# Méthodologie de surveillance dynamique à l'aide des réseaux neuro-flous temporels.

Nicolas Palluat

Année : **2006** N° ordre : **1135**

# THESE

présentée à

**L'UFR des Sciences et Techniques**
**de l'Université de Franche-Comté**

pour obtenir le

## GRADE DE DOCTEUR DE L'UNIVERSITE DE FRANCHE-COMTE

**en Automatique**
*(Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques)*

## Méthodologie de surveillance dynamique à l'aide des réseaux neuro-flous temporels

par

**Nicolas PALLUAT**

Soutenue le *12 janvier 2006* devant la Commission d'examen :

| | | |
|---|---|---|
| Rapporteurs : | Patrick BRÉZILLON | Chargé de recherche CNRS, Université de Paris VI |
| | Belkacem OULD-BOUAMAMA | Professeur à l'Université des Sciences et Technologies de Lille |
| Examinateurs : | Chengbin CHU | Professeur à l'UTT de Troyes |
| | Jean-Michel OLIVE | Maître de Conférences à l'Ecole Polytechnique de Marseille |
| Directeurs de thèse : | Noureddine ZERHOUNI | Professeur à l'ENSMM de Besançon |
| | Daniel RACOCEANU | Maître de Conférences à l'Université de Franche-Comté, Besançon |
| Invité : | Jean-Baptiste LÉGER | PDG Sté PREDICT, Vandœuvre-lès-Nancy |

# Remerciements

Je tiens à remercier ici toutes les personnes qui ont contribués directement et indirectement au bon déroulement de ces trois années (et quelque) de thèse.

En tout premier lieu je tiens à remercier les trois personnes sans qui tout ceci n'aurait pas été possible :

- M. Alain Bourjault, directeur du laboratoire d'Automatique de Besançon, pour m'avoir accepté au sein du laboratoire ;
- M. Noureddine Zerhouni, professeur à l'ENSMM, pour m'avoir accueilli au sein de l'équipe Maintenance et Sûreté de Fonctionnement ;
- M. Daniel Racoceanu, tout jeune HDR à l'université de Franche-Comté de Besançon, pour son aide et son soutien tout au long de ces trois années.

En second lieu, j'aimerais remercier les membres du jury pour avoir accepté de juger mon travail :

- M. Chengbin Chu, professeur à l'UTT de Troyes, pour m'avoir fait l'honneur de présider mon jury de soutenance ;
- M. Patrick Brézillon, chargé de recherche CNRS à l'Université de Paris VI, et M. Belkacem Ould-Bouamama, professeur à l'Université des Sciences et Technologies de Lille, pour leur lecture critique de mon manuscrit ;
- M. Jean-Michel Olive, Maître de Conférences à l'Ecole Polytechnique de Marseille, d'avoir accepté d'être examinateur lors de ma soutenance ;
- M. Jean-Baptiste Léger, Président Directeur Général de la société PREDICT de Vandœuvre-lès-Nancy, pour avoir accepté de participer à mon jury de soutenance.

Je pense aussi aux membres du laboratoire pour la bonne ambiance, pour les nombreux conseils qui m'ont apportés beaucoup pendant ces trois ans. Je pense notemment à Denis, Juju, Titi, Bruno, Béty, Laeti, Toufou, Cédric, Cécé, Rafael, Michael G, Ali, Guillaume, Arnaud, Jean-Yves, Mac Guyver, Micky, Farouk, Zabou ... mais aussi à ceux qui sont déjà partis Slava, Sylvain, Ry, Alex, Sid, Mehdi ... et enfin à tous les autres Claire, Maurice, Nanou, Timmy, Tutu ...

Merci enfin et surtout à ma famille pour tout le soutien qu'ils m'ont apporté et bien sûr à ma chère et tendre Fé.

# Sommaire

# Liste des Algorithmes

# Table des figures

# Glossaire des notations

| | |
|---|---|
| AdD : | Arbre de Défaillance |
| AMDEC : | Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité |
| DDA : | Dynamic Decay Adjustment, algorithme introduit par Berthold |
| GMAO : | Gestion de la Maintenance Assistée par Ordinateur |
| P-RCE : | Extension probabiliste à l'algorithme RCE |
| RBF : | Radial Basis Function neural network (idem RFR) |
| RCE : | Restricted Coulomb Energy, algorithme introduit par Reilly, Cooper et Elbaum |
| RFR : | Réseau de neurones à Fonction de base Radiale (idem RBF) |
| RRBF : | Recurrent Radial Basis Function neural network (idem RRFR) |
| RRFR : | Réseau de neurones Récurrents à Fonction de base Radiale (idem RRBF) |
| SdF : | Sûreté de Fonctionnement |

### Notations spécifiques au chapitre III

*Base d'apprentissage* :

| | |
|---|---|
| $F$ : | Ensemble des séquences de la base d'apprentissage |
| $n_F$ : | Nombre de séquences |
| $BF$ : | Ensemble des séquences de bon fonctionnement |
| $n_{BF}$ : | Nombre de séquences de bon fonctionnement |
| $N$ : | Nombre de vecteurs dans une séquence |
| $n$ : | Dimension d'un vecteur |
| $V$ : | Nombre de vecteurs dans la base d'apprentissage modifiée |

*Réseau de neurone dynamique - RRFR* :

| | |
|---|---|
| $e_i$ : | Entrée du neurone $i$ de la couche d'entrée |

| | |
|---|---|
| $s_i$ : | Sortie du neurone $i$ de la couche d'entrée |
| $a_i$ : | Activation du neurone $i$ de la couche d'entrée |
| $w_i$ : | Poids d'auto-connexion du neurone $i$ de la couche d'entrée |
| $k_i$ : | Sensibilité de neurone $i$ de la couche d'entrée |
| $S_0$ : | Seuil d'oubli pour la phase d'apprentissage de la couche d'entrée |
| $\delta$ : | Coefficient de mémoire de la couche d'entrée |
| $\Delta$ : | Précision de $\delta$ |
| $r_j$ : | Centre de la gaussienne du neurone $j$ de la couche cachée |
| $\sigma_j$ : | Rayon d'influence de la gaussienne du neurone $j$ de la couche cachée |
| $R_j$ : | Activation du neurone $j$ de couche cachée |
| $A_j^c$ : | Poids reliant le neurone $j$ de la couche cachée au neurone de sortie $c$ |
| $m$ : | Nombre de neurones dans la couche cachée |
| $p_j^c$ : | Prototype $j$ lié à la classe $c$ – correspond au neurone $j$ de la couche cachée |
| $m_b$ : | Nombre de prototypes liés à la classe $b$ |
| $\theta^+$ $\theta^-$ $\theta$ : | Seuils utilisés pour la phase d'apprentissage de la couche cachée |
| $\beta_{ji}$ : | Borne supérieure sur la dimension $i$ de l'hypercube $j$ |
| $\gamma_{ji}$ : | Borne inférieure sur la dimension $i$ de l'hypercube $j$ |
| $H_j$ : | Degré d'appartenance pour l'hypercube $j$ |
| $\eta$ : | Sensibilité de l'hypercube |
| $\mu$ : | Facteur pour la création d'un nouvel hypercube |

*"If anything can go wrong, it will"*
*Capt. Edward A. Murphy (1949)*

# Introduction générale

La complexité croissante des processus industriels et leur mise en oeuvre sur la base des nouvelles technologies de l'information et de la communication ont pour conséquence essentielle, lors de l'exploitation de ces systèmes, que leur coût d'arrêt ou d'indisponibilité est sans commune mesure avec le coût de leur réparation. Un des défis à résoudre relativement à ces systèmes est donc de chercher à maîtriser au mieux les coûts de possession en cohérence avec les objectifs de sécurité et de disponibilité de l'ensemble de l'entreprise de plus en plus étendue et nécessairement flexible.

En ce sens, une des solutions est d'implanter au sein de ces systèmes, des processus de maintenance principalement de type surveillance, diagnostic (aide au diagnostic), pronostic (aide au pronostic), compensation dont la finalité est de contribuer à garantir une disponibilité maximale des composants, compatible avec un maintien en exploitation. Ces processus n'ont pas pour vocation de remplacer l'homme mais bien de l'aider dans sa prise de décision finale. En effet, l'aide au diagnostic a pour objet de proposer, le plus rapidement possible et par un principe de causalité directe, la cause initiale du dysfonctionnement, alors que l'aide au pronostic a pour objet d'évaluer par propagation l'impact d'une défaillance sur le système en termes de sécurité, de disponibilité mais aussi de coût.

La modélisation précise et détaillée des systèmes de production étendus et flexibles devient de plus en plus difficile vu le nombre important de paramètres, d'aléas et de restructurations imposées par un environnement de plus en plus concurrentiel et exigeant. L'essor des systèmes d'aide à la décision dans ce contexte industriel soulève le problème de leur flexibilité, adaptabilité et de leur fiabilité, condition nécessaire à leur implantation dans des systèmes opérationnels. Il devient donc impératif pour les responsables de maintenance de pouvoir disposer d'un outil d'aide à la décision rapide, efficace et sûr de fonctionnement, capable d'intégrer des contraintes de production et de maintenance ainsi que de capitaliser un savoir-faire de plus en plus précieux.

Depuis l'émergence des premiers systèmes experts, l'automatisation du diagnostic a inspiré de nombreux travaux en intelligence artificielle. Vu la complexité croissante des systèmes : installations industrielles compliquées nécessitant un mélange de compétences (mécanique, automatique, électronique. . . ), l'apparition de grands réseaux et l'augmen-

tation des risques (centrales nucléaires), il est de plus en plus crucial d'assister l'homme dans les opérations de surveillance. Cette aide au diagnostic implique d'automatiser un ensemble de tâches : détecter un fonctionnement anormal du système, puis à partir des observations anormales, déterminer la cause du dysfonctionnement du système, afin de proposer éventuellement des actions à effectuer. Dans le cadre de ce mémoire, nous nous intéressons à la détection de défaillances et au diagnostic, c'est-à-dire à la recherche d'explication(s) et de cause(s) pour un ensemble de phénomènes anormaux observés.

Les premiers systèmes d'aide au diagnostic s'appuient sur la modélisation du raisonnement de l'expert pour résoudre le problème, souvent sous forme d'associations[1] ***si*** *symptôme1* ***et*** *symptôme2* ***et*** *... symptômeN* ***alors*** *panne*, ou sur l'utilisation par analogie de cas déjà résolus. Ces premières approches, bien que très efficaces – certaines se sont montrées aussi efficaces que des experts humains – ont été critiquées dés le début des années 80. Les principales reproches portent sur les difficultés d'acquisition et de validation du modèle, sur la nécessité de connaître à l'avance les pannes pouvant survenir, sur la difficulté à traiter les cas de pannes multiples, sur l'absence de justification des réponses proposées, ainsi que sur les problèmes de maintenance de la base de connaissances lorsque le système évolue.

Notre travail se situe dans le cadre des méthodes dites « sans modèle » par opposition aux méthodes « à base de modèle » qui reposent sur une modélisation « profonde » du comportement ou du fonctionnement du système. Au sein de ces approches « sans modèle », deux grands courants se dessinent : Les méthodes à base d'outils statistiques et celles à base d'outils symboliques. Notre approche consiste en l'utilisation des outils symboliques et plus précisant de méthodes de reconnaissance de forme et des modèles explicatifs pour l'étude d'un système dynamique d'aide à la surveillance. Ce choix est justifié par les qualités requises pour un système d'aide à la surveillance permettant ainsi l'intégration d'un apprentissage dynamique et d'une approche permettant la modélisation de l'incertitude et de l'imprécision.

Ce mémoire est organisé en quatre chapitres.

Le premier chapitre traite de la problématique de la surveillance dans l'environnement industriel. La surveillance industrielle est une partie intégrante de la sûreté de fonctionnement des systèmes et en constituant une fonction de plus en plus importante dans le pilotage des systèmes. Les besoins dans ces deux domaines évoluent constamment notemment en intelligence artificielle pour exploiter et préserver un savoir-faire et pour amener une intelligence répartie vers les niveaux opérationnels les plus vas. Les méthodologies de surveillance se divisent en deux groupes : méthodologies de surveillance avec et sans modèle. Les premières se basent sur l'existence d'un modèle formel de l'équipement et utilisent généralement les techniques de l'automatique et de l'industrie. Pour la deuxième catégorie de méthodologies, les techniques utilisées sont celles de traitement

[1] Ce qui vaut à ces modèles le nom de modèles associatifs.

du signal et celles de l'intelligence artificielle. De nombreux travaux ont été entrepris dans chacune de ces catégories. Nous en feront un état de l'art de ces travaux dans ce chapitre.

L'objet du deuxième chapitre est la présentation des différents outils de l'Intelligence Artificielle pour la surveillance. Ces méthodes se partagent en trois grandes familles : les méthodes à base de modèles comportementaux, les méthodes de reconnaissance de formes et les méthodes à base de modèles explicatifs. Parmi les différentes modalités de définition du concept de surveillance, nous avons mis en évidence deux étapes essentielles : la détection de défauts et le diagnostic de pannes. Le diagnostic peut lui même être décomposé en une localisation et une identification des causes. Cette définition nous amène au fait que toutes les méthodes dites de surveillance ne remplissent pas entièrement la fonction de recherche de cause qui nous parait primordiale. Selon la classification que nous avons effectuée, seules les techniques à base de modèles explicatifs répondent à ce critère. Ainsi, en respectant les fonctions naturelles de la surveillance, notre outil se décomposera en deux outils : un outil pour la détection et un autre pour le diagnostic.

Le troisième chapitre présente les deux outils que nous utilisons pour effectuer une aide à la surveillance. Pour chaque étape de la surveillance, nous avons défini un outil. L'outil de détection, devant prendre en compte le temps, est un réseau récurrent à fonction de base radiale, RRFR, tandis que l'outil de diagnostic est un réseau neuro-flou qui est développé spécialement pour cette action. Nous développons dans ce chapitre les techniques d'apprentissage des réseaux qui permettront de les configurer et de les initialiser.

Enfin, le dernier chapitre présente l'intégration du système d'aide au diagnostic dans un ordinateur industriel grâce au langage de programmation LabVIEW. Pour déterminer les différentes utilisations du système ainsi que ses interactions avec des outils, des méthodes et des opérateurs, nous utilisons le formalisme fourni par UML. Enfin, nous montrerons l'application de notre système d'aide à la surveillance sur un système flexible de production.

# Chapitre I

# Problématique de la surveillance dans l'environnement industriel

---

*Les besoins de l'automatique et de la sûreté de fonctionnement en matière d'intelligence artificielle évoluent. En automatique, c'est la préservation et l'exploitation rationnelle du savoir-faire nécessaire à l'élaboration de systèmes complexes qui crée ce besoin. En sûreté de fonctionnement c'est la sous-utilisation des différentes méthodes d'analyse fonctionnelle et prévisionnelle qui le crée. De plus, la surveillance, partie intégrante de la sûreté de fonctionnement, est de plus en plus utilisée en automatique. En effet, malgré la conception de commandes précises et évoluées, celles-ci n'évoluent pas en même temps que le procédé. On assiste alors à une lente dégradation des performances de la commande. Ceci impose quel que soit le degré de sophistication de la partie commande, un module de surveillance. Dans cette optique, les liens entre la surveillance et l'intelligence artificielle sont réels et de nombreux travaux ont déjà été entrepris. Dans ce sens, une classification des différentes techniques utilisées en surveillance a été établi afin de positionner et de mettre en évidence les techniques en intelligence artificielle.*

---

## I.1 Introduction

La surveillance industrielle est une partie intégrante de la sûreté de fonctionnement et en constituant une fonction de plus en plus importante en pilotage des systèmes. Les besoins dans ces deux domaines évoluent constamment. Ces besoins s'expriment principalement en intelligence artificielle soit pour exploiter et préserver un savoir-faire, soit pour amener une intelligence répartie vers les niveaux opérationnels les plus bas. Ce chapitre débute par une vue générique des évolutions technologiques et des tendances dans l'automatique pour établir une vue globale sur la sûreté de fonctionnement, et pour terminer par un état de l'art des différentes techniques de surveillance.

## I.2 Évolutions technologiques et tendances dans l'automatique

L'automatique concerne les tâches de commande et de surveillance des systèmes physiques. Elle est par nature une discipline transversale, qui s'intéresse à deux grands types de procédés : les systèmes continus (c'est-à-dire décrits en temps continu) et les systèmes à événements discrets. Ses objectifs sont multiples : d'une part faire mieux, ce qui peut signifier améliorer la qualité ou la quantité du produit fabriqué, mais ce qui implique aussi et avant tout de garantir la sécurité de l'installation, des hommes et de l'environnement ; d'autre part faire moins cher, cet objectif étant parfois combiné avec le précédant pour aboutir au concept de conduite optimale ; enfin innover, puisqu'un certain nombre de systèmes ne peuvent plus se concevoir sans leur automatisation, qui en est une partie intégrante.

Les outils de base de cette discipline sont doubles. D'abord, une formalisation mathématique poussée permet de représenter le système à automatiser et sa commande par un(des) modèle(s). Les étapes habituelles d'une automatisation sont l'analyse, la modélisation, la simulation et la commande du système. Pour cela, on est amené à élaborer plusieurs modèles de nature différente (par exemple statique continu pour optimiser et dynamique échantillonné pour commander autour d'un point de fonctionnement). Un modèle mathématique est en fait un outil de compression des données puisqu'il résume en peu de paramètres tous les comportements possibles du procédé.

Ensuite, les techniques informatiques permettent de mettre en œuvre ces modèles, de synthétiser des commandes, de vérifier leurs effets en simulation puis de les implanter à travers un système d'exploitation temps réel. L'automatisation complète d'une installation complexe passe par l'utilisation d'autres disciplines fortement connexes, comme le traitement du signal, la recherche opérationnelle et l'analyse numérique.

La conception de l'automatisation d'une installation et ses essais peuvent prendre de nombreux mois de travail d'ingénieurs. L'implantation temps réel s'organise autour de différents modules. Un module d'acquisition-traitement des signaux physiques issus de l'instrumentation du procédé en donne une image à chaque période d'échantillonnage. Un module de commande en temps réel élabore à chaque période d'échantillonnage les valeurs des signaux à appliquer sur les entrées du système pour assurer, malgré les perturbations, les valeurs choisies aux sorties (valeurs précises ou gamme de valeurs). Un module de visualisation-stockage permet d'obtenir, de mettre à la disposition des opérateurs humains et de conserver un historique du fonctionnement. Enfin, un module de contrôle permet de surveiller en permanence le procédé, de détecter des situations anormales, et s'il y a lieu d'activer des automatismes d'urgence (mise en route d'un organe de secours, arrêt automatique ...) ou de laisser les opérateurs reprendre la conduite en manuel si leur savoir-faire s'avère indispensable.

Les progrès de l'automatique, ces dernières années, ont porté essentiellement sur la partie commande. Les commandes classiques supposent le système stationnaire autour d'un point de fonctionnement, descriptible par des équations aux différences ou différentielles linéaires. Elles évoluent, grâce à l'effort des laboratoires de recherche, vers des commandes adaptatives (paramètres des équations variables), des commandes robustes (tenant compte des incertitudes des modèles) ou des commandes non-linéaires (pour tenir compte des plages de fonctionnement plus larges).

Soulignons l'existence, en ce qui concerne l'automatisation des procédés complexes, de deux situations qui, pour être caricaturales, n'en sont pas moins fréquentes. L'automatisation d'un procédé devrait évoluer en même temps que celui-ci est modifié, ce qui est rarement le cas dans l'industrie, compte-tenu de la longueur des études nécessaires et du manque de compétences sur site. On assiste alors à une lente dégradation des performances de la commande, à une nécessité de plus en plus fréquente de la reprise en manuel par les opérateurs, ce qui laisse croire à l'existence d'une nécessaire « expertise » humaine pour régler les problèmes de commande pourtant classiques. Les avancées des méthodes permettant d'englober des fonctionnements complexes dans des modèles mathématiques peuvent faire oublier que toutes ces représentations sont basées sur des hypothèses forcément restrictives et que de toutes façons, toutes les sources de pannes et de dysfonctionnements ne sont pas modélisables de façon exhaustive ; ceci impose, quel que soit le degré de sophistication de la partie commande, un module de surveillance, de détection de défaut et de prise de décision en cas de défaillance (Åström *et al.*, 1986). Par « défaut » on entend ici non seulement une panne franche, mais aussi une dégradation des performances (par exemple un écart transitoire par rapport à une trajectoire nominale, dû à une perturbation).

Sur un procédé industriel, la supervision peut recouvrir l'analyse de plusieurs centaines d'informations en moins d'une minute. Par exemple, une plate-forme pétrolière est surveillée par l'intermédiaire de 500 variables analogiques et 2500 variables logiques, une

avalanche d'alarmes peut mettre en jeu 500 variables en 1 minute et les opérateurs ont à régler un problème mineur toutes les demi-heures et un problème majeur par semaine. Compte tenu du coût de l'arrêt d'une telle installation, les industriels préfèrent investir actuellement dans un bon système de contrôle que dans une commande sophistiquée.

Si l'automatique fait appel à un arsenal mathématique parfois complexe, sa mise en application nécessite aussi un grand savoir-faire. Pour préserver et exploiter rationnellement un tel savoir-faire l'automaticien peut penser que l'intelligence artificielle lui sera utile (Boullart *et al.*, 1992). Les besoins de l'Automatique en matière d'Intelligence Artificielle semblent bien réels, notamment en conception assistée de systèmes de commande, et en supervision, mais aussi en régulation comme semble le suggérer l'effervescence actuelle autour de la commande floue, des réseaux neuro-mimétiques et des « smart-controllers » (régulateurs intégrant toute une logique de décision qui conduit à leur auto-réglage). En CAO, on assiste à une évolution des outils vers des progiciels intégrant le savoir-faire de l'automaticien dans des bases de connaissances couplées aux produits algorithmiques classiques. Ces outils commencent à évoluer à leur tour vers des systèmes expliquant ce savoir-faire, dans un souci de formation. En matière de supervision, la sécurité des hommes et de leur environnement dans les installations industrielles requiert une surveillance très poussée du procédé physique et de son système de contrôle-commande. Une modélisation purement quantitative et exhaustive de toutes les situations possibles est pratiquement infaisable. Le raisonnement qualitatif est envisagé pour proposer des méthodes de simulation ou de diagnostic à un niveau de précision adéquat.

En synthèse, les besoins de l'automatique évoluent vers une intelligence répartie embarquée de plus en plus vers les niveaux opérationnels les plus bas. Les besoins en supervision et donc en surveillance en matière d'Intelligence Artificielle sont par conséquent bien présents. S'il est vrai que la surveillance est un domaine de l'Automatique, il appartient essentiellement au domaine de la Sûreté de Fonctionnement que nous nous proposons d'étudier brièvement par la suite.

## I.3 Éléments de base de la sûreté de fonctionnement

La sûreté de fonctionnement a acquis sa notoriété et sa forme actuelle principalement au cours du dernier demi-siècle et dans les secteurs de la défense, de l'aéronautique, de l'espace, du nucléaire, puis des télécommunications et des transports. Ce domaine se montre de plus en plus indispensable à tous les secteurs de l'industrie.

Par définition, la sûreté de fonctionnement consiste à connaître, évaluer, prévoir, mesurer et maîtriser les défaillances des systèmes technologiques et les défaillances humaines. Nous allons présenter dans ce qui suit les notions fondamentales à la maîtrise de cette discipline et les outils et démarches associées.

## I.3.1 Notions fondamentales

La démarche, le raisonnement « sûreté de fonctionnement » s'appuient sur quelques notions de base. Parcourir ce vocabulaire de base est donc une introduction classique à la sûreté de fonctionnement.

### I.3.1.1 Sûreté de fonctionnement

La sûreté de fonctionnement est l'aptitude d'une entité à satisfaire une ou plusieurs fonctions requises dans des conditions données. On notera que ce concept peut englober la fiabilité, la disponibilité, la maintenabilité, la sécurité, la durabilité ... ou des combinaisons de ces aptitudes. Au sens large, la Sûreté de fonctionnement est considérée comme la science des défaillances et des pannes. (Villemeur, 1988)

La sûreté de fonctionnement est souvent définie comme :
- FMDS (Fiabilité, Maintenabilité, Disponibilité et Sécurité) ;
- Science des défaillances ;
- Analyse de risque.

Elle se caractérise à la fois par les études structurelles statiques et dynamiques des systèmes, du point de vue prévisionnel mais aussi opérationnel et expérimental (essais, accidents), en tenant compte des aspects probabilités et des conséquences induites par les défaillances techniques et humaines. Cette discipline intervient non seulement au niveau de systèmes déjà construits mais aussi au niveau conceptuel pour la réalisation des systèmes.

### I.3.1.2 Défaillance, Dégradation, Panne

Suivant l'AFNOR, une défaillance est l'altération ou la cessation de l'aptitude d'un ensemble à accomplir sa ou ses fonction(s) requise(s) avec les performances définies dans les spécifications techniques. L'ensemble est indisponible suite à la défaillance.

La cessation de l'aptitude conduit l'entité à être dans un état appelé panne.

Un ensemble est défaillant si ses capacités fonctionnelles sont interrompues (panne ou arrêt volontaire par action d'un système interne de protection ou une procédure manuelle équivalente). Dans le cas d'une dégradation sans perte totale de la fonction, on considère qu'il s'agit d'une défaillance si sa performance tombe au dessous d'un seuil défini, lorsqu'un tel seuil minimal est contenu dans les spécifications fonctionnelles du matériel.

Il s'ensuit qu'un ensemble est défaillant s'il est considéré ou déclaré incapable d'assurer les fonctions requises par l'exploitant utilisant des critères fonctionnels simples. Toute étude de fiabilité implique l'acceptation de deux états totalement exclusifs : le

fonctionnement normal et le fonctionnement défaillant. Les passages d'un état de fonctionnement normal à un état défaillant pouvant se manifester en fonction du temps de manière progressive, soudaine ou de façon aléatoire, la fiabilité ne connaît pas la notion de défaillance partielle ou progressive. La figure I.1 représente trois cas conduisant tous à une défaillance.

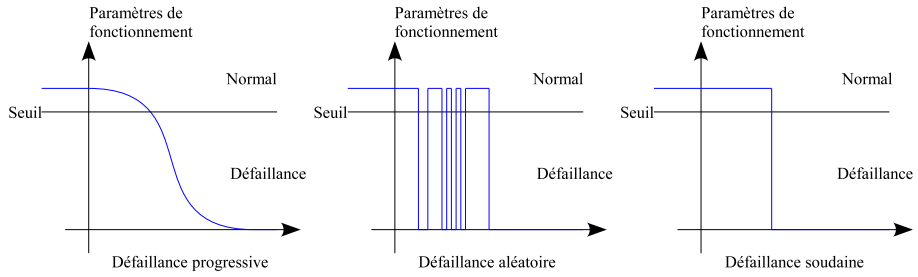



FIG. I.1 – Cas de figure conduisant à une défaillance

#### I.3.1.3 Maintenance

Suivant la norme AFNOR NF X 60010, la maintenance est définie comme toutes les activités destinées à maintenir ou à rétablir un bien dans un état ou dans des conditions données de sûreté de fonctionnement pour accomplir une fonction requise. Ces activités sont une combinaison d'activités techniques, administratives et de management.

