les k fenêtres temporelles postérieures à ω , l'occurrence de cette faute est prévisible dans un délai des k observables événements suivants, donc cette faute est k-prévisible.

Proposition 2 : la prévisibilité d'occurrence d'un événement $e_f \in F_\alpha$ dans les k fenêtres temporelles postérieures à w_i^k est équivalente à la probabilité de son occurrence dans w_{i+1}^k et w_{i+k}^k .

Preuve : soit $w_i^k = e_i^{d_i} e_{i,1}^{d_{i,1}} e_{i,2}^{d_{i,2}} \dots e_{i,n_i}^{d_{i,n_i}} e_{i+1}^{d_{i+1}} \dots e_{i+k}^{d_{i+k}} \in W^k$ la fenêtre temporelle de dimension k générée par le SED tel que $|P_o(w_i^k)| = k$ et $e_{i,j}^{d_{i,j}}$ ($j = 1, 2 \dots n_i$) sont les événements non observables qui se produisent entre les deux événements observables $e_i^{d_i}$ et $e_{i+1}^{d_{i+1}}$. Donc :

$$\begin{aligned}
 w_{i+1}^k &= e_{i+1}^{d_{i+1}} e_{i+1,1}^{d_{i+1,1}} e_{i+1,2}^{d_{i+1,2}} \dots e_{i+1,n_{i+1}}^{d_{i+1,n_{i+1}}} e_{i+2}^{d_{i+2}} \dots e_{i+1+k}^{d_k} \\
 w_{i+2}^k &= e_{i+2}^{d_{i+2}} e_{i+2,1}^{d_{i+2,1}} e_{i+2,2}^{d_{i+2,2}} \dots e_{i+2,n_i}^{d_{i+2,n_i}} e_{i+3}^{d_{i+3}} \dots e_{i+2+k}^{d_k} \\
 &\vdots \\
 w_{i+k-1}^k &= e_{i+k-1}^{d_{i+k-1}} e_{i+k-1,1}^{d_{i+k-1,1}} e_{i+k-1,2}^{d_{i+k-1,2}} \dots e_{i+k-1,n_i}^{d_{i+k-1,n_i}} e_{i+k}^{d_{i+k}} \dots e_{i-1+2k}^{d_k} \\
 w_{i+k}^k &= e_{i+k}^{d_{i+k}} e_{i+k,1}^{d_{i+k,1}} e_{i+k,2}^{d_{i+k,2}} \dots e_{i+k,n_{i+k}}^{d_{i+k,n_{i+k}}} e_{i+k+1}^{d_{i+k+1}} \dots e_{i+2k}^{d_k}
 \end{aligned}$$

De ceci, nous pouvons tirer deux remarques :

- w_{i+1}^k et w_{i+k}^k contiennent tous les évènements des k fenêtres temporelles postérieures à w_i^k .
- Les évènements des fenêtres temporelles w_{i+1}^k et w_{i+k}^k ont le même ordre que les évènements des k fenêtres temporelles postérieures à w_i^k .

Donc, la probabilité d'occurrence d'un évènement $e_f \in F_\alpha$ dans les k fenêtres temporelles postérieures à w_i^k est équivalente à sa probabilité d'occurrence dans w_{i+1}^k et w_{i+k}^k . Cette proposition permet de réduire la difficulté du calcul des paramètres $P_\alpha(\omega)$ que nous présentons par la suite.

Exemple 7 : en reprenant σ_1 et σ_2 de l'exemple 4. Nous pouvons remarquer que w_2^3 et w_4^3 contiennent tous les évènements (genre et ordre) des 3 fenêtres temporelles postérieures à w_1^3 . (w_2^3 , w_3^3 et w_4^3).

Sur la base de la proposition 2, nous pouvons définir l'ensemble $suf_k(\omega)$, défini par l'équation (III.4) qui regroupe tous les couples des fenêtres temporelles contenant k évènements observables qui peuvent représenter une continuation possible de la fenêtre temporelle ω .

$$suf_k(\omega) = \{(w_{i+1}^k, w_{i+k}^k) \mid P_o(w_i^k) = \omega\} \quad (III.4)$$

Exemple 8 : reprenons encore σ_1 et σ_2 de l'exemple 4 et considérons $\omega = e_2^1 e_1^{2,1} e_3^{4,0}$ comme étant la dernière fenêtre temporelle générée par le SED. W^3 contient trois fenêtres temporelles similaires à ω ($P_o^{-1}(\omega)$) : w_4^3 , w_5^3 et w_8^3 mais w_8^3 est la dernière fenêtre temporelle dans W^3 donc elle n'a pas de continuation. D'où ω a deux possibles continuations (w_5^3, w_7^3) et (w_6^3, w_8^3) ainsi, $suf_k(\omega) = \{(w_5^3, w_7^3) \text{ et } (w_6^3, w_8^3)\}$

À partir de modèle statistique, nous pouvons définir le degré de prévisibilité $P_\alpha^k(\omega)$ (2) qu'est un paramètre statistique qui caractérise la probabilité qu'une des fautes $e_f \in F_\alpha$ puisse se produire dans les k fenêtres temporelles postérieures à ω .

$$P_\alpha(\omega) = \frac{\text{nombre des } w_i \text{ similaire à } \omega \text{ prefixe d'un mot contenant } e_f \in F_\alpha}{\text{nombre des } w_i \text{ similaire à } \omega}$$

Plus formellement :

$$P_\alpha(\omega) = \begin{cases} \frac{\text{card}\{(\vartheta, \mu) \in \text{suf}_k(\omega) | \vartheta \in S_\alpha^k \text{ or } \mu \in S_\alpha^k\}}{\text{card } P_o^{-1}(\omega)} & \alpha \neq 0 \\ \frac{\text{card}\{(\vartheta, \mu) \in \text{suf}_k(\omega) | \vartheta \in S_0^k \text{ And } \mu \in S_0^k\}}{\text{card } P_o^{-1}(\omega)} & \alpha = 0 \end{cases} \quad (\text{III.5})$$

Où l'ensemble $\{(\vartheta, \mu) \in \text{suf}_k(\omega) | \vartheta \in S_\alpha^k \text{ or } \mu \in S_\alpha^k\}$ (respectivement $\{(\vartheta, \mu) \in \text{suf}_k(\omega) | \vartheta \in S_0^k \text{ And } \mu \in S_0^k\}$) est l'ensemble de tous les couples des mots temporisés, à partir de la base de données, qui peuvent représenté une continuation de ω , qui contiennent une faute appartenant à F_α (respectivement qui ne contiennent aucune faute). Les deux ensembles $\text{suf}_k(\omega)$ et $P_o^{-1}(\omega)$ sont respectivement défini par les équations (III.4) et (III.2).

Proposition 3 : Soit une fenêtre temporelle ω de dimension k, une faute $e_f \in F_\alpha$ est prévisible après l'occurrence de ω est généré par le SED si et seulement si $P_\alpha^k(\omega) = 1$ ou $P_\alpha^k(\omega) = 0$

Preuve : la preuve de la proposition 3 est similaire à la preuve de la proposition 1.

4.2.2 Temps restant avant l'occurrence d'une faute prévue

La description des SED avec des mots temporisés nous permet de caractériser chaque évènement par sa date d'occurrence. Ainsi, nous pouvons définir pour une fenêtre temporelle ω et pour chaque classe F_α la fonction, défini par l'équation (III.6), qui donne le temps restant avant l'occurrence de la faute e_{f_α} .

$$t_\alpha(\omega) = \begin{cases} -1 & , \quad P_\alpha(\omega) = 0 \\ \min\{T^\alpha(\omega)\} & , \quad P_\alpha(\omega) > 0 \end{cases} \quad (\text{III.6})$$

Où $T^\alpha(\omega) = \{T_{e_\alpha}(\mu) | \exists \vartheta \in W^k \text{ s. t. } ((\mu, \vartheta) \in \text{suf}_k(\omega) \text{ or } (\vartheta, \mu) \in \text{suf}_k(\omega)) \text{ and } \mu \in S_\alpha^k\}$ est l'ensemble de toutes les valeurs probables qui peuvent représenter le temps restant avant l'occurrence de la faute $e_f \in F_\alpha$ durant les k observables évènements postérieurs à ω . La fonction $t_\alpha(\omega)$ (III.6) prend le minimum de cet ensemble si l'occurrence de cette faute est probable sinon elle prend la valeur -1.

Exemple 9 : considérons la trace $\sigma_1 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5} e_3^{9.0}$ de l'exemple 4 et $\omega = e_2^1 e_3^{2.1} e_1^{5.0}$ comme la séquence des évènements générée par le SED.

$$W^3 = \{w_1^3 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, w_2^3 = e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}, w_3^3 = e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}, w_4^3 = e_2^{8.0} e_1^{8.5} e_3^{9.0}\}$$

$$P_o^{-1}(\omega) = \{w_1^3\}, \text{suf}_3(\omega) = \{(w_2^3, w_4^3)\}$$

$$t_{f1}(\omega) = \min\{T_1(w_2^3), T_1(w_4^3)\} = 6,3$$

$$t_{f2}(w_1^k) = -1$$

Ce qui signifie que l'occurrence de la faute e_{f1} est probable à la date 6,3, cependant l'occurrence de la faute e_{f2} n'est pas probable.

Selon la classe des fautes détectées ou prédites, la valeur des probabilités $D_\alpha(\omega)$ et $P_\alpha(\omega)$ et en se basant sur l'expertise, plusieurs actions peuvent être mises en place qui peut être un simple réglage des paramètres du SED et qui peut aller jusqu'à l'arrêt de SED.

5. Comparaison avec les méthodes classiques de diagnostic des SED

L'approche présentée dans la section III.3 peut être comparée avec les autres méthodes existantes dans la littérature sur trois niveaux :

- L'ensemble des mots temporisés, d'où est dérivé le modèle statistique utilisé pour décrire les modes de fonctionnements de SED peut être considéré comme le langage de l'automate fini temporisé qui peut modéliser le fonctionnement de SED, ceci justifie la validité de notre modèle statistique.
- La méthode de diagnostic classique [1, 7, 14], qui repose sur les automates finis et qui représente la base de quasiment la totalité des méthodes de diagnostic des fautes dans les SED, propose des diagnostiqueurs sous forme d'un automate fini où à chaque état nous affectons un des labels N, F ou U. Les paramètres $D_\alpha(\omega)$ sont équivalents à ces labels : $D_\alpha(\omega) = 1$ est équivalent au label F_α (état de faute), $D_0(\omega) = 1$ est équivalent au label N (état normal) et $0 < D_\alpha(\omega) < 1$ est équivalent au label U (état incertain, *uncertain* en anglais). Cependant ce paramètre est plus significatif dans le cas du label U, car il donne la probabilité d'occurrence de la faute.
- L'utilisation de cette approche permet la détection (respectivement la prévention) des fautes avant l'occurrence de k événements observables ce qui est équivalent à la k -diagnosticable/ k -prévisibilité (respectivement dans la limite des k événements observables futures) présenté dans [1, 92].

6. Efficacité maximum du diagnostic et du pronostic

6.1 Minimisation des dimensions des fenêtres temporelles

L'efficacité des opérations de la détection et de la prévention des fautes dans un SED est proportionnelle au nombre des événements prises en considération lors de la construction des fenêtres temporelles. Cependant, ces dernières vont représenter, dans le chapitre IV, les vecteurs d'entrée des RdN qui vont assurer le diagnostic et le pronostic des fautes. Si leur taille est grande, les architectures des RdN seraient plus complexes. Donc, c'est très judicieux de trouver un compromis entre le nombre des événements qu'il faut prendre en considération et le degré d'efficacité des opérations de diagnostic/pronostic des fautes.

Nous proposons, dans la Figure III.3, un algorithme itératif qui cherche les dimensions minimales m_d et m_p des fenêtres temporelles (nombre des évènements observables dans chaque fenêtre) qui assurent l'efficacité maximale respectivement du diagnostic et du pronostic de SED. En d'autres termes, cet algorithme donne les dimensions m_d et m_p qui maximisent respectivement les degrés de diagnosticabilité D_α et de prévisibilité P_α d'une classe des fautes cible (souvent la classe des fautes qui représentent des dangers sur les opérateurs ou sur le SED considéré) ou maximiser la diagnosticabilité de toutes les classes pour atteindre un certain degré de diagnosticabilité/prévisibilité.

Cet algorithme prend en entrée un ensemble des mots temporisés $[\sigma_j]$ et le paramètre k_{lim} qui représente la dimension maximale des fenêtres temporelles à ne pas dépasser. Dans chaque itération, la dimension k est incrémentée, les fenêtres temporelles sont conçues et les paramètres D_α et P_α sont calculés. Les sorties de cet algorithme sont les m_d (III.9) et m_p (III.10) qui représentent les dimensions minimales qui maximisent respectivement les fonctions coût F_D^k (III.7) et F_P^k (III.8) [91].

$$F_D^k = \text{Mean}_{\alpha=1, \dots, m} \left\{ \text{Mean}_{w \in S_\alpha^k} \{D_\alpha(w)\} \right\} \quad (\text{III.7})$$

$$F_P^k = \text{Mean}_{\alpha=1, \dots, m} \left\{ \text{Mean}_{w \in S_\alpha^k} \{P_\alpha(w)\} \right\} \quad (\text{III.8})$$

$$m_d = \min_{k=1, \dots, k_{lim}} \left\{ k \mid F_D^k = \text{Max}_{k=1, \dots, k_{lim}} \{F_D^k\} \right\} \quad (\text{III.9})$$

$$m_p = \min_{k=1, \dots, k_{lim}} \left\{ k \mid F_P^k = \text{Max}_{k=1, \dots, k_{lim}} \{F_P^k\} \right\} \quad (\text{III.10})$$

$F_D^{m_d}$ et $F_P^{m_p}$ caractérisent respectivement les degrés de diagnosticabilité et de prévisibilité du SED considéré.

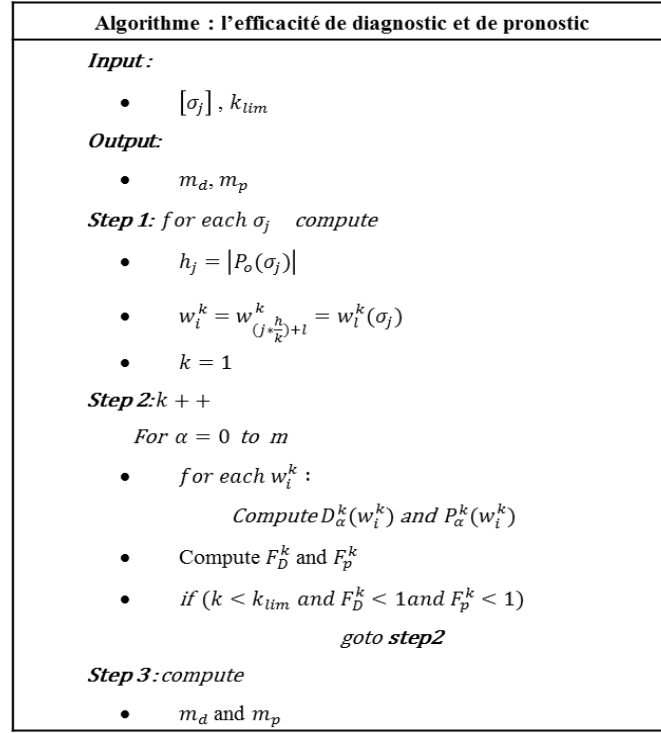


Figure III.3: Algorithme pour augmenter l'efficacité de diagnostic/pronostic.

L'algorithme présenté dans la Figure III.3 maximise les fonctions coûts $F_D^{m_d}$ et $F_p^{m_p}$. Cependant, nous pouvons prendre comme fonctions coûts les paramètres $\hat{D}_\alpha^k$ et $\hat{P}_\alpha^k$ définies par les équations (III.11) et (III.12) telle que F_α est la classe cible que nous voulons maximiser sa diagnosticabilité et sa prévisibilité. Souvent, cette classe cible est celle qui présente les dégâts les plus néfastes.

$$\hat{D}_\alpha^k = \text{Mean}\{D_\alpha(w)\}_{w \in S_\alpha^k} \quad (\text{III.11})$$

$$\hat{P}_\alpha^k = \text{Mean}\{P_\alpha(w)\}_{w \in S_\alpha^k} \quad (\text{III.12})$$

6.2 Intégration des lois de fiabilité

L'approche développée dans la section 3, qui se base sur des données statistiques, prend en compte l'ordre et les dates d'occurrence des événements générés par le SED et néglige le degré de dégradation et le vieillissement des composants du SED considéré. Cette dégradation peut être traduite par la fiabilité de ces composants, qui varie en fonction de leurs temps de bon fonctionnement. Dans la suite de cette section, nous visons améliorer l'approche de diagnostic/pronostic, développée dans la section III.3, en combinant (figure III.4), cette approche représentée par ses paramètres $D_\alpha(w_i)$ et $P_\alpha(w_i)$ avec celle des lois de fiabilité. Ceci sera élaboré en utilisant quelques notions de la logique floue.

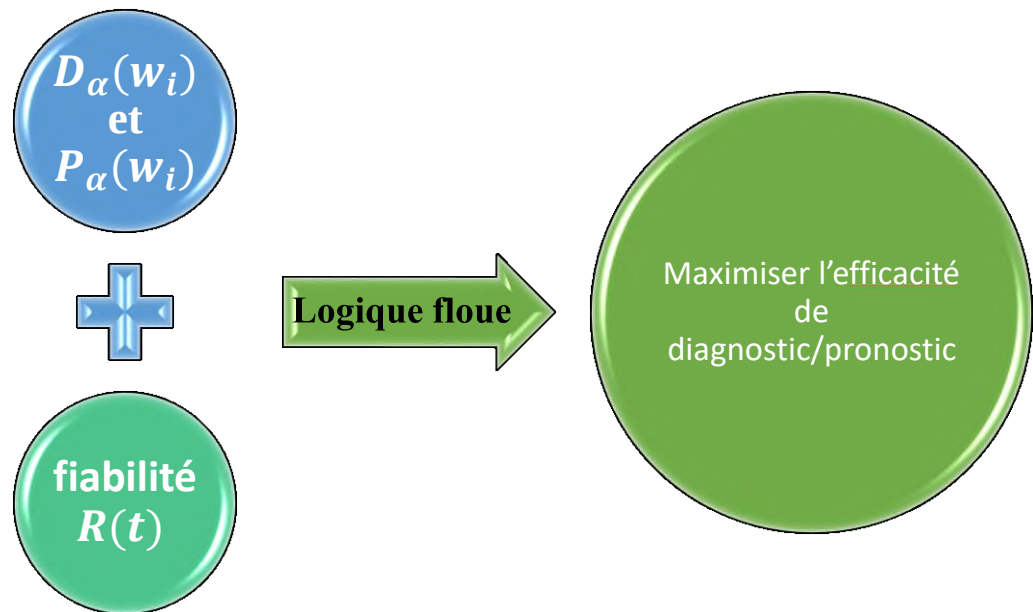


Figure III.4: Synoptique de la méthode développée.