On peut citer trois types de maintenance :

- la maintenance corrective qui est exécutée après la détection d'une panne et qui est destinée à remettre un bien dans un état dans lequel il peut accomplir une fonction requise. On peut trouver la maintenance corrective différée qui est retardée en accord avec des règles de maintenance données, et la maintenance corrective d'urgence qui est exécutée sans délai afin d'éviter des conséquences inacceptables.
- la maintenance préventive qui est destinée à réduire la probabilité de défaillance d'un bien ou la dégradation d'un service rendu. C'est une intervention de maintenance prévue, préparée et programmée avant la date probable d'apparition d'une défaillance. Elle peut être programmée ou systématique, c'est-à-dire une maintenance préventive effectuée selon un calendrier préétabli ou selon un nombre défini d'unités d'usage (sans contrôle préalable pour une maintenance préventive systématique).
- la maintenance conditionnelle fait l'objet d'une demande croissante dans un grand nombre d'applications industrielles. Cette maintenance est basée sur la surveillance en continu de l'évolution du système, afin de prévenir un dysfonctionnement avant qu'il n'arrive. Elle n'implique pas la connaissance de la loi de dégradation. La décision d'intervention préventive est prise lorsqu'il y a évidence expérimentale de

défaut imminent, ou approche d'un seuil de dégradation prédéterminé. Elle impose donc des traitements en ligne, au moins en partie.

#### I.3.1.4 Risque

Un risque est un événement redouté évalué en terme de fréquence et de gravité. Le diagramme fréquence-gravité, aussi appelé diagramme de Farmer (figure I.2), est la façon la plus pratique d'illustrer et de communiquer sur le risque. On y trouve la limite d'acceptation des risques qui relie les points à la frontière entre l'acceptable et l'inacceptable. Son tracé, sa forme, est l'expression très lisible des critères d'acceptation des risques. Un risque placé au-dessus d'elle doit être réduit pour devenir acceptable :

- soit on en réduit la fréquence, c'est faire de la prévention ;
- soit on en réduit la gravité, c'est faire de la protection.

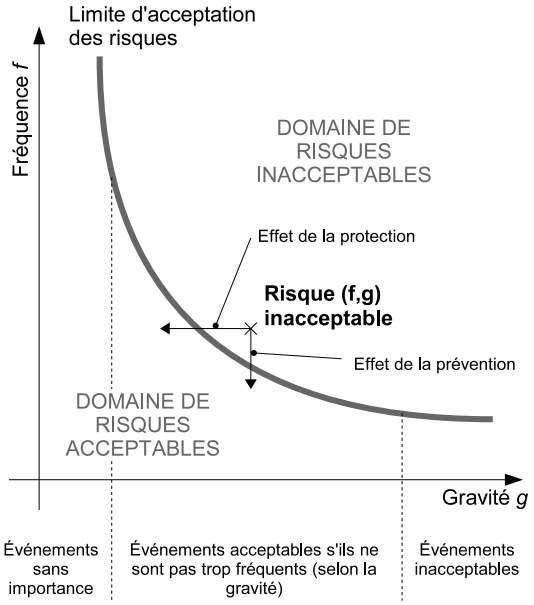



Fig. I.2 – Diagramme de Farmer

#### I.3.1.5 Fiabilité (*Reliability*)

La norme AFNOR NF X 60-500 définit la fiabilité comme l'aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné.

La notion de fiabilité s'interprète comme la probabilité $R(t)$ que l'entité considérée soit non défaillante sur la durée $[0, t]$ sachant qu'elle n'est pas défaillante à l'instant 0.

Un exemple de mesure de fiabilité est le taux de défaillance, inverse du MTTF (*Mean Time To Failure* : temps moyen jusqu'à la première défaillance)

#### I.3.1.6 Maintenabilité (*Maintenability*)

Dans les conditions données d'utilisation, l'AFNOR définit la maintenabilité comme étant l'aptitude d'une entité à être maintenue ou rétablie, sur un intervalle de temps donné, dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des procédures et des moyens prescrits.

On notera que la norme américaine MIL-STD-721C est presque identique dans sa formulation mais inclut le niveau requis de qualification des personnels : « La maintenabilité est la mesure de l'aptitude d'un dispositif (« item ») à être maintenu ou remis dans des conditions spécifiées lorsque la maintenance de celui-ci est réalisée par des agents ayant les niveaux spécifiés de compétence, utilisant les procédures et les ressources prescrites, à tous les niveaux prescrits de maintenance et de réparation ».

Elle se caractérise par la probabilité $M(t)$ que la maintenance de l'entité considérée accomplie dans des conditions données, avec des procédures et des moyens prescrits, soit effectuée sur la durée $[0, t]$ sachant qu'elle est défaillante à l'instant 0.

Un exemple de mesure de maintenabilité est le MTTR (*Mean Time To Recover* : temps moyen de réparation ou de restauration du système dans l'état de bon fonctionnement)

#### I.3.1.7 Disponibilité (*Availability*)

La norme AFNOR X 60-500 définit la disponibilité comme « l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens extérieurs nécessaires de maintenance soit assurée ».

Elle se caractérise par la probabilité A(t) d'être, à l'instant $t$, en état d'accomplir les fonctions requises.

#### I.3.1.8 Sécurité (*Safety*)

« Aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques. » (Villemeur, 1988)

Il est important de noter qu'en français, la terminologie ne fait pas la différence entre « security » et « safety ». Le terme « security » concerne les aspects réglementaires de la

sécurité (respects des normes, contrôle des accès à des locaux ou à des systèmes informatiques) tandis que le terme « safety » enseigné aux États-Unis sous le nom « d'industrial safety » recouvre les aspects techniques de la sécurité. En sûreté de fonctionnement, nous sommes plus proches de « safety ».

#### I.3.1.9 Études de sûreté de fonctionnement

Les études de sûreté visent essentiellement à évaluer la probabilité de l'occurrence d'un événement indésirable en prenant en compte dès la conception tous les facteurs initiateurs :

- facteurs techniques : matériels et produits manipulés (incluant les problèmes de conception, de fabrication, d'assurance qualité, de conduite et de maintenance) ;
- facteurs humains : qualité de la formation, ergonomie, procédures ;
- facteurs environnementaux : risques naturels, milieux ambiants (poussières, gaz, électricité statique ...).

Les études de sécurité, où la maintenance joue un rôle non négligeable dans la mesure où de nombreux accidents sont liés à des défaillances techniques ou humaines, couvrent un spectre technique étendu. Les méthodes utilisées en sécurité doivent en particulier :

- identifier les modes de fonctionnement anormaux pouvant conduire à une situation dangereuse ;
- analyser la combinaison et l'enchaînement d'événements peu probables, pris isolément, qui conduisent à des accidents. L'expérience montre, en effet, que de nombreuses catastrophes ont été le résultat de séquences de défaillances mineures (techniques ou humaines) ;
- évaluer la probabilité d'occurrence d'un accident et lui assigner une gravité sur une échelle appropriée pour juger si le risque est acceptable économiquement ou écologiquement compte tenu des enjeux de la mission ;
- maintenir le risque à son niveau acceptable grâce, par exemple, à la maîtrise de la fiabilité des matériels obtenue par des politiques efficaces de maintenance.

La maîtrise des risques représente une discipline à part entière et fait l'objet d'ouvrages spécialisés et de techniques spécifiques des industries concernées (transport, chimie, nucléaire ...).

La figure I.3 résume les liens entre fiabilité, maintenabilité, disponibilité et sécurité.

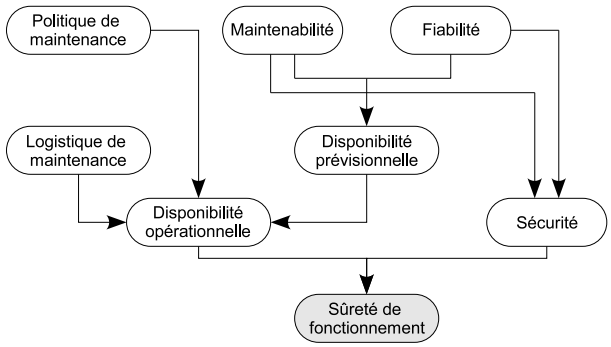



FIG. I.3 – Relations entre fiabilité, maintenabilité, disponibilité et sécurité

### I.3.2 Démarches et méthodes d'une approche Sûreté de Fonctionnement

Ces méthodes s'encadrent dans l'analyse prévisionnelle des dysfonctionnements des systèmes qui consiste à identifier les conditions qui peuvent conduire à des défaillances et à prévoir leurs conséquences sur la fiabilité, la maintenabilité, la disponibilité et la sécurité des systèmes en cours de conception ou déjà opérationnels. Elle est réalisée à partir d'informations diverses dont le tri et l'analyse permettent de concevoir un modèle du système.

Les informations nécessaires à l'analyse sont :

- la description du système réel : structures physiques et fonctionnelles ;
- les caractéristiques des composants du système et leurs interactions (modes de défaillance et leurs conséquences ...) ;
- les relations entre le système et son environnement ;
- la prise en compte des erreurs humaines en phase d'exploitation.

#### I.3.2.1 Méthodes d'analyse fonctionnelle

Les méthodes d'analyse fonctionnelle sont indispensables pour réaliser une décomposition fonctionnelle et matérielle d'une installation industrielle en cours de conception ou en fonctionnement. L'utilisation de ces méthodes, par leurs caractères systématiques et exhaustifs, représente une garantie formelle pour décomposer une installation industrielle en niveaux fonctionnels et matériels nécessaires pour identifier les modes de défaillances et leurs conséquences sur les objectifs opérationnels retenus pour l'installation ou l'équipement concerné.

Les méthodes d'analyse fonctionnelle permettent :

- de décrire le besoin d'un utilisateur en termes de fonctions, en faisant abstraction des solutions pouvant les réaliser ;
- de décrire les choix technologiques que l'on impose au concepteur, en termes de contraintes ;
- de s'assurer, pour chaque fonction, de sa bonne expression en termes d'objectifs et de sa stabilité dans le temps ;
- de décrire le produit envisagé comme solution, en termes de fonctions de services et en termes de fonctions techniques ou de conceptions ;
- d'initialiser l'optimisation du produit aussi bien du point de vue coût que du point de vue fiabilité.

Les résultats des analyses fonctionnelles sont matérialisés par trois éléments :

- le cahier des charges fonctionnel (CdCF), pour l'expression des besoins ;
- le bloc diagramme fonctionnel (BdF), pour la relation fonctions / solutions ;
- le tableau d'analyse fonctionnelle (TAF), qui sert de trame pour la réalisation d'analyses des modes de défaillance, de leurs effets et de leurs criticité (AMDEC).

Les principales méthodes d'analyse fonctionnelle sont :

- La méthode FAST :
  - Principe général : Instrument graphique de communication entre les intervenants, FAST porte sur les relations entre produits et fonctions. Cet instrument permet de représenter la logique des relations entre les fonctions par répétition des trois questions « Pourquoi ? », « Comment ? » et « Quand ? » posés à chaque étape de l'analyse. FAST présente les relations existant entre produits et fonctions à l'intérieur d'un domaine strictement délimité.
  - Originalités : FAST produit un diagramme qui permet au concepteur d'expliquer et de justifier les solutions techniques.
- la méthode RELIASEP® :
  - Principe général : A partir du besoin exprimé en terme de fonctions, rechercher les sous-fonctions nécessaires à la satisfaction de ce besoin (arbre fonctionnel). Cette recherche est progressive et adaptée aux phases de développement.
  - Originalités : Très orientée Sûreté de Fonctionnement, cette méthode intègre les exigences de SDF à toutes les étapes de la vie d'un produit. L'arbre fonctionnel sert de base à l'Analyse des Modes de Défaillance, de leurs Effets et leur Criticité (AMDEC). Cette méthode permet de visualiser le cheminement fonctionnel des dégradations. RELIASEP se révèle efficace pour l'étude de systèmes complexes plutôt « matériels » que « logiciels ».
- la méthode SADT® :
  - Principe général : Modélise des systèmes existants ou futurs pour en comprendre le fonctionnement et envisager des solutions. Cette modélisation porte sur les actions du système analysé (actigrammes), et sur les données que ce système doit traiter (datagrammes) dans une structure arborescente de ce système.

  - Originalités : Plutôt orientée systèmes d'information, logiciels et automatismes. SADT met en jeu deux types d'acteurs : un « auteur » qui conçoit le système et un « lecteur » qui le « critique ». Ces travaux permettent de préciser et d'optimiser le système par approche itérative.
- la méthode APTE® :
  - Principe général : Ensemble d'outils méthodologiques mis à disposition des équipes de projets pour maîtriser la cohérence entre les différents éléments d'un projet, choisir les technologies performantes et fiables pour satisfaire les besoins au moindre coût. Cette méthode propose des logiques de formalisation communes pour les phases conceptuelles de projets complexes.
  - Originalités : Très systémique dans le sens où le produit est conçu à partir de son milieu environnant en privilégiant le point de vue du concepteur et de l'utilisateur, la méthode APTE comporte ses propres outils internes de validation (besoin, fonctions, contraintes ...). APTE propose une méthode d'animation de groupes efficace. L'emploi de cette méthode est très répandu dans la DGA.

D'autres méthodes de description des systèmes peuvent être également mises en œuvre :

- MERISE : pour décrire et concevoir des systèmes d'informations ;
- ASA, langage LSA : utilisée pour la formalisation de spécifications et le maquettage de systèmes temps réel industriels) ;
- réseaux de PETRI adaptés à la méthode des espaces des états, pour les arbres d'événements et les diagrammes causes-conséquences ;
- méthode MIL-STD-1629 pour la réalisation des AMDEC ;
- méthode des flux pour la réalisation de blocs-diagrammes fonctionnels et de tableaux d'analyse fonctionnelle ;
- méthode SA (structured analysis) ou méthode de Yourdon-De Marco pour les systèmes de traitement de l'information.

#### I.3.2.2 Méthodes d'analyse prévisionnelle

Les méthodes d'analyse prévisionnelle se répartissent en deux grandes familles qui se différencient par les techniques de raisonnement :

- les méthodes inductives (bottom to top) partent des causes des défaillances et remontent jusqu'aux conséquences que l'on souhaite éviter ;
- les méthodes déductives sont au contraire des méthodes descendantes (top to bottom) : on part de l'événement indésirable et on recherche toutes les causes susceptibles d'entraîner cet événement.

Une famille est exclusive de l'autre, mais une approche par une méthode inductive est judicieusement complétée par une méthode déductive et réciproquement.

Il n'y a pas de recherche de quantification sans analyse qualitative. Par contre, il peut y avoir analyse qualitative sans quantification. Nous avons donc qualifié de quantitatives les méthodes qui offraient une possibilité importante de quantification (de fréquence) et de qualitatives les méthodes qui l'excluaient ou dans lesquelles cet aspect est marginal.

Le tableau I.1 regroupe les principales méthodes d'analyse prévisionnelle en indiquant leur technique de raisonnement (inductive ou déductive), si elles sont quantitatives ou qualitatives et leurs fonction. Cette dernière peut être très réductrice.

Tab. I.1 – Méthodes d'analyse prévisionnelles

| Méthodes | Inductive/ Déductive | Quantitative/ Qualitative | Fonction |
|---|---|---|---|
| Analyse des Modes de Défaillance et de leurs Effets (AMDE) | Inductive | Qualitative | Recenser les conséquences des défaillances |
| Analyse des Modes de Défaillance et de leurs Effets (AMDEC) | Inductive | Quantitative | Évaluer les conséquences des défaillances |
| Analyse Préliminaire des Risques (APR) | Inductive | Qualitative | Repérer a priori les risques à étudier |
| Arbre de causes | Déductive | Qualitative | Organiser les éléments ayant contribué à un accident |
| Arbre d'événements | Inductive | Quantitative | Évaluer les conséquences possibles d'un événement |
| Arbre de Défaillances | Déductive | Quantitative | Évaluer les scénarios d'un accident potentiel |
| Graphes d'état | Inductive | Quantitative | Évaluer les états possibles d'un système réparable |

Il n'y a pas de règle qui fabrique la méthode universellement idéale en assemblant ces méthodes, mais leurs caractéristiques majeures leur assignent des rôles privilégiés.

– L'APR est une démarche introductive qui permet de mieux focaliser ses efforts d'analyse de risque. Elle est précieuse en sécurité quand la préoccupation d'être exhaustif dans l'identification des risques est forte et difficile à satisfaire.
– L'AMDEC est la démarche qui permet de réunir et d'exploiter des connaissances disponibles sur les défaillances des éléments composant le système étudié. Elle a la réputation d'être exhaustive, ce qui est justifié si la connaissance des modes de

défaillance au niveau où l'analyse est conduite est elle-même exhaustive et si la prise en compte des modes de défaillance suffit à atteindre les objectifs de l'étude.
- L'arbre de causes est la méthode la plus propre à réunir l'ensemble des éléments explicatifs d'un accident (sans se limiter à ce qui peut être qualifié de causes).
- L'arbre d'événements est adapté pour traiter la question « que peut-il arriver si ... ? ».
- L'arbre de défaillances est une représentation synthétique des scénarii d'accidents. Contrairement à l'AMDEC, il rend bien compte des combinaisons de pannes ou d'événements. Comme l'AMDEC il ne rend pas compte des aspects temporels. Il ne peut être construit que si on sait répondre aux questions : « qu'est ce qui peut conduire à ... ? ».
- Les graphes d'états sont particulièrement adaptés à l'évaluation de la disponibilité d'un système réparable ; les méthodes précédentes, prenant difficilement en compte les réparations, sont plus adaptées à l'évaluation de la fiabilité (ou de la maintenabilité) ou à une approche sécurité.

En synthèse, nous pouvons dire que les méthodes de la sûreté de fonctionnement représentent bien le système même si elles représentent de nombreuses heures de travail. Pourtant, leur utilisation est délicate et fastidieuse pour les opérateurs. Néanmoins, elles peuvent apporter une aide précieuse à leur déduction dans la recherche de causes d'une défaillance, et même pour la surveillance d'un système.

## I.4 Méthodes de surveillance industrielle

Dans un grand nombre d'applications industrielles, une demande croissante est apparue en matière de remplacement des politiques de maintenance curative par des stratégies de maintenance préventive. Cette mutation d'une situation où on « subit les pannes » à une situation où on « maîtrise les pannes », nécessite quelques moyens technologiques ainsi que la connaissance de techniques d'analyse appropriées. La fonction surveillance en continu de l'évolution de l'équipement à travers des données quantifiables et qualifiables, permet ainsi de prévenir un dysfonctionnement avant qu'il n'arrive et d'écarter les fausses alarmes qui peuvent ralentir la production (Basseville *et al.*, 1996).

Suivant (Lefebvre, 2000), la surveillance est un dispositif passif, informationnel qui analyse l'état du système et fournit des indicateurs. La surveillance consiste notamment à détecter et classer les défaillances en observant l'évolution du système puis à les diagnostiquer en localisant les éléments défaillants et en identifiant les causes premières.

Nous présentons ci-après et sur la figure I.4, les techniques les plus courantes en surveillance d'équipements industriels.

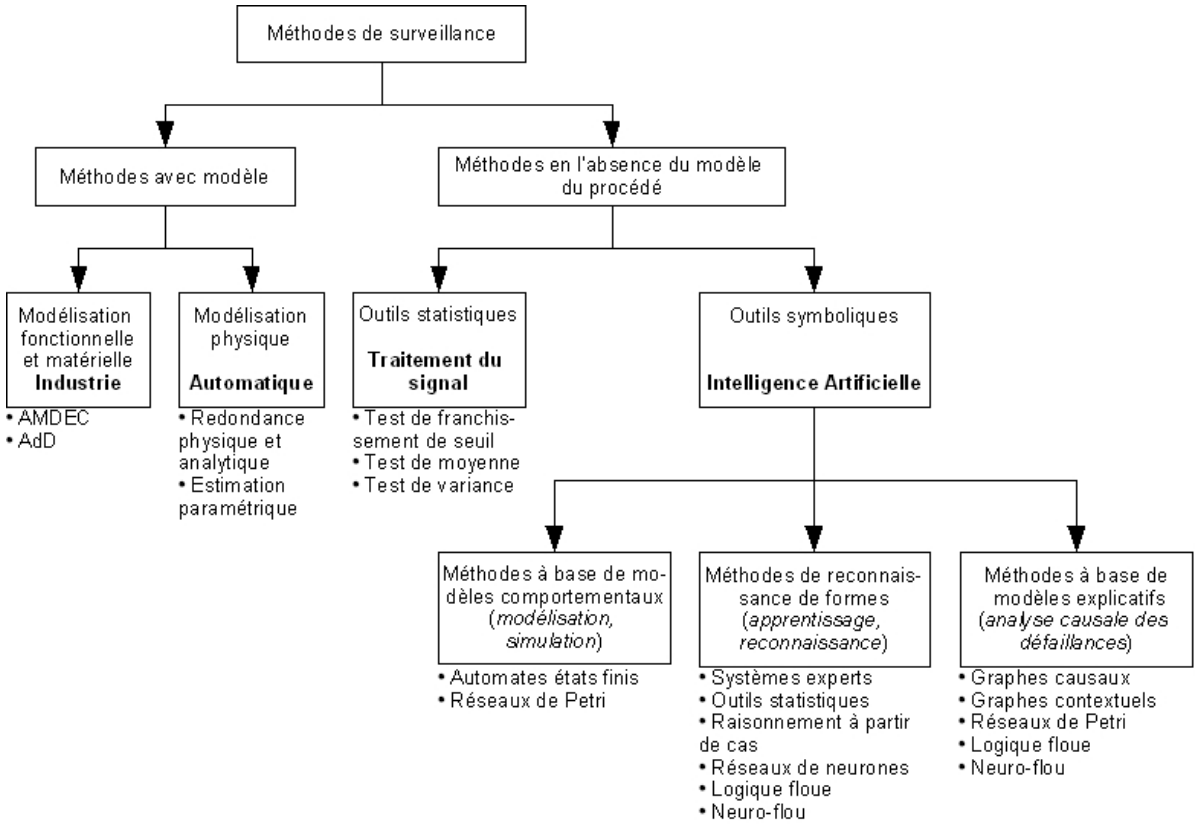



FIG. I.4 – Classification des méthodes de surveillance industrielle

L'existence d'un modèle formel ou mathématique de l'équipement détermine la méthode de surveillance utilisée (Dash *et al.*, 2000; Monnin, 2004; Zemouri, 2003). La surveillance avec modèle se compose essentiellement de deux types : méthodes par modélisation fonctionnelle et matérielle, et les méthodes par modélisation physique. Mais, nombreuses sont les applications industrielles dont le modèle est difficile, voire impossible à obtenir suite à une complexité accrue ou à de nombreuses reconfigurations intervenants durant le processus de production. Pour ce type d'applications industrielles, les seules méthodes de surveillance opérationnelles sont celles sans modèle (Dubuisson, 2001). Ces méthodes se divisent en deux catégories : méthodes utilisant des outils statistiques et méthodes symboliques de l'Intelligence Artificielle.

### I.4.1 Méthodes par modélisation fonctionnelle et matérielle

Le principe de ces méthodes consiste à établir *a priori* et de la manière la plus complète possible, les liens entre les causes initiales des défaillances et leurs effets mesurables. Les méthodes les plus couramment rencontrées sont l'Analyse des Modes de

Défaillance, de leurs Effets et de leurs Criticité (AMDEC) et les Arbres de Défaillances (AdD). Nous les avons déjà vu comme des méthodes d'analyse prévisionnelle de sûreté de fonctionnement (Chapitre I.3.2.2).

## I.4.2 Méthodes par modélisation physique

Les méthodes à base de modèle sont les plus familières aux automaticiens. Elles supposent une connaissance approfondie du procédé sous forme de modèle numérique. Elles ont pour principe de comparer les mesures effectuées sur le système aux informations fournies par le modèle (Frank, 1990). Tout écart est alors synonyme d'une défaillance. Les outils de la théorie de la décision sont ensuite utilisés pour déterminer si cet écart est dû à des aléas normaux comme, par exemple, le bruit de mesure ou s'il traduit une défaillance du système. Ces méthodes peuvent être séparées en deux techniques : techniques de redondance physique et analytique et techniques d'estimation paramétrique (Zemouri, 2003). Ces deux techniques seront présentées brièvement.

### I.4.2.1 Redondances physiques et analytiques

La redondance physique permet de distinguer les défaillances capteurs des défaillances système afin de rendre fiable la détection des défaillances à partir des signaux mesurés. Physiquement, il s'agit de doubler ou tripler des composantes de mesure du système. Si ces composantes identiques placées dans le même environnement émettent des signaux identiques, on considère que ces composants sont dans un état de fonctionnement nominal et, dans le cas contraire, on considère qu'une défaillance capteur s'est produite dans au moins une des composantes (Zhang, 1999). Si cette méthode est simple, elle a un inconvénient certain, c'est son coût de mise en œuvre.

Par opposition la redondance analytique utilise les mesures disponibles sur le système. Ces algorithmes sont ou non basés sur un modèle du système. Ces méthodes fournissent une estimation de l'état du système. La comparaison avec son état réel fournit alors une quantité appelée résidu qui va servir à déterminer si le système est dans un état défaillant ou non. Elles doivent bien sûr tenir compte de variations normales du comportement du système, des bruits de mesures, de perturbations externes ainsi que des erreurs de modélisation, pour éviter les fausses alarmes ou les manques à la détection.

### I.4.2.2 Méthodes d'estimation paramétrique

Les méthodes d'estimation paramétrique supposent l'existence d'un modèle paramétrique décrivant le comportement du système et la connaissance des valeurs de ces paramètres en fonctionnement nominal. Elles consistent alors à identifier les paramètres

caractérisant le fonctionnement réel, à partir de mesures des entrées et des sorties du système (Willsky, 1976). On dispose ainsi d'une estimation des paramètres du modèle, effectuée à partir des mesures prises sur le système et de leurs valeurs théoriques. La différence entre les méthodes de redondance analytique et les méthodes d'estimation paramétrique se situe au niveau de la comparaison entre les estimations et la théorie. Dans les premières méthodes, on compare les états tandis que dans les secondes, on étudie la différence entre les paramètres. Les méthodes d'estimation paramétrique requièrent donc l'élaboration d'un modèle dynamique précis du système à surveiller. Ceci restreint leur utilisation à des procédés bien définis.