Ces lois de fiabilité donnent des bonnes estimations de l'état de fonctionnement de système en se basant sur les durées de vie de ses composants. Cependant ces lois ne prennent pas en considération l'état de fonctionnement actuel réel de système qui peut être traduit par les événements générés par le SED, leur ordre et leurs dates d'occurrence, et qui peut être estimé par l'approche développée dans la section III.3. Donc, cette combinaison est primordiale pour avoir une efficacité maximale des opérations de diagnostic/pronostic [93].

6.2.1 Lois de fiabilité

La courbe en baignoire, représentée dans la figure III.5, donne la variation de taux de défaillance en fonction de temps de bon fonctionnement, ceci montre que La durée de vie de la majorité des composants de système peut être caractérisée par trois phases différentes :

- La phase de rodage ou de jeunesse : elle se caractérise par une décroissance rapide de taux de la défaillance. Ce dernier est très élevé au début de cette phase et s'explique par le degré de compatibilité entre le nouveau composant installé et les autres composants de système.
- La vie utile ou phase productive : ici, le taux de défaillance est quasiment constant. En général, c'est la phase la plus longue de toutes les autres phases.
- La phase de vieillissement : se caractérise par un taux de défaillance progressivement croissant dû au vieillissement du matériel.

Remarque 4 : on parle de la durée de vie pour les composants irréparables et la durée de bon fonctionnement pour les composants réparables.

La décision de diagnostic/pronostic doit obligatoirement prendre en compte la zone où se situe le composant dans la courbe fiabilité (courbe en baignoire).

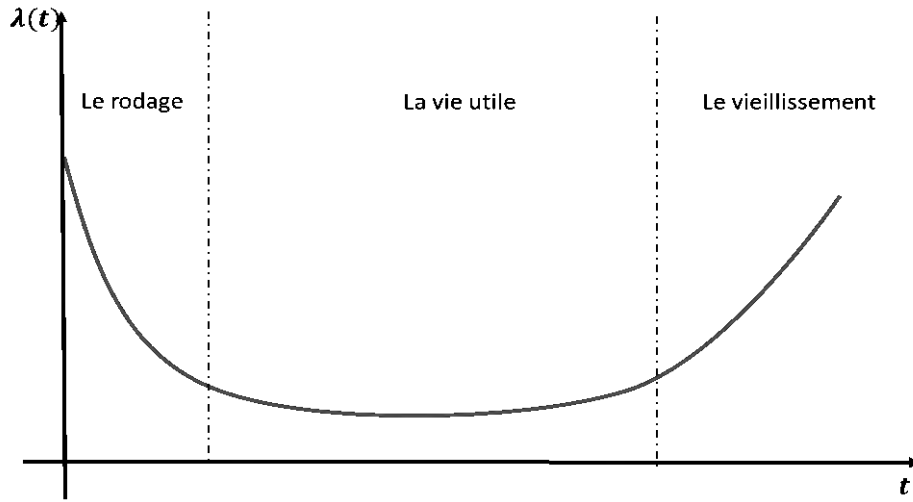


Figure III.5: Profil en baignoire.

La fiabilité d'un équipement est la probabilité de n'avoir aucune défaillance pendant sa durée de bon fonctionnement t après sa dernière défaillance. Elle est notée $R(t) \in [0,1]$. Cependant, $F(t) = 1 - R(t)$ représente la probabilité d'avoir au moins une défaillance pendant la durée t . En général, $R(t)$ d'un équipement s'écrit sous la forme de l'équation (III.13).

$$R(t) = e^{-\int \lambda(t) dt} \quad (\text{III.13})$$

$\lambda(t)$ est le taux de défaillance. Selon la forme de $\lambda(t)$, la loi de fiabilité $R(t)$ d'un composant de système C^α peut avoir plusieurs formes, selon les lois :

- La forme exponentielle où le taux de défaillance est considéré constant et égale à $\lambda = \frac{1}{MTTF}$ tel que MTTF est le temps moyen avant défaillance, ainsi $R(t)$ a la forme de l'équation (III.14).

$$R(t) = e^{-\lambda t} \quad (\text{III.14})$$

- La forme normale, où $R(t)$ est définie par l'équation (III.15), dont les paramètres sont la moyenne m et l'écart-type σ .

$$R(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-m)^2}{2\sigma^2}} \quad (\text{III.15})$$

- La forme de Weibull qui représente la loi la plus utilisée pour les équipements industriels où $R(t)$ est définie par l'équation (III.16), dont les paramètres β (paramètre de forme), η (paramètre d'échelle) et γ (paramètre de position) peuvent être déterminés en utilisant l'ajustement graphique, le papier de Weibull, la résolution mathématique de la régression linéaire ou l'ajustement par la méthode du maximum de vraisemblance.

$$R(t) = e^{-\left|\left(\frac{t-\gamma}{\eta}\right)^\beta\right|} \quad (\text{III.16})$$

Dans la littérature, plusieurs tests sont disponibles pour vérifier le degré d'adéquation d'une loi au fonctionnement d'un équipement, nous citons : le test de χ^2 , le test de Lilliefors (loi normale), le test de Bartlett (loi exponentielle) et le test de Mann (la loi de Weibull)...

La loi de Weibull, en raison de sa flexibilité, est très utile pour modéliser la fiabilité des divers équipements à partir d'un retour d'expérience. Ceci est due à son expression qui recouvre toute une famille de lois. C'est, en particulier, le cas de la loi exponentielle ($\beta = 1$), la loi normale ($3 \leq \beta \leq 4$) et la loi de Rayleigh ($\beta = 2$) [94].

Dans la suite de cette section, nous considérons un SED composé de n composants et qui fonctionne selon $m + 1$ états de fonctionnement (m états de fonctionnement anormal et un état de fonctionnement normal) et décrit par une base de données contenant des enregistrements des événements générés pendant une durée significative. Ainsi, nous relierons chaque classe de faute F_α aux défaillances d'un composant C^α qui a après une durée de bon fonctionnement t , mesuré à partir de sa dernière défaillance, la probabilité de défaillance est $F^\alpha(t)$ (ou bien une fiabilité $R^\alpha(t) = 1 - F^\alpha(t)$) calculée en considérant la loi de fiabilité la plus adéquate au fonctionnement de ce composant. Avant de présenter comment l'intégration entre les deux approches sera faite, nous rappelons quelques notions de la logique floue qui seront utilisées par la suite.

Remarque 5 : si un composant est composé de plusieurs sous composants, chaque sous composant a sa propre fiabilité. La fiabilité totale de composant global peut être calculée en utilisant les méthodes proposées dans [95].

6.2.2 Logique floue

La logique floue est déterminée comme étant un ensemble des principes mathématiques dédiés à la représentation des connaissances basée sur les degrés d'appartenance plutôt que sur l'appartenance booléenne de la logique binaire classique [96]. Ainsi, la logique floue, définit le degré d'appartenance d'un élément ω à une classe A par $\mu_A(\omega) \in [0,1]$ contrairement à la logique booléenne qui le définit par $\mu_A(\omega) \in \{0,1\}$. Un ensemble flou A est défini sur un univers U par l'équation (III.17).

$$A = \{\langle x, \mu_A(x) \rangle | x \in U\} \quad (\text{III.17})$$

Semblablement à la logique des ensembles classique, la logique des ensembles flous comporte plusieurs opérations comme l'union, l'intersection... La définition de ces opérateurs sur les ensembles flous peut être trouvée dans [97]. Parmi ces opérateurs on trouve les haies [98] qui sont des opérateurs unaires permettant la modification des valeurs de la fonction d'appartenance et permettent la génération des nouveaux ensembles flous à partir d'un ensemble flou original. Selon leurs impacts sur les fonctions d'appartenances, les haies sont de plusieurs types, nous citons :

- La concentration d'un ensemble flou, illustré dans la figure III.6, génère un nouvel ensemble avec des degrés d'appartenance relativement inférieurs à ceux de l'ensemble d'origine. La concentration d'un ensemble A est noté $con(A)$ et définie par l'équation (III.18), avec $\mu_{con(A)}(x) = \mu_A(x)^2$.

$$con(A) = \{ \langle x, \mu_{con(A)}(x) \rangle | x \in U \} \quad (III.18)$$

- La dilatation d'un ensemble, illustré dans la figure III.6, est l'opération inverse de la concentration. Le résultat de la dilatation d'un ensemble flou A est un nouvel ensemble flou noté $dil(A)$ avec des degrés d'appartenance supérieurs à ceux de l'ensemble original A. la dilatation de A est définie par l'équation (III.19), avec $\mu_{dil(A)}(x) = \sqrt{\mu_A(x)} = (\mu_A(x))^{0.5}$.

$$dil(A) = \{ \langle x, \mu_{dil(A)}(x) \rangle | x \in U \} \quad (III.19)$$

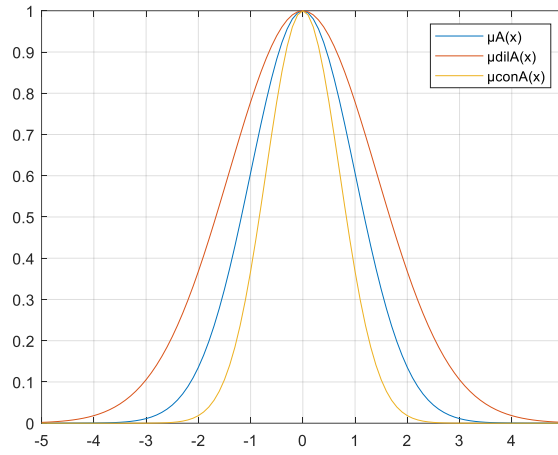


Figure III.6: Fonction d'appartenance d'un ensemble flou (loi normale), sa dilatation et sa concentration.

Remarque 6 : La dilatation et la concentration peuvent être appliquées sur les ensembles flous avec des fonctions d'appartenance continues et aussi sur les ensembles flous avec des fonctions d'appartenance discrète.

De même nous pouvons définir la dilatation d'ordre r qui permet de construire en fonction de r différents ensembles flous dérivés de l'ensemble original.

Définition 10: la dilatation d'ordre r ($r \in [0,1]$) d'un ensemble flou A est un nouvel ensemble flou dérivé de l'ensemble original A, noté $dil^r(A)$ est défini par l'équation (III.20) avec $\mu_{dil^r(A)}(x) = (\mu_A(x))^r$ [93].

$$dil^r(A) = \{ \langle x, \mu_{dil^r(A)}(x) \rangle | x \in U \} \quad (III.20)$$

Cette nouvelle notion sera utilisée pour l'amélioration des opérations de diagnostic/pronostic.

Remarque 7 : on peut même définir une concentration d'ordre r , mais ça ne serait pas intéressant dans le reste de ce travail.

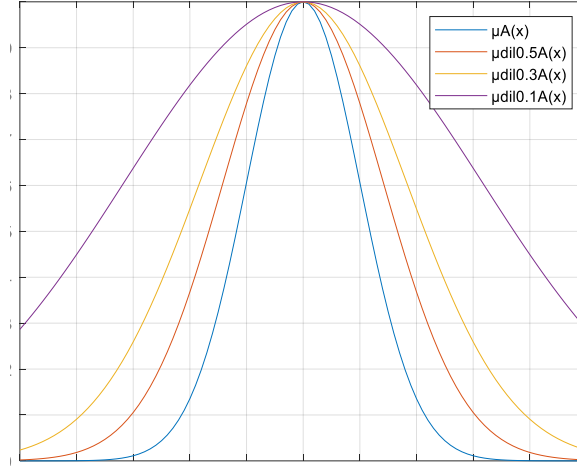


Figure III.7 : Fonction d'appartenance d'un ensemble flou, ses dilatations d'ordre $r = 0.5, 0.3$ et 0.1 .

6.2.3 Intégration des lois de fiabilité dans le diagnostic des fautes

L'état de fonctionnement d'un SED après l'occurrence de la séquence des événements observables ω peut-être décrit par des ensembles flous définis par l'équation (III.21) (avec $0 \leq \alpha \leq m$).

$$E_\alpha(\omega) = \{(\omega, D_\alpha(\omega)) | \omega \in W^k\} \quad (\text{III.21})$$

Ces ensembles flous sont construits à partir du modèle statistique W^k associé au SED considéré, qui regroupe l'ensemble des mots temporisés de dimension k que le SED peut générer, $D_\alpha(\omega)$ représentent les degrés d'appartenance de ω aux états de fonctionnement E_α . Ainsi, ces ensembles donnent l'état de fonctionnement de SED selon la dernière séquence des événements générés par le SED ω et même la probabilité d'être dans cet état de fonctionnement.

La fiabilité des composants est inversement proportionnelle à la durée de fonctionnement, donc, il est très judicieux que le diagnostic d'une fenêtre temporelle ω à l'instant t , soit différent que son diagnostic à l'instant $t + \Delta t$. Afin de prendre en compte le vieillissement des équipements, qui rend l'occurrence d'une faute plus probable, lorsque la même séquence d'événements est générée, nous faisons recours à la notion de la dilatation d'ordre r de l'ensemble flou E_α , tel que l'ordre r dépend de la fiabilité du composant C^α et il est défini par $r = F^\alpha(t) = 1 - R^\alpha(t)$. Donc, la dilatation est plus forte lorsque le composant C^α est moins fiable, donc l'occurrence des fautes $e_f \in F_\alpha$ est plus probable. Ainsi l'ensemble E_α devient :

$$E_\alpha(\omega, t) = \{(\omega, \hat{D}_\alpha(\omega, t)) | \omega \in W^k\} \quad (\text{III.22})$$

Avec $\hat{D}_\alpha(\omega, t) = (D_\alpha(\omega))^{F^\alpha(t)}$ est la nouvelle probabilité d'occurrence d'une faute $e_f \in F_\alpha$ lors de l'enregistrement de la séquence des événements observables ω à l'instant t , en tenant en considération

l'ordre des évènements observables générés, leurs natures et la fiabilité du composant C^α responsable des fautes appartenant à la classe des fautes F_α .

Remarque 8 : le temps t peut être remplacé dans certains cas par le nombre des cycles de fonctionnement N .

6.2.4 Intégration des lois de fiabilité dans le pronostic

Les ensembles flous définis par l'équation (III.23) donnent les états de fonctionnement futurs que le SED peut avoir après la génération de la séquence des évènements a été générée ω .

$$EF_\alpha = \{\langle \omega, P_\alpha(\omega) \rangle | \omega \in W^k\} \quad (III.23)$$

Ces ensembles flous, construits à partir du modèle statistique W^k , associées au SED considéré, décrivent les états futurs que peut avoir le système après la génération de la séquence des évènements observables ω . Telle que $P_\alpha(w_i)$ (III.5) représente le degré d'appartenance de ω à l'état de fonctionnement EF_α et ainsi il représente la probabilité d'occurrence d'une faute appartenant à la classe des fautes F_α dans le futur. En prenant en considération la fiabilité à l'instant $t + t_\alpha$ du composant C^α où la faute est probable, et après une dilatation d'ordre $R(t)$, les ensembles flous EF_α devient :

$$EF_\alpha(\omega, t) = \{\langle \omega, \hat{P}_\alpha(\omega, t) \rangle | \omega \in W^k\} \quad (III.24)$$

Telle que $\hat{P}_\alpha(\omega, t) = (P_\alpha(\omega))^{F_\alpha(t+t_\alpha)}$ est la nouvelle probabilité d'occurrence d'une faute $e_f \in F_\alpha$ dans les k fenêtres temporelles postérieures à ω à l'instant t et lorsque ω a été généré, en tenant en considération l'ordre des évènements observables générés, leurs natures et la fiabilité du composant C^α responsable des fautes appartenant à la classe des fautes F_α à l'instant $t + t_\alpha$ où la faute est probable.

7. Validité des contraintes temporelles dans un SED

Le bon fonctionnement d'un système à évènements discrets est jugé par la génération des évènements dans le bon ordre et à leurs dates appropriées. L'exécution de ces évènements dans leurs dates appropriées, représente une contrainte temporelle (que nous notons par la suite CT) que le SED doit vérifier. Dans cette section, nous présentons une approche de vérification de la validité de ces contraintes temporelles. Mais, dans un premier temps nous présentons comment les contraintes temporelles peuvent être extraite de modèle statistique. Seules les fautes qui se traduisent par un retard ou une avance d'exécution seront prises en considération. Dans cette section nous adopterons la forme des mots temporisés décrite dans la remarque 1.

7.1 Contraintes temporelles

Considérons un SED qui fonctionne selon un seul cycle répétitif, durant lequel le SED exécute plusieurs tâches. Chaque exécution de ce cycle est décrite par un mot temporisé contenant l'ensemble des

événements définissant des transactions entre les différentes taches successives. Ces événements sont couplés avec leurs dates d'occurrences.

Nous considérons les hypothèses :

H1 : On suppose que les événements observables se produisent toujours dans le bon ordre. Cette hypothèse reste valable tant que nous pouvons utiliser les méthodes de reconnaissance des formes existantes dans la littérature pour la détection des séquences qui contiennent des événements dans le mauvais ordre, ou plus précisément la méthode décrit dans la section III.3.

H2 : on suppose que le SED fonctionne selon un seul cycle répétitif où nous vérifions le degré de respect des CT après l'occurrence de chaque événement observable. Pour les SED qui fonctionnent selon différentes plusieurs cycles nous pouvons envisager un diagnostiqueur pour chaque type de cycle. Dans ce cas le SED est considéré dans un fonctionnement normal si un des diagnostiqueurs a jugé que tous les CT sont respectés.