### I.4.3 Méthodes par outils statistiques

Les outils statistiques de détection de défaillances consistent à supposer que les signaux fournis par les capteurs possèdent certaines propriétés statistiques. On effectue alors quelques tests qui permettent de vérifier si ces propriétés sont présentes dans un échantillon des signaux mesurés de taille $n$ (appelé fenêtre d'observation glissante). Nous ne présentons que trois tests statistiques, mais une grande variété de tests, applicables sur un échantillon de mesures, peut être trouvée dans (Basseville, 1988).(Zemouri, 2003)

#### I.4.3.1 Test de franchissement de seuil

Le test le plus simple est de comparer ponctuellement les signaux avec des seuils préétablis. Le franchissement de ce seuil par un des signaux capteurs génère une alarme.

#### I.4.3.2 Test de moyenne

Contrairement à la méthode précédente, le test de comparaison est effectué sur la moyenne du signal contenu dans une fenêtre de $n$ valeurs plutôt que sur une valeur ponctuelle.

#### I.4.3.3 Test de variance

On peut également calculer la variance d'un signal. Tant que cette variance se situe dans une bande située autour de sa valeur nominale, l'évolution du système est supposée normale.

### I.4.4 Méthodes par outils symboliques

Ces méthodes s'appuient largement sur les techniques de l'Intelligence Artificielle (IA) et font appel à des connaissances symboliques, familières ou au moins partageables par l'opérateur (Basseville *et al.*, 1996). En effet, l'utilisation de l'Intelligence Artificielle permet de pallier à la complexité des systèmes à surveiller. De plus, d'une manière générale, l'Intelligence Artificielle de part ces caractéristiques est relativement bien adaptée aux problèmes de surveillance. En effet, l'IA peut se caractériser par la capacité de traiter (Monnin, 2004) :

- une grande quantité d'information,
- des données non homogènes (numériques/symboliques),
- des données dépendant du contexte,
- des données incomplètes.

Globalement, ces méthodes seront regroupées sous l'expression : « Méthodes par modélisation symbolique ». Elles sont caractérisées par des mots clés qui rendent mieux compte de leurs objectifs et leurs particularités respectifs. On distingue donc parmi les modèles symboliques, les méthodes de reconnaissance, les méthodes à base de modèles comportementaux et les méthodes à base de modèles explicatifs. (Monnin, 2004)

#### I.4.4.1 Méthodes à base de modèles comportementaux

Ces méthodes se caractérisent par la possibilité notamment de simuler le comportement du système, à partir d'une modélisation de son comportement. Le plus souvent il s'agit de modèles « *de bon fonctionnement* » qui contrairement aux modèles numériques ne sont pas basés sur la physique du système mais sur une considération en terme de mode de fonctionnement. Ces méthodes se caractérisent par les termes *modélisation* et *simulation*. Elles regroupent notamment des outils tels que :

- les réseaux de Petri,
- les automates d'états finis.

#### I.4.4.2 Méthodes de reconnaissance

Ces méthodes regroupent les modèles associatifs et les méthodes de reconnaissance dans le sens où elles sont caractérisées par les termes *apprentissage* et *reconnaissance* qui s'appliquent aussi bien aux systèmes de reconnaissance de formes qu'aux systèmes à base de règles tels que les systèmes experts. Dans ces méthodes on considérera particulièrement les outils suivants :

- les réseaux neuronaux,
- la logique floue,
- les réseaux neuro-flous,

- les systèmes experts,
- le raisonnement à partir de cas,
- les outils statistiques.

#### I.4.4.3 Méthodes à base de modèles explicatifs

Ces méthodes reprennent le terme de modèles explicatifs introduit dans (Basseville *et al.*, 1996), et englobent de part leur définition même les notions de modèles de pannes de (Aghasaryan *et al.*, 1997). Il s'agit donc ici des méthodes qui permettent de fournir une représentation de l'analyse causale des liens entre les défaillances et leurs causes. On retrouve principalement dans ces méthodes des applications basées sur des outils tels que :

- les graphes d'influence,
- les graphes causaux,
- les graphes contextuels,
- les Bond Graph,
- la logique floue.

En synthèse, nous avons choisi de définir que le classement des techniques de surveillance est fonction de l'existence ou non d'un modèle formel de l'équipement à surveiller. Nous avons donc présenté, d'une part, les méthodes qui ne se basent pas sur l'existence de ce modèle, c'est-à-dire les outils statistiques et les techniques de l'intelligence artificielle et, d'autre part, celles qui l'utilisent, à savoir les méthodes de modélisation fonctionnelle et matérielle et les méthodes de modélisation physique. Ces dernières techniques ont pour principe de comparer l'état théorique du système fourni par le modèle avec son état courant donné par les observations. Ces techniques sont pourtant difficiles à mettre en œuvre et ont des problèmes d'incertitudes dus à la complexité des systèmes et de l'influence de l'extérieur sur ceux-ci. Les techniques de l'Intelligence Artificielle ne se basent pas sur le modèle de l'équipement et prennent en compte les perturbations ainsi que les bruits de mesure, d'une manière implicite. La surveillance à base de modèle est souvent opérée hors ligne, empêchant ainsi des traitements temps réel. En revanche, l'Intelligence Artificielle offre des outils totalement découplés de la structure du système, permettant un suivi temps réel de l'évolution de celui-ci. Le raisonnement en ligne fait que l'approche de l'Intelligence Artificielle est plus robuste à des changements de modes opératoires comme pour les systèmes ayant plusieurs configurations. Elle est donc évolutive.

Les systèmes de surveillance par outils de l'Intelligence Artificielle peuvent donc représenter d'excellents systèmes d'aide à la décision pour l'expert humain.

## I.5 Conclusion

Dans ce chapitre, nous avons pu voir dans un premier temps l'évolution dans l'Automatique qui conduit à concevoir des systèmes de surveillances performants. Ces systèmes de surveillances doivent traiter un nombre important de données, et doivent pouvoir exploiter le savoir-faire des automaticiens. Nous en avons conclu que les besoins en Intelligence Artificielle étaient bien présents. Puis nous avons abordé le point de vue de la sûreté de fonctionnement où nous avons présenté divers outils nécessaires pour améliorer ses quatre caractéristiques, la fiabilité, la maintenabilité, la disponibilité et la sécurité. Ces techniques ont les désavantages d'être coûteuses en temps, et de ne pas être souvent utilisées par les opérateurs de maintenance, d'où la question de savoir quels sont les outils utilisés en surveillance. Cette partie a été abordée dans la dernière section de ce chapitre. Nous en avons conclu que les systèmes d'intelligence artificielle étaient la meilleure solution pour l'aide à la décision pour l'opérateur de maintenance. Donc, nous avons un besoin d'outils en Intelligence Artificielle qui devront exploiter aussi bien le savoir-faire des automaticiens que les données des méthodes de sûreté de fonctionnement, dans un cadre de surveillance industrielle.

Dans ce sens, le chapitre suivant sera consacré à la présentation de différentes techniques d'intelligence artificielle rencontrées dans le domaine de la surveillance. Ceci en vue d'en extraire les points forts et les points faibles de chacune de ces techniques et d'évaluer dans quelle mesure elles répondent aux exigences de la surveillance.

# Chapitre II

# Outils de l'Intelligence Artificielle pour la surveillance

---

*De nombreuses techniques d'Intelligence Artificielle sont utilisées en surveillance. Nous en avons extrait trois grandes familles : les méthodes à base de modèles comportementaux, les méthodes de reconnaissance de formes et les méthodes à base de modèles explicatifs. La surveillance peut être définie de plusieurs manières. Nous avons fait le choix de la décrire en deux étapes : la détection de défaut et le diagnostic de pannes. Le diagnostic peut lui même être défini par une localisation et une identification des causes. Cette définition nous amène au fait que toutes les méthodes dites de surveillance ne remplissent pas entièrement la fonction de recherche de causes qui nous paraît primordiale. Selon la classification que nous avons effectuée, seules les techniques à base de modèles explicatifs répondent à ce critère. Ainsi, en respectant les fonctions naturelles de la surveillance, notre outil se décomposera en deux outils : un outil pour la détection et un autre pour le diagnostic.*

---

# II.1 Introduction

La diversité des outils de l'Intelligence Artificielle rencontrés dans des applications de diagnostic ne permet pas à priori de se faire une idée claire sur les avantages et les inconvénients de chacun de ces outils. Après avoir présenté quelques précisions sur la surveillance, nous donnons donc dans ce chapitre une présentation des différents outils de l'intelligence artificielle suivant la classification des méthodes de surveillance proposée au chapitre précédant. Dans un premier temps, nous présentons les méthodes à base de modèles comportementaux, puis les méthodes de reconnaissance de formes et enfin les méthodes à base de modèles explicatifs.

# II.2 Précision sur la surveillance

Dans cette partie nous intégrons un nouveau niveau dans la définition de la surveillance, c´est l´intégration des notions de détection, de diagnostic, de localisation de défaut et d´identification des causes. Nous avons vu dans la partie précédente la définition suivante de la surveillance :

*La surveillance est un dispositif passif, informationnel qui analyse l'état du système et fournit des indicateurs. La surveillance consiste notamment à* détecter *et classer les défaillances en observant l'évolution du système puis à les* diagnostiquer *en* localisant *les éléments défaillants et en* identifiant *les causes premières. (Lefebvre, 2000)*

La surveillance peut aussi se définir par deux fonctions qui rejoignent la définition proposée par Lefebvre. Ces fonctions sont « Voir » et « Comprendre ». La première, la fonction de perception et donc de détection, s'appuie sur les données acquises par les capteurs du système à surveiller et transmet à l'opérateur des informations plus ou moins élaborées. L'analyse de ces informations reçues constitue la fonction « Comprendre » et donc le diagnostic.

L'architecture d´un système de surveillance peut ainsi se résumer par la figure II.1.

Nous allons donc définir les différentes notions de la surveillance.

## II.2.1 Détection

*Pour détecter les défaillances du système, il faut être capable de classer les situations observables comme étant normales ou anormales. . . . (Lefebvre, 2000)*

Cette classification n'est pas triviale, étant donné le manque d'information qui caractérise généralement les situations anormales. Une simplification communément adoptée consiste à considérer comme anormale toute situation qui n'est pas normale.

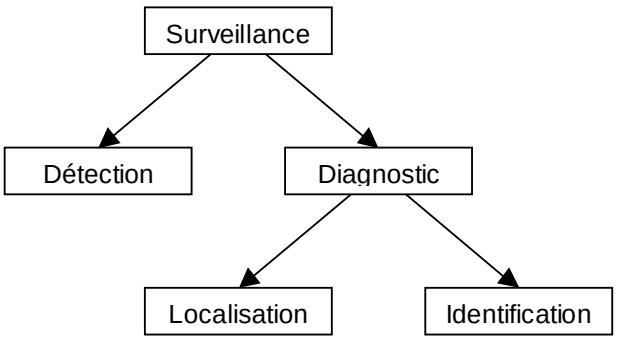



FIG. II.1 – Architecture d´un système de surveillance

### II.2.2 Diagnostic

Une première définition donnée dans (Dubuisson, 2001) considère le diagnostic de la manière suivante :

*Un problème de diagnostic peut se définir comme un problème de reconnaissance de formes. L'ensemble des états est homologue à un ensemble de classes et le vecteur forme est le vecteur composantes des paramètres observés sur le système. Cette définition s'applique à de nombreuses applications de diagnostic dans lesquelles le diagnostic est réalisé par des méthodes de reconnaissance de formes.*

Il existe cependant d'autres approches du diagnostic, en effet, étymologiquement diagnostic vient du grecque et signifie (dia : par , Gnosis : connaissance). La définition donnée par (Peng *et al.*, 1990), s'en inspire et se formule ainsi :

*Étant donné un ensemble de manifestations observées (symptômes, constatations, etc), il s'agit d'expliquer leur présence, de remonter aux causes, en utilisant un savoir sur le système considéré.*

Nous considérerons cette définition pour la suite du travail, elle semble mieux adaptée aux problèmes de diagnostic dans la mesure où elle met en évidence le raisonnement qui doit être suivi pour réaliser la fonction « Comprendre » et l'approche considérée est largement reprise par d'autres auteurs, on peut citer notamment (Zwingelstein, 1995; Grosclaude, 2001; Bouchon-Meunier *et al.*, 2003).

De plus, la décomposition du diagnostic en deux fonctions que sont *la localisation* qui permet de déterminer le sous-ensemble fonctionnel défaillant et *l'identification* qui doit déterminer les causes qui ont menées à une situation anormale s'applique également à cette définition. Enfin, dans cette définition, le diagnostic est réalisé à partir d'observations, ce qui revient bien à considérer la détection comme ne faisant pas partie de la phase de diagnostic.

Au regard de ces définitions, comme la surveillance, le diagnostic traite à la fois de

données numériques (exploitation des observations s'il s'agit de capteurs par exemple) et de données symboliques (connaissances sur le système considéré). Dans (Dubuisson, 1990), ces deux types de données sont considérés comme des connaissances nécessaires à l'opération de diagnostic. L'auteur définit deux types de connaissances :

- la connaissance globale que l'on peut qualifier de connaissance *a priori* sur le système, cette connaissance reposant sur le passé du système,
- la connaissance instantanée qui correspond à l'ensemble des éléments dont on dispose à un instant donné pour prendre une décision et l'exploiter ; cette connaissance repose donc sur les observations qui peuvent être numériques ou symboliques.

Une autre notion se dégage également de la définition proposée dans (Peng *et al.*, 1990). Elle rend compte de l'exploitation des connaissances causales sur le système. En effet, un « dysfonctionnement » peut être décrit de façon simple par des relations associant ses causes initiales (défaillances de composants, etc.) à ses manifestations observables, les symptômes. Si on dispose d'une théorie modélisant de telles relations, un problème de diagnostic consiste à rechercher à l'aide de cette théorie des explications satisfaisantes aux symptômes observés. L'inférence de base mise en jeu dans ce type de raisonnement qui « remonte des effets aux causes » est appelée abductive[1]. On peut la schématiser ainsi :

Étant donné le fait « $B$ » et l'association (la relation de causalité) « $A \rightarrow B$ » (« $A$ » cause « $B$ »), inférer « $A$ possible ».

Le diagnostic ainsi réalisé sera qualifié de *Diagnostic Abductif.*

Un point très important pour le diagnostic est soulevé ici, en effet dans quelle mesure peut-on considérer que les connaissances sur le système qu'elles soient causales, globales ou instantanées sont suffisantes ?, et comment les acquérir et les valider ?

Compte tenu de la complexité de la tâche, de nombreuses méthodes existent pour réaliser une surveillance faisant appel à des techniques variées. Nous avons vu dans le chapitre précédent une classification de ces différentes méthodes. Nous allons maintenant approfondir ces méthodes en mettant en évidence les méthodes de détection et les

[1]Les principaux types de raisonnement sont : inductif, déductif ou abductif. L'induction infère quelque chose de différent que ce qui est observé ; ce raisonnement qui va des cas particuliers à la règle permet ainsi en quelque sorte de découvrir des lois ou régularités. La déduction s'inscrit dans un processus de démonstration où on conclut des propositions prises pour prémisses, déjà démontrées ou autoposées, d'autres propositions. Quant à l'abduction, elle est définie comme « ... une méthode pour former une prédiction générale sans assurance positive qu'elle réussira dans un cas particulier ou d'ordinaire, sa justification étant qu'elle est le seul espoir possible de régler rationnellement notre conduite future, et que l'induction fondée sur l'expérience pestée nous encourage fort à espérer qu'elle réussira. » (Peirce, 1978) L'abduction qui est présentée peut être vue comme une synthèse ordonnée des deux autres types de raisonnements qui prend en compte le décalage entre les objets et les règles que l'on pourrait en tirer. Il suggère les hypothèses ou idées générales que la déduction développe et que l'induction met à l'épreuve (Deledalle, 1990).

méthodes de diagnostic c'est-à-dire les méthodes accomplissant la fonction de recherche de causes.

## II.3 Les méthodes à base de modèles comportementaux

Il existe deux approches principales pour la construction de ces modèles qui utilisent soit les automates d'états finis, soit les réseaux de Petri.

### II.3.1 Les automates d'états finis

Ils permettent de modéliser directement le fonctionnement du système, grâce à un automate global. Ce dernier est obtenu par composition d'automates élémentaires correspondant à des systèmes locaux (composants du système). Cette représentation est donc directement adaptée à la simulation et à la détection. Il existe cependant des systèmes pour lesquels cette représentation est également utilisée pour le diagnostic.

Dans (Sampath *et al.*, 1996), les auteurs développent une méthode de diagnostic qui se caractérise par deux étapes. Dans un premier temps, il s'agit de développer le modèle du système à l'aide des automates d'états finis, puis dans un deuxième temps, un outil de diagnostic correspondant également à un automate d'états finis est construit à partir du modèle global. Ce dernier effectue un diagnostic en observant en ligne une séquence d'événements. Pour chaque événement consécutif, l'outil fournit une estimation de l'état du système et des événements non observés, d'où les occurrences des pannes sont déduites.

En général, les utilisations des automates d'états finis en diagnostic sont dédiées au diagnostic des réseaux de télécommunications.

### II.3.2 Les réseaux de Petri

La deuxième grande approche utilisant les méthodes à base de modèles opérationnels est basée sur le formalisme des réseaux de Petri. Les réseaux de Petri sont des outils mathématiques et graphiques qui s'appliquent à un grand nombre d'applications où les notions d'événements et d'évolutions simultanées sont importantes. Ils constituent les modèles les plus utilisés lorsqu'il s'agit de systèmes à événements discrets. Ils ont toutefois été enrichis sous plusieurs aspects (RdP temporisés, stochastiques, flous), de manière à mieux rendre compte de la dynamique des systèmes à événements discrets. Utilisés

dans un premier temps comme modèles générateurs, ils permettent la réalisation de simulation ainsi que la détection dans une optique d'utilisation en diagnostic de systèmes. Dans ce cadre, les réseaux de Petri peuvent être qualifiés de modèle de bon fonctionnement. Dans (Anglano *et al.*, 1994), un raisonnement de type chaînage arrière sur les réseaux de Petri est défini. Les réseaux de Petri considérés (BPN – Behavioral Petri Net) constituent un modèle de comportement du système à diagnostiquer. Les auteurs introduisent 2 types de jetons (normal et inhibiteur). Ces 2 types de jetons permettent d'introduire 3 types de marquages, à savoir : true, false, unknown. Les BPN sont des réseaux saufs et déterministes. Des règles de tirage arrière sont définies et permettent de rendre compte d'éventuelles incohérences dans le raisonnement de recherche de causes. Dans cette approche le diagnostic est réalisé par un tirage arrière du réseaux qui permet de remonter au marquage source (causes) à partir du marquage correspondant à l'état observé.

### II.3.2.1 Règles de chaînage arrière

La figure II.2 donne une représentation graphique des différentes règles de tirage arrière pour chaque type de transitions considéré. Ces règles donnent les différents marquages qu'il est possible d'obtenir pour la ou les places amonts après le tirage arrière d'une transition en fonction de son architecture et du marquage de la ou des places avales.

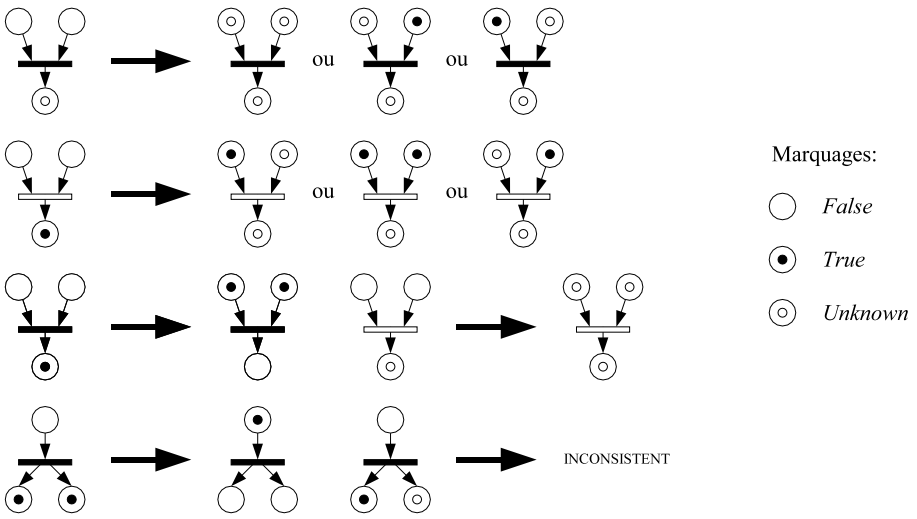



Fig. II.2 – Règles de chaînage arrière.

### II.3.2.2 Exemple de diagnostic

Un exemple d'application au diagnostic de pannes d'un moteur de voiture est présenté. Le BPN correspondant est celui représenté sur la figure II.3, les places définissent des événements avec des attributs qui conditionnent le marquage.

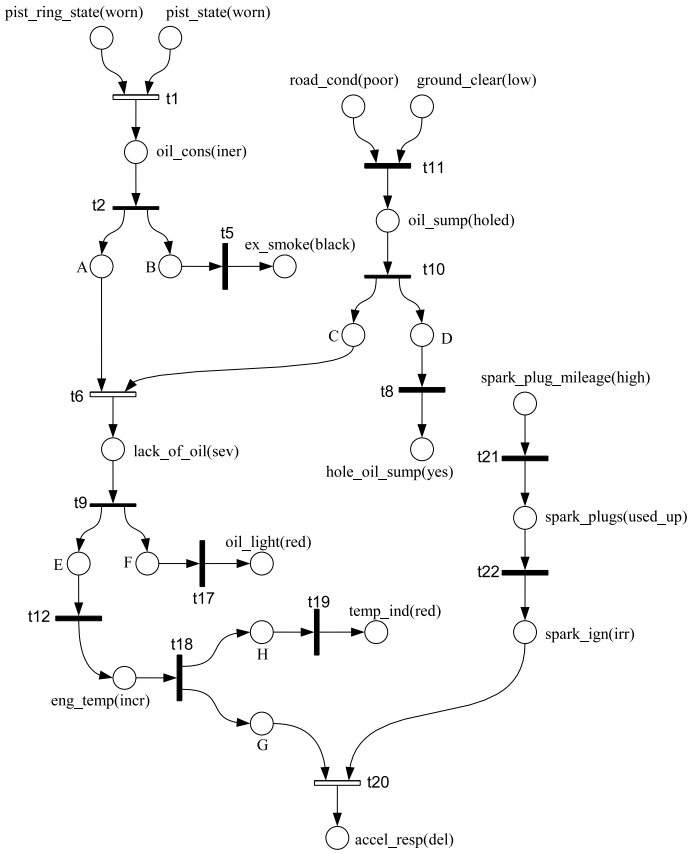



Fig. II.3 – Réseau de Petri comportemental.

Considérons un ensemble d'observations correspondant à des places du RdP soit par exemple, OBS={ex_smoke(normal), oil_light(red), temp_ind(red)}. On cherche ensuite l'ensemble des places pouvant être les causes initiales des observations de manière à avoir un ensemble de causes possibles.

Se pose ensuite le problème du marquage initial pour réaliser l'analyse par chaînage arrière du réseau. Les auteurs définissent le marquage de la façon suivante :

- marquage « true » noté b (black) si l'observation est cohérente avec la place correspondante, ensemble des places notées $P^+$,
- marquage « unknown » noté w (white) si l'observation est incohérente avec la place correspondante, ensemble de places notées $P^-$,

– marquage « false » partout ailleurs.

Le diagnostic obtenu par chaînage arrière est le marquage initial qui conduit au marquage final suivant :

– 1 pour les places de l'ensemble $P^+$,
– 0 pour les places de l'ensemble $P^-$,

Dans l'exemple considéré, le chaînage arrière est donné par le graphe de la figure II.4.

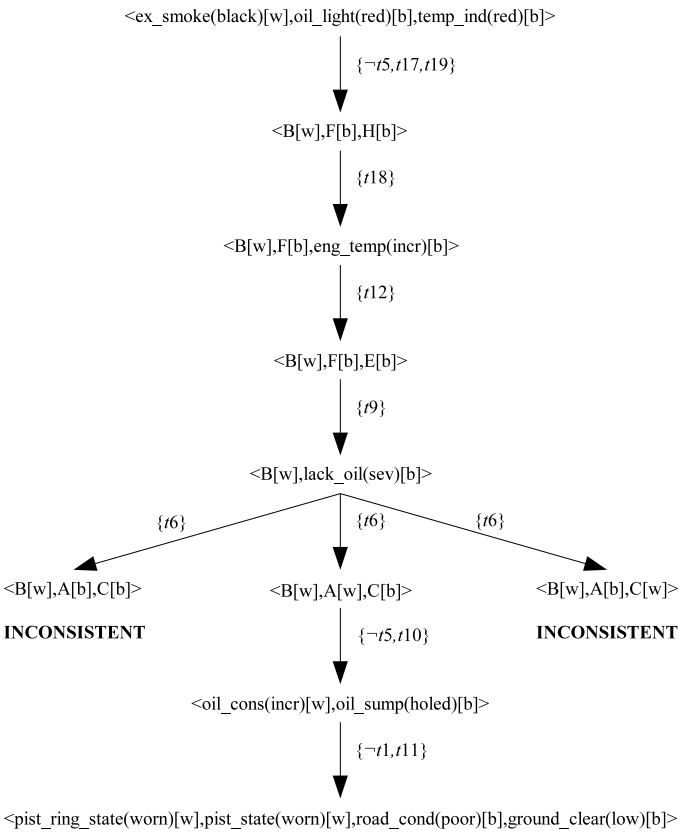



FIG. II.4 – Graphe du tirage arrière.

Ce graphe résume les transitions franchies à partir du marquage des observations et les différents marquages correspondants. Les trois marquages possibles ainsi que les règles de tirage arrière permettent d'éliminer certaines branches contradictoires du réseau dans la phase d'analyse et fournissent le marquage qui permet de revenir au marquage des observations. On obtient donc les états qui ont précédés aux états observés en fonction des différentes évolutions possibles prises en compte par l'architecture du réseau.

Le raisonnement de type chaînage arrière sur les réseaux de Petri présenté ici est particulièrement intéressant puisqu'il permet à priori de réaliser une réelle recherche de

causes grâce au chaînage arrière. Cependant, le mécanisme présenté ne l'est qu'à titre illustratif, et ne régit aucune application réelle. En ce sens, les contraintes liées à une application réelle ne sont pas prises en compte et la validité du diagnostic n'est pas évaluée. De plus, l'approche pour le diagnostic est complètement déterministe. Il serait donc intéressant d'apporter de la finesse au diagnostic réalisé afin d'apporter une aide au diagnostic en prenant en compte les imprécisions et les incertitudes inhérentes à la réalité industrielle.

### II.3.3 Autres formalismes

Il existent également d'autres formalismes à rattacher aux méthodes à base de modèles opérationnels tels les modèles de la physique qualitative qui permettent d'obtenir un modèle par abstraction du modèle numérique (Basseville *et al.*, 1996), ou les approches en logique classique ou linéaire (utilisées également avec le réseaux de Petri (Valette *et al.*, 1994)).

### II.3.4 Synthèse sur les méthodes à base de modèles comportementaux

Les automates d'états finis et les réseaux de Petri constituent donc des outils relativement bien adaptés pour construire des mécanismes de détection lorsque le fonctionnement normal du système est décrit par ces formalismes. En revanche, leurs utilisations en diagnostic sont encore limitées. Pour les automates, les principales difficultés étant liées à la taille importante de l'espace d'état, ceci conduit donc à des problèmes de mémoire et de vitesse d'exécution du diagnostic. Cependant, comme il est souligné dans (Valette *et al.*, 1994), les réseaux de Petri constituent un outil puissant de modélisation et peuvent être considérés comme un outil parmi d'autres pour décrire la connaissance nécessaire au diagnostic.

Nous allons donc maintenant aborder les méthodes de reconnaissance afin de mettre en avant quelles techniques de l'IA sont utilisées en surveillance.

## II.4 Les méthodes de reconnaissance de formes

Ces méthodes supposent qu'aucun modèle n'est disponible pour décrire les relations de cause à effet. La seule connaissance repose sur l'expertise humaine confortée par un solide retour d'expérience (Zwingelstein, 1995). La plupart de ces méthodes sont basées sur l'Intelligence Artificielle avec en particulier des outils tels que les systèmes experts, les outils statistiques, le raisonnement à partir de cas (RàPC), les réseaux neuronaux,

la logique floue et les réseaux neuro-flous. Nous reviendrons donc sur leur utilisation et en particulier nous aborderons : le RàPC, les réseaux neuronaux, la logique floue, les réseaux neuro-flous.

## II.4.1 Le raisonnement à partir de cas

Le raisonnement à partir de cas est une approche récente pour résoudre et apprendre des problèmes. Il signifie : « résoudre un nouveau problème en se rappelant une situation précédente similaire et en réutilisant les informations et les connaissances de cette situation » (Aamodt *et al.*, 1994). Il constitue donc une méthodologie pour modéliser le raisonnement et la pensée humaine ou pour développer des systèmes intelligents.

Le RàPC recouvre un ensemble de méthodes de résolution de problèmes à partir d'expériences passées plutôt qu'à partir de connaissances générales comme le raisonnement à partir de règles. (Malek, 1996).

### II.4.1.1 Principes de fonctionnement du RàPC

Globalement, le RàPC fonctionne de la manière suivante : il stocke les expériences précédentes (cas) dans une mémoire afin de résoudre un nouveau problème, puis lors d'un nouveau cas :

- il retrouve l'expérience similaire au nouveau problème dans la mémoire,
- il réutilise cette expérience dans le contexte de la nouvelle situation (complètement, partiellement ou en l'adaptant selon les différences),
- il mémorise la nouvelle expérience dans la mémoire (apprentissage).

Le raisonnement à partir de cas constitue donc un processus cyclique pour la résolution de problèmes, (Aamodt *et al.*, 1994; Malek, 1996; Lamontagne *et al.*, 2002). On identifie dans ce processus les phases suivantes (figure II.5) :

**Remémoration :** elle consiste en la recherche des cas les plus similaires au problème en cours de traitement. Elle est basée sur un calcul de similarité qui rend compte de l'appariement (de la ressemblance) entre les cas.

**Réutilisation :** elle correspond à l'utilisation de la connaissance du (ou des) cas remémoré(s) pour la résolution du problème. Deux méthodes existent pour cette phase : on peut « copier » la solution du cas réutilisé sans la modifier ou « adapter » la solution du cas réutilisé (l'adaptation pouvant être réalisée soit par l'utilisateur, soit automatiquement).

**Révision :** il s'agit ici d'une évaluation de la solution afin de faire un bilan du cas avant la mémorisation. A ce niveau, si le résultat de l'évaluation n'est pas bon, la solution est réparée en utilisant des connaissances sur le domaine si elles existent

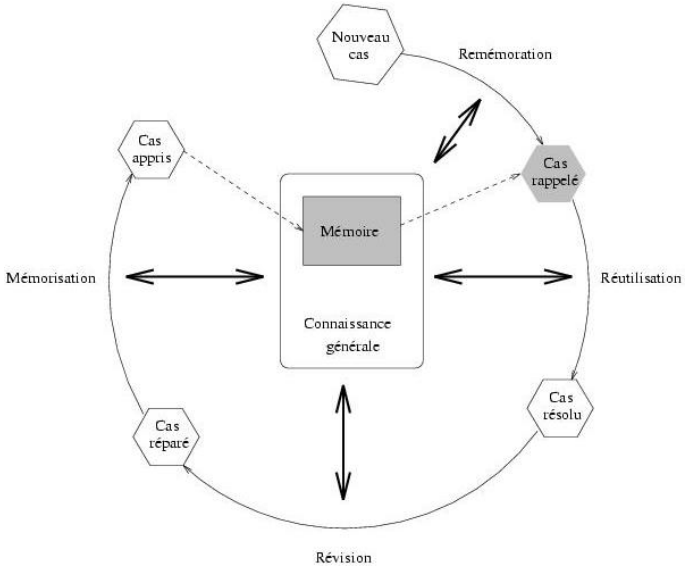



FIG. II.5 – Cycle du Raisonnement à Partir de Cas.

ou en se référant à l'expert du domaine. Les critères utilisés pour la révision sont généralement la justesse et la qualité de la solution auxquelles peuvent être ajoutées par exemple des préférences de l'utilisateur.

**Mémorisation :** elle consiste à extraire et mémoriser la partie utile (nouvelle) de cette expérience. Elle doit donc décider quelles informations sont à retenir et sous quelle forme, mais également comment indexer le nouveau cas dans la base pour une utilisation future.

Un nouveau problème ou cas est résolu en se rappelant (pendant la remémoration) des cas similaires déjà pré-analysés et stockés dans la mémoire. La solution trouvée est ensuite adaptée (réutilisée) au nouveau problème. Le nouveau cas est ensuite révisé ou réparé (par l'expert ou par l'utilisation de la connaissance générale du système). Ce nouveau cas peut aussi être appris au système comme nouvelle expérience.

Le RàPC s'articule donc autour d'une base de connaissance qui contient deux parties :

- La connaissance générale, souvent représentée par une base de règles et qui peut intervenir dans toutes les phases du RàPC.
- La mémoire, qui contient les cas et les structures nécessaires qui constituent l'expérience du système ; elle est utilisée lors des phases de remémoration et de mémorisation.

### II.4.1.2 Les caractéristiques

Un système de RàPC se caractérise donc par : la structure des cas utilisés, la structure de la base de cas (indexation des cas) et l'organisation de la mémoire.

#### a) Structure des cas

La structure des cas va dépendre des domaines d'utilisation et des tâches à accomplir, mais d'une manière générale, les cas sont définis pas trois champs qui donnent la structure du cas.

Structure du cas :

**Problème** ↔ description
**Solution** ↔ résolution du problème (plusieurs solutions possibles)
**Conclusion** ↔ évaluation de la solution, conséquences, commentaires

Un cas décrit donc une situation particulière et contient plusieurs attributs définis par leurs valeurs, propres à la situation.

Tab. II.1 – Exemple de structure d'un cas (Bergmann, 1998-2000)

| | |
|---|---|
| Cas 1 | Problème (Symptôme)<br>- *Problème* : feu avant en panne<br>- *Voiture* : VW Golf II 1.6L<br>- *Année* : 1996<br>- *Batterie* : 13.6V<br>- *État des feux* : OK<br>- *État de l'interrupteur* : OK |
| | Solution<br>- *Diagnostic* : fusible des feux avant défectueux<br>- *Action* : changer de fusible |

#### b) Structure de la base de cas (indexation des cas)

Plusieurs méthodes sont disponibles pour l'indexation des cas, on trouve principalement des méthodes :

– selon les attributs,
– basées sur la différence,
– basées sur la similarité,
– basées sur l'explication,
– par apprentissage inductive.

**c) Organisation de la mémoire**

La mémoire peut être soit une *mémoire plate*, soit une *mémoire hiérarchique*, cette dernière étant utilisée pour les domaines dans lesquels on utilise une représentation orientée objet, où les cas correspondent à un ensemble d'objets.

### II.4.1.3 RàPC et diagnostic

L'utilisation du RàPC en diagnostic remonte à une quinzaine d'années, on peut citer à titre d'exemple les systèmes MOLTKE et PATDEX (1988-1991), (Bergmann, 1998-2000). Pour une application en diagnostic, la principale singularité du RàPC tient à la définition de la structure des cas. En effet, le mécanisme est identique à celui présenté plus haut, et permet à partir de la description d'une panne d'en retrouver la ou les causes et de proposer une action pour une éventuelle intervention de maintenance. Adaptée au diagnostic, la structure des cas est donc la suivante :

Structure du cas :

**Problème** ↔ Symptômes (description de la situation particulière de diagnostic)
**Solution** ↔ Origines (plusieurs origines possibles)
**Conclusion** ↔ Actions (stratégie de maintenance).

Le RàPC constitue donc une technique pour résoudre des problèmes basés sur l'expérience et est donc relativement bien adapté au problème de diagnostic pour lesquels la notion d'expérience est relativement importante. Pour ce faire, la résolution s'organise en quatre phases pour proposer une solution. Dans le cadre du RàPC, plusieurs techniques sont également disponibles d'une part pour représenter les connaissances, en particulier pour représenter les cas en fonction du domaine d'utilisation, et d'autre part pour réaliser les différentes phases du cycle. Enfin, le RàPC, présente également les avantages suivants :

- il réduit les efforts d'acquisition de connaissances,
- il est relativement facile à maintenir,
- l'efficacité de la résolution des problèmes augmente à mesure qu'il est utilisé,
- il permet d'utiliser des données existantes comme des bases de données,
- il peut s'adapter aux changements de son environnement.

Son utilisation en diagnostic apparaît donc comme relativement aisée, avec comme propriété, une structure de cas adaptée. Cependant, la difficulté tient justement à cette structure de cas et aux informations qu'elle doit contenir. En effet, l'extraction des connaissances et leurs représentations sont primordiales dans ce type d'application.

## II.4.2 Les réseaux neuronaux

Les réseaux de neurones artificiels sont un outil bien adapté pour les problèmes de perception, de classification et de prédiction. Leur utilisation s'inscrit particulièrement dans les méthodologies de surveillance en l'absence de modèle du procédé. Leur utilisation est principalement guidée par leurs propriétés suivantes :

- capacité d'apprentissage,
- parallélisme dans le traitement,
- adaptés aux non-linéarités des systèmes,
- rapidité de traitement.

Nous verrons donc ici les architectures les plus couramment utilisées pour ces problèmes de surveillance des systèmes industriels et dans quelle mesure ils peuvent réaliser un diagnostic.

### II.4.2.1 Généralités

Deux éléments principaux constituent un réseau de neurones artificiels, tout d'abord le(les) modèle(s) de neurones utilisés pour constituer le réseau et ensuite, l'architecture du réseau. Chaque neurone artificiel est un processus élémentaire qui reçoit un certain nombre d'entrées de neurones amonts. A chacune de ces entrées est associé un poids représentatif de la force de connexion entre les neurones correspondants. Ceci met donc en avant deux caractéristiques propres à chaque neurone (voir figure II.6) :

- un « potentiel » ou « activation » égal(e) à la somme pondérée des entrées
- une fonction de transfert qui donne la sortie du neurone en fonction de son « activation ».

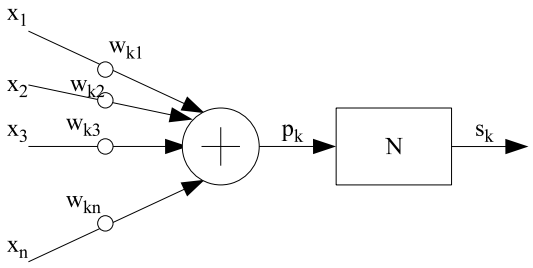



FIG. II.6 – Modèle de neurone.

Les calculs du potentiel $p_k$ et de la sortie $s_k$ s'expriment par les relations suivantes :

$p_k = \sum_i w_{ki} \cdot x_i$ et : $s_k = N(p_k)$

$k$ : numéro d'ordre du neurone cible
$i$ : numéro d'ordre du neurone émetteur
où $N$ est la fonction d'activation du neurone, ou fonction neurone, linéaire ou non.

Les principales fonctions d'activation que l'on retrouve dans les différentes architectures sont représentées sur la figure II.7.

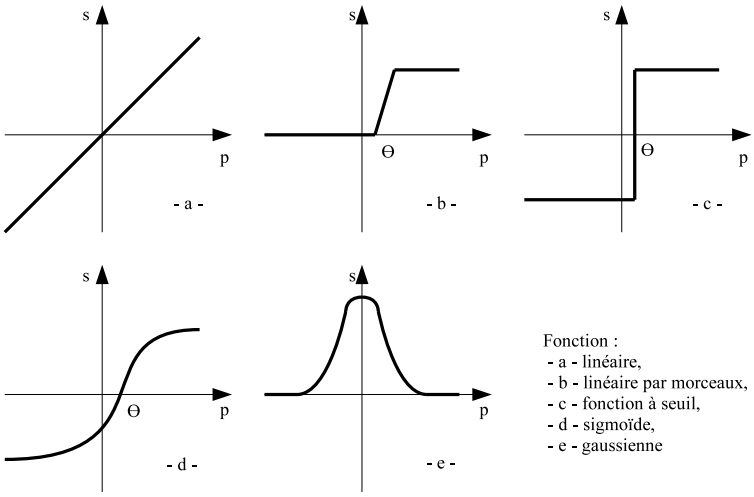



FIG. II.7 – Les différentes fonctions d'activation.

Chaque neurone réalise donc une fonction simple, les propriétés globales de l'outil émergent de sa structure. L'architecture des différents réseaux de neurones s'organise en couches, lesquelles sont interconnectées. Les principaux types de connexions sont donnés dans la figure II.8.

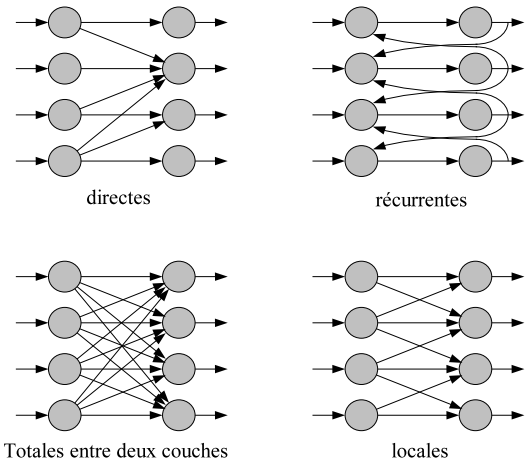



FIG. II.8 – Architectures des connexions entre les couches.

Toutes les caractéristiques des réseaux de neurones sont ensuite exploitées à travers la propriété principale des réseaux de neurones qu'est l'apprentissage. En effet, les mécanismes d'apprentissage sont à l'origine des capacités de résolution de problèmes

des réseaux neuronaux. Cet apprentissage permet de configurer les poids synaptiques ainsi que les fonctions d'activation afin d'adopter un comportement désiré. Deux types d'apprentissage sont utilisés :

- *Apprentissage supervisé* : l'apprentissage supervisé permet de déterminer les poids synaptiques à partir d'exemples étiquetés auxquels un expert a associé des réponses du réseau. Les paramètres du réseau sont donc modifiés de manière à minimiser l'erreur entre la sortie cible (fournie par l'expert) et la sortie réelle du réseau.
- *Apprentissage non-supervisé* : les données fournies en entrée ne contiennent pas d'information sur la sortie désirée. L'apprentissage est réalisé à l'aide de règles qui modifient les paramètres du réseau en fonction des exemples fournis en entrée.

#### II.4.2.2 Le modèle de Hopfield

Le modèle de Hopfield introduit en 1982, est constitué de processeurs élémentaires qui effectuent une somme pondérée de toutes les entrées et d'une fonction d'activation à seuil qui fournit une réponse égale à 0 ou à 1. Chaque neurone est connecté avec tous les autres de façon bidirectionnelle, sauf avec lui même, (Zwingelstein, 1995). Ce modèle est basé sur le concept de mémoire adressée par le contenu : la mémoire associative. L'architecture du réseau est symétrique, c'est-à-dire le poids $w_{ij}$ de la connexion entre le neurone $i$ et le neurone $j$ est identique à $w_{ji}$, poids de la connexion entre les neurones $j$ et $i$. On utilise une représentation matricielle pour donner les poids des connexions entre les neurones d'un réseau à $n$ cellules, par exemple, la matrice $n \times n$ des connexions :

$$w = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix}$$

Il existe deux méthodes de mise à jour du réseau : une évolution synchrone qui conduit à calculer l'état de tous les neurones du réseau à chaque unité de temps, une évolution asynchrone qui consiste à mettre à jour un seul neurone à la fois.

Chaque information mémorisée dans le réseau représente un point stable de l'espace d'état vers lequel l'évolution du système aboutit à partir d'un point voisin, correspondant à une version déformée de l'information mémorisée. L'espace d'état comporte donc des attracteurs qui correspondent aux informations mémorisées. L'évolution du réseau de Hopfield vers un état stable est caractérisée par une fonction d'énergie qui dépend à la fois du vecteur d'état et de la matrice $w$. Pour obtenir les états stables mémorisés par le réseau, il faut donc trouver la matrice $w$ qui minimise cette fonction d'énergie (Zemouri, 2003).

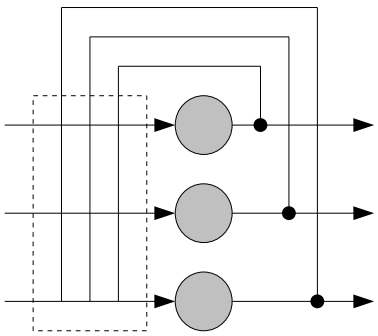


FIG. II.9 – Architecture du réseau de Hopfield.

#### II.4.2.3 Le réseau de Kohonen

Il existe des zones du cerveau (dans le cortex visuel par exemple) qui présentent la même topologie que les capteurs sensoriels. C'est à dire deux zones proches dans le cortex visuel correspondent à deux zones proches dans la rétine, comme l'ont démontré Hubel et Wiesel en 1959. Ces propriétés se retrouvent avec le bulbe olfactif, ou l'appareil auditif.

Il est intéressant de souligner que ces dispositions au sein du cerveau ne sont pas génétiques, mais sont dues à un apprentissage. A la suite de ces observations, Kohonen proposa un modèle de carte topologique auto-adaptative où seules les entrées modifient le processus, il s'agit donc d'un apprentissage non-supervisé. Une représentation d'une carte auto-adaptative est donnée dans la figure II.10.

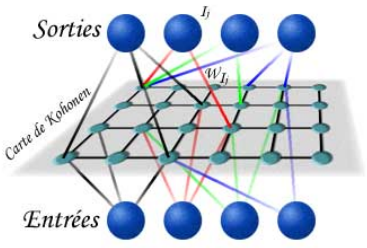



FIG. II.10 – Carte topologique auto-adaptative de Kohonen.

Chaque neurone de la couche d'entrée est relié à tous les neurones de la carte, dans la plupart des applications, les cartes sont à deux dimensions. A chaque cellule est associé un voisinage qui dépend de la géométrie du maillage de la carte et auquel correspondent des mécanismes d'interaction latérale. Cette interaction latérale, souvent choisie sous la forme d'un « chapeau mexicain » (figure II.11), montre que les neurones proches apportent une activation, alors que les neurones plus éloignés ont une action inhibitrice.

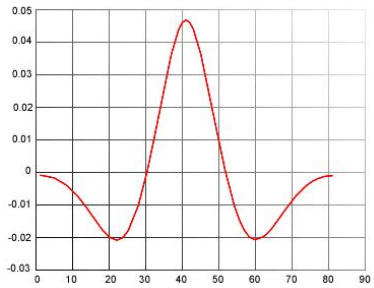



FIG. II.11 – Liaison latérale de type « chapeau mexicain » ou DOG (Difference of Gaussians).

Le fonctionnement du réseau de Kohonen s'effectue en deux étapes. Dans la première, on identifie la cellule $I_i$ la plus sensible à l'entrée présentée et dans la deuxième étape, on modifie les poids des cellules pour obtenir à la fin de l'apprentissage une partition des cellules en zones particulières sensibles à un sous-espace donné de l'ensemble des formes présentées à l'entrée. Après un long temps de convergence, le réseau évolue donc de manière à représenter au mieux la topologie de l'espace de départ.

Les réseaux de Kohonen ont trouvé de nombreuses applications pour la classification de signatures vibratoires de machines. Ils présentent cependant l'inconvénient majeur d'exiger un temps relativement long pendant la phase d'apprentissage (Zwingelstein, 1995).

Dans (Zemouri, 2003), deux architectures sont mises en avant pour la surveillance des systèmes, il s'agit du Perceptron Multicouche (PMC ou MLP pour Multi Layer Perceptron) et du Réseau à Fonctions de base Radiales (RFR ou RBF pour Radial Basis Function).

#### II.4.2.4 Le Perceptron Multicouche

Le perceptron multicouche est un réseau orienté de neurones artificiels organisé en couches et où l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie. La figure II.12 donne l'exemple d'un réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système. Elle ne contient aucun neurone. Les couches suivantes sont des couches de neurones. Les sorties des neurones de la dernière couche correspondent toujours aux sorties du système. Dans le cas général, un perceptron multicouche peut posséder un nombre de couches quelconque et un nombre de neurones (ou d'entrées) par couche également quelconque.

Les neurones sont reliés entre eux par des connexions pondérées. Ce sont les poids

de ces connexions qui gouvernent le fonctionnement du réseau et « programment » une application de l'espace des entrées vers l'espace des sorties à l'aide d'une transformation non linéaire. La création d'un perceptron multicouche pour résoudre un problème donné passe donc par l'inférence de la meilleure application possible telle que définie par un ensemble de données d'apprentissage constituées de paires de vecteurs d'entrées et de sorties désirées. Cette inférence peut se faire, entre autre, par l'algorithme dit de rétropropagation, (Parizeau, 2004).

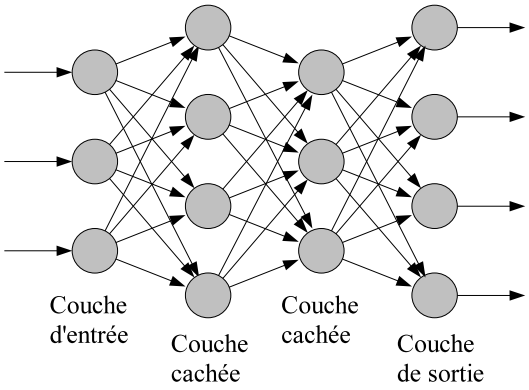



Fig. II.12 – Architecture du Perceptron Multicouche.

L'algorithme de rétropropagation, va donc réaliser l'apprentissage du réseau en modifiant les poids des connexions neurone par neurone en commençant par la couche de sortie. Le critère d'apprentissage étant la minimisation de la racine de l'erreur quadratique moyenne.

Le modèle perceptron multicouche se caractérise donc par une architecture globale (tous les neurones participent à la sortie après la phase d'apprentissage), qui lui confère de mauvaises propriétés en classification et donc le rend moins intéressant pour une utilisation en surveillance (Zemouri, 2003). Cependant il existe un nombre non négligeable de travaux en surveillance et diagnostic qui utilise cette architecture (Keller *et al.*, 1994; Meador *et al.*, 1991; Wu *et al.*, 1994). D'autres références pourront être retrouvées dans (Zemouri, 2003).

### II.4.2.5 Les réseaux à fonction de base radiale (RFR)

Les réseaux de neurones à fonction de base radiales sont également des réseaux à propagation avant. Ils possèdent une seule couche cachée composée de fonctions noyaux Gaussiennes (figure II.13). La principale particularité de ces réseaux réside dans le fait qu'ils sont capables de fournir une représentation locale de l'espace grâce aux fonctions noyaux Gaussiennes. L'influence de ces fonctions noyaux est restreinte à certaines zones

de cet espace, elles ne fournissent une réponse utile que pour un domaine de valeurs restreint, leur champ récepteur. Deux paramètres caractérisent ces fonctions noyaux : un vecteur de référence $\mu_j$ appelé centre ou prototype et la dimension $\sigma_i$ du champ d'influence appelé rayon d'influence. Chaque neurone de la couche cachée est donc caractérisé par les équations suivantes :

Distance du vecteur d'entre $x$ au vecteur prototype $\mu_i$ : $r_i(x) = \|x - \mu_i\|$

Fonction gaussienne d'activation : $\Phi_i(x) = \exp\left(\frac{-r_i(x)^2}{\sigma_i^2}\right)$

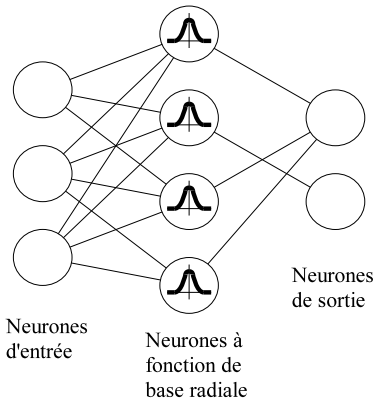



FIG. II.13 – Architecture du RFR.

L'apprentissage des réseaux RFR permet donc de déterminer la position et le nombre des prototypes $\mu_i$, ainsi que la taille du champ récepteur $\sigma_i$.

Pour une utilisation en classification, l'espace des entrées est divisé en différentes classes, chaque classe possédant un ou plusieurs prototypes. La classification consiste ensuite à évaluer la distance entre le vecteur d'entrée et les prototypes et à déterminer à quel champ récepteur appartient le vecteur d'entrée.

Appliqué au diagnostic, cet outil s'inscrit dans les méthodes de reconnaissance de formes. Plus généralement, leur utilisation en surveillance ne permet pas en l'état de traiter des données dynamiques, il existe donc des architectures qui permettent de prendre en compte la notion temporelle (Chappelier, 1996; Chappelier *et al.*, 2001; Zemouri *et al.*, 2001; Palluat *et al.*, 2005*b*), et donc la réalisation d'applications industrielles telles que la détection dynamique, le diagnostic et le pronostic.

#### II.4.2.6 Les réseaux de neurones temporels

On trouve principalement deux familles de réseaux de neurones temporels (Chappelier, 1996; Chappelier *et al.*, 2001) : les réseaux de neurones temporels avec une représentation externe du temps, et ceux avec une représentation interne.

La représentation externe du temps transforme les informations temporelles que contiennent les données en information spatiale ; on parle alors de représentation spatiale du temps. Un mécanisme externe est donc chargé de retarder ou de retenir les données jusqu'au moment de leur utilisation dans le réseau. Les réseaux de neurones qui utilisent cette représentation du temps sont principalement utilisés dans le domaine de le reconnaissance de la parole.