En prenant en considération ces deux hypothèses, les mots temporisés générés lors des cycles de fonctionnement du SED auront la même dimension et les mêmes événements dans le même ordre. Ainsi, la notion des fenêtres temporelles ne sera pas utilisée dans cette section : le modèle statistique sera composé des mots temporisés enregistrés sans les décortiquer en fenêtres temporelles.

Dans les SED, chaque tache a une durée bien spécifique. L'accomplissement de chaque tâche dans sa durée nominale représente une CT que le SED doit satisfaire. Ceci nécessite que les événements annonçant la fin et le début de la tache doivent se produire dans leurs dates nominales à partir de l'exécution de leurs événements précédents. Chaque événement e_α est associé à une contrainte temporelle, que nous notons par $CT(e_\alpha)$, ceci est défini par un intervalle de temps pendant lequel l'occurrence de cet événement est souhaitable. Plus formellement $CT(e_\alpha)$ est défini par l'équation (III.25), où $t_{\alpha,min}$ est la date d'exécution de l'événement e_α au plus tôt et $t_{\alpha,max}$ est sa date d'exécution au plus tard. Ceux-ci peuvent être déterminés en prenant, à partir de modèle statistique, la plus petite date et la plus grande date auxquelles cet événement est généré. Une CT non vérifiée signifie que l'événement associé à cette contrainte s'est exécuté en dehors de cet intervalle.

$$CT(e_\alpha) = [t_{\alpha,min} , t_{\alpha,max}] \quad (III.25)$$

La vérification des CT dans un SED peut être vue comme un problème de reconnaissance des formes, où les formes à reconnaître sont des vecteurs dont les éléments sont les dates d'occurrences des différents événements observables enregistrés pendant un cycle de fonctionnement du SED. À partir de ces vecteurs, il faut déterminer le degré de respect des CT où déterminer la CT violée dans le cas contraire.

Les fautes, qui font l'objet de cette section, sont celles que nous pouvons détecter par un retard d'occurrence d'un événement (vérin bloqué, moteur en surcharge, chute d'un colis...) ou par son

occurrence avant le bon moment (mauvais freinage lors de la descente d'un ascenseur...). Toutes les trois catégories des fautes citées dans la section III.1 peuvent être le résultat d'une violation d'une CT, en effet :

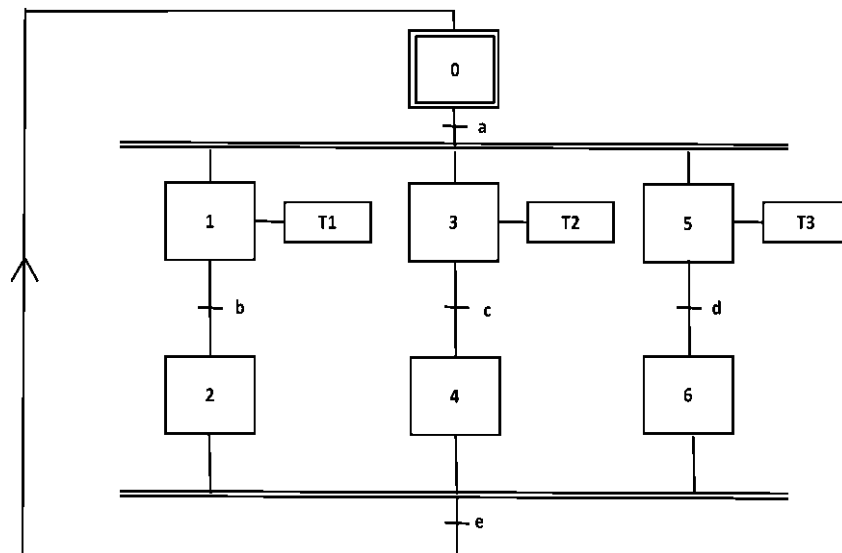
- Un moteur qui accélère ou ralenti un processus est une faute actionneur
- Le blocage d'un capteur ou son déclenchement à des moments aléatoires est le résultat d'une faute capteur
- La chute d'un produit d'un tapis roulant est une faute processus qui peut retarder l'arrivée du produit à sa destination.

Ce type des fautes ne peut pas être traité par les méthodes usuelles qui prennent en compte seulement le bon ordre des événements, et qui sont basées sur les automates finis, les réseaux de Pétri ou même l'approche présentée dans la section III.3. Ces fautes ne peuvent être détectées que par la vérification des CT, ce qui rend cette notion très intéressante dans les opérations de diagnostic/pronostic. Cependant, cette notion a été peu discutée dans la littérature [99].

7.2 Construction des CT à partir du modèle statistique

Le modèle statistique contient un nombre important des mots temporisés (événements couplés avec leurs dates d'occurrences) enregistrés pendant des cycles répétitifs qui traduisent la même séquence des tâches. Durant un cycle de fonctionnement, un événement peut se produire à différentes dates, et ces dates dépendent des dates des événements générés précédemment. Autrement dit, $CT(e_\alpha)$ dépend de toutes les $CT(e_i)$ ($i = 1, 2, \dots, \alpha - 1$).

Exemple 10 : considérons un SED, modélisé par le GRAFCET (une extension des réseaux de Pétri) de la figure III.8, qui exécute les tâches T1, T2 et T3 en parallèle. Considérons que le mot temporisé $\omega = a^{t_1}c^{t_2}d^{t_3}b^{t_4}$ est généré par ce SED. La bonne exécution de la tâche T1, qui est délimitée par les deux événements a et b, dépend des durées t_2 et t_3 (la durée $t_2 + t_3$ représente la durée de la tâche T1). Ainsi



en vérifiant les contraintes temporelles propres aux événements c et d on peut vérifier la bonne exécution de la tâche T1.

Figure III.8: SED avec trois tâches en parallèle.

7.3 Perte d'observabilité

Considérons le SED, illustré dans la figure III.9, qui contient seulement des événements observables et supposons que le capteur responsable de la génération de la transition t_1 a cessé de fonctionner. Dans ce cas la transition t_1 devient non observable et ainsi le diagnostiqueur va nous délivrer des informations inexactes sur l'état de fonctionnement de ce SED : il va nous informer que le SED est coincé dans la place P1 cependant le SED peut être dans la place p2. Cette faute peut être détectée en calculant le temps écoulé dans chaque place, autrement dit, calculer le temps entre l'exécution de la transition t_2 et la transition t_1 , si ce temps dépasse une certaine valeur limite cela signifie que le capteur responsable de la génération de la transition t_1 est en panne. L'approche développée dans cette section, qui permet la vérification des CT, est valable aussi pour la détection de la perte d'observabilité.

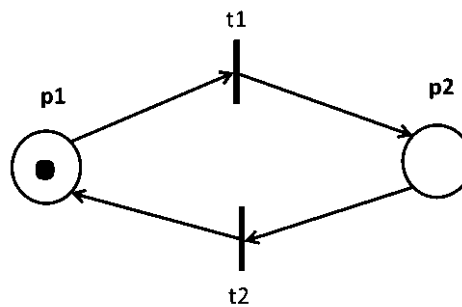


Figure III.9 : SED modélisé par réseau de Pétri.

La vérification de respect des CT peut être considérée comme un problème de classification, où nous essayons de classer les fonctionnements de SED à des fonctionnements normaux et autres anormaux, en fonction des durées écoulées entre la génération de chaque deux événements successifs. En général le bon fonctionnement de SED peut dépendre de la relation existante entre les différents CT des différents événements générés par le SED. Cette dépendance, que nous pouvons extraire à partir de modèle statistique associé au SED, rend le problème de la vérification des CT un problème non linéaire. Ceci nécessite un outil de l'intelligence artificielle pour résoudre un tel problème. Dans le chapitre IV, nous allons utiliser les RdN pour la vérification de ces CT.

8. Conclusion

Dans ce chapitre, nous avons pu présenter une nouvelle approche pour le diagnostic et le pronostic des SED larges et complexes dont l'application des méthodes de diagnostic/pronostic usuellement utilisé pour les SED est une tâche complexe. Dans un premier temps la notion des fenêtres temporelle a été utilisée pour construire un modèle statistique qui a permis de décrire tous les modes de fonctionnement de SED. Ensuite, ce modèle statistique est utilisé pour calculer des paramètres en relation avec le diagnostic/pronostic des fautes dans ce SED. Ces paramètres permettent de caractériser la probabilité

qu'une faute a été produite lors de l'enregistrement la dernière fenêtre temporelle et la probabilité de son occurrence dans la future.

L'efficacité de l'approche présentée a été améliorée en introduisant les lois de fiabilité et un algorithme qui permet de chercher les dimensions des fenêtres temporelles qui vont permettre de maximiser les probabilités de détection et de prédiction des fautes les plus critiques.

A la fin de ce chapitre, la notion des contraintes temporelles a été abordée, ceci a constitué une notion très intéressante pour la détection de quelque type des fautes comme la perte d'observabilité des événements observables et les fautes qui peuvent être le résultat d'un retard ou d'une exécution en avance d'un événement.

Dans le chapitre suivant, nous allons utiliser les RdN pour pouvoir apprendre les relations existantes entre les fenêtres temporelles du modèle statistique et les différents paramètres du diagnostic/pronostic afin de pouvoir mettre en place des diagnostiqueurs/pronostiqueurs des fautes en temps réel.

Chapitre IV:
Les réseaux de neurones pour le diagnostic et le pronostic des fautes
dans les SED

1. Introduction

Les RdN sont des outils très puissants de l'intelligence artificielle qui peut apprendre, à partir d'une base de données, les relations existantes entre un ensemble d'éléments d'entrée et un ensemble des éléments de sortie. Ce chapitre utilise les résultats des approches développées dans le chapitre III afin de construire les bases de données qui vont servir à l'apprentissage des différents RdN qui vont assurer les opérations de diagnostic et de pronostic. Ainsi, quatre réseaux de neurones vont être mis en place et qui vont assurer en temps réel le diagnostic de l'état de fonctionnement de SED, le pronostic des états de fonctionnement futurs de SED, la prédiction de temps restant avant l'occurrence d'une faute probable et la vérification du respect des contraintes temporelles. Des exemples et des cas d'application seront présentés.

2. Réseaux de neurones

2.1 Implémentation des RdN dans le diagnostic/pronostic des SED

Dans les SED, éviter l'occurrence d'une faute ou limiter au maximum les dégâts qui peuvent survenir après son occurrence est une nécessité majeure que les méthodes de diagnostic/pronostic doivent satisfaire. C'est pour une telle raison que ces méthodes doivent être précises, rapides et efficaces. Cependant, dans le cas des SED larges est complexes, nous pouvons avoir des modèles statistiques énormes dans lesquels la recherche des fenêtres temporelles, qui ont la même séquence des événements observables à celle générée par le SED, nécessaires pour le calcul des paramètres de diagnostic/pronostic peut représenter une tâche très complexe. Ceci demande un grand temps de calcul même avec des calculateurs très performants, qui peut influencer le temps de réponse des diagnostiqueurs/pronostiqueurs. L'objectif de ce chapitre est de synthétiser le modèle statistique d'un SED en construisant des bases de données qui servent à l'apprentissage des RdN qui vont assurer les opérations du diagnostic et du pronostic dans les SED. L'utilisation des RdN va permettre d'apprendre les relations existantes entre les différents éléments de ce modèle statistique et ainsi donner en temps réel des résultats précis de diagnostic/pronostic.

L'implémentation des RdN dans les problèmes de diagnostic/pronostic peut être de différentes formes [54, 70-72]. Dans cette thèse, l'implémentation des RdN dans le processus de diagnostic/pronostic, comme il est montré sur les figures IV.1 et IV.2, sera sur deux niveaux différents :

- L'apprentissage, représenté dans la figure IV.1, qui consiste à configurer, à partir d'une base de données, les paramètres de RdN en fonction des relations existantes entre un ensemble de vecteurs d'entrée (formé à partir des fenêtres temporelles) et de sortie (les paramètres de diagnostic ou de pronostic). Ceci nécessite la collecte des données du SED (mots temporisés enregistrés lors des cycles de fonctionnement de SED), les formater sous forme des fenêtres temporelles, calculer les paramètres en relation avec le diagnostic et le pronostic et ranger

l'ensemble de ces fenêtres temporelles et ces paramètres calculés dans des bases de données qui seront utilisées pour l'apprentissage.

- L'utilisation des RdN configurés lors de la phase d'apprentissage, comme montre la figure IV.2 pour traiter les informations reçues du SED afin de mettre en place un diagnostic/pronostic en temps réel. Ceci peut être sous deux formes différentes :
 - Une interpolation qui consiste à approximer quelques paramètres de diagnostic/pronostic comme les probabilités d'occurrence d'une faute ou la date de son occurrence dans le futur.
 - Une classification qui consiste à classifier un mode de fonctionnement traduit par une fenêtre temporelle à l'état de fonctionnement le plus probable qui consiste à vérifier le degré de respect des CT.

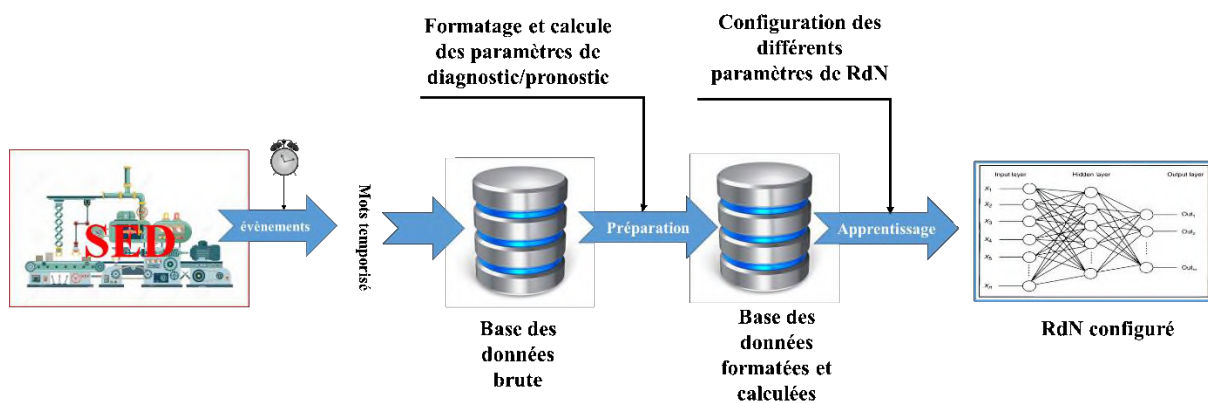


Figure IV.1 : processus d'apprentissage des RdN pour assurer le diagnostic/pronostic

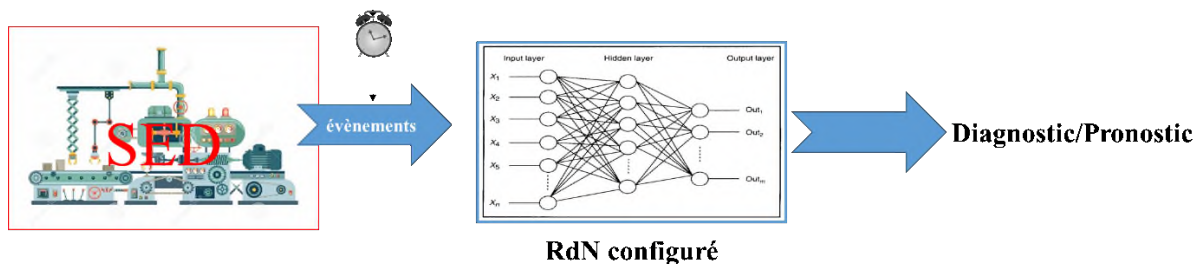


Figure IV.2: Implémentation des RdN dans le diagnostic/pronostic en temps réel.

2.2 Choix du RdN

À l'issue de la comparaison présentée dans le chapitre II, nous pouvons conclure que le RBFNN (réseau de neurones à base des fonctions radiales) est le RdN le plus le plus convenable pour la mise en œuvre de notre approche de diagnostic/pronostic vu le nombre important des avantages qui il présente. Ces avantages peuvent être résumés comme suit :

- Une grande précision résultante de la représentation locale de l'espace ;
- Un temps d'apprentissage réduit ;

- Une architecture simple et réduite à trois couches.

2.3 Apprentissage et choix des paramètres des RBFNN

Nous considérons un SED avec $m + 1$ classes de fonctionnement, une classe de fonctionnement normal et m classes de fonctionnement anormal. Ces modes de fonctionnement sont décrits par un large modèle statistique à partir duquel l'ensemble des paramètres qui traduisent le diagnostic et le pronostic des fautes sont calculés.

L'apprentissage de RBFNN consiste à établir une relation entre les entrées (généralement dans notre cas sont les fenêtres temporelles générées par le SED) et les sorties (les paramètres de diagnostic et de pronostic des fautes) des RdN

Pour chaque RBFNN, nous utilisons un apprentissage hybride dans lequel les neurones des couches cachées cherchent leurs centres et leurs rayons d'influence en utilisant un apprentissage non supervisé (le centre et le rayon d'influence d'un neurone de la couche cachée sont deux vecteurs qui ont la même dimension que les vecteurs d'entrée). Après, un apprentissage supervisé est utilisé pour la configuration des poids des différentes connexions entre les différents neurones [63]. Lors de ce dernier apprentissage et lorsqu'une nouvelle entrée est présentée au RBFNN les poids sont mis à jour en utilisant l'algorithme de la descente de gradient présenté dans la section II.3.2.3.

Comme le montre la figure IV.3, Le RBFNN comporte plusieurs paramètres que nous configurons comme suit :

- Le nombre des entrées n et le nombre des sorties m sont définis respectivement par le nombre des événements pris en compte pour assurer le diagnostic (vecteur d'entrée) et le nombre des paramètres que le RdN doit calculer (vecteur de sortie).
- Le nombre des neurones dans la couche cachée est déterminé d'une façon incrémentale, après chaque itération nous augmentons le nombre des neurones dans la couche cachée jusqu'à l'obtention d'une erreur quadratique moyenne minimale. L'erreur quadratique est calculée selon l'équation (IV.1), où c_i est la sortie cible, y_i est la sortie de RBFNN, N est la taille de la base d'apprentissage.

$$E = \frac{1}{N} \sum_{i=1}^N \|c_i - y_i\|^2 \quad (\text{IV.1})$$

- Les centres et les rayons d'influence (μ_i, σ_i) des neurones de la couche cachée sont choisis lors de la phase d'apprentissage non supervisé. Dans cette phase une des méthodes de regroupement (clustering) est utilisée comme la méthode K-Means.