Dans la représentation interne du temps, le temps est pris en compte par les effets qu'il produit, le réseau possède alors des propriétés dynamiques, on parle alors de représentation dynamique du temps. De tels réseaux ont la capacité de mémoriser des informations temporelles. Nous y trouvons les représentations internes explicite du temps et les représentations internes implicite du temps (réseaux de neurones dynamiques).

La prise en compte implicite du temps est réalisée en réintroduisant en entrée du réseau l'état précédent du réseau (ou une partie de celui-ci). C'est le cas des réseaux récurrents d'Elman (Elman, 1990) et de Jordan (Jordan, 1986).

Représenter explicitement le temps dans le modèle neuromimétique employé peut se faire au seul niveau des liaisons du réseau ou au niveau du neurone lui-même. Comme exemples d'architectures introduisant le temps au niveau des retards synaptiques, nous pouvons citer entre autres les travaux suivants : (Béroule, 1987; Amit, 1988; Jacquemin, 1994). Pour des architectures introduisant le temps au niveau du neurone, nous pouvons nous intéresser aux travaux réalisés sur les cartes auto-organisatrices de Kohonen, dont une vue d'ensemble est donnée dans (Barreto *et al.*, 2001).

Dans (Palluat *et al.*, 2005*b*), nous avons montré que les architectures à représentation interne implicite du temps, appelées communément réseaux de neurones récurrents, sont les mieux adaptées pour la surveillance et le diagnostic. Afin d'affiner cette première classification de réseaux de neurones temporels, nous avons ajouté un critère architectural, le type local ou global de la réponse du neurone. Par exemple, on trouve une réponse globale pour les neurones utilisant une fonction d'activation sigmoïdale, et une fonction d'activation gaussienne pour les réponses locales. Le choix d'une réponse locale ou d'une réponse globale déterminera les caractéristiques principales du réseau. En effet, un réseau temporel à représentation externe du temps donnant une réponse globale, comme le NETtalk (Sejnowski *et al.*, 1986), possède une très bonne capacité de généralisation. Toutefois, lors d'un nouvel apprentissage, tous les paramètres du réseau doivent être recalculés. On utilisera de préférence ce type de réseaux pour les problèmes d'interpolation. Par contre, pour un réseau donnant une réponse locale, comme le TDRBF (Berthold, 1994), la sortie du réseau est non nulle si le vecteur d'entrée est proche d'un vecteur appris, et nulle sinon. Lors d'un nouvel apprentissage, seule une partie des paramètres du réseau est générée, car seuls quelques neurones ont une réponse non nulle. Ces réseaux pourront être utilisés pour résoudre les problèmes d'extrapolation. A noter la possibilité de réponses mixtes pour les réseaux possédant à la fois des neurones à réponses

globales et des neurones à réponses locales, tels que les réseaux RRBF (Zemouri, 2003) et DGNN (Ferariu *et al.*, 2002).

Nous retrouvons cette classification dans la figure II.14.

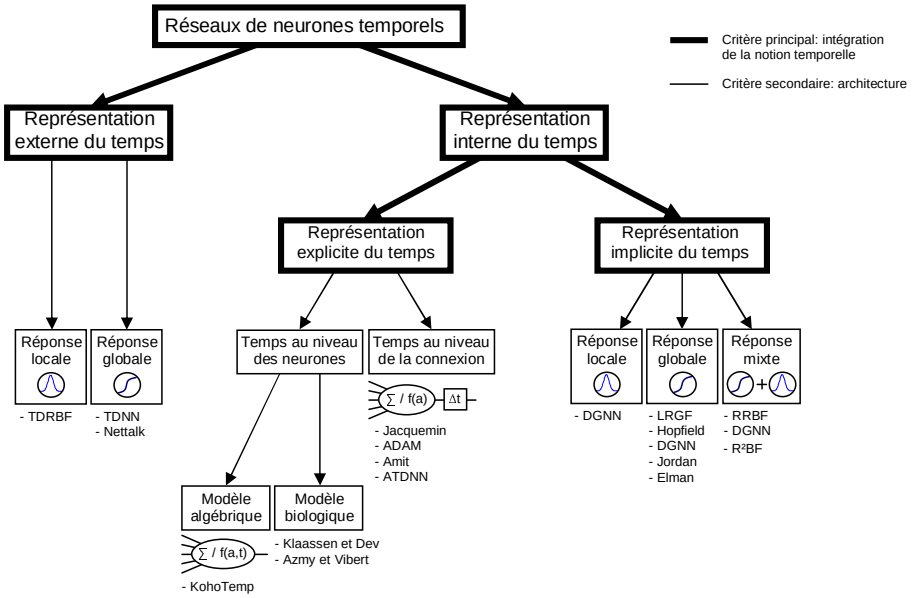



FIG. II.14 – Classification des réseaux de neurones temporels à l'aide de deux critères (temporel et architectural)

Une étude de surveillance, correspondant à un problème d'extrapolation, nous nous sommes attardés sur les réseaux à réponses locales et mixtes. Nous avons montré dans (Palluat *et al.*, 2005*b*), à travers un exemple de classification et un de prédiction, que le réseau récurrent à fonction de base radiale (RRFR) est le plus adapté.

Dans le RRFR, l'architecture de base est celle du RFR à laquelle sont ajoutées des connexions récurrentes au niveau de la couche d'entrée pour la prise en compte du temps. La figure II.15 présente les deux architectures.

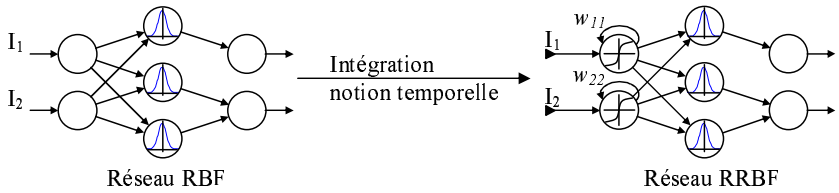



FIG. II.15 – Réseaux RFR et RRFR.

L'auto-connexion au niveau de la couche d'entrée a pour effet la prise en compte d'un certain passé des données en entrée. Le RRFR peut donc être vu comme comportant

deux types de mémoire : la mémoire statique de la couche cachée qui comprend les prototypes et les champs d'influence et une mémoire dynamique constituée par l'ensemble des neurones bouclés. De plus, l'algorithme d'apprentissage est extrêmement flexible. Cet algorithme permet un temps d'entraînement relativement court et possède très peu de paramètres à optimiser, (Palluat *et al.*, 2005*b*). D'autres propriétés du RRFR sont présentées à travers une application de surveillance dynamique d'un processus dans (Zemouri, 2003).

#### II.4.2.7 Analyse des différents types de réseaux de neurones

Au regard de ces différents travaux, l'architecture RRFR semble donc la mieux adaptée pour des applications de pronostic et de surveillance dynamique. De plus grâce à son architecture simple (séparation de la mémoire dynamique et de la mémoire statique) et des temps d'apprentissage relativement courts, elle autorise l'apprentissage en ligne, ce qui la rend particulièrement intéressante dans le cadre de la surveillance dynamique. Appliquée au diagnostic, ses capacités de classification seront mises en avant. En effet, les réseaux neuronaux appartiennent aux méthodes de diagnostic par reconnaissance de formes. Cependant, même si l'outil neuronal est capable d'identifier des modes de défaillances d'un système, il n'explique pas réellement les causes à l'origine de ces modes de défaillances.

Pour une application de surveillance, les réseaux de neurones et en particulier le réseau RRFR semble plutôt s'inscrire dans la phase de détection qui vient en amont du diagnostic dans la surveillance des systèmes.

Les différents résultats obtenus avec les réseaux de neurones temporels en font des outils particulièrement intéressants pour des applications de surveillance industrielle. Leurs capacités de détection et de classification permettraient de les utiliser dans le cadre d'une application de diagnostic pour la génération d'alarmes intelligentes dans le sens où la détection qu'ils réalisent est suivi d'un traitement et permet une classification. En effet, ils pourraient constituer un bon outil qui permettrait d'obtenir des symptômes associés à une défaillance qu'un système de diagnostic utiliserait pour réaliser la localisation et l'identification des causes de cette défaillance.

### II.4.3 La logique floue

Introduite par Zadeh en 1965, la logique floue permet de formaliser la représentation et le traitement de connaissances imprécises ou approximatives. Elle offre la possibilité de traiter des systèmes d'une grande complexité dans lesquels sont, par exemple présents des facteurs humains. Elle intervient dans la manipulation de connaissances imparfaites. Son utilisation dans des domaines tels que l'aide à la décision ou le diagnostic semble

donc naturel dans la mesure où elle fournit un outil puissant pour assister de façon automatique des actions humaines, naturellement empreintes d'imprécisions (Bouchon-Meunier, 1995).

La logique floue est considérée dans (Bouchon-Meunier, 1995), comme le seul cadre dans lequel on peut traiter des imprévisions et des incertitudes et qui autorise également le traitement de certaines incomplétudes. C'est également le seul cadre dans lequel on peut traiter des connaissances numériques et des connaissances exprimées symboliquement par des qualifications du langage naturel.

### II.4.3.1 Généralités

Nous donnons ici rapidement les principes de la logique floue. Le premier point à introduire en logique floue est celui de sous-ensemble flou. En effet le concept de sous-ensemble flou permet d'éviter le passage brusque d'une classe à une autre (de classe noire à classe blanche par exemple). Un élément peut être ni complètement blanc ni complètement noir ou encore être à la fois partiellement noir et partiellement blanc. Un sous-ensemble flou sera donc défini de la façon suivante :

#### a) Définition

Un sous-ensemble flou $A$ de $X$ est défini par une fonction d'appartenance qui associe à chaque élément $x$ de $X$, le degré $\mu_A(x)$, compris entre 0 et 1, pour lequel $x$ appartient à $A$ :

$\mu_A : X \rightarrow [0,1]$

De la même manière, on définit des opérations sur les sous-ensembles flous qui caractérisent l'égalité, l'inclusion, l'intersection et l'union de deux sous-ensembles flous.

1- égalité : Soit deux sous-ensembles flous $A$ et $B$ dans un univers $X$, $A$ et $B$ sont égaux ($A = B$) si leurs fonctions d'appartenance prennent la même valeur en tout point $x$ de $X$ :

$\mu_A(x) = \mu_B(x) \quad \forall x \in X$

2- Inclusion : Soit deux sous-ensembles flous $A$ et $B$ dans un univers $X$, $A$ est inclus dans $B$ noté $A \subseteq B$ si leurs fonctions d'appartenance sont telles que :

$\mu_A(x) \leqslant \mu_B(x) \quad \forall x \in X$

3- Intersection : L'intersection de deux sous-ensembles flous $A$ et $B$ de $X$ est le sous-ensemble flou constitué des éléments de $X$ affectés du plus petit de leurs deux degrés d'appartenance, donnés par $\mu_A$ et $\mu_B$. Elle est définie comme l'élément $C = A \cup B$ de $\mu(x)$ tel que :

$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) \quad \forall x \in X$

min désignant l'opérateur de minimisation. Intuitivement, un élément $x$ ne peut appartenir à $A$ et à $B$ à la fois, c'est-à-dire à $A \cap B$, moins fortement qu'il n'appartient à chacun d'eux.

4- Union : l'union de deux sous-ensembles flous $A$ et $B$ de $X$ est le sous-ensemble flou constitué des éléments de $X$ affectés du plus grand de leurs deux degrés d'appartenance, donnés par $\mu_A$ et $\mu_B$. Elle est définie comme l'élément $D = A \cap B$ de $\mu(x)$ tel que :

$\mu_D(x) = \max(\mu_A(x), \mu_B(x)) \quad \forall x \in X$

D'autres opérateurs sont également disponibles pour les opérations d'intersection, d'union et de complémentation ; il s'agit des normes et conormes triangulaires (tableau II.2).

Tab. II.2 – Normes et conormes triangulaires

| T-norme | T-conorme | Négation | Noms |
|---|---|---|---|
| $\min(x, y)$ | $\max(x, y)$ | $1 - x$ | Zadeh |
| $x \cdot y$ | $x + y - x \cdot y$ | $1 - x$ | Probabiliste |
| $\max(x + y - 1, 0)$ | $\max(x + y, 1)$ | $1 - x$ | Lukasiewicz |
| $\frac{x \cdot y}{\gamma + (1-\gamma)(x+y-x \cdot y)}$ | $\frac{x+y-x \cdot y-(1-\gamma) \cdot x \cdot y}{1-(1-\gamma) \cdot x \cdot y}$ | $1 - x$ | Hamacher $(\gamma > 0)$ |
| $\begin{cases} x \; si \; y = 1 \\ y \; si \; x = 1 \\ 0 \; sinon \end{cases}$ | $\begin{cases} x \; si \; y = 0 \\ y \; si \; x = 0 \\ 1 \; sinon \end{cases}$ | $1 - x$ | Weber |

### b) Implication floue et Modus Ponens

Implication floue :

L'implication floue entre deux propositions floues élémentaire « $V$ est $A$ » et « $W$ est $B$ » est une proposition floue concernant le couple de variables $(V, W)$, dont la valeur de vérité est donnée par la fonction d'appartenance $\mu_R$ d'une relation floue $R$ entre $X$ et $Y$ définie pour tout $(x, y)$ de $X \times Y$ par :

$V$ est $A \xrightarrow{\mu_R} W$ est $B$

Avec :

$\mu_R(x, y) = \Phi(\mu_A(x), \mu_B(y))$ pour une fonction $\Phi$ de $[0, 1] \times [0, 1] \rightarrow [0, 1]$

Modus Ponens :

Pour une règle de la forme « Si $V$ est $A$ alors $W$ est $B$ », une conclusion doit être obtenue relativement à $W$ lorsque la donnée disponible est de la forme « $V$ est $A'$ », pour

$A'$ plus ou moins différent de $A$ (fonctions d'appartenances de $A$ et $A'$ peu différentes). Le raisonnement approximatif introduit par Zadeh permet donc de tenir compte de l'aspect graduel des caractérisations floues : si la donnée est très proche de la prémisse de la règle, la conclusion sera très proche de la conclusion de la règle. Le modus ponens généralisé est donc défini de la façon suivante :

| | | | | |
|---|---|---|---|---|
| *Règle floue* : | si | $V$ est $A$ | alors | $W$ est $B$ |
| Fonctions d'appartenance : | | $\mu_A$ | | $\mu_B$ |
| *Fait observé* : | | $V$ est $A'$ | | |
| Fonction d'appartenance : | | $\mu_{A'}$ | | |
| *Conclusion* : | | | | $W$ est $B'$ |
| Fonction d'appartenance : | | | | $\mu_{B'}$ |

La fonction d'appartenance de $B'$ est donc calculée comme une combinaison de $\mu_R$ et de $\mu_{A'}$ de la forme :

$\mu_{B'}(y) = \sup_{x \in X} \top (\mu_{A'}(x), \mu_R(x, y)) \quad \forall y \in Y$

pour une t-norme $\top$ appelée opérateur de modus ponens généralisé.

### II.4.3.2 Aide au diagnostic et aide à la décision

Les applications de la logique floue sont extrêmement nombreuses et variées. Les plus courantes sont la commande floue, les systèmes experts flous, le raisonnement à partir de cas et la reconnaissance floue de formes. Dans le cadre de la surveillance et du diagnostic, on trouve principalement les systèmes experts, le raisonnement à partir de cas et la reconnaissance de formes. Dans ces différents contextes (aide au diagnostic, aide à la décision), l'expert humain exprime des connaissances ou des données dans un langage naturel fondamentalement imprécis ; la logique floue permet donc d'une part de prendre en compte les imprécisions inhérentes aux données et d'autre part de rendre compte de l'expression des règles qui permettent de formuler un diagnostic ou de déterminer une action. On trouve par exemple dans (Rahnamai *et al.*, 1998) le schéma de l'architecture d'un outil de détection/diagnostic d'antennes, dans lequel la logique floue intervient à trois niveaux différents pour réaliser le diagnostic.

Dans le cadre de la surveillance des systèmes industriels, la logique floue se trouve également associée à d'autres outils et techniques d'analyses. Dans (Minca *et al.*, 2002; Minca, 2004), la logique floue est utilisée en association avec les réseaux de Petri pour la conception d'un outil de surveillance temps réel. Le réseau de Petri est utilisé pour modéliser un arbre de défaillance et la logique floue est utilisée pour traduire sous forme d'une base de règles floues l'expression logique que représente l'arbre de défaillance. Le modèle RdPFS (Réseau de Petri Flou pour la Surveillance) ainsi construit permet la

propagation d'un marquage flou représentatif de symptômes à travers le réseau qui donne une connaissance avancée sur les dégradations du système durant le fonctionnement et fournit une analyse dynamique permanente de son état de dégradation. Dans une certaine mesure, cet outil permet donc de réaliser un pronostic sur l'état du système. La logique floue a trouvé également d'autres applications avec les arbres de défaillances dans lesquelles elle est utilisée pour évaluer l'apparition de l'événement sommet de l'arbre (Cheng, 2000).

Dans ces diverses applications, l'utilisation de la logique floue est assez naturelle dans la mesure où elle permet de traiter l'imprécision, l'incertitude et l'incomplétude liées aux connaissances du domaine. Cependant même si la logique floue fournit dans tous ces exemples, des résultats satisfaisants, on ne peut considérer ces applications comme de réelles applications de la logique floue pour le diagnostic dans la mesure où ces différents outils ne s'appliquent pas à la localisation et à l'identification des causes expliquant un défaut. Utilisée avec les arbres de défaillances, cette dernière devrait fournir une évaluation sur l'occurrence ou la présence des événements de base de l'arbre de défaillances qui sont eux à l'origine de l'événement sommet. On obtiendrait ainsi l'évaluation des causes à l'origine d'un dysfonctionnement.

### II.4.4 Les réseaux neuro-flous

Les réseaux neuro-flous sont nés de l'association des réseaux de neurones avec la logique floue, de manière à tirer profit des avantages de chacune de ces deux techniques. La principale propriété des réseaux neuro-flous est leur capacité à traiter dans un même outil des connaissances numériques et symboliques d'un système (Uppal *et al.*, 2002). Ils permettent donc d'exploiter les capacités d'apprentissage des réseaux de neurones d'une part et les capacités de raisonnement de la logique floue d'autre part. Différentes combinaisons de ces deux techniques d'intelligence artificielle existent et mettent en avant des propriétés différentes. On peut identifier les combinaisons suivantes (Ould Abdeslam, 2002) :

- Réseau flou neuronal
- Système neuronal/flou simultanément
- Modèles neuro-flous coopératifs
- Modèles neuro-flous hybrides

Des structures neuro-floues pour la modélisation, la prédiction, le contrôle ou le diagnostic peuvent être réalisées par une grande variété d'architectures pour un même type de combinaison donnée (Dourado *et al.*, 1999). On trouve par exemple dans (Wang *et al.*, 2002) une utilisation d'un système neuro-flou R-SANFIS (Recurrent Self-Adaptative Neuro-Fuzzy Inference System) pour la commande d'un véhicule sous-marin autonome. Une autre utilisation des réseaux neuro-flous est présentée dans (Nauck *et al.*, 1998)

où l'architecture NEFPROX (NEuro Fuzzy function apPROXimator) est utilisée pour l'approximation de fonctions.

Dans des applications de diagnostic, on trouve principalement des modèles neuro-flous hybrides, pour lesquels réseau de neurones et système flou sont combinés de manière homogène. Une définition d'un tel système neuro-flou est donnée dans (Palade *et al.*, 2002) :

*Un système neuro-flou est un réseau de neurones qui est topologiquement équivalent à la structure d'un système flou. Les entrées/sorties du réseau ainsi que les poids sont des nombres réels, mais les nœuds implémentent des opérations spécifiques aux systèmes flous : fuzzyfication, opérateurs flous (conjonction, disjonction), défuzzyfication. En d'autres termes, un système neuro-flou peut être vu comme un système flou pour lequel les opérations sont implémentées de façon parallèle par un réseau de neurones.*

*Applications des réseaux neuro-flous au diagnostic*

On retrouve principalement deux utilisations des réseaux neuro-flous en diagnostic. Ces utilisations en diagnostic reposent la plupart du temps sur la structure de diagnostic suivante (Palade *et al.*, 2002; Uppal *et al.*, 2002).

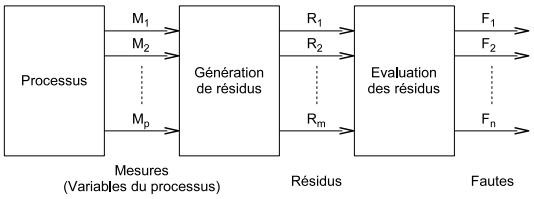



FIG. II.16 – Structure générale d'un système de diagnostic.

Dans cette structure de diagnostic, le diagnostic est basé sur l'étude des résidus qui sont générés par différence d'un signal estimé donné par un observateur avec les valeurs réelles du signal. Ces résidus sont ensuite classés et évalués. Le neuro-flou intervient donc dans ces deux étapes du diagnostic, l'idée étant d'obtenir un classificateur de défauts interprétable.

Une première application est décrite dans (Palade *et al.*, 2002), il s'agit du diagnostic d'une turbine à gaz industrielle. Dans cette application, les résidus sont générés grâce à un observateur neuro-flou TSK. Les résidus sont choisis en fonction de leur représentation des dysfonctionnements possible de la turbine. Dans l'exemple deux défauts de la turbine sont considérés un pour l'actionneur et un pour le compresseur. Compte tenu de la sensibilité des résidus, deux modèles TSK sont utilisés. Un exemple de schéma correspondant à la génération de résidus est donné par la figure II.17.

Les relations entre les résidus et les défauts à diagnostiquer sont données par la table II.3.

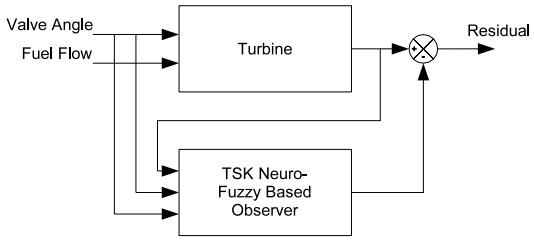



FIG. II.17 – Observateur neuro-flou pour la génération de résidus.

TAB. II.3 – Identification/localisation des défauts.

| | Défaut de l'actionneur | Défaut du compresseur |
|---|---|---|
| R1 | 1 | 1 |
| R2 | 0 | 1 |

Comme le montre la table II.3, pour établir formellement un diagnostic, il faut interpréter les résidus pour remonter jusqu'au défaut concerné, les résidus permettant la détection de l'état de panne. Pour ce faire, les auteurs proposent l'utilisation d'un classificateur neuro-flou qui doit permettre d'identifier sans ambiguïté compte tenu de la valeur des résidus, s'il s'agit d'un défaut de l'actionneur ou du compresseur. Il s'agit donc d'une architecture NEFCLASS qui permet à son utilisateur d'obtenir de manière interactive un classificateur flou avec un bon compromis entre précision et interprétation. Dans cet exemple, l'apprentissage du réseau neuro-flou est réalisé avec 150 motifs des résidus pour chaque défaut.

Une autre application des réseaux neuro-flous est décrite dans (Uppal *et al.*, 2002), il s'agit du diagnostic d'une vanne éléctro-pneumatique. Le principe est exactement identique au précédent : des modèles TSK et Mamdani sont utilisés pour la génération de résidus et un autre réseau neuro-flou est utilisé pour la classification des résidus. En raison de sa plus grande interprètabilité, c'est un modèle Mamdani qui est utilisé pour la classification.

Les réseaux neuro-flous apparaissent donc comme des outils puissants combinant des grandes capacités d'approximation pour la modélisation des systèmes dynamiques non-linéaires pour lesquels le modèle mathématique est inconnu avec la possibilité d'obtenir des résultats possédant un certain niveau d'interprétation.

Plusieurs méthodes sont disponibles pour développer de tels systèmes mais quelques points négatifs sont tout de même à souligner.

Tout d'abord au niveau de la conception de l'outil, il est nécessaire d'avoir des connaissances suffisantes sur le système à diagnostiquer pour déterminer les variables d'entrées, les fonctions d'appartenances, ainsi que les règles. De plus les systèmes flous

sont relativement gourmands en ressources, même si les réseaux de neurones viennent améliorer leurs performances, il serait intéressant d'avoir des précisions sur la possibilité d'effectuer un diagnostic en ligne.

La plupart des applications rencontrées sont basées sur l'établissement d'un diagnostic à partir de la classification de résidus, elles nécessitent donc de pouvoir établir un modèle du système. Des possibilités d'établir des modèles avec des techniques neuro-floues sont présentées ici mais les applications restent toutefois limitées. Il serait donc intéressant d'employer ces techniques compte tenu de leurs capacités en s'affranchissant complètement d'un modèle du système à diagnostiquer.

### II.4.5 Synthèse sur les méthodes de reconnaissances de formes

Nous avons abordé dans cette partie les différentes techniques de l'Intelligence Artificielle qui permettent de réaliser un diagnostic basé sur les méthodes de reconnaissances de formes. Leurs applications sont nombreuses et pour certaines, les résultats sont globalement satisfaisant. Cependant, ces méthodes sont pour la plupart des méthodes qui effectuent une classification par reconnaissance de forme. Le diagnostic revient donc à identifier un mode de fonctionnement du processus qui reflète l'état de panne. En ce sens, le diagnostic réalisé ne permet pas d'identifier formellement les causes du dysfonctionnement à moins que celles si soient explicitement décrites dans le mode identifié ; ce que réalise le raisonnement à partir de cas. Pour les autres outils, les applications s'apparentent plus à de la « détection intelligente », pour laquelle la sortie du système de diagnostic est porteuse d'information sur l'état du système, mais n'en donne pas les causes. Ces outils semblent donc mieux adaptés à la réalisation d'un module de détection dans une architecture complète de surveillance.

Il existent encore une autre classe de méthodes de diagnostic sans modèle numérique du système, basée sur des méthodes à base de modèles explicatifs. Nous verrons dans la section suivante les principaux outils de l'Intelligence Artificielle qui s'intègrent dans ces méthodes à base de modèles explicatifs, et dans quelle mesure ces derniers réalisent un diagnostic.

## II.5 Les méthodes à base de modèles explicatifs

Ces méthodes sont principalement basées sur la représentation des relations entre les différents états de pannes et leurs effets (éventuellement observables). Elles reposent donc sur une analyse profonde du système, de manière à avoir les connaissances suffisantes à l'expression de ces relations de cause à effet. Les modèles ainsi obtenus permettent pour certains une approche abductive qui consiste à remonter aux causes des pannes à partir

des observations correspondant aux symptômes. Plusieurs outils de l'Intelligence Artificielle permettent une telle formalisation des connaissances disponibles sur un système. Il s'agit notamment des graphes causaux, des graphes contextuels, on retrouve également des approches basées sur la logique floue ou les réseaux de Petri. Nous verrons donc dans cette partie comment ces outils par leurs capacités de modélisation et d'expression permettent de fournir des modèles explicatifs dans le cadre d'applications au diagnostic.