- Les poids des connexions entre les différents neurones et les biais θ_i des neurones de la couche de sortie font l'objet de la deuxième phase d'apprentissage. Celle de l'apprentissage supervisé où ils sont configurés de tel sorte à minimiser l'erreur quadratique moyenne.
- Le pas d'apprentissage ε , utilisé dans l'algorithme de la descente de gradient (présenté dans la section 3.2.3 du chapitre II), est inversement proportionnel au temps d'apprentissage : un grand pas d'apprentissage peut le rendre instable, ainsi ce paramètre est choisi le plus petit possible.

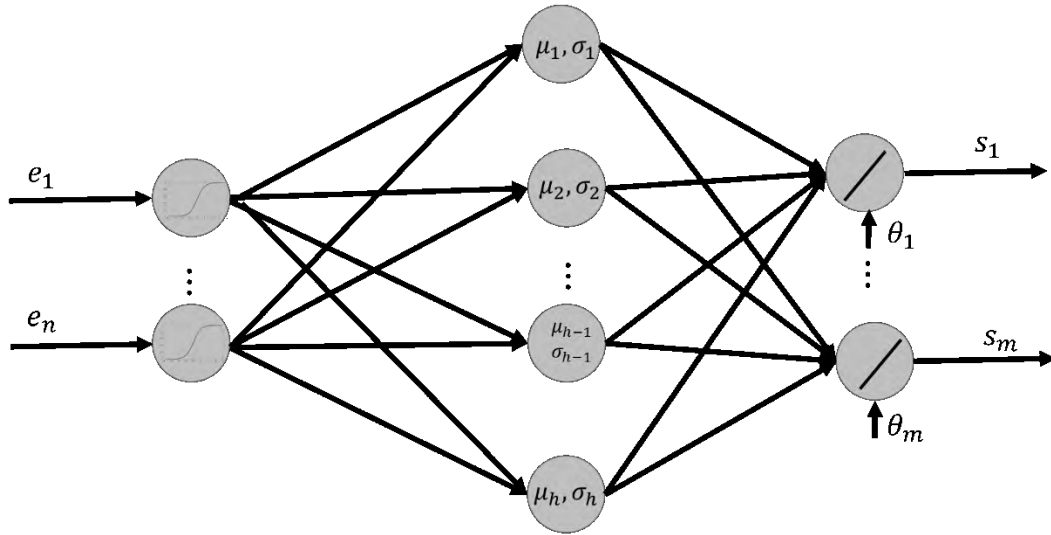


Figure IV.3 : Ensemble des paramètres d'un RBFNN.

3. Diagnostic/pronostic des fautes en utilisant les RBFNN

3.1 Diagnostic

Le diagnostic des SED en utilisant les RdN a comme objectif l'approximation des paramètres $D_\alpha(\omega)$ pour chaque ω généré par le SED (avec $|\omega| = m_d$ le nombre des événements observables pris en considération). Le paramètre m_d , calculé par l'algorithme de la figure III.3, représente la dimension (nombre des événements observables) minimale des fenêtres temporelles qu'il faut prendre en considération pour assurer l'efficacité maximale des opérations de diagnostic des SED. L'apprentissage de ce RdN nécessite une base des données déduite à partir du modèle statistique et des paramètres $D_\alpha(w_i^{m_d})$. Les lignes de cette base de données sont définies par l'équation (IV.2). Où $P_o(w_i^{m_d})$ est le vecteur composé des événements observables de la $i^{\text{ème}}$ fenêtre temporelle de modèle statistique et D_i défini par l'équation (IV.3) est le vecteur composé de la probabilité dans l'état de fonctionnement normal ($\alpha = 0$) et les probabilités d'être dans un des états de fonctionnement anormal E_α ($\alpha = 1, \dots, m$), défini par l'équation (III.22), après l'enregistrement d'une fenêtre temporelle $\omega = P_o(w_i^{m_d})$.

$$L_i = [P_o(w_i^{m_d}) \quad D_i] \quad (IV.2)$$

$$D_i = [D_0(w_i^{m_d}) \quad D_1(w_i^{m_d}) \quad \dots \quad D_m(w_i^{m_d})] \in [0,1]^{1 \times m+1} \quad (IV.3)$$

Le vecteur $P_o(w_i^{m_d})$ contient m_d éléments et il représente le vecteur d'entrée de RdN. Le vecteur D_i contient $m + 1$ éléments et il représente les valeurs que le RdN doit approximer. Donc le RdN qui va assurer le diagnostic des fautes va contenir m_d entrées est $m + 1$ sorties. La détermination des autres paramètres de ce RdN a fait l'objectif de la section IV.2.3 .

3.2 Pronostic

3.2.1 Pronostic des fautes

Le pronostic des SED en utilisant les RdN a comme objectif l'approximation des paramètres $P_\alpha(\omega)$ pour chaque ω généré par le SED (avec $|\omega| = m_p$ le nombre des évènements observables pris en considération). Le paramètre m_p calculé par l'algorithme de la figure III.3, représente la dimension (nombre des évènements observables) minimale des fenêtres temporelles qu'il faut prendre en considération pour assurer l'efficacité maximale des opérations de pronostic des fautes dans les SED. L'apprentissage de ce RdN nécessite une base des données déduite à partir du modèle statistique et des paramètres $P_\alpha(w_i^{m_d})$. Les lignes de cette base de données sont définies par l'équation (IV.5), où $P_o(w_i^{m_d})$ est le vecteur composé des évènements observables de la $i^{\text{ème}}$ fenêtre temporelle de modèle statistique et P_i défini par l'équation (IV.6) est le vecteur composé des probabilités d'être dans un des états EF_α ($\alpha = 1, \dots, m$), défini par l'équation (IV.4), dans les k fenêtres temporelles générées après l'enregistrement d'une fenêtre temporelle $\omega = P_o(w_i^{m_d})$.

$$L_i = [P_o(w_i^{m_p}) \quad P_i] \quad (IV.5)$$

$$P_i = [P_o(w_i^{m_d}) \quad P_1(w_i^{m_d}) \dots P_m(w_i^{m_d})] \in [0,1]^{1 \times m+1} \quad (IV.6)$$

Le vecteur $P_o(w_i^{m_p})$ contient m_p éléments et il représente le vecteur d'entrée de RdN. Le vecteur P_i contient $m + 1$ éléments et il représente les valeurs que le RdN doit approximer. Donc le RdN qui va prédire l'occurrence des éventuelles fautes va contenir m_d entrées est $m + 1$ sorties. La détermination des autres paramètres de ce RdN va faire l'objectif de la section IV.2.3 .

3.2.2 Prédiction de temps restant avant l'occurrence d'une faute

Lorsque le pronostic a donné comme résultat que l'occurrence d'une faute $e_f \in F_\alpha$ est probable, nous pouvons prévoir le temps restant avant l'occurrence de cette faute en utilisant un RdN qui va nous approximer les temps t_α (III.6). L'apprentissage de ce RdN nécessite une base de données que ses lignes sont décrites par l'équation (IV.7), où T_i défini par l'équation (IV.8) est le vecteur contenant les dates d'occurrence des fautes probables. Ce vecteur contient les valeurs que le RdN doit approximer. Le RdN qui va nous permettre de calculer le temps restant avant l'occurrence d'une faute probable aura m_p entrées et m sorties.

$$L_i = [P_o(w_i^{m_p}) \quad T_i] \quad (IV.7)$$

$$T_i = [t_1(w_i^{m_p}) \quad \dots \quad t_m(w_i^{m_p})] \in [0,1]^{1 \times m} \quad (IV.8)$$

Le vecteur $P_o(w_i^{m_p})$ contient m_p éléments et il représente le vecteur d'entrée de RdN. Le vecteur T_i contient m éléments et il contient les valeurs que le RdN doit approximer. Donc le RdN qui va assurer la prédiction de temps restant avant l'occurrence des fautes probables va contenir m_d entrées et m sorties. La détermination des autres paramètres de ce RdN a fait l'objectif de la section IV.2.3 .

3.3 Cas d'application

Reprenons le SED de l'exemple 4 (chapitre III) défini par les ensembles $\Sigma_o = \{e_1, e_2, e_3\}$, $\Sigma_f = \{e_{f1}, e_{f2}\}$, $\Sigma_{reg} = \{e_{r1}, e_{r2}\}$. Avec deux classes des fautes $F_1 = \{e_{f1}\}$ et $F_2 = \{e_{f2}\}$ et considérons σ_1, σ_2 et σ_3 les trois mots temporisés enregistrés lors des trois premiers cycles

$$\sigma_1 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5} e_3^{9.0}$$

$$\sigma_2 = e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}$$

$$\sigma_3 = e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5} e_3^{9.0}$$

Nous reprenons $m_d = m_p = 3$:

$$\begin{aligned} W^3 = \{ & w_1^3 = e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, w_2^3 = e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}, w_3^3 = e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}, w_4^3 \\ & = e_2^{8.0} e_1^{8.5} e_3^{9.0}, w_5^3 = e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0}, w_6^3 = e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0}, w_7^3 \\ & = e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5}, w_8^3 = e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}, w_9^3 = e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}, w_{10}^3 \\ & = e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}, w_{11}^3 = e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}, w_{12}^3 = e_2^{8.0} e_1^{8.5} e_3^{9.0} \} \end{aligned}$$

Pour chaque fenêtre temporelle de modèle statistique W^3 , nous calculons la probabilité d'occurrence d'une faute lors de l'enregistrement de cette fenêtre D_α , la probabilité de son occurrence dans les k fenêtres temporelles suivantes et la date d'occurrence d'une faute probable P_α . En regroupant toutes ces données dans le tableau IV.1, nous construisons la base de données qui peut servir à l'apprentissage des RBFNN qui vont assurer le calcul des différents paramètres de diagnostic/pronostic.

Afin de simplifier la compréhension des informations présentées dans ce tableau et le calcul des différents paramètres, nous mettons les événements de faute Σ_f en rouge, les événements non observables réguliers Σ_{reg} en vert et les fenêtres temporelles ayant les mêmes séquences des événements observables au même couleur.

Remarque 9: si $m_d \neq m_p$, il faut établir deux modèles statistiques $W^{m_d} \neq W^{m_p}$ et par la suite il faut construire deux tableaux à la place de tableau IV.1, un pour les paramètres de diagnostic dont les colonnes sont $w_i^3, P_o(w_i^3), P^{-1}(w_i^3)$ et D_i et un autre pour les paramètres de pronostic donc les colonnes sont $w_i^3, P_o(w_i^3), P^{-1}(w_i^3), P_i$ et T_i .

		w_i^3	$P_o(w_i^3)$	$P^{-1}(w_i^3)$	D_i			P_i			T_i	
					$P_0(w_i^3)$	$P_1(w_i^3)$	$P_2(w_i^3)$	$P_0(w_i^3)$	$P_1(w_i^3)$	$P_2(w_i^3)$	$t_1(w_i^3)$	$t_2(w_i^3)$
σ_1	$i = 1$	$e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}$	$e_2^1 e_3^{2.1} e_1^{5.0}$	w_1^3, w_9^3	$\frac{1}{2} = 0.5$	0	$\frac{1}{2} = 0.5$	0	$\frac{2}{2} = 1$	0	6.3	-1
	$i = 2$	$e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}$	$e_3^{2.1} e_1^{5.0} e_2^{8.0}$	w_2^3, w_{10}^3	0	$\frac{2}{2} = 1$	0	0	$\frac{2}{2} = 1$	0	6.3	-1
	$i = 3$	$e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}$	$e_1^{5.0} e_2^{8.0} e_1^{8.5}$	w_3^3, w_{11}^3	0	$\frac{2}{2} = 1$	0	$\frac{2}{2} = 1$	0	0	-1	-1
	$i = 4$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	$\frac{4}{4} = 1$	0	0	$\frac{3}{4} = 0.75$	$\frac{1}{4} = 0.25$	0	6.3	-1
σ_2	$i = 5$	$e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0}$	$e_2^1 e_1^{2.1} e_3^{4.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	$\frac{4}{4} = 1$	0	0	$\frac{3}{4} = 0.75$	$\frac{1}{4} = 0.25$	0	6.3	-1
	$i = 6$	$e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0}$	$e_1^{2.1} e_3^{4.0} e_2^{7.0}$	w_6^3	$\frac{1}{1} = 1$	0	0	1	0	0	-1	-1
	$i = 7$	$e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5}$	$e_3^{4.0} e_2^{7.0} e_1^{7.5}$	w_7^3	$\frac{1}{1} = 1$	0	0	0	1	0	6.3	-1
	$i = 8$	$e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}$	$e_2^{7.0} e_1^{7.5} e_3^{9.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	$\frac{4}{4} = 1$	0	0	$\frac{3}{4} = 0.75$	$\frac{1}{4} = 0.25$	0	6.3	-1
σ_3	$i = 9$	$e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}$	$e_2^1 e_3^{2.1} e_1^{5.0}$	w_1^3, w_9^3	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{2}{2} = 1$	0	6.3	-1
	$i = 10$	$e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}$	$e_3^{2.1} e_1^{5.0} e_2^{8.0}$	w_2^3, w_{10}^3	0	$\frac{2}{2} = 1$	0	0	$\frac{2}{2} = 1$	0	6.3	-1
	$i = 11$	$e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}$	$e_1^{5.0} e_2^{8.0} e_1^{8.5}$	w_3^3, w_{11}^3	0	$\frac{2}{2} = 1$	0	$\frac{2}{2} = 1$	0	0	-1	-1
	$i = 12$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	$\frac{4}{4} = 1$	0	0	$\frac{3}{4} = 0.75$	$\frac{1}{4} = 0.25$	0	6.3	-1

Tableau IV.1 : Exemple de construction d'une base de données pour l'apprentissage des trois RdN décrits dans les sections IV.1.4 et IV.1.5.

3.4 Fonction de marquage

Dans des cas bien spécifiques où nous voulons juste déterminer le mode de fonctionnement le plus probable et où les probabilités ne sont pas nécessaires, les problèmes des sections 3.1 et 3.2.1 peuvent être simplifié en passant d'un problème d'interpolation qui cherche à approximer les paramètres D_α et P_α à un problème de classification qui classifie le mode de fonctionnement de SED, traduit par ses fenêtres temporelles générées, au mode de fonctionnement le plus probable. Nous définissons la fonction de marquage (IV.9) qui assigne pour chaque fenêtre temporelle du modèle statistique 1 à la classe qui représente le mode de fonctionnement le plus probable et assigne 0 aux autres classes.

Remarque 10 : Si deux classes ont la même probabilité et cette probabilité représente la probabilité maximale, nous favorisons la classe qui représente le mode de fonctionnement le plus critique. Sinon nous favorisons l'état de fonctionnement normal.

$$F_\alpha^{\mathcal{P}}(\omega) = \begin{cases} 1, & \mathcal{P}_\alpha(\omega) = \underset{\alpha=0,1\dots m}{Max} \{ \mathcal{P}_\alpha(\omega) \} \\ 0, & else \end{cases} \quad (IV.9)$$

Où $\mathcal{P}_\alpha(\omega)$ peut être $D_\alpha(\omega)$ dans le cas de diagnostic ou $P_\alpha(\omega)$ dans le cas de pronostic.

Ainsi les vecteurs D_i et P_i des équations (IV.2) et (IV.3) deviennent :

$$D_i = [F_0^D(w_i^{m_d}) \quad \dots \quad F_m^D(w_i^{m_d})] \in \{0,1\}^{1 \times (m+1)} \quad (IV.10)$$

$$P_i = [F_0^P(w_i^{m_d}) \quad \dots \quad F_m^P(w_i^{m_d})] \in \{0,1\}^{1 \times (m+1)} \quad (IV.11)$$

Exemple 11 :

Dans le tableau IV.2, nous utilisons la fonction de marquage IV.5 pour convertir le problème de cas d'application présenté dans la section 03.3 d'un problème d'interpolation qui essaie de calculer les paramètres de diagnostic/pronostic à un problème de classification qui classifie les fenêtres temporelles générées par le SED a ceux qui représentent l'état de fonctionnement normal et ceux qui représentent un des états de fonctionnement anormaux. Ceci rend le problème moins complexe.

		w_i^3	$P_0(w_i^3)$	$P^{-1}(w_i^3)$	D_i			P_i		
					$P_0(w_i^3)$	$P_1(w_i^3)$	$P_2(w_i^3)$	$P_0(w_i^3)$	$P_1(w_i^3)$	$P_2(w_i^3)$
σ_1	$i = 1$	$e_2^1 e_{f2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}$	$e_2^1 e_3^{2.1} e_1^{5.0}$	w_1^3, w_9^3	0	0	1	0	1	0
	$i = 2$	$e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}$	$e_3^{2.1} e_1^{5.0} e_2^{8.0}$	w_2^3, w_{10}^3	0	1	0	0	1	0
	$i = 3$	$e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}$	$e_1^{5.0} e_2^{8.0} e_1^{8.5}$	w_3^3, w_{11}^3	0	1	0	1	0	0
	$i = 4$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	1	0	0	1	0	0
σ_2	$i = 5$	$e_2^1 e_1^{2.1} e_{r1}^{3.0} e_3^{4.0}$	$e_2^1 e_1^{2.1} e_3^{4.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	1	0	0	1	0	0
	$i = 6$	$e_1^{2.1} e_{r1}^{3.0} e_3^{4.0} e_{r2}^{5.3} e_2^{7.0}$	$e_1^{2.1} e_3^{4.0} e_2^{7.0}$	w_6^3	1	0	0	1	0	0
	$i = 7$	$e_3^{4.0} e_{r2}^{5.3} e_2^{7.0} e_1^{7.5}$	$e_3^{4.0} e_2^{7.0} e_1^{7.5}$	w_7^3	1	0	0	0	1	0
	$i = 8$	$e_2^{7.0} e_1^{7.5} e_{r1}^{8.0} e_3^{9.0}$	$e_2^{7.0} e_1^{7.5} e_3^{9.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	1	0	0	1	0	0
σ_3	$i = 9$	$e_2^1 e_{r2}^{1.5} e_3^{2.1} e_{r1}^{4.0} e_1^{5.0}$	$e_2^1 e_3^{2.1} e_1^{5.0}$	w_1^3, w_9^3	0	0	1	0	1	0
	$i = 10$	$e_3^{2.1} e_{r1}^{4.0} e_1^{5.0} e_{f1}^{6.3} e_2^{8.0}$	$e_3^{2.1} e_1^{5.0} e_2^{8.0}$	w_2^3, w_{10}^3	0	1	0	0	1	0
	$i = 11$	$e_1^{5.0} e_{f1}^{6.3} e_2^{8.0} e_1^{8.5}$	$e_1^{5.0} e_2^{8.0} e_1^{8.5}$	w_3^3, w_{11}^3	0	1	0	1	0	0
	$i = 12$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$e_2^{8.0} e_1^{8.5} e_3^{9.0}$	$w_4^3, w_5^3, w_8^3, w_{12}^3$	1	0	0	1	0	0

Tableau IV.2: Conversion de problème de l'exemple 1 à un problème de classification.