## II.5.1 Les graphes causaux

L'exploitation de connaissances causales est assez naturelle pour le diagnostic. En effet, un « dysfonctionnement » peut être assez simplement décrit par les relations associant ses causes à ses manifestations observables. Les graphes causaux constituent un formalisme bien adapté à la représentation de ces liens causaux.

### II.5.1.1 Généralités et principes

Dans une utilisation de diagnostic ils permettent d'exprimer les enchaînements causaux régissant le fonctionnement du système à surveiller en cas de panne. Il s'agit de graphes orientés acycliques. Les nœuds du graphe correspondent aux causes et effets et sont reliés par des arcs orientés. (Grosclaude, 2001)

Cet outil appartient donc aux méthodes à base de modèles explicatifs leur but étant de fournir une explication à des observations d'un fonctionnement anormal d'un système. Ils représentent des connaissances profondes du système capable de donner les relations causales entre les différents états de pannes (Grosclaude *et al.*, 2001). La construction du modèle repose sur une expertise telle que des catalogues de pannes, des AMDEC, ou des arbres de défaillances (Basseville *et al.*, 1996), qui expriment pour une panne l'ensemble des symptômes observables.

Leur principale utilisation en diagnostic consiste en un diagnostic abductif. Le graphe causal est utilisé abductivement, « à partir des symptômes, on recherche les causes » de manière à relier les symptômes observés à un ensemble de causes possibles (Grosclaude *et al.*, 2000). Pour ce faire, plusieurs extensions des graphes causaux existent.

La première extension repose sur l'introduction de contraintes temporelles, on parle alors de « diagnostic abductif temporel » (Brusoni *et al.*, 1995). Une relation causale relie alors une occurrence de la cause à une occurrence de l'effet, des contraintes temporelles décrivent la durée de la cause, le délai entre l'occurrence de la cause et celle de l'effet, ou encore la durée de l'effet (Grosclaude, 2001). Des algorithmes existent pour retrouver dans le graphe les causes expliquant des observations (Brusoni *et al.*, 1995; Brusoni *et al.*, 1997).

Une autre approche des graphes causaux repose sur l'introduction de probabilités. On parle alors de réseaux bayésiens pour lesquels les nœuds représentent des variables aléatoires et les arcs, les dépendances causales et probabilistes entre les variables aléatoires.

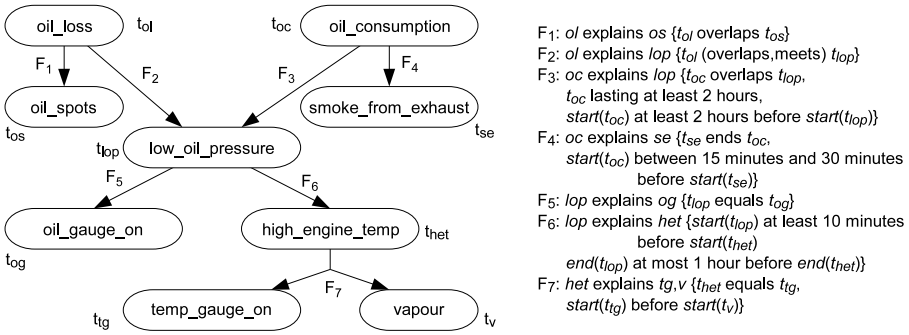



FIG. II.18 – Graphe causal temporel (Brusoni *et al.*, 1995).

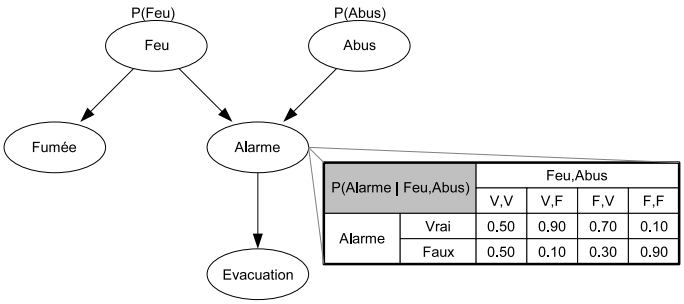



FIG. II.19 – Exemple de réseau bayésien.

### II.5.1.2 Graphes causaux et diagnostic

On trouve plusieurs applications des graphes causaux en diagnostic : dans le domaine médical ainsi que dans le domaine industriel.

Une application dans le domaine industriel réalise le diagnostic de moto-pompe primaire d'une centrale nucléaire EDF. Le graphe causal utilisé compte 500 nœuds et 1000 arcs. Les algorithmes de diagnostic abductif temporel ainsi que les interfaces ont été implémentés sous le langage JAVA (Grosclaude, 2001; Grosclaude *et al.*, 2001). Dans cette application, d'autres extensions ont été apportées aux graphes causaux notamment par la prise en compte de pannes interagissant et l'expression d'effets contradictoires (qui permettent de rendre compte d'action de maintenance par exemple).

Dans (Grosclaude, 2001), les graphes causaux servent de base pour compiler des scénarii utilisés pour le diagnostic. L'abduction sur le graphe causal n'est utilisée que lorsque la reconnaissance de scénarii a échoué pour le diagnostic. Cette approche permet de limiter l'utilisation du modèle causal qui est plus lourd à gérer.

Le graphe causal est un outil particulièrement intéressant pour le diagnostic dans le sens où il peut apporter une justification du diagnostic proposé par le système au travers du chemin causal suivi dans le graphe. De plus, les algorithmes de diagnostic abductif permettent à partir de l'observation de symptômes de rechercher un ensemble de causes possibles qui expliquent par le biais de relations causales les observations. Enfin, l'introduction de contraintes temporelles, d'effets contradictoires et la prise en compte des interactions entre les pannes rend compte de manière plus juste la réalité physique du système à diagnostiquer.

Les graphes causaux reposent donc sur la formalisation des liens causaux qui régissent les états de pannes et nécessitent de ce fait une grande connaissance du système pour établir ces liens causaux ainsi que les contraintes temporelles. De plus, les algorithmes de diagnostic abductif temporel sont relativement complexes et imposent des longs temps de calculs. Un diagnostic en ligne est donc difficilement réalisable en utilisant le modèle causal directement.

Dans (Grosclaude, 2001), des possibilités de pronostic sont envisagées : l'objectif étant de déterminer à partir du diagnostic de l'état courant les évolutions possibles du système. Une possibilité d'obtenir des arbres de décision à partir du modèle causal est également étudiée afin d'orienter le diagnostic au fur et à mesure de l'apparition des symptômes. Dans (Grosclaude *et al.*, 2001), on envisage également l'introduction d'heuristiques de manière à réduire le temps de calcul.

### II.5.2 Les graphes contextuels

Dans un contexte d'aide à la décision, les outils créés doivent assister un opérateur dans ses activités quotidiennes, sans toutefois prendre de décision à sa place. Pour ce faire, l'analyse de l'activité des opérateurs permet de mettre en évidence la structure des raisonnements suivis ainsi que l'organisation des sous-tâches et des actions. Cette structure peut être modélisée par des graphes. Les graphes ainsi créés donnent les différentes méthodes pour atteindre un but et ces méthodes se distinguent par le contexte dans lequel elles s'appliquent ; on parle alors de graphes contextuels. Il s'agit en effet d'un formalisme bien adapté car il est basé sur une expression du raisonnement que suivent les opérateurs en pratique. De nombreux travaux ont été réalisés sur l'utilisation du contexte et des graphes contextuels en vue de la création d'un Système d'Aide à la Gestion d'Incidents dans le Métro. Les graphes contextuels et leur utilisation seront donc présentés au travers de cette application.

### II.5.2.1 Principes

Les graphes contextuels ont été introduits à partir des arbres de décision pour lesquels on passe d'une représentation basée sur le contexte de l'incident à une représentation basée sur le contexte de résolution de l'incident. Les branches de l'arbre qui conduisent à la même action terminale sont rassemblées dans le graphe et un branchement temporel est introduit dans le graphe pour rendre compte des actions et des décisions qui peuvent être réalisées en parallèle.

Le graphe est constitué des éléments suivants :

**Des actions élémentaires :** Elles correspondent aux plus simples actions pouvant être identifiées dans le domaine concerné. Leur organisation dans le graphe contextuel décrit des tâches plus complexes.

**Des activités :** Ces activités sont similaires aux actions élémentaires si ce n'est que leur réalisation est détaillée et induit une planification.

**Des nœuds contextuels :** Les nœuds contextuels correspondent à des éléments représentant les choix.

**Des nœuds de recombinaisons :** La réalisation des actions réduit l'écart entre les stratégies, si bien que plus ou moins rapidement, les stratégies sont similaires. Les nœuds de recombinaisons permettent ainsi de représenter cette convergence de stratégie guidée par la politique globale des opérateurs.

**Des branchements temporels :** Les branchements temporels représentent des groupes d'actions/décisions pouvant être menés en parallèle.

Ces éléments conduisent à la définition suivante :

*Un graphe contextuel est un graphe orienté acyclique ayant une unique source et un unique puits. Les sommets du graphe sont des instances des éléments décrits plus hauts. Ils sont reliés par des arcs montrant la précédence temporelle.*

Remarque : Le fait que le graphe soit acyclique garantit la finitude de l'algorithme de parcours du graphe.

Un graphe contextuel représente ainsi un raisonnement contextuel local construisant une stratégie pour la réalisation d'un but en fonction de la situation et de son évolution. En effet les graphes contextuels sont conçus pour évoluer au cours de leur utilisation.

### II.5.2.2 Application des graphes contextuels

Une application des graphes contextuels est décrite dans (Pasquier, 2002). Ils sont utilisés dans la conception d'un Système d'Aide à la Gestion d'Incidents dans le Métro (SAGIM). L'expérience dans ce domaine montre que les conséquences d'un incident sont

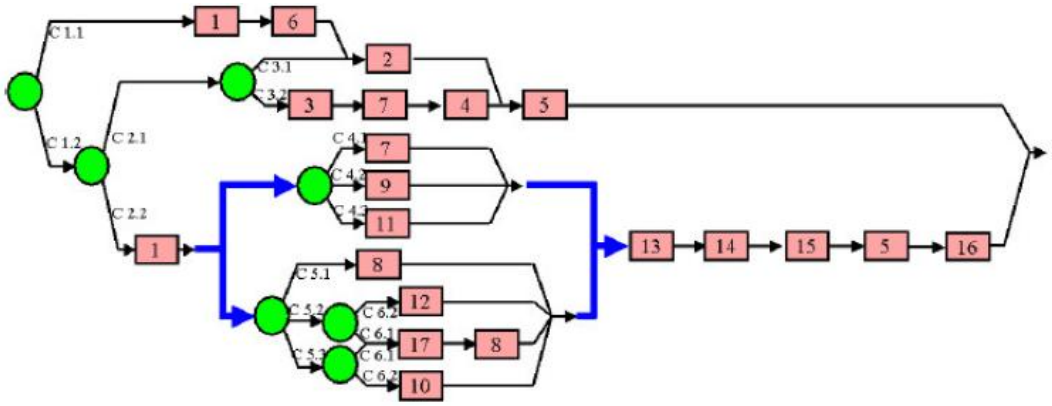



FIG. II.20 – Graphe contextuel.

fortement dépendantes du contexte dans lequel l'incident se produit (e.g. heure de pointe ou non) (Brézillon, 2003). Pour ce faire, les graphes contextuels permettent d'organiser les tâches nécessaires à la résolution d'un incident autour de la notion de contexte. Ceci permet donc de choisir la méthode la plus appropriée en fonction de la situation courante. Les méthodes sont organisées dans une structure montrant l'ordonnancement des actions et des activités secondaires, mais aussi les éléments contextuels sur lesquels se basent les choix stratégiques. Un autre intérêt des graphes contextuels concernant cette application est de permettre l'introduction aisée des nouvelles pratiques d'opérateurs. Généralement, une nouvelle pratique correspond à une pratique connue avec quelques changements introduits par des nœuds contextuels. Ainsi, un système basé sur les graphes contextuels comprend des pratiques utilisées par les opérateurs ou les acquière si besoin. Il n'est donc plus nécessaire de développer *a priori* une représentation exhaustive de la résolution d'un incident (Brézillon, 2003). L'algorithme d'acquisition des connaissances est au cœur du système, ce dernier est donc capable de faire évoluer ses connaissances sur les pratiques des opérateurs (Pasquier, 2002). Dans SAGIM, les graphes contextuels sont associés au raisonnement à partir de cas de manière à former un système hybride. Les deux modes de raisonnement utilisés se complètent et permettent de produire des connaissances plus pertinentes.

Une description de l'implémentation du système est donnée dans (Pasquier, 2002) dans laquelle on retrouve les différentes interfaces et la fonction du système de gestion des incidents. Le prototype ainsi créé a été testé dans des conditions réelles d'exploitation du métro.

Les graphes contextuels permettent donc la représentation d'actions multiples dépendantes du contexte. De plus cette représentation prend en compte la dynamique du contexte dans son évolution. Ils présentent également l'avantage de pouvoir traiter des grandes structures telles que les applications industrielles. Leur représentation est

compréhensible par des opérateurs puisqu'elle est similaire à leur mode de raisonnement. Enfin, leur flexibilité et leur modularité permettent l'acquisition incrémentale de connaissances de manière à intégrer des nouvelles pratiques.

Les graphes contextuels apparaissent donc comme un outil adapté pour la modélisation d'activités comportant une dualité procédure/pratique. Ils sont donc applicables dans des domaines où une interprétation ou une adaptation de règles générales est nécessaire pour prendre en compte la richesse du contexte réel d'application.

Dans le cadre d'une application de supervision, ils pourraient s'appliquer dans des cas où le contexte prend une place importante dans le lien entre diagnostic de défauts et les actions de reprise.

### II.5.3 Les réseaux de Petri

Les réseaux de Petri permettent également dans le cadre du diagnostic une approche en terme de modèles de pannes. Les places constituent alors des états de pannes et l'architecture du réseau permet de rendre compte des relations existant entre ces pannes. Plusieurs techniques se basant sur un modèle de pannes ont été développées, avec en particulier, l'utilisation de réseaux de Petri stochastiques.

Dans (Aghasaryan *et al.*, 1997; Aghasaryan, 1998; Tromp, 2000; Benveniste *et al.*, 2001), une approche par réseaux de Petri aux problèmes de détection de pannes et de diagnostic est proposée. Dans cette approche, le diagnostic est présenté de la manière suivante :

Les alarmes générées par le processus de détection souvent n'identifient pas les pannes et ne déterminent pas exactement leur localisation. Le problème de diagnostic consiste donc à trouver une explication à la présence de symptômes en utilisant les connaissances du domaines (Aghasaryan, 1998). Les réseaux de Petri partiellement stochastiques seront présentés à travers deux approches originales pour le diagnostic. La première concerne une application à la gestion des pannes dans les réseaux de télécommunication, la deuxième traite des systèmes industriels complexes.

Les réseaux de Petri sont des modèles puissants pour l'expression de la concurrence ; des événements concurrents pouvant apparaître sur l'axe du temps dans n'importe quel ordre, ou simultanément. Dans un RdP, les événements sont seulement partiellement ordonnés, ce qui définit une sémantique d'ordre partiel. Cette sémantique d'ordre partiel est particulièrement bien adaptée pour les environnements répartis comme le sont les réseaux de télécommunication. De plus, l'approche est fondée sur une description explicite de la propagation des alarmes et des défauts en utilisant les réseaux de Petri saufs. Cette approche peut donc être vue comme appartenant aux méthodes à base de modèles explicatifs, (Aghasaryan, 1998).

Dans (Tromp, 2000), l'approche pour la construction du modèle est un peu différente, les réseaux de Petri utilisés pour le diagnostic sont établis à partir des arbres de défaillances du système considéré. De plus, l'approche globale du travail est hybride puisqu'elle utilise aussi les résidus issus d'un modèle continu du système.

### II.5.3.1 Expression du non-déterminisme

L'outil proposé ici est probabiliste, les réseaux de Petri ayant une bonne capacité d'expression du non-déterminisme. Le modèle proposé utilise les réseaux de Petri partiellement stochastiques (PSPN). Une vraisemblance est associée aux ordres partiels, le temps est partiellement ordonné, ce qui est approprié aux grands systèmes répartis comme les réseaux de télécommunication. Les deux intérêts des PSPN sont :

- ils conservent l'indépendance probabiliste entre les événements concurrents,
- ils évitent l'exploration explicite de l'espace d'état car seuls les contextes locaux sont décisifs pour la propagation d'événements ou l'évaluation de vraisemblance.

Dans (Tromp, 2000), l'auteur s'inspire de ces PSPN pour construire des PSPN hybrides qui permettent de considérer des événements déterministes, ce modèle étant mieux adapté aux systèmes industriels.

### II.5.3.2 Modélisation

Les réseaux de Petri permettent de modéliser les différentes dépendances entre les défauts (figure II.21).

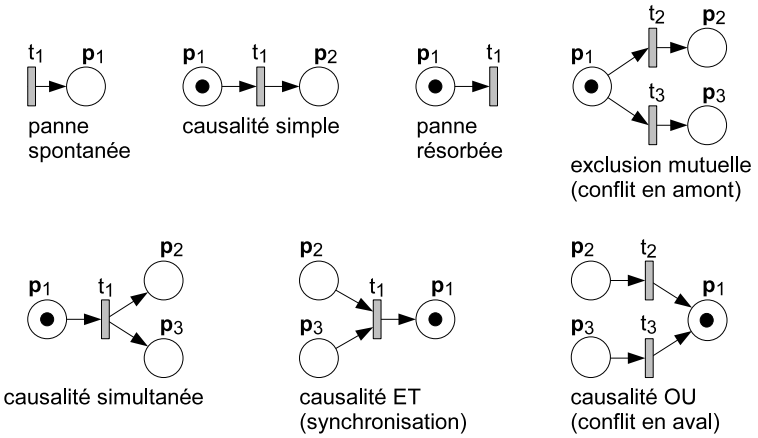



Fig. II.21 – Configurations élémentaires de causalité entre pannes.

Dans l'utilisation qui est faite des réseaux de Petri, les alarmes sont associées aux transitions, les places qui correspondent aux états de pannes ont une capacité de 1, avec une fonction d'étiquetage.

Ces descriptions élémentaires sont la base pour la construction d'un « réseau de défauts » (FAULT NET), pour lequel les places sont bornées à un jeton et les arcs ont un poids de 1. Un franchissement exprime donc une propagation élémentaire de panne et est accompagné par l'observation d'une alarme choisie dans une étiquette.

On trouve également une construction des RdP à partir des arbres de défaillances, dans (Tromp, 2000) (figure II.22).

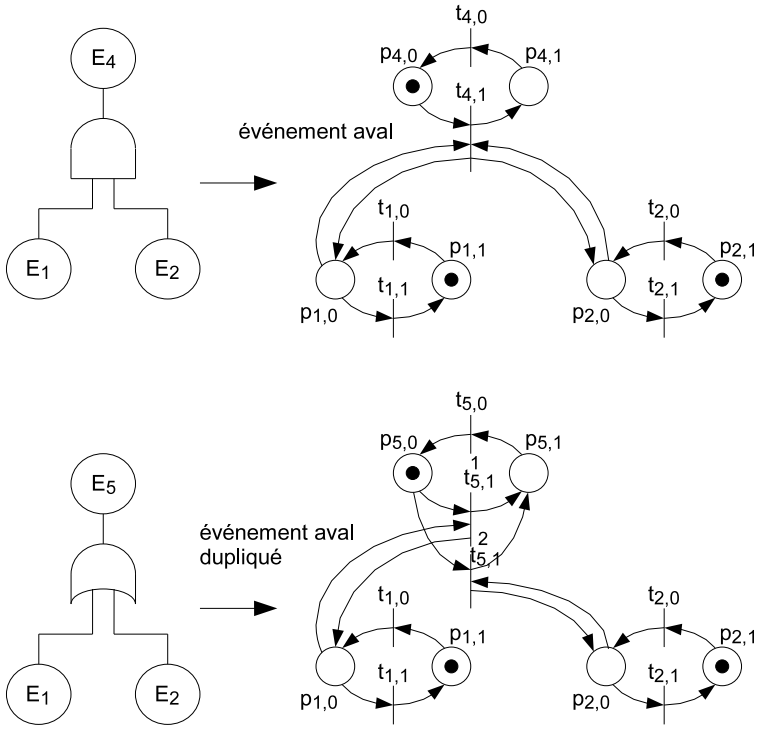



FIG. II.22 – Conversion des portes logique de l'arbre de défaillance.

### II.5.3.3 Trajectoire de réseaux de Petri

Le réseau de Petri définit donc une relation de causalité basée sur le passage de ressource dans une séquence de franchissement. Cette relation de causalité définit un graphe orienté acyclique (graphe de causalité).

Le graphe de causalité (à droite sur la figure II.23) est obtenu à partir de la séquence S={t0,t1,t2,t3,t4,t1}. La sémantique d'ordre partiel du RdP est obtenue à partir de ce

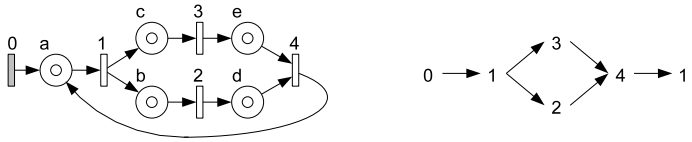



FIG. II.23 – RdP et graphe de causalité.

graphe de causalité. La trajectoire du RdP est un ordre partiel obtenu par fermeture transitive de la relation de causalité à partir du graphe de causalité.

Pour représenter une trajectoire du réseau partiellement stochastique et pour calculer sa vraisemblance, une technique de dépliage du temps est utilisée (figure II.24). Sur ce réseau déplié, le graphe de causalité est obtenu en ne gardant que les connexions qui correspondent aux transitions tirées.

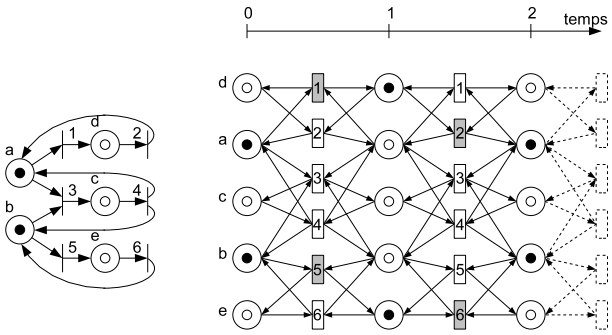



FIG. II.24 – Dépliage du temps et trajectoire de RdP.

Cette méthode de diagnostic offre plusieurs aspects intéressants, tout d'abord les travaux ont été étendus au diagnostic décentralisé. Dans ce cas, les solutions locales sont vues comme des alarmes intelligentes puis mises ensembles au niveau d'un superviseur central.

L'outil développé permet d'exprimer trois notions essentielles pour le diagnostic :

- la concurrence,
- la causalité,
- le non-déterminisme

L'utilisation d'un modèle de réseaux de Petri partiellement stochastique permet de conserver l'indépendance probabiliste des événements concurrents, et ne nécessite pas l'exploration de tout l'espace d'état car seul les contextes locaux sont nécessaires à la propagation des événements ou à l'estimation de vraisemblance. De plus cet aspect de contexte local rend la méthode plus robuste aux changements de configuration du réseau, l'outil est dans une certaine mesure relativement bien évolutif.

En revanche, les aspects de l'approche locale posent des problèmes quant aux résultats dans (Tromp, 2000) et conduisent à des résultats expérimentaux qui ne sont pas satisfaisants.

Une perspective pour l'application aux réseaux de télécommunication serait la création d'un RdP de haut niveau dans lequel les places et les transitions seraient des abstractions de sous-réseaux plus détaillés, ceci permettrait de rendre compte de l'organisation hiérarchique dans le système de diagnostic.

En ce qui concerne l'application industrielle de (Tromp, 2000), plusieurs modifications sont proposées de manière à améliorer les résultats du système de diagnostic.

## II.5.4 La logique Floue

Une autre utilisation de la logique floue en diagnostic est décrite dans (Bouchon-Meunier *et al.*, 2003). Il s'agit ici d'un diagnostic orienté explication qui permet d'expliquer la présence d'un ensemble de symptômes et de remonter aux causes à l'origine de ces observations.

Pour ce faire on utilise la modélisation d'une règle graduelle ($\mathcal{R}$) de causes à effets entre les dysfonctionnements et les symptômes. De la même façon, il sera aussi plus juste de prendre en compte l'incertitude ou le caractère vague de l'observation de certains symptômes.

Selon cette approche, le diagnostic est réalisé grâce à la confrontation entre les connaissances et les observations. Le diagnostic est réalisé en deux étapes : il utilise un indice de cohérence raffiné ensuite par un indice de pertinence.

### II.5.4.1 Diagnostic par cohérence

Le diagnostic par cohérence permet de rejeter tous les dysfonctionnements incohérents avec les observations. Ce premier principe se base sur le modus tollens qui s'énonce comme suit : si $d \Longrightarrow s$ et $\neg s$, alors $\neg d$. Ainsi, seuls les dysfonctionnements pour lesquels les observations ne contredisent aucun de leurs symptômes sont gardés. Le diagnostic par cohérence est permis par un indice de cohérence ($coh$) qui évalue le degré d'intersection entre deux ensembles flous, au sens du maximum de cohérence entre les éléments communs à ces deux ensembles flous :

$$\forall d \in \mathcal{D},\ coh\left(d\right) = \min_{i=1}^{n} \sup_{u \in U_i} \min\left(\pi_i^o\left(u\right), \pi_i^d\left(u\right)\right)$$

avec :

$\mathcal{D} = \{d_1, \ldots, d_p\}$, un ensemble de $p$ dysfonctionnements possibles,

$\mathcal{A} = \{X_1, \ldots, X_n\}$, un ensemble de $n$ attributs observables,

$U_i$, le domaine associé à l'attribut $X_i$,

$\pi_i^d : U_i \to [0,1]$, la distribution de possibilité qui exprime les valeurs plus ou moins possibles de l'attribut $X_i$ lorsque le dysfonctionnement $d$ (seul) est présent,

$\pi_i^d$ représente ainsi l'effet de $d$ sur $X_i$.

Ainsi, avec l'indice de cohérence, un dysfonctionnement sera d'autant plus rejeté que son indice de cohérence sera petit. Ensuite, pour orienter le diagnostic vers les dysfonctionnements les plus plausibles, on met en œuvre un test abductif.