4. Vérification du respect des contraintes temporelles

4.1 Vérification en temps réel des CT et fonction de marquage

La construction du vecteur complet des CT propres à tous les événements du cycle de fonctionnement nécessite l'accomplissement du cycle en cours afin de prendre en considération les dates d'occurrences de tous ses événements. Ceci ne permet de vérifier les CT qu'à la fin de chaque cycle ce qui supprime l'aspect « temps réel » de la vérification des CT. Afin de pouvoir construire un RdN capable de vérifier en temps réel le degré de respect des CT par un SED, nous définissons une fonction de marquage qui permettra de construire le vecteur d'entrée à n'importe quel moment t de fonctionnement de SED (cette date t est calculée à partir de l'instant de l'exécution du dernier événement). Nous supposons que chaque événement e_i va se produire à sa date normale (calculé à partir de l'instant de l'exécution de son événement antérieur e_{i-1}) tant qu'il ne s'est pas encore produit et la durée t n'a pas encore excédé sa valeur maximale $t_{i,max}$. La fonction de marquage (IV.12) assigne à chaque événement e_i de σ_j sa date d'occurrence t_i s'il s'est déjà produit et il lui assigne sa date d'occurrence normale $t_{i,moy}$ s'il n'a pas encore produit et la durée t , calculée à partir de l'exécution de l'événement e_{i-1} , si t n'a pas encore dépassé la date maximale représentée dans $CT(e_i)$. La date d'occurrence normale d'un événement e_i est la moyenne de toutes les dates de son occurrence existante dans le modèle statistique.

$$\mathcal{M}(e_i) = \begin{cases} t_i & \text{si } occur(e_i) = 1 \\ t_{i,moy} & \text{si } occur(e_i) = 0 \text{ and } t < t_{i,max} \\ t & \text{si } occur(e_i) = 0 \text{ and } t > t_{i,max} \end{cases} \quad (IV.12)$$

Où $occur(e_i) = 1$ si l'événement e_i s'est déjà produit et $occur(e_i) = 0$ si l'événement e_i ne s'est pas encore produit. Nous définissons aussi la fonction de marquage qui prend la valeur 1 si une CT a été respecté et 0 dans le cas contraire.

$$\widehat{CT}(e_i) = \begin{cases} 1 & \text{si } t_i \in CT(e_i) \\ 0 & \text{sinon} \end{cases} \quad (IV.13)$$

Les lignes de la base d'apprentissage de RdN qui va assurer la vérification des CT sont définies par (IV.14), telle que $E = [\mathcal{M}(e_i)]_{i=1,...,n}$ est le vecteur d'entrée de RdN sera comme dans l'équation (IV.12). Donc, on aura un RdN avec n entrées et n sorties. Chaque sortie i indique si la CT associée à l'événement e_i est respectée ou non.

$$L_i = [E \quad \widehat{CT}] \quad (IV.14)$$

Remarque 11 : pour les SED où la cadence de fabrication est réglable, la durée de certaines taches peut être variable en fonction de la cadence. Donc, en plus des n éléments de vecteur d'entrée nous pouvons ajouter une entrée supplémentaire E_{n+1} qui prend en charge la variation de la cadence.

4.2 Apprentissage des RBFNN pour vérifier les CT

Chaque neurone de la couche cachée d'un RBFNN représente une mémoire statique qui mémorise les informations en relation avec un des modes de fonctionnement ou une partie de ces informations (souvent un mode de fonctionnement est représenté par plusieurs neurones dans la couche cachée). Ces informations sont traduites avec des couples de type (centre, écart-type). La fonction d'activation de ces neurones est la fonction de Gausse (II.4) qui est une fonction radiale qui donne des réponses en fonction de distance euclidienne entre le vecteur d'entrée et le centre de ce neurone qui peut être calculé dans le cas des CT (III.23) en utilisant $\frac{t_{min}+t_{max}}{2}$. Chaque intervalle lié à une CT peut être représenté un neurone de la couche cachée. Lorsqu'un vecteur d'entrée est représenté au RBFNN chaque neurone de la couche cachée doit donner le degré de respect de CT propres à son mode de fonctionnement.

Exemple 12 :

Nous reprenons le SED de l'exemple 1 qui peut fonctionner selon deux différentes cadences C_1 et C_2 . Lors de chaque cycle de fonctionnement ce SED génère le mot temporisé des événements observables suivants :

$$\sigma = e_{p0}^{t1} e_{pA}^{t2} e_{pB}^{t3} e_{pC}^{t4} e_{pB}^{t5} e_{pA}^{t6} e_{p0}^{t7}$$

Le tableau IV.3 donne les informations propres aux CT des différents événements de mot temporisé σ . Les données de ce tableau peuvent être traduites par les durées des différents déplacements, représenté dans la figure IV.4 que le chariot peut effectuer lors de l'enregistrement de σ .

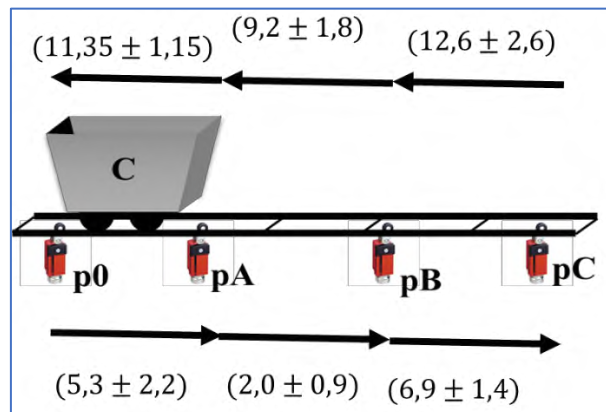


Figure IV.4: les durées des différents déplacements de chariot

e_i	CT(e_i)		Centre	Rayon d'influence ¹
$e_1 = e_{p0}$	[2.2 , 5.8]		4	1.8
$e_2 = e_{pA}$	[3.1 , 7.5]		5.3	2.2
$e_3 = e_{pB}$	[1.1 , 2.9]		2	0.9
$e_4 = e_{pC^2}$	C_1	[2.0 , 6.0]	4	2
	C_2	[4.0 , 9.8]	6.9	1.4
$e_5 = e_{pB}$	[10.0, 15.2]		12.6	2.6
$e_6 = e_{pA}$	[7.4 , 11.0]		9.2	1.8
$e_7 = e_{p0}$	[10.2 , 12.5]		11.35	1.15

Tableau IV.3: Des informations sur les CT des différents événements de σ

Supposons que ce SED fonctionne selon la cadence C_2 et a généré le mot temporisé des événements observables suivants :

$$\tilde{\sigma} = e_{p0}^{3.3} e_{pA}^{5.3} e_{pB}^{2.2}$$

À l'instant de la génération du dernier événement e_{pB} par le SED nous pouvons faire un diagnostic/pronostic en temps réel en prenant comme vecteur d'entrée de RdN le vecteur suivant :

$$[3.3 \quad 5.3 \quad 2.2 \quad 6.9 \quad 12.6 \quad 9.2 \quad 11.35]$$

Où les trois premières valeurs sont les dates d'occurrence des événements de $\tilde{\sigma}$ et les autres valeurs sont les centres des CT propres aux derniers quatre événements pris à partir de tableau IV.3.

5. Conclusion

Dans ce chapitre, les bases de données qui servent à l'apprentissage des RdN ont été construites en utilisant les résultats des approches développées dans le chapitre III. Ainsi, la capacité d'apprentissage et de généralisation des RdN a été utilisée pour mettre en place un diagnostiqueur, un pronostiqueur, un RdN qui peut prédire le temps restant avant l'occurrence d'une faute probable et un RdN qui vérifie le degré de respect des CT par le SED. Ceux-ci ont permis de synthétiser les bases de données énormes qui peuvent résulter de modèle statistique. Ainsi, le temps des opérations de diagnostic/pronostic sera réduit et leurs précisions seront améliorées.

Dans le chapitre suivant, nous essaierons de mettre en pratique l'ensemble des approches développées le long de cette thèse en développant une application informatique capable de réaliser l'ensemble des opérations de diagnostic/pronostic d'un SED.

¹ Le rayon d'influence est calculé en utilisant la formule $\frac{t_{max}-t_{min}}{2}$

² La CT de l'événement e_4 change en fonction de la cadence de fonctionnement de SED.

À la suite de ce travail présenté dans ce chapitre, nous avons l'intention de développer dans nos futurs travaux une nouvelle approche d'apprentissage des RbN qui rend les RdN plus rapide, plus efficace et qui peut assurer une meilleure précision.

Chapitre V:

Mise en application de notre approche

1. Introduction

Afin de mettre en évidence la pertinence de notre approche nous visons, dans ce chapitre, développer une application informatique qui permettra d'assurer les différentes opérations de diagnostic/pronostic. Cette application va permettre de construire le modèle statistique du SED, le calcul des différents paramètres de diagnostic/pronostic et la construction des RdN qui vont permettre de réaliser un diagnostiqueur/pronostiqueur. Le développement de cette application sera fait sous langage de programmation Python vu le nombre des avantages que possède ce langage de programmation. À la fin de ce chapitre, un test d'application sera présenté, dans lequel nous utilisons l'application développée pour assurer le diagnostic/pronostic en temps réel d'un SED. Ensuite cette application est utilisée pour la mise en place de notre approche à un cas d'application réel.

2. Mise en place de l'approche développée dans un SED

Comme montre figure V.1, la mise en place de l'approche développée dans les sections précédentes de cette thèse dans un SED peut être faite en 4 étapes :

Étape 1 : Initialisation et collecte des données qui consiste à :

- La définition des éléments des différents ensembles des événements que le SED peut générer à savoir : l'ensemble des événements observables, l'ensemble des événements de faute, l'ensemble des événements non observables réguliers et aussi la définition des différentes classes des fautes ;
- La collecte des différents mots temporisés enregistrés lors des différents cycles et lors des différents modes de fonctionnement.

Étape 2 : Construction du modèle statistique qui consiste à :

- Le formatage des données qui consiste à calculer les dimensions permettant de maximiser l'efficacité des opérations de diagnostic/pronostic et ensuite transformer les mots temporisés collectés du SED sous la forme des fenêtres temporelles.
- Le calcul des paramètres qui traduisent la probabilité d'occurrence d'une faute, la probabilité de son occurrence dans le futur et le temps restant avant l'occurrence d'une faute probable.

Étape 3 : la construction des RdN, qui consiste à :

- Construire, à partir du modèle statistique et des paramètres calculés lors de l'étape 2, les bases de données qui vont servir à l'apprentissage ;
- Utiliser ces bases de données pour l'apprentissage des différents RBFNN qui vont assurer les différentes opérations de diagnostic/pronostic.

Étape 4 : En-ligne diagnostic/pronostic qui consiste à donner en temps réel les probabilités D_α et P_α de chaque classe des fautes d'occurrence d'une faute et à vérifier le degré du respect des CT par ce SED à la suite de l'occurrence de n'importe quel évènement observable dans ce SED.

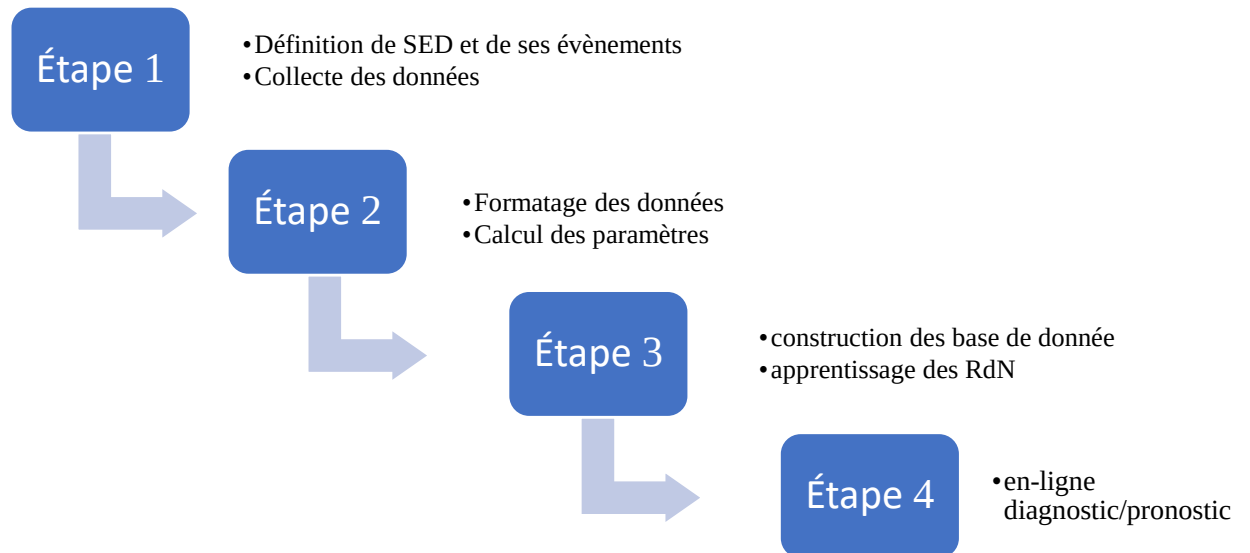


Figure V.1: Les étapes de la mise en place de l'approche diagnostic/pronostic des SED développée dans cette thèse.

La mise en place de ces étapes nécessite la programmation d'une application informatique, que nous pouvons décrire par l'algorithme de la figure V.2, qui sera capable d'assurer d'une façon automatique les différentes opérations de diagnostic/pronostic de n'importe quel SED. Il y'a plusieurs langages de programmation qui permettent la mise en œuvre d'une telle application. Dans la section suivante, nous essaierons de choisir le langage de programmation le plus adéquat à la mise en place de notre approche de diagnostic/pronostic.

```

charger_InfoData_SED()
mode <= lire('choisissez un mode')
Si mode == « collecte des données »:
    Modbus.connecter()
    Modbus.readInputs()
    tantque event.disponible()
        Save_event()
        Save_date_event()
    dataBase.save()
    Modbus.deconnecter()
Si mode == « calcul des paramètres »:
    klim <= lire('Donner le nombre des évènements maximal: ')
    md, mp <= algorithmme(klim, dataBase)
    formater_data(dataBase, md)
    calculer_diag_parameters(md, dataBase)
    data_diagnostic.save()
    formater_data(dataBase, mp)
    calculer_prog_parameters(mp, dataBase)
    data_pronostic.save()
Si mode == « apprentissage des réseaux de neurones » :
    Appr_diag_rbfnn(md, nbr_classe, data_diagnostic)
    Appr_prog_rbfnn(mp, nbr_classe, data_pronostic)
    pour tout type_cycle
        Appr_CT_rbfnn(nbr_event, nbr_classe, dataBase)
Si mode == en-ligne:
    Modbus.connecter()
    Modbus.readInputs()
    tantque event.disponible()
        Save_event()
        Save_date_event()
        Si ndr_recorded_event() >= md
            diag_rbfnn(derniere_md_event)
        Si ndr_recorded_event >= mp
            prog_rbfnn(derniere_mp_event)
        construire_CT_vecteur()
        Pour chaque type_cycle
            CT_rbfnn(CT_vecteur, type_cycle)
        Afficher_resultats()
    Modbus.deconnecter()

```

Figure V.2: Algorithme de l'application informatique.

3. Le langage de programmation : le Python

Dans la littérature, il existe un très grand nombre des langages de programmation qui peut servir au développement de notre application pour assurer le diagnostic/pronostic des SED. Vu ce nombre important qui ne cesse d'augmenter, il est intéressant de mettre des critères de choix du langage de programmation le plus approprié à notre application, un langage de programmation peut être choisi selon plusieurs critères que nous résumons comme suit :

- La disponibilité de la documentation et des codes « Open source » ;
- La facilité de la syntaxe et la lisibilité de code ;
- La portabilité : la capacité de pouvoir s'exécuter sur différent système d'exploitation et sur différent environnements matériels sans la nécessité de connaître ou d'installer des composants logiciels ou matériels propres à ce langage de programmation ;
- La stabilité : la capacité de résister aux changements que peut avoir le système d'exploitation ou l'environnement matériel ;
- Le coût qui englobe l'achat de la licence et les coûts de maintenance.

La majorité de ces langages de programmation présentent deux inconvénients majeurs qui rendent leurs utilisations un peu délicates. Ces inconvénients sont :

- Ils nécessitent l'achat de licence.
- Ils ne sont pas portables ou bien ils sont liés à des systèmes d'exploitation spécifiques (comme Windows) et ne sont pas fonctionnels sur les autres systèmes d'exploitation.

Cependant, le langage de programmation Python présente beaucoup d'avantages que nous pouvons résumer comme suit :

- L'interpréteur Python est un logiciel gratuit que son utilisation ne demande pas l'achat d'une licence ;
- Le code Python a une structure basée sur une syntaxe très facile et organisée avec des tabulations ce qui rend le code très lisible ;
- Les scripts écrits sous Python sont portables et peuvent fonctionner sur n'importe quel système d'exploitation (Linux, Windows, MAC...) et ils sont même fonctionnels sur des cartes embarquées (comme Raspberry) ;
- Le langage Python dispose d'une large bibliothèque des packages (Numpy, Matplotlib, Openpyxl...) permettant le traitement, l'analyse, la simulation et l'affichage des différents types de données ;
- Le couplage de Python avec les autres logiciels des calculs mathématique (comme Matlab) est très simple ce qui rend l'utilisation de Python très favorable pour le développement des solutions

scientifiques surtout dans le domaine de l'intelligence artificielle qui demande beaucoup de calcul mathématique ;

- Python dispose d'une large communauté des utilisateurs qui développe des solutions scientifiques très puissantes ;
- La possibilité de générer des fichiers exécutables permet l'utilisation des scripts conçus sous Python sans une connaissance préalable de ce langage de programmation. Ceci rend ces scripts utilisables par n'importe quel utilisateur.

Dans le tableau V.1 nous classerons les dix langages de programmation les plus populaires selon le taux de leurs utilisations par les développeurs pour la conception des applications de bureau et des applications scientifiques lors de l'année 2018. Ce classement est tiré d'une application publiée par « IEEE Spectrum »³ et il représente les dix langages de programmation les plus convenables pour le développement de notre application informatique.

Langage de programmation	Utilisation scientifique (popularité)	Marché de travail	Open source	Tendance
Python	1	1	1	1
C++	2	4	2	2
Java	3	2	3	3
C	4	3	4	4
C#	5	5	5	7
R	6	9	8	9
Go	7	-	6	5
Matlab	8	8	-	-
Ruby	9	7	9	6
Shell	10	6	7	8

Tableau V.1 : Classement des langages de programmation utilisés pour le développement des applications scientifiques

Ce classement a pris en considération quatre critères qui sont :

- La popularité traduite par le taux des développeurs qui utilisent ce langage pour le développement des applications de bureau et les applications scientifiques ;
- La disponibilité des codes préprogrammés et des bibliothèques open source sur internet ;
- La tendance traduit par le changement du taux d'accroissement de langage entre l'année actuelle et l'année précédente.

³ IEEE Spectrum est un magazine phare et site Web de l'IEEE qui informe ses membres des principales tendances et évolutions en matière de technologie.

Emploi : le langage le plus demandé dans le marché de travail.

Ce classement a prouvé, en plus des avantages cités, que le langage **Python** est le plus adéquat pour le développement de notre application informatique.

4. Programmation de l'application

Cette application, que nous présentons son algorithme dans la figure V.3 comporte quatre modes de fonctionnement :

- Mode collecte des données : Dans ce mode, un réseau de communication est créé entre les automates qui commandent le SED et l'ordinateur sur lequel est installé cette l'application. À travers ce réseau l'application accède aux différentes données des automates (l'état des entrées, l'état des sorties, l'état des registres, l'état des variables internes, l'état des conteurs...) chaque changement de ces données constitue un évènement que nous devons les enregistrer. Ces évènements avec leurs dates d'occurrence enregistrées lors des différents modes de fonctionnement sous forme de mots temporisés.

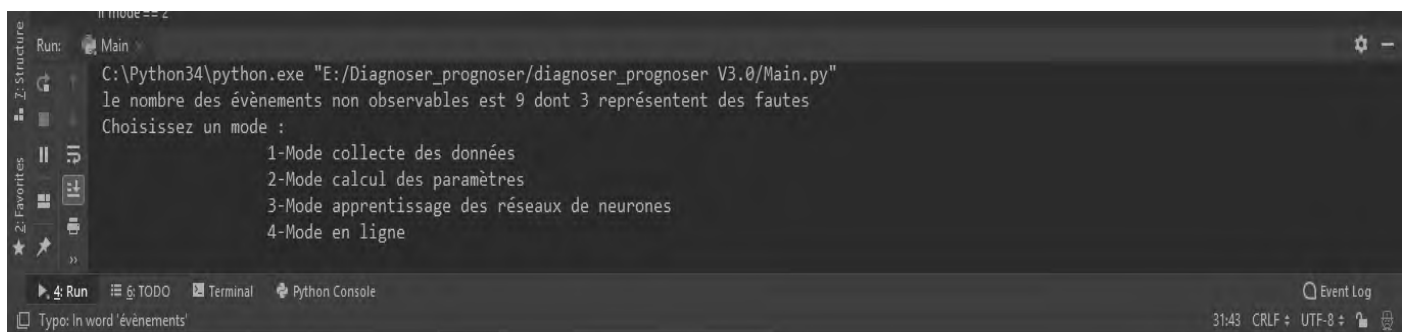


Figure V.3: Capture d'écran de l'interface principale de l'application.

- Mode formatage des données (figure V.4): Dans ce mode, les fenêtres temporelles sont construites à partir des mots temporisés colletés lors du mode « collecte des données ». Lors de cette phase, l'algorithme présenté dans la figure III.3 est utilisé pour maximiser l'efficacité des opérations de diagnostic/pronostic.

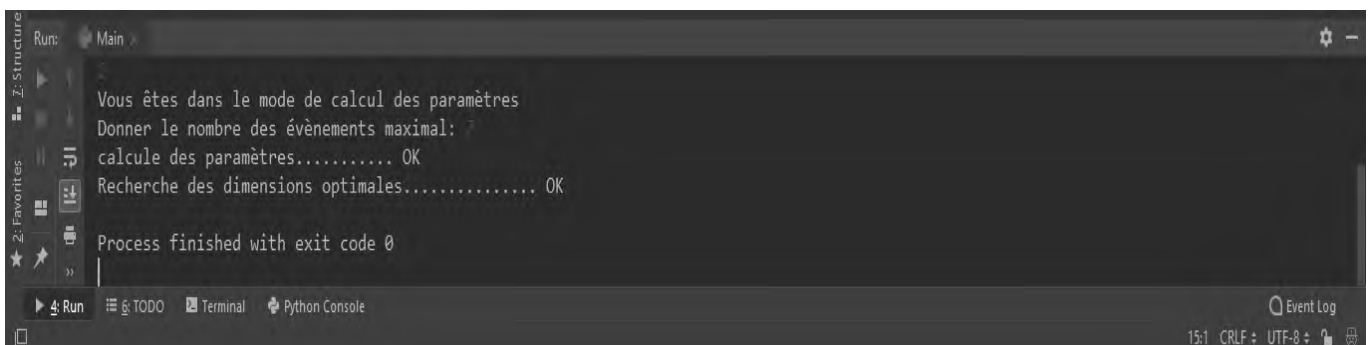


Figure V.4 : Capture d'écran de mode de calcul des paramètres de diagnostic/pronostic.

- Mode apprentissage des RdN (figure V.5) : Dans ce mode, les données collectées et formatées sous forme des fenêtres temporelles, ainsi que les paramètres calculés lors du deuxième mode, sont utilisées pour l'apprentissage des RdN. Lors de ce mode, nous utilisons le logiciel de calcul mathématique MATLAB pour la construction des RBFNN en utilisant l'algorithme de la rétropropagation du gradient. Dans ce mode, nous cherchons à optimiser l'architecture des RBFNN en changeant à chaque fois le nombre des neurones dans la couche cachée jusqu'à l'obtention de l'erreur minimale.

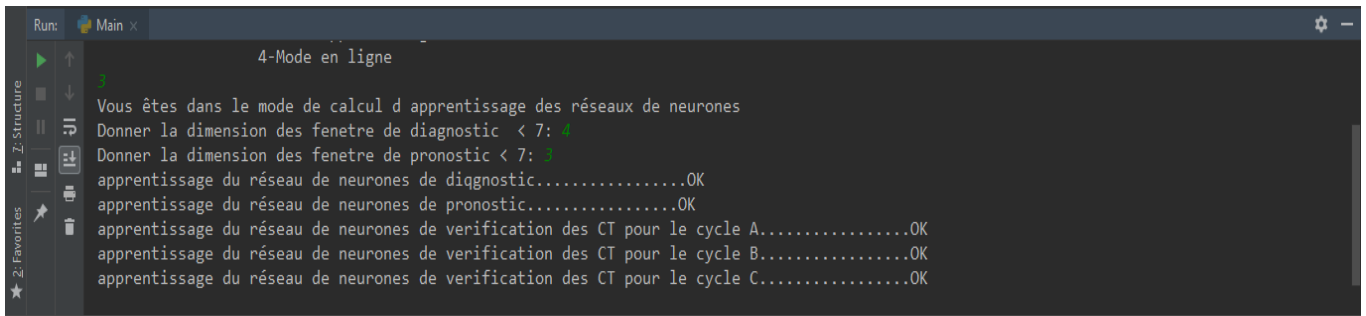


Figure V.5 : Capture d'écran de mode d'apprentissage des RBFNN.

- Mode en-ligne : Dans ce mode, le réseau de communication est utilisé pour collecter les fenêtres temporelles générées par le SED en temps réel. Après les RBFNN construits lors du troisième mode sont utilisés pour traiter ces fenêtres temporelles et ainsi effectuer les opérations de diagnostic/pronostic en temps réel des SED.

Chacun de ces modes nécessite obligatoirement que tous les modes qui le précèdent soient déjà exécutés au moins une seule fois.

5. Cas d'application

5.1 Description de système

Dans l'objectif de montrer la pertinence de l'approche de diagnostic/pronostic des SED que nous avons développé le long de cette thèse, nous appliquons cette approche, dans cette partie, un cas d'application. Nous considérons la maquette d'essai d'un chariot suspendu (figure V.6) disponible dans le laboratoire des techniques industrielles à la faculté des sciences et techniques FÈS, USMBA Maroc.

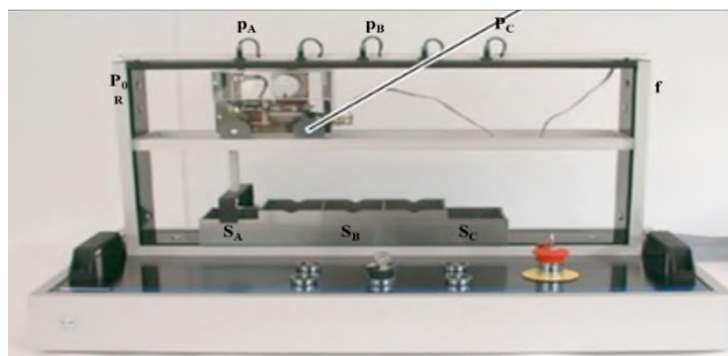


Figure V.6 : chariot suspendu.

La fonction principale de ce chariot est le transfert de trois types de produits A, B et C de la position de chargement P_0 aux stocks S_A , S_B ou S_C . un lecteur code-barres fait la distinction entre les trois produits considérés.

Le trajet de chariot est composé de trois zones notées 0A (entre les positions P_0 et P_A), AB (entre les positions P_A et P_B) et BC (entre les positions P_B et P_C). Dans chacune d'elles, différentes opérations manuelles pourraient être envisagées. Le produit A est soumis à des opérations dans la zone 0A et ensuite il est déposé dans le stock S_A . Le produit B est soumis à des opérations dans les zones 0A et AB et ensuite il est déposé dans le stock S_B . Le produit C est soumis aux opérations dans les zones 0A, AB et BC et ensuite il est déposé dans le stock S_C . ces opérations sont assurées par des déplacements horizontaux et verticaux de chariot que nous illustrons par le grafctet de la figure V.7.

Le tableau V.2 donne une description de tous les événements possibles que le chariot peut générer. Nous aurons les ensembles suivants : $\Sigma_o = \{e_{p0}, e_{pA}, e_{pB}, e_{pC}, e_A, e_B, e_C, e_{SA}, e_{SB}, e_{SC}, e_l\}$, $\Sigma_{reg} = \{e_{0A}, e_{AB}, e_{BC}\}$ et $\Sigma_f = \{e_{f0}, e_{f1}, e_{f2}\}$. Nous considérons aussi deux classes de fautes $F_1 = \{e_{f0}, e_{f1}\}$ et $F_2 = \{e_{f2}\}$.

Observable	Position	e_{p0}	Chariot dans la position de chargement
		e_{pA}	Chariot dans la position A
		e_{pB}	Chariot dans la position B
		e_{pC}	Chariot dans la position C
		e_l	Chariot dans la position limite
	Types de produit	e_A	Chargement de chariot par un produit de type A
		e_B	Chargement de chariot par un produit de type B
		e_C	Chargement de chariot par un produit de type C
	Stocks	e_{SA}	Produit déposé dans le stock A
		e_{SB}	Produit déposé dans le stock B
		e_{SC}	Produit déposé dans le stock C
Non observable	Fautes	e_{f0}	Une action erronée
		e_{f1}	Un produit dans le mauvais stock
		e_{f2}	Capteur en défaillance
	Opérations régulières	e_{0A}	Les opérations manuelles dans la zone 0A sont terminées
		e_{AB}	Les opérations manuelles dans la zone AB sont terminées
		e_{BC}	Les opérations manuelles dans la zone 0A sont terminées

Tableau V.2: Les événements générés par le chariot suspendu.

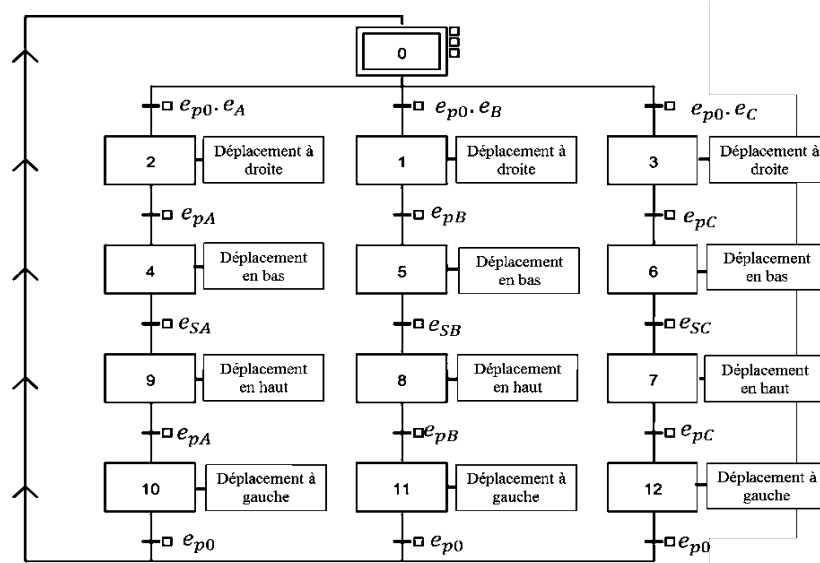


Figure V.7: Graphe de fonctionnement de chariot suspendu.

5.2 Contraintes temporelles

Le chariot suspendu, et selon le type de produit détecté, peut exécuter un des cycles de fonctionnement suivants :

- Produit A : lorsqu'un produit de type A est détecté dans la position de chargement il est transféré vers le stock S_A . Ceci donne le mot temporisé des événements observables suivant : $\sigma_A = e_{p0}^{t1} e_A^{t2} e_{pA}^{t3} e_{SA}^{t4} e_{pA}^{t5} e_{p0}^{t6}$.
- Produit B : lorsqu'un produit de type B est détecté dans la position de chargement, il est transféré vers le stock S_B . Ceci donne le mot temporisé des événements observables suivant : $\sigma_B = e_{p0}^{t1} e_B^{t2} e_{pA}^{t3} e_{pB}^{t4} e_{SB}^{t5} e_{pB}^{t6} e_{pA}^{t7} e_{p0}^{t8}$.
- Produit C : lorsqu'un produit de type C est détecté dans la position de chargement, il est transféré vers le stock S_C . Ceci donne le mot temporisé des événements observables suivant : $\sigma_C = e_{p0}^{t1} e_C^{t2} e_{pA}^{t3} e_{pB}^{t4} e_{pC}^{t5} e_{SC}^{t6} e_{pC}^{t7} e_{pB}^{t8} e_{pA}^{t9} e_{p0}^{t10}$.

Remarque 12 : plusieurs événements de ces mots temporisés n'apparaissent pas dans le graphe de fonctionnement de la figure V.7.

Les dates des occurrences possibles de chaque événement de ces trois mots temporisés sont regroupées dans des bases de données qui contiennent l'enregistrement de ces mots dans les états de fonctionnement normal et anormal de chariot suspendu. Dans ce travail, nous considérons trois types de fautes qui peuvent résulter d'un non-respect des CT, qui sont :

- Faute 1 (F1) : le lecteur code-barres en défaillance ou il met beaucoup de temps pour détecter le type de produit. Cette faute peut être détectée par un retard d'exécution de e_A , e_B ou e_C .

- Faute 2 (F2) : le moteur qui assure le déplacement horizontal est très rapide ou très lent. Cette faute peut être détectée par une exécution d'un événement qui reflète la position horizontale de chariot (e_{pA} , e_{pB} ou e_{pC}) en dehors de leurs dates appropriées.
- Faute 3 (F3) : le moteur qui assure le déplacement vertical est très rapide ou très lent. Cette faute peut être détectée par une exécution d'un événement qui reflète la position verticale de chariot (e_{SA} , e_{SB} ou e_{SC}) en dehors de leurs dates appropriées.

5.3 Collecte des données

La collecte des données peut être faite en utilisant un réseau Modbus, comme le montre la figure V.8, qui va permettre à l'ordinateur, sur lequel est installée notre application qui va assurer les différentes opérations de diagnostic/pronostic, de lire les données existantes sur les différents équipements connectés au réseau.

Le chariot suspendu, qui fait l'objet de cette étude de cas, est commandé par un automate programmable (API) de type TSX micro 3705 de Schneider, les pilotes de cet automate et permettant à l'ordinateur de reconnaître cet API afin de construire un réseau Modbus assurant la communication entre cet automate et l'ordinateur n'est fonctionnelle que sous Windows XP. Cependant, notre application nécessite des bibliothèques et des fichiers qui ne sont disponibles que sur les versions les plus récentes de Windows (Windows 8 et 10).