### II.5.4.2 Diagnostic abductif

L'approche abductive permet la sélection des dysfonctionnements qui expliquent effectivement les observations. Elle peut se formuler de la façon suivante : sachant $d \Longrightarrow S$ et ayant observé l'ensemble de symptômes $S$, on conclut que $d$ est une explication pertinente des observations. Ainsi, parmi les dysfonctionnements sélectionnés par la cohérence, l'abduction met l'accent sur les seuls dysfonctionnements qui sont pertinents. Pour ce faire, on utilise un indice d'inclusion flou. Ces indices ont été largement étudiés ; ils sontgénéralement définis à partir d'un indice d'implication flou (Dubois *et al.*, 2000). Le diagnostic abductif est alors possible avec l'indice *per* (pertinence des observations) :

$$\forall d \in \mathcal{D},\ per\,(d) = \min_{i=1}^{n} \inf_{u \in U_i} \pi_i^o\,(u) \to \pi_i^d\,(u)$$

où $\to$ est une implication floue telle que l'implication forte de Dienes :

$$a \underset{D}{\to} b = \max\,(1-a, b)$$

Le diagnostic est alors établi de la manière suivante : *coh* permet tout d'abord de rejeter les dysfonctionnements incohérents avec les observations. Puis *per* vient affiner le diagnostic afin de sélectionner et de classer par ordre de suspicion les dysfonctionnements restants.

Lors d'une phase courante de diagnostic, nous pouvons considérer des attributs binaires : ils ne peuvent prendre que deux valeurs $a_i$ et $p_i$, pour l'absence et la présence du symptôme $i$. Nous avons ainsi : $U_i = \{a_i, p_i\}$.

L'indice de cohérence et l'indice abductif s'écrivent alors :

$$coh\,(d) = \min_{i=1}^{n}\ \max\left(\min\left(\pi_i^o\,(a_i)\,, \pi_i^d\,(a_i)\right), \min\left(\pi_i^o\,(p_i)\,, \pi_i^d\,(p_i)\right)\right);$$

$$per\,(d) = \min_{i=1}^{n}\ \min\left(\pi_i^o\,(a_i) \to \pi_i^d\,(a_i)\,, \pi_i^o\,(p_i) \to \pi_i^d\,(p_i)\right).$$

Si ces équations s'appuient sur les possibilités, elles peuvent être transformées en s'appuyant sur les nécessités ou certitudes (Cayrac *et al.*, 1996; Dubois *et al.*, 1995). Notons $\Gamma_d^+(s_i)$ et $\Gamma_d^-(s_i)$ les degrés de certitude (respectifs de présence et d'absence du symptôme $i$ lorsque le dysfonctionnement $d$ est présent. Notons $\gamma_d^+(s_i)$ et $\gamma_d^-(s_i)$ les degrés de certitude (respectifs) d'avoir observé la présence ou l'absence du symptôme $i$. Ces degrés sont liés aux distributions de possibilités par le système suivant :

$$\begin{cases} \pi_i^d(p_i) = 1 - \Gamma_d^-(s_i) \\ \pi_i^d(a_i) = 1 - \Gamma_d^+(s_i) \\ \pi_i^o(p_i) = 1 - \gamma^-(s_i) \\ \pi_i^o(p_i) = 1 - \gamma^+(s_i) \end{cases}$$

Par distributivité de min sur max et en appliquant les conditions de normalisation, on obtient :

$$coh(d) = 1 - \max\left(coh\left(\gamma^+, \Gamma_d^-\right), coh\left(\Gamma_d^+, \gamma^-\right)\right),$$

où $coh(F,G)$ est le degré max-min de cohérence entre les ensembles flous $F$ et $G$.

$$coh(F,G) = \max_i \min(F(s_i), G(s_i)).$$

En ce qui concerne l'indice abductif, on obtient l'écriture suivante :

$$per(d) = \min\left(incl\left(\Gamma_d^+, \gamma^+\right), incl\left(\Gamma_d^-, \gamma^-\right)\right),$$

où $incl(F,G)$ est le degré d'inclusion floue de l'ensemble flou $F$ dans l'ensemble flou $G$.

$$incl(F,G) = \min_i (F(s_i) \rightarrow G(S_i)).$$

A travers les différents exemples décrits, la logique floue apparaît donc comme un outil puissant pour traiter de l'imprécision, de l'incertitude, et du raisonnement. Les possibilités qu'elle offre de traiter à la fois des données symboliques et numériques en font un outil tout indiqué pour le diagnostic. En effet, le diagnostic nécessite de traiter avec des connaissances et des raisonnements humains exprimés de ce fait sous forme linguistique. L'utilisation de ces connaissances à travers les raisonnements doit se faire de façon qualitative mais aussi quantitative pour fournir une aide au diagnostic pertinente et efficace.

D'autres travaux sur la logique floue s'intéressent également à la réalisation d'un raisonnement abductif par l'inversion du modus ponens. Étant donné une règle floue du type « si $U$ est $A$ alors $V$ est $B$ », et une observation de type $V$ est $B'$, les auteurs cherchent à caractériser les hypothèses de type $U$ est $A'$ répondant à la question « pourquoi $V$ est-il $B'$ ? » Cette approche abductive basée sur l'inversion du modus ponens généralisé doit permettre de retrouver les hypothèses satisfaisant une observation donnée. Si la règle de départ modélise une relation causale pour laquelle la prémisse $A$ est

une cause et la conclusion $B$ est un effet. L'approche abductive partant de l'observation $B'$ de l'effet, permet de caractériser la cause par l'hypothèse $A'$ et réalise donc un diagnostic, (Mellouli *et al.*, 2000*a*; Mellouli *et al.*, 2000*b*; Mellouli *et al.*, 2003).

## II.6 Conclusion

Pour conclure ce chapitre de présentation des différents outils de l'IA appliqués à la surveillance rencontrés dans la littérature, nous avons pu constater que la plupart des techniques rencontrées effectuaient soit une détection (plus ou moins intelligente), soit un diagnostic (tel qu'il est défini dans (Peng *et al.*, 1990)). Nous allons revenir sur quatre points essentiels énoncés dans (Basseville *et al.*, 1996) pour évaluer et comparer les différentes solutions proposées. En effet, ces quatre points clés reviennent dans quasiment tous les outils présentés dans ce chapitre et permettent de mettre en avant les avantages et les inconvénients des différentes méthodes de surveillance rencontrées. Ils peuvent être énoncés de la façon suivante :

1. les difficultés liées à l'acquisition des informations nécessaires et en particulier des modèles,
2. la capacité à prendre en compte l'incertain et l'imprécision,
3. la généricité des outils et leur capacité à évoluer avec le système,
4. la validation (ou l'évaluation) des résultats obtenus.

– *Acquisition des modèles*

Les problèmes liés à l'acquisition des modèles sont donc communs à toutes les méthodes de surveillances rencontrées.

Plus particulièrement, pour les méthodes à base de modèles opérationnels, les modèles décrivent le système dans son fonctionnement normal (avec éventuellement des états de panne) qui est donc le mieux connu, ce qui rend l'acquisition du modèle un peu moins délicate et notamment pour la complétude du modèle. En revanche la pertinence du modèle dépend fortement du niveau d'abstraction donné à ce modèle.

Pour les méthodes de reconnaissance telles que celles basées sur les réseaux de neurones, la logique floue ou les réseaux neuro-flous, la définition du vecteur forme est un point crucial qui nécessite des connaissances suffisantes pour définir les éléments les plus pertinents qui constitueront les composantes du vecteur. De plus, pour les structures à couches comme les réseaux de neurones ou les réseaux neuro-flous, le nombre d'éléments par couche entre également dans les difficultés rencontrées lors de l'acquisition du modèle. Pour ces méthodes, du modèle va également dépendre l'interprétabilité des résultats de diagnostic. Pour exemple, pour un système de RàPC, dans la structure

des cas, la définition des symptômes et des origines est fortement liée au niveau d'analyse auquel on se place pour les définir et de cette définition va également dépendre l'efficacité du système. En effet, la définition des cas est primordiale et conditionne complètement le système, elle nécessite donc une très bonne expertise du système et de ces dysfonctionnements. En revanche, un point fort pour toutes ces méthodes est la notion d'apprentissage qui permet de garantir une meilleure complétude du système puisque les cas non rencontrés peuvent être appris au cours de l'utilisation.

Les méthodes à base de modèles explicatifs reposent sur l'expression des relations causes-effets entre les dysfonctionnements, les pannes et leurs effets observables. Le diagnostic réalisé par le modèle dépend donc directement de l'expression de ces relations. Un des problèmes sera donc de trouver le bon formalisme pour traduire de façon efficace et exploitable la connaissance de ces relations. Cependant il apparaît que même si le modèle dépend uniquement des connaissances de son comportement en cas de dysfonctionnement, des outils tels que l'AMDEC, les arbres de défaillances ou les catalogues de pannes facilitent son acquisition et garantissent une certaine complétude. Le problème se complique en revanche lorsqu'on y ajoute des contraintes temporelles qui elles, ne sont pas fréquemment explicitées dans les outils cités plus haut.

L'extraction de la connaissance disponible sur un système et sa formalisation constitue donc une étape très importante dans la conception d'un système de surveillance. De cette étape va dépendre l'efficacité du système, et surtout son interprètabilité.

– *Prise en compte de l'incertain*

Dans les activités de surveillance, les imprécisions et les incertitudes interviennent à différents niveaux. Un système de surveillance qui se veut efficace se doit donc de les prendre en compte. Nous avons vu que plusieurs solutions sont proposées dans la littérature. Les réseaux neuronaux par exemple de part leur nature même permettent de traiter des données bruitées ou incomplètes. De plus, la plupart des réseaux rendent également compte des incertitudes part la valeur de la sortie du réseau qui correspond à une valeur qui peut être comprise entre 0 et 1 en fonction de l'appartenance du vecteur d'entrée à une classe. Avec les réseaux de Petri partiellement stochastiques, on trouve des algorithmes qui permettent d'associer une vraisemblance à une trajectoire du réseau et constituent donc un modèle probabiliste. Associées aux graphes causaux, les probabilités constituent les réseaux Bayésiens, ce qui permet la prise en compte des incertitudes. Cependant, comme pour les réseaux de Petri stochastiques, les algorithmes associés aux réseaux Bayésiens sont souvent coûteux en terme de ressources et donc difficilement applicables à la surveillance.

La logique floue tient une place un peu particulière en ce qui concerne la prise en compte de l'incertain. En effet, en plus des systèmes basés uniquement sur la logique floue comme la reconnaissance floue de forme, il est fréquent qu'elle soit associée à

d'autres outils pour introduire la prise en compte des incertitudes et des imprécisions sur le système à surveiller. On trouve par exemple les réseaux neuro-flous pour lesquels elle fournit une meilleure interprètabilité des résultats tout comme pour les systèmes d'aide à la décision utilisant la logique floue. Elle est également introduite en extension des réseaux de Petri pour la prise en compte de l'incertitude et enfin, dans les méthodes à base de modèles explicatifs, elle permet d'introduire des niveaux de certitude sur les liens entre des dysfonctionnements et leurs symptômes associés introduisant ainsi l'imprécision sur les données au cœur même du raisonnement à la base du diagnostic. Elle apparaît donc comme indispensable à tout système de diagnostic.

– *Généricité des outils*

Comme nous l'avons vu, l'acquisition du modèle est une tâche complexe, de plus la surveillance nécessite des connaissances qui sont le plus souvent propres aux systèmes considérés et nécessite donc une bonne expertise du système en particulier pour les méthodes de reconnaissance. En effet, pour ces méthodes, il est difficile de mesurer l'impact d'une modification du système sur l'ensemble de l'expertise. Ce problème se retrouve également pour les méthodes à base de modèles explicatifs. A plusieurs reprises, les réseaux de neurones sont mis en avant pour leur capacité de modélisation et d'apprentissage. Ils permettent par exemple d'obtenir un modèle global du système par composition de modèles élémentaires et limitent ainsi les répercussions d'une modification du système.

– *Validation des résultats*

Le diagnostic doit fournir une explication aux différents états de pannes du système, une première solution consiste donc à confronter le diagnostic avec un retour d'expérience s'il est disponible. Se pose alors le problème des pannes éventuelles qui ne sont encore jamais survenues sur le système. En effet, il peut être dangereux de fournir une explication dans tous les cas, si les pannes sont encore inconnues dans l'historique du système, et que les connaissances liées à ces pannes sont trop incomplètes. La confrontation des diagnostics à ceux émis par les experts et les opérateurs de conduite est une des solutions les plus couramment employées (Basseville *et al.*, 1996).

La surveillance d'un système industriel se décompose en deux parties, la détection et le diagnostic. Nous avons pu constater au cours de nos recherches qu'aucun outil n'était capable de tout faire à part dans quelque cas très particulier, mais nous nous classons plus dans un cas de détection intelligente que dans le cas d'une détection suivi d'un diagnostic (c'est-à-dire une localisation et une recherche de causes). Nous en avons donc déduit qu'il ne devait pas y avoir qu'un seul outil pour faire toute la surveillance, mais deux.

Pour la partie détection, où le temps est très important, nous nous sommes intéressés aux réseaux de neurones temporels et plus particulièrement aux réseaux de neurones

récurrents pour leur capacité d'apprentissage, leur parallélisme dans le traitement, leur capacité de faire face à des problèmes inhérents à la non-linéarité des systèmes, et leur rapidité de traitement quand ils sont implémentés en circuit intégré.

Pour la partie diagnostic, deux outils semblent donc être particulièrement intéressants. Les réseaux de neurones d'une part et la logique floue d'autre part qui semble indispensable pour fournir une aide au diagnostic pertinente par ses capacités à formaliser les connaissances nécessaires aux systèmes de diagnostic et à prendre en compte l'imprécision et l'incertitude.

Le chapitre suivant présente quelques perspectives de ces outils à travers la proposition d'un outil de surveillance basé sur une utilisation conjointe des réseaux de neurones récurrents et des réseaux neuro-flous. Nous verrons dans quelle mesure il est possible de réaliser une aide au diagnostic.

# Chapitre III

# Utilisation des réseaux neuro-flous pour la surveillance

---

*La surveillance industrielle se décompose en deux actions distinctes : la détection de défauts et le diagnostic de pannes. Pour chaque action, nous avons défini un outil approprié. L'outil de détection devant prendre en compte le temps est un réseau récurrent à fonction de base radiale, RRFR, tandis que l'outil de diagnostic est un réseau neuro-flou qui est développé spécialement pour cette action. Le réseau RRFR gère le temps via sa première couche constituée de neurones à fonction de base sigmoïdale et à retour local de la sortie. Elle agit en temps que mémoire dynamique. La couche cachée constituée de neurones gaussiens représente la mémoire statique du réseau. Les algorithmes d'apprentissage que nous avons établi permettent de configurer et d'initialiser le réseau à l'aide de deux paramètres (un degré d'oubli et un degré de pertinence) et d'une base d'apprentissage. Cette base d'apprentissage contient l'ensemble des modes de fonctionnement du système à surveiller ainsi que les séquences de mesure des différents capteurs permettant d'obtenir les modes de fonctionnement. L'apprentissage du réseau neuro-flou utilise quant à lui l'arbre de défaillance et l'AMDEC du système.*

---

# III.1 Introduction

Dans de nombreuses applications de surveillance industrielle avec des outils de l'intelligence artificielle, nous retrouvons souvent la même erreur : considérer une situation de surveillance comme une action de diagnostic, voire même comme une action de détection. Pour notre part, nous avons fait le choix de considérer qu'une surveillance se décompose en une phase de détection et une phase de diagnostic. Dans ce chapitre, nous présentons dans un premier temps notre outil d'aide à la surveillance composé de deux outils : un outil de détection et un outil de diagnostic. L'outil de détection occupe la deuxième section de ce chapitre dans laquelle nous présentons son architecture basée sur le réseau récurrent RRBF, ainsi que ses algorithmes d'apprentissage. Enfin, nous présentons notre outil d'aide au diagnostic basé sur une architecture neuro-floue, son architecture, les principes d'apprentissage ainsi que ses liaisons avec l'outil de détection.

# III.2 Présentation de l'outil

La surveillance industrielle se décompose en deux tâches : la détection et le diagnostic de défaillances/dégradations (localisation et identification des causes) (Wan *et al.*, 1999; Pencolé, 2002; Tromp, 2000). Un outil de surveillance efficace doit pouvoir être facilement mis à jour afin de suivre l'évolution du système et intégrer le retour d'expérience des opérateurs.

Afin de créer notre outil de surveillance, nous utilisons les informations de la maintenance et de la production. Ces informations sont fournies par :

- l'analyse des modes de défaillances, de leurs effets et de leur criticité – AMDEC,
- l'arbre de défaillances – AdD,
- l'analyse fonctionnelle,
- le retour d'expérience des opérateurs et responsables production et maintenance,
- la gestion de la maintenance assistée par ordinateur – GMAO,
- les systèmes d'acquisition et de contrôle de données – SCADA,
- ...

La méthode proposée concerne toutes les phases de la fonction de surveillance : la détection et le diagnostic de défaut.

- *L'outil de détection dynamique.* En entrée de l'outil de détection, nous trouvons les informations données par les capteurs. Ces données sont traitées dynamiquement. Les sorties donnent les modes opératoires (symptômes) de l'équipement surveillé. Conformément à nos conclusions du chapitre précédent, nous utilisons les réseaux de neurones récurrents.

– *L'outil de diagnostic.* L'entrée de l'outil de diagnostic reprend les sorties du système de détection : les modes opératoires. Nous trouvons aussi en entrée des données de type qualitative ou quantitative que l'opérateur pourra ajouter pour améliorer le diagnostic. En sortie, nous trouverons les différentes causes possibles associées à un degré de crédibilité et un degré de sévérité pour chacune d'elles. Ces degrés aideront le responsable maintenant à évaluer et planifier les actions de maintenance. En suivant les conclusions du chapitre précédent nous utilisons les réseaux de neurones ainsi que la logique floue dans un seul et même outil appelé réseau neuro-flou.

La figure suivante représente de façon schématique notre outil de surveillance.

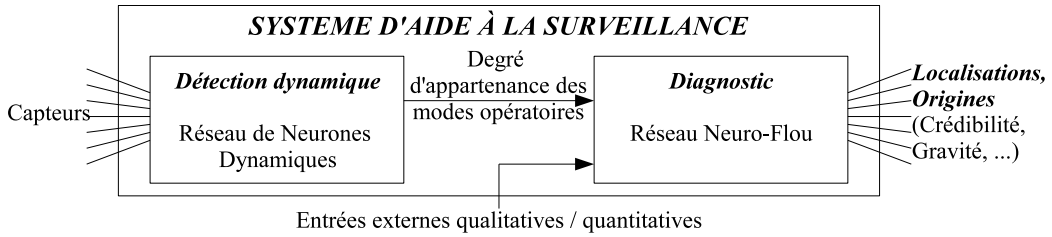



FIG. III.1 – Schéma du système d'aide à la surveillance

En fonctionnement, le système de détection scrute en permanence le système. Lorsqu'une panne ou une défaillance se produit, une alarme est déclenchée et un diagnostic est lancé. En utilisant les informations provenant du système de détection, le système d'aide au diagnostic propose à l'opérateur les causes possibles de ce problème, ainsi que les interprétations floues de ces causes.

Nous allons voir par la suite la description des outils pour chaque phase. Dans un premier temps, nous nous intéressons à la phase de détection, puis dans un deuxième temps à la phase de diagnostic.

## III.3 Phase de Détection

Suite à notre étude sur les réseaux de neurones temporels (Palluat *et al.*, 2005*b*), brièvement présentée au chapitre II.4.2.6, nous appliquerons, pour cette phase, un réseau de neurones dynamiques en l'occurrence le réseau de neurones RRFR (Zemouri, 2003) utilisé en classification.

Le réseau RRBF est un réseau RBF dont on a modifié la couche d'entrée afin d'introduire la notion dynamique. En effet, les neurones d'entrée ne sont plus linéaires mais bouclés avec une sigmoïde comme fonction d'activation (figure III.2).

Le réseau de neurones RRBF (Zemouri *et al.*, 2001) utilise une représentation interne implicite du temps. Cet aspect dynamique est obtenu par une récurrence des connexions

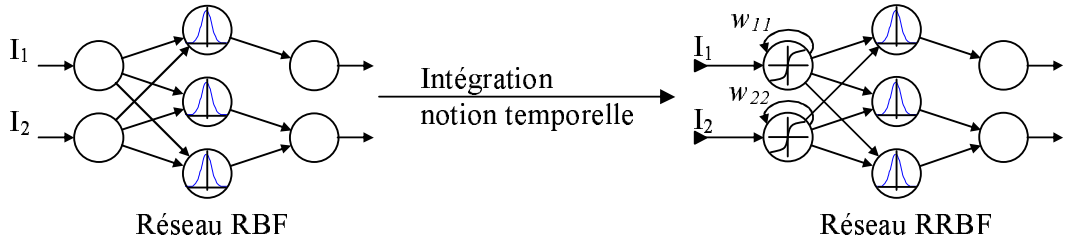



FIG. III.2 – Du réseau RFR au réseau RRBF

au niveau des neurones de la couche d'entrée. Ces auto-connexions procurent aux neurones d'entrée une capacité de prise en compte d'un certain passé des données en entrée. Nous pouvons ainsi qualifier l'ensemble de ces neurones bouclés de *mémoire dynamique* du réseau de neurones. Le réseau RRFR est donc doté de deux types de mémoires : une mémoire *dynamique* (couche d'entrée) pour la prise en compte de la dynamique des données en entrée, et une mémoire *statique* (couche cachée) pour mémoriser les prototypes. La couche de sortie représente la couche de décision. (Zemouri *et al.*, 2002)

L'avantage principal de cette représentation est la séparation de la mémoire statique et de la mémoire dynamique, nous pouvons donc partager l'apprentissage en deux parties et utiliser pour l'apprentissage de la mémoire statique et de la couche de décision les techniques classiques d'apprentissage pour les réseaux de neurones statiques. Nous verrons par la suite que ces techniques ne sont pas optimales. Nous introduirons alors un nouvel algorithme.

### III.3.1 Mémoire dynamique

La couche dynamique est constituée de neurones à fonction d'activation sigmoïdale et retour local de la sortie (figure III.3).

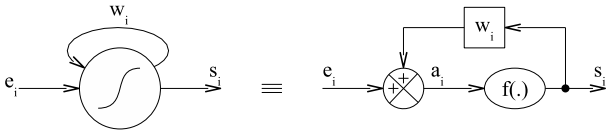



FIG. III.3 – Neurone Bouclé

L'activation du neurone bouclé $i$ à l'instant $t$ est la somme de son entrée $e_i(t)$ avec sa sortie à l'instant précédent $s_i(t-1)$ pondérée par le poids de l'auto-connexion $w_i$.

$$a_i(t) = e_i(t) + w_i \cdot s_i(t-1)$$

La sortie du neurone $i$ est définie par :

$$s_i(t) = f(a_i(t))$$

La fonction d'activation $f(.)$ est la sigmoïde (Figure III.4) définie par :

$$f(x) = \frac{1 - \exp(-k \cdot x)}{1 + \exp(-k \cdot x)}$$

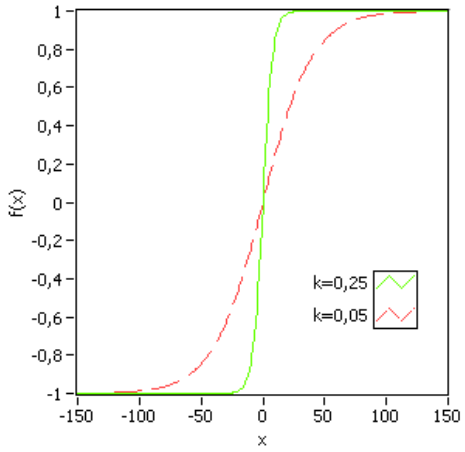



FIG. III.4 – Fonction d'activation du neurone bouclé

Donc le neurone bouclé $i$ est régi par l'équation suivante :

$$s_i(t) = \frac{1 - \exp(-k_i \cdot e_i(t) - k_i \cdot w_i \cdot s_i(t-1))}{1 + \exp(-k_i \cdot e_i(t) - k_i \cdot w_i \cdot s_i(t-1))}$$

Les paramètres $k_i$ et $w_i$ définissent la dynamique du système. Dans (Zemouri, 2003), l'auteur définit que le produit $k_i \cdot w_i$ correspond à la mémoire du système avec un phénomène d'oubli pour $k_i \cdot w_i < 2$ et un phénomène de mémorisation sinon.

La phase d'apprentissage de cette couche consistera donc en la détermination de ces deux paramètres pour chaque entrée du réseau. Avec cette base, nous avons développé une méthode d'apprentissage de ces paramètres dans le cadre d'une utilisation en classification dynamique.

Les hypothèses que nous utilisons pour ce type d'applications sont les suivantes. Le réseau est construit à partir d'une *base d'apprentissage* qui est constitué d'un ensemble de *séquences* auxquelles sont associées des *classes*, représentant les *modes de fonctionnement*, que nous pourrons regrouper en deux grandes familles : les classes de bon fonctionnement et les classes de mauvais fonctionnement. Ces classes s'excluent

mutuellement. Une séquence est constituée d'un ensemble de *vecteurs*. Chaque vecteur est constitué de *points* représentant les valeurs numériques de chaque entrée du réseau. Les données pouvant être de différentes natures, nous nous proposons de les normaliser. Cette normalisation se justifie aussi par le fait de la nature de la fonction d'activation, la sigmoïde. L'utilisation d'une telle fonction suppose que les variations des entrées se situent autour de 0. La référence qui nous servira pour la normalisation des données provient des séquences de bon fonctionnement. Les données sont normalisées en supposant que toutes les valeurs de bon fonctionnement se situent dans un rayon de 0,1 autour de 0 (algorithme III.1).

**Fonction** Normalise_$e_i$( $e_i$ : **matrice de réels** ; $n_{BF}$, $N$ : **entiers**) : **matrice de réels**

*[$e_i$ entrée $i$ à normaliser, $n_{BF}$ le nombre de séquences de bon fonctionnement, et $N$, le nombre de vecteurs dans une séquence.]*

*[Définition des variables locales]*

$e_i^{norm}$ : **matrice de réels** ; *[entrée $i$ normalisée]*

$e_i^{max}$, $e_i^{moy}$, $e_i^{min}$ : **réels** ;

*[initialisation des valeurs de la référence]*

$$e_i^{max} \leftarrow \max_{l \in BF,\ v=1,\dots N} \left(e_{iv}^{l}\right) ;$$

$$e_i^{min} \leftarrow \min_{l \in BF,\ v=1,\dots N} \left(e_{iv}^{l}\right) ;$$

$$e_i^{moy} = \frac{1}{n_{BF} \cdot N} \sum_{l=1}^{n_{BF}} \sum_{v=1}^{N} e_{iv}^{l} ;$$

*[Boucle de calcul]*

**Si** $(e_i^{max} - e_i^{moy} \geqslant e_i^{moy} - e_i^{min})$ **Alors**

$$e_i^{norm} \leftarrow \frac{0{,}1 \cdot (e_i - e_i^{moy})}{e_i^{max} - e_i^{moy}} ;$$

**Sinon**

$$e_i^{norm} \leftarrow \frac{0{,}1 \cdot (e_i - e_i^{moy})}{e_i^{moy} - e_i^{min}} ;$$

**Fin Si**

**Retourner** $e_i^{norm}$ ;

**Fin**

Alg III.1: Normalisation de la base d'apprentissage

Après cette première étape de normalisation, nous déterminons les paramètres de la mémoire dynamique. Le premier élément important à connaître est la taille de la mémoire, c'est-à-dire la durée du souvenir d'un événement. Classiquement, un réseau de neurones artificiels comme celui que nous utilisons fonctionne en temps discret, le temps est échantillonné. Donc si nous pouvons connaître le temps entre deux échantillons et en connaissant le nombre d'échantillons nous pouvons déterminer la durée du « souvenir ».