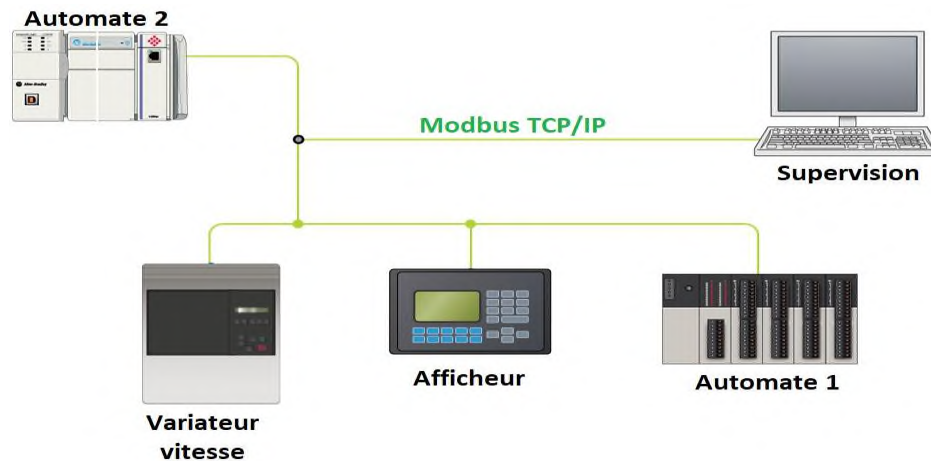


Figure V.8: Réseau Modbus entre l'ordinateur et d'autres équipements industriels.

Cette incompatibilité logicielle nous a obligés à concevoir un modèle qui va simuler le fonctionnement de ce chariot suspendu. À partir d'une base de données conçue manuellement, ce modèle va générer les événements que le SED peut générer dans les différents modes de fonctionnement et ainsi les opérations de diagnostic/pronostic peuvent être mises en place.

5.4 Calcul des paramètres de diagnostic/pronostic et formatage des données

Le SED est décrit par une base de données qui contient tous les mots temporisés enregistrés lors de tous les cycles de fonctionnement possibles. L'application de l'algorithme de la figure III.3, développée dans

la section précédente de cette thèse, donne les résultats résumés dans le tableau 1. Nous pouvons remarquer, et en prenant en considération qu'une utilisation d'une grande dimension rend l'apprentissage des RBFNN plus complexe, qu'en utilisant la dimension $m_d = 4$ les degrés de diagnosticabilité $F_D^{m_d}$ atteignent des valeurs acceptables. Cependant, l'utilisation d'une dimension m_p supérieure à 3 rend la classe des fautes F_3 non prévisible. Donc nous adopterons $m_d = 4$ pour le diagnostic et $m_p = 3$ pour le pronostic.

	$\hat{D}_0^k$	$\hat{D}_1^k$	$\hat{D}_2^k$	$\hat{D}_3^k$	F_D^k	$\hat{P}_0^k$	$\hat{P}_1^k$	$\hat{P}_2^k$	$\hat{P}_3^k$	F_P^k
k = 2	83,56%	37,50%	66,67%	100,00%	71,93%	88,61%	35,29%	42,86%	29,03%	48,95%
k = 3	68,52%	63,16%	66,67%	100,00%	74,59%	80,65%	51,85%	56,41%	33,33%	55,56%
k = 4	55,56%	72,00%	69,57%	100,00%	74,28%	83,33%	66,67%	73,33%	33,33%	64,17%
k = 5	50,00%	75,00%	75,00%	100,00%	75,00%	75,00%	75,00%	81,82%	0,00%	57,96%

Tableau V.3: Résultats de l'application de l'algorithme sur le chariot suspendu.

En utilisant ces dimensions cinq bases de données seront créées et qui seront utilisé pour :

- L'apprentissage du RBFNN qui va assurer les opérations de diagnostic avec des fenêtres temporelles de dimension $m_d = 4$;
- L'apprentissage du RBFNN qui va assurer les opérations de pronostic (pronostic des fautes et prédiction de temps restant avant l'occurrence d'une faute probable) avec des fenêtres temporelles de dimension $m_p = 3$;
- L'apprentissage des trois RBFNN qui vont assurer la vérification des CT lors des trois différents cycles.

5.5 Apprentissage des RdN

Le tableau V.4 résume quelques informations sur les RBFNN utilisés pour assurer les différentes opérations de diagnostic/pronostic des fautes dans le chariot suspendu ainsi que des informations sur les bases de données utilisées pour leurs apprentissages. La base de données d'apprentissage du RBFNN qui va assurer le diagnostic (respectivement le pronostic) des fautes dans le chariot suspendu est composée de 300 (respectivement 335) fenêtres temporelles (100 des fenêtres temporelles sont enregistrées lors de mode de fonctionnement normal, 90 des fenêtres temporelles sont enregistrées lors de mode de fonctionnement F_1 et 80 des fenêtres temporelles sont enregistrées lors de mode de fonctionnement F_2 et 30 des fenêtres temporelles sont enregistrées lors de mode de fonctionnement F_3). Quant aux RBFNN qui vont assurer la vérification des CT, nous construisons trois RBFNN, chaque RBFNN est utilisé pour la vérification des CT d'un seul type des cycles mentionnés au-dessus. Nous désignons, dans le tableau V.4 qui va aussi donner des informations à propos des RBFNN qui vont vérifier la validité des CT, par A (respectivement B et C) le RBFNN qui va assurer la vérification des CT pour le cycle de traitement de produit de type A (respectivement de type B et de type C).

Nous traçons dans la figure V.10 la précision, traduite par l'erreur quadratique moyenne, des deux RBFNN (qui vont assurer le diagnostic et le pronostic) en fonction du nombre des neurones dans la couche cachée. Nous regroupons dans le tableau V.4 quelques caractéristiques des architectures de ces RBFNN qui ont donné une précision de 100% après l'apprentissage. Nous pouvons conclure qu'une précision de 100% a été atteinte avec :

- 69 neurones dans la couche cachée pour le diagnostic ;
- 45 neurones dans la couche cachée pour le pronostic.

Quant à la vérification des CT, la précision, que nous traçons dans la figure V.11, est mesurée par le pourcentage des CT bien classé. On a une précision de 100% avec :

- 74 neurones dans la couche cachée dans le cas de cycle A ;
- 60 neurones dans la couche cachée dans le cas de cycle B ;
- 60 neurones dans la couche cachée dans le cas de cycle C.

RBFNN	Nombre des neurones			Taille des bases de données				
	Entrée	Cachée	Sortie	N	F1	F2	F3	Total
Diagnostic	4	24	4	100	90	80	30	300
Pronostic	3	14	4	185	60	60	30	335
CT	A	6	74	4	20	20	20	80
	B	8	60	4	20	20	40	100
	C	10	60	4	20	20	60	120

Tableau V.4 : Caractéristiques des RBFNN et de leurs bases d'apprentissage

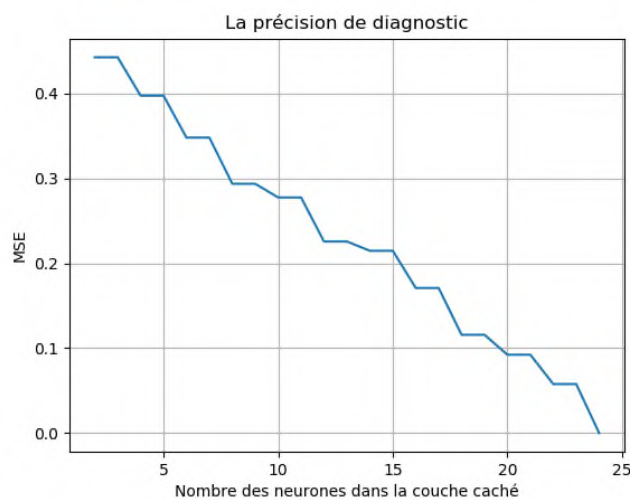


Figure V.9:Précision de diagnostic en fonction de nombre des neurones dans la couche cachée

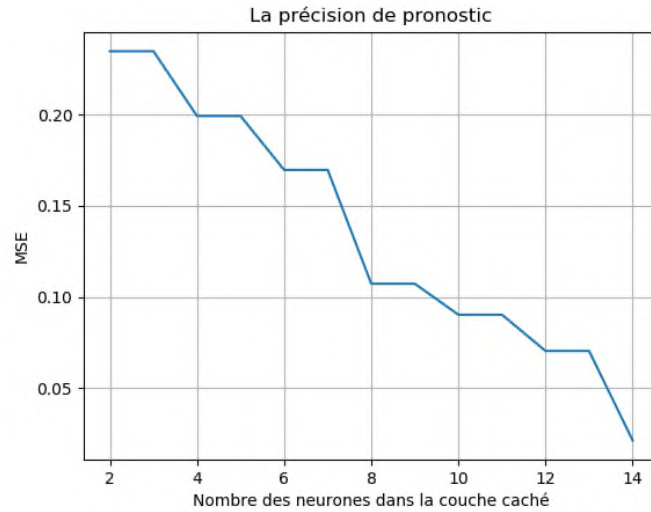


Figure V.10: Précision de pronostic en fonction de nombre des neurones dans la couche cachée

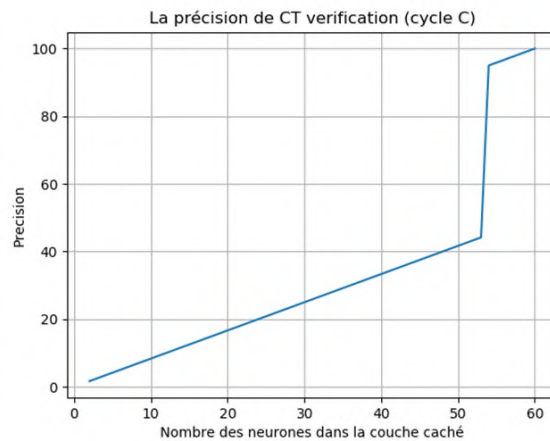
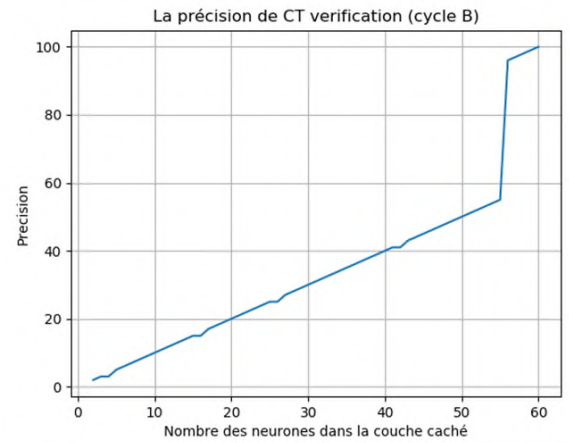
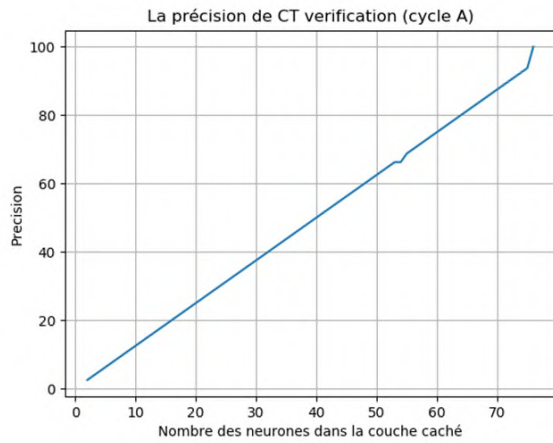


Figure V.11: Précision de la vérification des CT en fonction de nombre des neurones dans la couche cachée des trois types de cycle A, B et C.

5.6 En-ligne diagnostic, pronostic et vérification des CT

Lors de mode en-ligne les événements générés par le SED sont collectés, organisés en forme des fenêtres temporelles et ensuite présentées aux différents RBFNN si leurs dimensions atteignent les dimensions

calculées lors de la section V.4.4 (m_d pour le diagnostic, m_p pour le pronostic cependant la vérification des CT commence dès l'occurrence de premier évènement). Les résultats de diagnostic/pronostic des fautes dans ce SED sont donnés sous forme des probabilités. La figure V.12 donne la dernière fenêtre temporelle générée par le SED ($\omega = e_{p0}e_c e_{pA}e_{pB}$) à partir de laquelle le diagnostic, pronostic et la vérification des CT ont été fait. Les données de cette figure montrent que tous les CT sont respectés, la probabilité d'être dans un état de fonctionnement normal est de 100%, la probabilité d'être dans un des états de fonctionnement anormaux est de 0%, la probabilité d'être, dans le futur (à la limite des 3 évènements observables postérieurs à la dernière fenêtre temporelle enregistrée ω), dans l'un des états de fonctionnement (normal ou anormal) est de 33.33%.

```

***** Résultat de pronostic *****
Vecteur de prognostic: ['ep0', 'eC', None]
Données insuffisantes pour le pronostic
***** verification de respect des CT *****
L contrainte temporelle 1 est respecté
L contrainte temporelle 2 est respecté
L contrainte temporelle 3 est respecté
évènement généré est : epA
***** Résultat de diagnostic *****
Vecteur de diagnostic : ['ep0', 'eC', 'epA', None]
Données insuffisantes pour le diagnostic
***** Résultat de pronostic *****
Vecteur de prognostic: ['ep0', 'eC', 'epA']
60.0% on aura l'état de fonctionnement normal dans les 3 évènements postérieurs
40.0% on aura l'état de fonctionnement anormal F1 dans les 3 évènements postérieurs
40.0% on aura l'état de fonctionnement anormal F2 dans les 3 évènements postérieurs
40.0% on aura l'état de fonctionnement anormal F3 dans les 3 évènements postérieurs
***** verification de respect des CT *****
L contrainte temporelle 1 est respecté
L contrainte temporelle 2 est respecté
L contrainte temporelle 3 est respecté
évènement généré est : epB
***** Résultat de diagnostic *****
Vecteur de diagnostic : ['ep0', 'eC', 'epA', 'epB']
100.0% on est dans l'état de fonctionnement normal
0.0% on est dans l'état de fonctionnement anormal F1
0.0% on est dans l'état de fonctionnement anormal F2
0.0% on est dans l'état de fonctionnement anormal F3
***** Résultat de pronostic *****
Vecteur de prognostic: ['eC', 'epA', 'epB']
33.33% on aura l'état de fonctionnement normal dans les 3 évènements postérieurs
33.33% on aura l'état de fonctionnement anormal F1 dans les 3 évènements postérieurs
33.33% on aura l'état de fonctionnement anormal F2 dans les 3 évènements postérieurs
33.33% on aura l'état de fonctionnement anormal F3 dans les 3 évènements postérieurs
***** verification de respect des CT *****

```

Figure V.12: Capture d'écran de résultat de mode en-ligne.

6. Conclusion

Dans ce chapitre, l'ensemble des approches détaillées dans les chapitres précédents ont été testées dans un cas d'application pratique afin de mettre en évidence leurs pertinences. Dans ce chapitre, une application informatique, programmée en Python, a été utilisée pour assurer les différentes opérations de diagnostic/pronostic des SED. Cette application est utilisée pour élaborer le diagnostic/pronostic en temps réel d'un chariot suspendu qui est considéré comme un SED. Dans un premier temps une description de fonctionnement de ce SED est faite et après les résultats des différentes opérations du diagnostic, du pronostic et de la vérification de respect des CT sont présentés.

Lors de la mise en place de notre approche de diagnostic/pronostic sur le chariot suspendu disponible au laboratoire des techniques industrielles nous avons rencontré deux contraintes majeures qui sont :

- L'impossibilité de l'établissement d'un réseau Modbus entre l'ordinateur et l'API disponible dans notre laboratoire vu l'incompatibilité logicielle ;
- Le manque des données en relation avec la fiabilité des différentes composantes de SED que nous avons pris comme cas d'application

Pour faire face à ces contraintes, nous visons, dans la suite de nos travaux, acheter une nouvelle gamme d'API qui sera compatible avec les dernières versions de Windows afin de pouvoir établir un réseau Modbus entre cet automate et l'ordinateur et ainsi mettre en place la solution complète de diagnostic/pronostic au chariot suspendu ainsi que les approches qui seront développées par notre équipe de recherche à l'avenir. Nous visons aussi appliquer notre approche sur un système industriel réel où des informations sur la fiabilité de ses composants sont disponibles.

Conclusion générale et perspectives

Le diagnostic est le pronostic des fautes dans les systèmes industriels jouent un rôle primordial dans les systèmes industriels. Ceci en préservant ces systèmes de commettre des fautes dangereuses qui peuvent mettre en danger la sécurité des hommes est des équipements. L'implémentation des approches de diagnostic/pronostic conventionnelles (AMDEC, arbre des défaillances...) dans les systèmes à événements discrets industriels est confrontée à pas mal de problèmes et elle devient parfois impossible notamment lorsqu'il s'agit des systèmes industriels larges est complexe qui ont reconnu dernièrement un grand progrès technologique.

Ce travail de recherche a permis de développer une nouvelle approche de diagnostic/pronostic des fautes dans les SED indépendamment des outils de modélisation de systèmes étudiés ce qui rend cette approche l'approche la plus adéquate aux SED large et complexe où l'utilisation des outils de modélisation conventionnelle pour les SED (automate fini, réseau de Petri...) est une tâche très complexe et parfois impossible.

Le comportement du SED est décrit par des fenêtres temporelles contenant les événements générés par le SED couplé avec leurs dates d'occurrences. Un algorithme permettant de calculer les dimensions de ces fenêtres temporelles qui vont permettre la maximisation de l'efficacité des opérations de diagnostic/pronostic est proposé. À la base de ces fenêtres temporelle la probabilité d'occurrence d'une faute dans la dernière fenêtre temporelle (diagnostic), la probabilité de son occurrence dans les k fenêtres temporelles postérieures (pronostic) et le temps restant avant l'occurrence d'une faute probable sont calculés. Les contraintes temporelles que le SED doit respecter lors de son fonctionnement sont aussi traitées.

L'approche de diagnostic/pronostic proposée dans cette thèse est équivalente aux approches de k -diagnosticabilité/ k -prédictabilité proposées dans la littérature. Cependant, notre approche, au contraire des approches proposées dans la littérature, donne des résultats sous forme des probabilités.

Des réseaux de neurones à base de fonction radiale (RBFNN) sont utilisés pour apprendre les différentes relations existantes entre ces fenêtres temporelles est les différents paramètres de diagnostic/pronostic cités au-dessus. Ces réseaux vont permettre de mettre en place un en-ligne diagnostiqueur, un en-ligne pronostiqueur, un en-ligne estimateur de temps restant avant l'occurrence d'une faute et des en-ligne vérificateurs de respects des contraintes temporelles.

L'approche présentée dans cette thèse a plusieurs points forts, nous citons :

- Elle donne les résultats du diagnostic/ pronostic sous forme des probabilités ce qui remplace les états incertains donnés par les diagnostiqueurs basés sur les automates finis ;

- Elle permet de détecter le temps restant avant l'occurrence d'une faute probable. Ce temps est très nécessaire pour préparer les actions de maintenance correctives ou bien pour arrêter le système avant l'écoulement de ce temps afin d'éviter les dégâts dangereux causés par l'occurrence de cette faute ;
- Elle est applicable aux SED larges et complexes où les outils de diagnostic/pronostic usuels sont insuffisants, ainsi elle est utile pour une large gamme des systèmes industriels ;
- L'intégration des lois de fiabilité a permis à notre approche de prendre en considération, en plus du comportement actuel de SED, les degrés de fiabilité des composants de ce SED.

Comme extension de ce travail, nous visons à :

- Mettre en place un nouvel automate programmable capable d'établir un réseau Modbus avec l'ordinateur et ainsi tester en temps réel l'approche de diagnostic/pronostic sur la maquette du chariot suspendu.
- Implémenter l'approche du diagnostic/pronostic dans un SED industriel large où des informations sur la fiabilité des composants sont disponibles.
- Développer une formule ou une fonction coût qui permet de traduire la complexité des RBFNN en fonction du nombre des éléments d'entrée (les dimensions des fenêtres temporelles) ainsi nous pouvons mieux choisir les dimensions des fenêtres temporelles qui vont maximiser l'efficacité de l'approche de diagnostic/pronostic ;
- Intégrer des outils d'interprétabilité qui vont permettre d'expliquer les différentes décisions prises par les RdN ;
- Développer une architecture des RdN et une méthode d'apprentissage des RdN plus convenables à l'analyse des fenêtres temporelles ;
- Utiliser les RdN récurrentes ou les RdN profonds pour l'analyse des événements générés par le SED et comparer les résultats obtenus avec ceux de l'approche développée le long de cette thèse.

Références

- [1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on automatic control*, vol. 40, pp. 1555-1575, 1995.
- [2] J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Introduction to automata theory, languages, and computation," *Acm Sigact News*, vol. 32, pp. 60-65, 2001.
- [3] G. Lamperti and M. Zanella, "Flexible diagnosis of discrete-event systems by similarity-based reasoning techniques," *Artificial Intelligence*, vol. 170, pp. 232-297, 2006.
- [4] L. Rozé and M.-O. Cordier, "Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks," *Discrete Event Dynamic Systems*, vol. 12, pp. 43-81, 2002.
- [5] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard, "Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets," *Discrete event dynamic systems*, vol. 8, pp. 203-231, 1998.
- [6] Ş. Genç and S. Lafortune, "A distributed algorithm for on-line diagnosis of place-bordered petri nets," *IFAC Proceedings Volumes*, vol. 38, pp. 68-73, 2005.
- [7] C. G. Cassandras, *Introduction to discrete event systems*: Springer Science & Business Media, 2008.
- [8] M. Combacau, P. Berruet, E. Zamai, P. Charbonnaud, and A. Khatab, "Supervision and Monitoring of Production Systems," *IFAC Proceedings Volumes*, vol. 33, pp. 849-854, 2000/07/01/ 2000.
- [9] N. Jerbi, S. C. Dutilleul, E. Craye, and M. Benrejeb, "Robustesse d'un atelier à tâches sans assemblage en fonctionnement répétitif," *Séminaire d'Automatique-Industrie (SAI'06), Gabès-Matmata, CD ROM*, 2006.
- [10] A. Drogoul, "De la simulation multi-agents à la résolution collective de problèmes," Thesis at University of Paris IV, 1993.
- [11] G. Frey and L. Litz, "Formal methods in PLC programming," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, 2000, pp. 2431-2436.
- [12] F. Cassez and S. Tripakis, "Fault Diagnosis with Dynamic Diagnosers," in *9th Workshop on Discrete-Event Systems*, Goteborg, Sweden, 2008, pp. 212--217.
- [13] F. Lin, "Diagnosability of discrete event systems and its applications," *Discrete Event Dynamic Systems*, vol. 4, pp. 197-212, 1994/05/01 1994.
- [14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE transactions on control systems technology*, vol. 4, pp. 105-124, 1996.
- [15] C. M. Ozveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE transactions on automatic control*, vol. 35, pp. 797-806, 1990.
- [16] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, "A polynomial algorithm for testing diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 46, pp. 1318-1321, 2001.
- [17] T.-S. Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially observed discrete-event systems," *IEEE Transactions on automatic control*, vol. 47, pp. 1491-1495, 2002.
- [18] M. V. Moreira, T. C. Jesus, and J. C. Basilio, "Polynomial time verification of decentralized diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 56, pp. 1679-1684, 2011.
- [19] T. Ushio, I. Onishi, and K. Okuda, "Fault detection based on Petri net models with faulty behaviors," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, 1998, pp. 113-118.
- [20] S. Genc and S. Lafortune, "Distributed diagnosis of discrete-event systems using Petri nets," in *International Conference on Application and Theory of Petri Nets*, 2003, pp. 316-336.
- [21] S. Genc and S. Lafortune, "Distributed diagnosis of place-bordered Petri nets," *IEEE Transactions on Automation science and Engineering*, vol. 4, pp. 206-219, 2007.

- [22] B. Liu, M. Ghazel, and A. Toguyéni, "On-the-Fly and Incremental Technique for Fault Diagnosis of Discrete Event Systems Modeled by Labeled Petri Nets," *Asian Journal of Control*, vol. 19, pp. 1659-1671, 2017.
- [23] A. Benveniste, E. Fabre, S. Haar, and C. Jard, "Diagnosis of asynchronous discrete-event systems: a net unfolding approach," *IEEE Transactions on Automatic Control*, vol. 48, pp. 714-727, 2003.
- [24] S. Haar, C. Rodríguez, and S. Schwoon, "Reveal your faults: It's only fair!," in *Application of Concurrency to System Design (ACSD), 2013 13th International Conference on*, 2013, pp. 120-129.
- [25] M. P. Cabasino, A. Giua, M. Pocci, and C. Seatzu, "Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems," *Control Engineering Practice*, vol. 19, pp. 989-1001, 2011.
- [26] A. Giua and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, 2005, pp. 6323-6328.
- [27] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, pp. 1531-1539, 2010.
- [28] K. R. Boel and G. Jiroveanu, "Distributed contextual diagnosis for very large systems," *IFAC Proceedings Volumes*, vol. 37, pp. 333-338, 2004.
- [29] G. Jiroveanu and R. K. Boel, "The diagnosability of Petri net models using minimal explanations," *IEEE Transactions on Automatic Control*, vol. 55, pp. 1663-1668, 2010.
- [30] A. Boussif, "Contributions to Model-Based Diagnosis of Discrete-Event Systems," Université de Lille1-Sciences et Technologies, 2016.
- [31] A. Boussif, B. Liu, and M. Ghazel, "An Experimental Comparison of Three Diagnosis Techniques for Discrete Event Systems," in *DX'17-28th International Workshop on Principles of Diagnosis*, 2017, p. p8.
- [32] M. P. Cabasino, A. Giua, and C. Seatzu, "Diagnosability of discrete-event systems using labeled petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 11, pp. 144-153, 2014.
- [33] J. Prock, "A new technique for fault detection using Petri nets," *Automatica*, vol. 27, pp. 239-245, 1991.
- [34] Y. Wen, C. Li, and M. Jeng, "A polynomial algorithm for checking diagnosability of Petri nets," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, 2005, pp. 2542-2547.
- [35] A. Ramírez-Treviño, E. Ruiz-Beltrán, I. Rivera-Rangel, and E. Lopez-Mellado, "Online fault diagnosis of discrete event systems. A Petri net-based approach," *IEEE Transactions on Automation Science and Engineering*, vol. 4, pp. 31-39, 2007.
- [36] S.-L. Chung, "Diagnosing PN-based models with partial observable transitions," *International Journal of Computer Integrated Manufacturing*, vol. 18, pp. 158-169, 2005.
- [37] A. Madalinski, F. Nouioua, and P. Dague, "Diagnosability verification with Petri net unfoldings," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 14, pp. 49-55, 2010.
- [38] M. P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu, "A new approach for diagnosability analysis of Petri nets using verifier nets," *IEEE Transactions on Automatic Control*, vol. 57, pp. 3104-3117, 2012.
- [39] H.-E. Gougam, A. Subias, and Y. Pencolé, "Supervision patterns: formal diagnosability checking by Petri net unfolding," *IFAC Proceedings Volumes*, vol. 46, pp. 73-78, 2013.
- [40] H.-E. Gougam, A. Subias, and Y. Pencolé, "Discriminability Analysis of Supervision Patterns by Net Unfoldings," *IFAC Proceedings Volumes*, vol. 47, pp. 459-464, 2014.
- [41] Y. Wu and C. N. Hadjicostis, "Algebraic approaches for fault identification in discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 50, pp. 2048-2055, 2005.
- [42] D. Lefebvre and C. Delherm, "Diagnosis of DES with Petri net models," *Automation Science and Engineering, IEEE Transactions on*, vol. 4, pp. 114-118, 2007.
- [43] O. Contant, S. Lafortune, and D. Teneketzis, "Diagnosability of discrete event systems with modular structure," *Discrete Event Dynamic Systems*, vol. 16, pp. 9-37, 2006.

- [44] Y. Pencolé, "Diagnosability analysis of distributed discrete event systems," in *ECAI*, 2004, p. 43.
- [45] A. Schumann and Y. Pencolé, "Scalable Diagnosability Checking of Event-Driven Systems," in *IJCAI*, 2007, pp. 575-580.
- [46] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part III: Process history based methods," *Computers & chemical engineering*, vol. 27, pp. 327-346, 2003.
- [47] M. K. Shahzad, S. Hubac, A. Siadat, and M. Tollenaere, "An interdisciplinary fmea methodology to find true dfm challenges," in *12th European Advanced Process Control and Manufacturing Conference, Grenoble, France*, 2012.
- [48] S. Sahin, M. R. Tolun, and R. Hassanpour, "Hybrid expert systems: A survey of current approaches and applications," *Expert systems with applications*, vol. 39, pp. 4609-4617, 2012.
- [49] B. Li, T. Han, and F. Kang, "Fault diagnosis expert system of semiconductor manufacturing equipment using a Bayesian network," *International Journal of Computer Integrated Manufacturing*, vol. 26, pp. 1161-1171, 2013.
- [50] Z. Liu, Y.-F. Li, L.-P. He, Y.-J. Yang, and H.-Z. Huang, "A new fault tree analysis approach based on imprecise reliability model," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 228, pp. 371-381, 2014.
- [51] G. Merle, J.-M. Roussel, and J.-J. Lesage, "Improving the efficiency of dynamic fault tree analysis by considering gate FDEP as static," in *European Safety and Reliability Conference (ESREL 2010)*, 2010, pp. pp. 845-851.
- [52] K. D. Rao, V. Gopika, V. S. Rao, H. Kushwaha, A. K. Verma, and A. Srividya, "Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment," *Reliability Engineering & System Safety*, vol. 94, pp. 872-883, 2009.
- [53] B.-S. Yang, M.-S. Oh, and A. C. C. Tan, "Fault diagnosis of induction motor based on decision trees and adaptive neuro-fuzzy inference," *Expert Systems with Applications*, vol. 36, pp. 1840-1849, 2009.
- [54] H. Koivo, "Artificial neural networks in fault diagnosis and control," *Control engineering practice*, vol. 2, pp. 89-101, 1994.
- [55] S. Khomfoi and L. M. Tolbert, "Fault diagnostic system for a multilevel inverter using a neural network," *IEEE Transactions on Power Electronics*, vol. 22, pp. 1062-1069, 2007.
- [56] Z. Gao, C. Cecati, and S. X. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 3757-3767, 2015.
- [57] Z. Gao, C. Cecati, and S. X. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part II: Fault Diagnosis With Knowledge-Based and Hybrid/Active Approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 3768-3774, 2015.
- [58] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115-133, 1943.
- [59] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, p. 386, 1958.
- [60] J. L. McClelland, D. E. Rumelhart, and P. R. Group, "Parallel distributed processing," *Explorations in the microstructure of cognition*, vol. 2, 1986.
- [61] Y. Pao, "Adaptive pattern recognition and neural networks," 1989.
- [62] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," DTIC Document 1988.
- [63] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural computation*, vol. 1, pp. 281-294, 1989.
- [64] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, pp. 59-69, 1982.
- [65] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines* vol. 3: Pearson Education Upper Saddle River, 2009.
- [66] R. Sathya and A. Abraham, "Comparison of supervised and unsupervised learning algorithms for pattern classification," *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, pp. 34-38, 2013.

- [67] M. K. S. Alsmadi, K. B. Omar, and S. A. Noah, "Back propagation algorithm: the best algorithm among the multi-layer perceptron algorithm," *International Journal of Computer Science and Network Security*, vol. 9, pp. 378-383, 2009.
- [68] Y. Xiong, L. Wang, and D. Li, "Training Feedforward Neural Networks by Pruning Algorithm Based on Grey Incidence Analysis," in *Intelligent Information Technology Application, 2008. IITA'08. Second International Symposium on*, 2008, pp. 535-539.
- [69] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on communications*, vol. 28, pp. 84-95, 1980.
- [70] H. Wang and P. Chen, "Intelligent diagnosis method for rolling element bearing faults using possibility theory and neural network," *Computers & Industrial Engineering*, vol. 60, pp. 511-518, 2011.
- [71] R. Patton, J. Chen, and T. Siew, "Fault diagnosis in nonlinear dynamic systems via neural networks," in *Control, 1994. Control'94. International Conference on*, 1994, pp. 1346-1351.
- [72] H. R. DePold and F. D. Gass, "The application of expert systems and neural networks to gas turbine prognostics and diagnostics," in *ASME 1998 International Gas Turbine and Aeroengine Congress and Exhibition*, 1998, pp. V005T15A009-V005T15A009.
- [73] D. Tobon-Mejia, K. Medjaher, and N. Zerhouni, "CNC machine tool's wear diagnostic and prognostic by using dynamic Bayesian networks," *Mechanical Systems and Signal Processing*, vol. 28, pp. 167-182, 2012.
- [74] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *Control Systems Magazine, IEEE*, vol. 10, pp. 18-23, 1990.
- [75] J. Na, X. Ren, C. Shang, and Y. Guo, "Adaptive neural network predictive control for nonlinear pure feedback systems with input delay," *Journal of Process Control*, vol. 22, pp. 194-206, 2012.
- [76] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, 2004.
- [77] M. I. Jordan, "Serial order: A parallel distributed processing approach," *Advances in psychology*, vol. 121, pp. 471-495, 1997.
- [78] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, pp. 179-211, 1990.
- [79] P. Frasconi, M. Gori, M. Maggini, and G. Soda, "Representation of finite state automata in recurrent radial basis function networks," *Machine Learning*, vol. 23, pp. 5-32, 1996.
- [80] R. Zemouri, D. Racocanu, and N. Zerhouni, "Recurrent radial basis function network for time-series prediction," *Engineering Applications of Artificial Intelligence*, vol. 16, pp. 453-463, 2003.
- [81] T. J. Sejnowski and C. R. Rosenberg, *NETtalk: A parallel network that learns to read aloud*: MIT Press, 1988.
- [82] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, pp. 328-339, 1989.
- [83] M. R. Berthold, "A time delay radial basis function network for phoneme recognition," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, 1994, pp. 4470-4472, 4472a vol. 7.
- [84] M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schlüter, and H. Ney, "Comparison of feedforward and recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8430-8434.
- [85] M. Fatima and S. Hamid, "Comparaison de méthodes de classification réseau RBF, MLP et RVFLNN1," *Damascus University Journal Vol.(25)-No.(2)*, 2009.
- [86] J.-W. Park, R. Harley, and G. Venayagamoorthy, "Comparison of MLP and RBF neural networks using deviation signals for on-line identification of a synchronous generator," in *Power Engineering Society Winter Meeting, 2002. IEEE, 2002*, pp. 274-279.
- [87] D. van Schrick, "Remarks on terminology in the field of supervision, fault detection and diagnosis," *IFAC Proceedings Volumes*, vol. 30, pp. 959-964, 1997.
- [88] R. Alur and T. A. Henzinger, "Back to the future: towards a theory of timed regular languages," in *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, 1992, pp. 177-186.

- [89] P. Bouyer, F. Chevalier, and D. D'Souza, "Fault diagnosis using timed automata," in *International Conference on Foundations of Software Science and Computation Structures*, 2005, pp. 219-233.
- [90] F. Cassez and A. Grastien, "Predictability of event occurrences in timed systems," in *International Conference on Formal Modeling and Analysis of Timed Systems*, 2013, pp. 62-76.
- [91] M. MSAAF and F. BELMAJDOUN, "Fault diagnosis and prognosis in discrete event systems using statistical model and neural networks," *International Journal of Mechatronics and Automation*, vol. 6, pp. 173-182, 2018.
- [92] S. Genc and S. Lafortune, "PREDICTABILITY IN DISCRETE-EVENT SYSTEMS UNDER PARTIAL OBSERVATION 1," *IFAC Proceedings Volumes*, vol. 39, pp. 1461-1466, 2006.
- [93] M. Msaaf and F. Belmajdoub, "An Improvement of Faults Diagnosis and Prognosis of Discrete Event Systems Based on Reliability Laws," in *International Journal of Engineering Research in Africa*, 2019, pp. 148-155.
- [94] P. Crow-AMSAA, "The New Weibull Handbook," 1993.
- [95] K. Kołowrocki, "Reliability of large systems," *Encyclopedia of quantitative risk analysis and assessment*, vol. 4, 2008.
- [96] L. Zadeh, "Fuzzy Sets," *Inform Control*, vol. 8, pp. 338-353, 1965.
- [97] S. K. De, R. Biswas, and A. R. Roy, "Some operations on intuitionistic fuzzy sets," *Fuzzy Sets and Systems*, vol. 114, pp. 477-484, 2000/09/16/ 2000.
- [98] L. A. Zadeh, "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges," *Journal of Cybernetics*, vol. 2, pp. 4-34, 1972/01/01 1972.
- [99] M. Fri, F. Belmajdoub, and D. Lefebvre, "Fault Diagnosis of Discrete Event Systems under Time Constraints," in *Systems Informatics, Modelling and Simulation (SIMS), International Conference on*, 2016, pp. 81-86.