Le temps entre deux échantillons dépendant de l'implantation du réseau, nous nous pencherons sur le problème du nombre d'échantillons. Sachant que la base d'apprentissage est constituée de séquences, il est supposé que la longueur de la séquence est suffisante pour reconnaître un mode de fonctionnement, nous considérerons donc que le nombre de vecteurs dans une séquence correspondra au nombre d'échantillons. Pourtant la durée du souvenir est fonction d'une autre variable : l'oubli. En effet, il est nécessaire de définir un niveau pour lequel l'information devient inutilisable. Nous utilisons pour cela le seuil d'oubli.

Pour simuler un comportement simple d'oubli, nous faisons les hypothèses suivantes :

– à l'instant $t = -1$, nous produisons un événement en entrée tel que $s\,(0) = 1$ ;
– pour les instants $t > 0$, $e\,(t) = 0$

L'équation régissant le neurone bouclé $i$ s'écrit alors :

$$s_i\,(t) = \frac{1 - \exp\left(-k_i \cdot w_i \cdot s_i\,(t-1)\right)}{1 + \exp\left(-k_i \cdot w_i \cdot s_i\,(t-1)\right)}$$

Sachant que la taille de la séquence ne varie pas suivant l'entrée, nous posons :

$$\delta = k_i \cdot w_i, \quad \text{pour tout } i$$

L'équation de sortie s'écrit alors :

$$s\,(t) = \frac{1 - \exp\left(-\delta \cdot s\,(t-1)\right)}{1 + \exp\left(-\delta \cdot s\,(t-1)\right)}$$

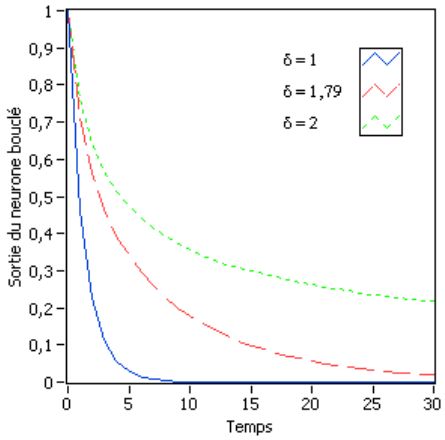



FIG. III.5 – Evolution de la sortie du neurone bouclé en fonction de $\delta$ et du temps

Sur la figure III.5, nous pouvons observer l'évolution de la sortie du neurone bouclé en fonction de $\delta$ et du temps. La valeur désirée de $\delta$ répondra à l'équation :

$$s\left(N\right) = S_o$$

avec $N$ le nombre de vecteurs dans une séquence, et $S_o$ le seuil d'oubli.

Pour déterminer $\delta$, nous avons mis au point deux méthodes, une méthode par récurrence (algorithme III.2), et une méthode par approximation en déterminant la fonction $\delta = f\left(S_o, N\right)$.

Cette fonction peut être établie en effectuant un développement limité de :

$$f\left(x\right) = \frac{1 - \exp\left(-\delta \cdot x\right)}{1 + \exp\left(-\delta \cdot x\right)}$$

Pour cela, on utilise la formule de Taylor :

$$f\left(x\right) = f\left(x_0\right) + \sum_{p=1}^{+\infty} \frac{\left(x - x_0\right)^p}{p!} f^{(p)}\left(x_0\right)$$

Pour $\delta < 2$, nous avons un comportement d'oubli, $\lim\limits_{t \to +\infty} \left(s\left(t\right)\right) = 0$. Nous posons donc $x_0 = 0$ et la formule de Taylor devient celle de Mac-Laurin :

$$f\left(x\right) = f\left(0\right) + \sum_{p=1}^{+\infty} \frac{\left(x\right)^p}{p!} f^{(p)}\left(0\right)$$

Nous allons déterminer les dérivées successives de $f\left(x\right)$ jusqu'à l'ordre 2 :

$$\begin{aligned}
f\left(x\right) &= \frac{1 - \exp\left(-\delta \cdot x\right)}{1 + \exp\left(-\delta \cdot x\right)} \\
f^{(1)}\left(x\right) &= \frac{2 \cdot \delta \cdot \exp\left(-\delta \cdot x\right)}{\left(1 + \exp\left(-\delta \cdot x\right)\right)^2} \\
f^{(2)}\left(x\right) &= \frac{-2 \cdot \delta^2 \cdot \left(1 - \exp\left(-\delta \cdot x\right)\right) \cdot \exp\left(-\delta \cdot x\right)}{\left(1 + \exp\left(-\delta \cdot x\right)\right)^3} \\
&\dots
\end{aligned}$$

**Fonction** Calcul_$\delta$( $N$ : **entier** ; $S_o$ : **réel**) : **réel**
*[Définition des variables locales]*
$\delta$, $\delta^{new}$, $\delta_{max}$, $\delta^{new}_{max}$, $\delta_{min}$, $\delta^{new}_{min}$, $s_{n-1}$, $s_n$, $\Delta$ : **réel** ;
*[initialisation des variables locales]*
$\delta \leftarrow 1$ ;
$\delta_{max} \leftarrow 2$ ;
$\delta_{min} \leftarrow 0$ ;
$s_{n-1} \leftarrow 1$ ; *[initialisation de la sortie]*
$\Delta \leftarrow 0{,}01$ ; *[Précision de $\delta$]*
*[Boucle de calcul]*
**Répéter**
**Pour** i **de** 1 **à** N **faire**
$s_n \leftarrow \dfrac{1 - \exp(-\delta \cdot s_{n-1})}{1 + \exp(-\delta \cdot s_{n-1})}$ ;
$s_{n-1} \leftarrow s_n$ ;
**Fin Pour**
*[ajustement des valeurs de $\delta$, $\delta_{max}$ et $\delta_{min}$]*
**Si** $(s_{n-1} \leq S_o)$ **Alors**
$\delta^{new}_{max} \leftarrow \delta$ ;
$\delta^{new} \leftarrow \dfrac{\delta_{max} + \delta}{2}$ ;
$\delta \leftarrow \delta^{new}$ ;
$\delta_{max} \leftarrow \delta^{new}_{max}$ ;
**Sinon**
$\delta^{new}_{min} \leftarrow \delta$ ;
$\delta^{new} \leftarrow \dfrac{\delta + \delta_{min}}{2}$ ;
$\delta \leftarrow \delta^{new}$ ;
$\delta_{min} \leftarrow \delta^{new}_{min}$ ;
**Fin Si**
**jusqu'à ce que** $(\delta_{max} - \delta_{min} \leq \Delta)$
**Retourner** $\delta$ ;
**Fin**

Alg III.2: Calcul de $\delta$ en fonction de $N$ et $S_o$

Pour $x = x_0 = 0$, nous avons :

$$f(0) = 0$$
$$f^{(1)}(0) = \frac{\delta}{2}$$
$$f^{(2)}(0) = 0$$
$$\dots$$

D'où :

$$f(x) \cong \frac{\delta}{2} \cdot x$$

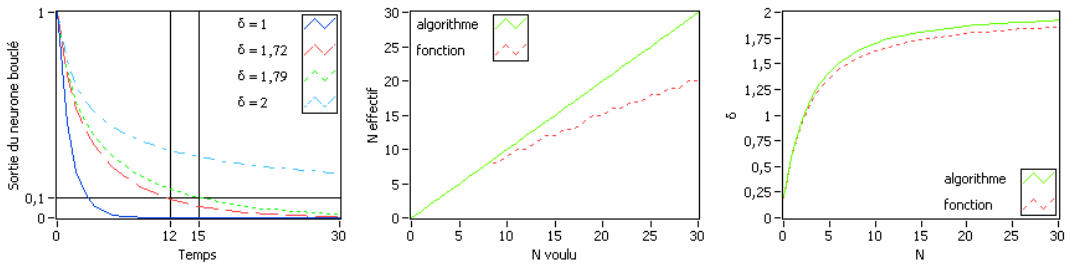
Donc l'équation peut être simplifiée en une suite géométrique de raison $\frac{\delta}{2}$. D'où :

$$s(t) = \left(\frac{\delta}{2}\right)^{t} \cdot s(0)$$

Or puisque $s(0) = 1$ et $s(N) = S_o$, nous pouvons déterminer $\delta$ par l'expression suivante :

$$\delta = 2 \cdot \sqrt[N]{S_o}$$

La figure III.6(a) montre le cas où $N = 15$ et $S_o = 0{,}1$. Dans cette situation, nous trouvons $\delta = 1{,}79$ avec notre algorithme, et $\delta = 1{,}72$ avec l'équation. Si la valeur de l'approximation est proche de celle de l'algorithme (erreur de 4 % environ), l'erreur au niveau de la mémoire effective est beaucoup plus importante (pour notre exemple $N_{reel} = 12$ soit une erreur de 20 %). En faisant varier le N désiré, nous nous sommes apperçus que plus le N désiré augmente plus l'erreur vis à vis du N obtenu augmente (figure III.6(b)), l'erreur entre les deux deltas restant faible (figure III.6(c)). Pour diminuer cette erreur, il faudrait augmenter l'ordre dans la formule de Mac Laurin, mais cela ne nous permet pas de trouver une solution du type $\delta = f(S_o, N)$. Nous utiliserons donc l'algorithme III.2 pour déterminer $\delta$.




(a) Sortie du neurone bouclé en fonction du temps et de $\delta$

(b) N effectif en fonction du N voulu et de la méthode utilisée

(c) $\delta$ en fonction de N et de la méthode utilisée

FIG. III.6 – Simulation du comportement d'oubli avec les deux méthodes

Une fois la taille de la mémoire définie, nous devons déterminer la sensibilité de notre système. Pour cela, il est nécessaire de spécifier l'influence de l'entrée sur la sortie du neurone bouclé.

Les hypothèses sont les suivantes :

- A l'instant $t < 0$, nous nous trouvons dans un état idéal de bon fonctionnement ($s(t-1) = 0$).
- A l'instant $t = 0$, une entrée d'amplitude $e_i$ est présentée.

L'équation régissant la sortie $s_i$ s'écrit alors :

$$s_i = \frac{1 - \exp(-k_i \cdot e_i)}{1 + \exp(-k_i \cdot e_i)}$$

La sensibilité du neurone $i$ est donc déterminée par le paramètre $k_i$. La figure III.7 représente la sortie $s_i$ en fonction de l'entrée $e_i$ et du paramètre $k_i$.

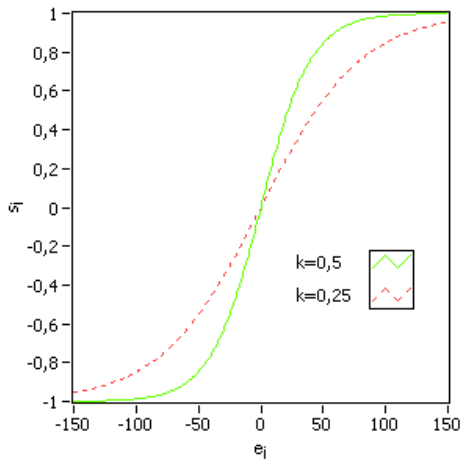
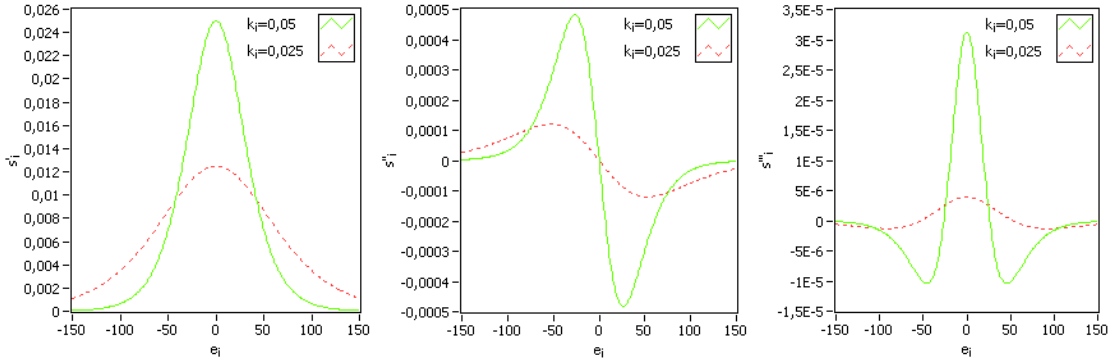



FIG. III.7 – Simulation de la sortie $s_i$ pour la sensibilité du neurone

Pour déterminer le paramètre $k_i$, nous faisons l'hypothèse que quelque soit la variation que nous pouvons avoir en entrée, nous nous trouvons dans la partie linéaire de la sigmoïde. Nous déterminons ainsi les limites de la partie linéaire ainsi que la variation maximale de la base d'entrée.

Les limites de la partie linéaire de la sigmoïde seront déterminées grâce aux dérivées successives de $s_i$ en fonction de l'entrée $e_i$ (figure III.8).

$$\frac{ds_i}{de_i} = \frac{2 \cdot k_i \cdot \exp\left(-k_i \cdot e_i\right)}{\left(1 + \exp\left(-k_i \cdot e_i\right)\right)^2}$$
$$\frac{d^2 s_i}{de_i^2} = \frac{-2 \cdot k_i^2 \cdot \exp\left(-k_i \cdot e_i\right) \cdot \left(1 - \exp\left(-k_i \cdot e_i\right)\right)}{\left(1 + \exp\left(-k_i \cdot e_i\right)\right)^3}$$
$$\frac{d^3 s_i}{de_i^3} = \frac{2 \cdot k_i^3 \cdot \exp\left(-k_i \cdot e_i\right) \cdot \left(1 - 4 \cdot \exp\left(-k_i \cdot e_i\right) + \exp\left(-2 \cdot k_i \cdot e_i\right)\right)}{\left(1 + \exp\left(-k_i \cdot e_i\right)\right)^4}$$
$$\ldots$$




(a) Dérivée d'ordre 1 (b) Dérivée d'ordre 2 (c) Dérivée d'ordre 3

FIG. III.8 – Dérivées successives de la sortie $s_i$

La dérivée première (figure III.8(a)) nous permet de déterminer le point d'inflexion de $s_i$. L'étude de la dérivée seconde (figure III.8(b)) nous permet d'obtenir les points d'inflexion de la dérivée première, c'est-à-dire les limites de la zone linéaire de la sigmoïde. Sachant que les points qui nous intéressent sont les extrêma de la dérivée seconde, ils peuvent être déterminés grâce à la dérivée troisième (figure III.8(c)) en déterminant les points l'annulant.

$$\frac{d^3 s_i}{de_i^3} = 0 \Leftrightarrow \frac{2 \cdot k_i^3 \cdot \exp\left(-k_i \cdot e_i^{lim}\right) \cdot \left(1 - 4 \cdot \exp\left(-k_i \cdot e_i^{lim}\right) + \exp\left(-2 \cdot k_i \cdot e_i^{lim}\right)\right)}{\left(1 + \exp\left(-k_i \cdot e_i^{lim}\right)\right)^4} = 0$$

Or $\exp\left(-k_i \cdot e_i^{lim}\right) > 0$, d'où :

$$1 - 4 \cdot \exp\left(-k_i \cdot e_i^{lim}\right) + \exp\left(-2 \cdot k_i \cdot e_i^{lim}\right) = 0$$

En posant $X = \exp\left(-k_i \cdot e_i^{lim}\right)$, on obtient :

$$1 - 4 \cdot X + X^2 = 0$$

D'où :

$$X = 2 \pm \sqrt{3}$$

Donc, les limites valent :

$$e_i^{lim} = \frac{\ln\left(2 \pm \sqrt{3}\right)}{k_i}$$

Or :

$$\ln\left(2 + \sqrt{3}\right) + \ln\left(2 - \sqrt{3}\right) = 0$$

Donc les limites sont symétriques et valent :

$$e_i^{lim} = \pm\frac{\ln\left(2 + \sqrt{3}\right)}{k_i}$$

Sachant que ces limites sont symétriques, la variation maximale peut aisément être déterminée en parcourant la base d'entrée normalisée et en déterminant le maximum absolu sur chaque entrée.

$$e_i^M = \max_{l \in F,\ v=1,\ldots N} \left|e_{iv}^l\right|$$

avec $F$, l'ensemble des séquences et $N$ le nombre de vecteurs dans une séquence.

En utilisant l'hypothèse que nous avons posée précédemment, $e_i^{lim}$ vaut $e_i^M$ donc le paramètre $k_i$ est donné par la relation suivante :

$$k_i = \frac{\ln\left(2 + \sqrt{3}\right)}{\max\limits_{l \in F,\ v=1,\ldots N} \left|e_{iv}^l\right|}$$

Pour terminer l'apprentissage de la mémoire dynamique, il nous reste à déterminer le paramètre $w_i$ qui est aisément déterminé par la relation suivante :

$$w_i = \frac{\delta}{k_i}$$

Une fois les paramètres de la mémoire dynamique définie, nous pouvons passer à la mémoire statique. Pour cela nous créons une nouvelle base d'apprentissage qui est déterminée à partir de la base d'apprentissage statique. Chaque séquence est présentée à la couche d'entrée, créant ainsi un ensemble de *vecteurs* que nous qualifions de statiques.

### III.3.2 Mémoire statique et couche de décision

La mémoire statique est constituée de neurones à fonction d'activation gaussienne. Cette fonction d'activation est définie par :

$$R(x) = \exp\left(\frac{-\|x-r\|^2}{2\cdot\sigma^2}\right)$$

avec $r$ le centre de la gaussienne, et $\sigma$ son rayon d'influence.

En ce qui concerne la couche de décision, la sortie étant la classe, nous avons :

$$s_c(x) = \frac{\sum_{j=1}^{m} A_j^c \cdot R_j(x)}{\sum_{j=1}^{m} A_j^c}$$

avec $m$ le nombre de gaussiennes et $A_j^c$ le poids de la gaussienne $j$ pour la classe $c$.

De nombreux algorithmes sont utilisés pour entraîner ce type de réseau, cependant, ils nécessitent une architecture fixée, pour laquelle le nombre de neurones dans la couche cachée doit être déterminé avant le début de l'apprentissage. L'algorithme d'apprentissage RCE, introduit par Reilly, Cooper et Elbaum (Reilly *et al.*, 1982), et son extension probabiliste, l'algorithme P-RCE, utilise l'avantage d'une structure évolutive qui permet l'ajout d'un neurone dans la couche cachée uniquement lorsque c'est nécessaire. De part sa nature, cet algorithme permet d'atteindre une stabilité plus rapidement que dans l'algorithme de rétropropagation du gradient. Malheureusement les réseaux P-RCE ne peuvent adapter le rayon d'influence de chaque prototype individuellement. Ils utilisent une seule valeur globale pour ce paramètre. Le choix de l'algorithme DDA (Berthold *et al.*, 1995) (algorithme III.3) se justifie par l'utilisation conjointe de la structure évolutive du P-RCE et de la possibilité de gérer le rayon de chaque prototype individuellement. Ce rayon est fonction des neurones les plus proches et de leurs classes.

Cet algorithme nécessite la définition de deux paramètres : $\theta^+$ et $\theta^-$ qui permettent de réduire les zones de conflits entre prototypes. Pour assurer la convergence de l'algorithme, le réseau devra vérifier les deux inégalités ci-dessous pour chaque vecteur d'apprentissage

**Répéter**
*[Initialisation des poids de sortie]*
**Pour** tout prototype $j$ de classe $b$ $p_j^b$ **faire**
$A_j^b \leftarrow 0$ ;
**Fin Pour**
*[Itération d'apprentissage]*
**Pour** tout vecteur d'apprentissage $x$ de classe $c$ **faire**
**Si** $(\exists p_j^c : R_j^c(x) \geq \theta^+)$ **Alors**
$A_j^c \leftarrow A_j^c + 1$ ;
**Sinon**
*[Création d'un nouveau prototype]*
ajouter un nouveau prototype $p_{m_c+1}^c$ avec :
$r_{m_c+1}^c \leftarrow x$ ;
$\sigma_{m_c+1}^c \leftarrow \max_{b \neq c \wedge 1 \leqslant d \leqslant m_b} \left\{\sigma : R_{m_c+1}^c\left(r_d^b\right) < \theta^-\right\}$ ;
*[A noter pour un premier neurone, le rayon aura la valeur qui permet de couvrir le maximum l'espace admissible]*
$A_{m_c+1}^c \leftarrow 1$ ;
$m_c \leftarrow m_c + 1$ ;
**Fin Si**
*[Ajustement des zones de conflits]*
**Pour** tout $b \neq c$, $1 \leqslant d \leqslant m_b$ **faire**
$\sigma_d^b \leftarrow \max\left\{\sigma : R_d^b(x) < \theta^-\right\}$ ;
**Fin Pour**
**Fin Pour**
**jusqu'à ce que** (plus de modifications du réseau (ajout de prototype et/ou modification des rayons d'influences))

Alg III.3: Algorithme DDA

$x$ de classe $c$.

$$\exists j : R_j^c(x) \geqslant \theta^+$$
$$\forall b \neq c, 1 \leqslant d \leqslant m_k : R_d^b(x) < \theta^-$$

Ainsi, sur la figure III.9, nous pouvons observer que l'algorithme ne créera pas un nouveau prototype pour le vecteur $x$ ($R_0^B(x) > \theta^+$). Le seuil $\theta^-$ permet de réduire les zones de conflits par les relations suivantes : $R_0^B\left(p_0^A\right) < \theta^-$, $R_0^A(x) < \theta^-$, $R_0^A\left(p_0^B\right) < \theta^-$

Pourtant l'utilisation de l'algorithme DDA est particulièrement efficace lorsque les classes sont distinctes, or nous pouvons trouver des cas où cette condition n'est pas vérifiée.

Prenons l'exemple de l'apprentissage d'une dégradation. La figure III.10 montre deux

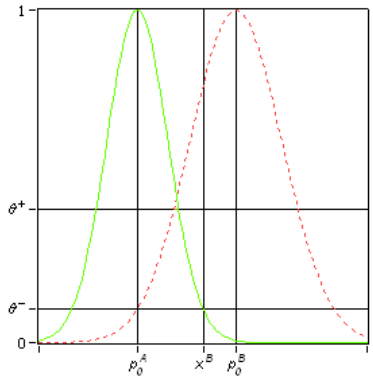
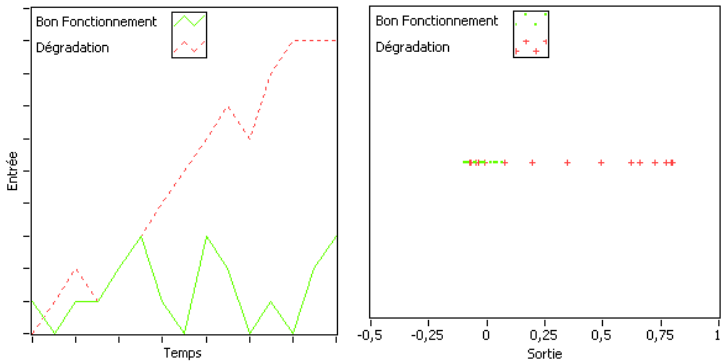



FIG. III.9 – Ajustement des rayons d'influence avec deux seuils $\theta^+$ et $\theta^-$

séquences, une de bon fonctionnement et une représentant une dégradation, avant le neurone bouclé (figure III.10(a)) et après le neurone bouclé (figure III.10(b)). Nous pouvons voir que certains points statiques sont très proches, voire identiques, les uns des autres mais de classes différentes. Cela s'explique par le fait que les deux séquences débutent par un mode de bon fonctionnement.




(a) Entrée du neurone bouclée en fonction du temps

(b) Sortie du neurone bouclé

FIG. III.10 – Exemple d'une séquence de dégradation

L'utilisation de l'algorithme DDA sur cette base de données donnera lieu à la création de nombreux neurones gaussiens ayant un rayon d'influence faible voire nul. Dans le cas extrême du rayon d'influence nul (points identiques mais classes différentes), l'algorithme ne se termine jamais.

Nous proposons de modifier l'algorithme DDA sur trois points importants :

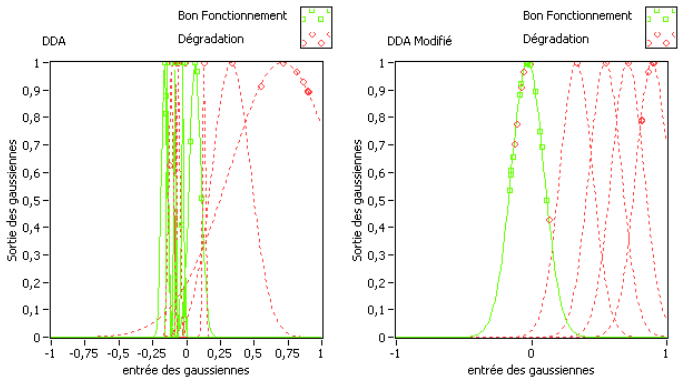
La première modification se situe au niveau de la vérification d'un prototype existant reconnaissant le vecteur d'apprentissage. Dans l'algorithme original, le prototype devait avoir la même classe que le vecteur d'apprentissage. Cette vérification ne se fait plus dans le nouvel algorithme.

La seconde consiste à ajouter un rayon initial, ce rayon est important car trop grand il risque de couvrir deux classes distinctes. Comment déterminer le rayon maximal ? Une réponse pourrait être dans l'étude de la base d'entrée. En effet, grâce à la normalisation de la base d'entrée, nous savons que tous les points correspondant à du bon fonctionnement se trouvent centrés en 0 dans un rayon de 0,1. Donc, on peut déterminer le rayon d'influence maximal par la relation suivante :

$$\sigma_{\max} = \sqrt{\frac{-(0{,}1)^2}{2 \cdot \ln(\theta^+)}}$$

Enfin, l'ajustement des zones de conflits ne s'effectuent que lors de la création d'un nouveau prototype.

L'algorithme obtenu (algorithme III.4) est comparé à l'algorithme DDA sur l'exemple de la figure III.10. Les résultats sont présentés sur la figure III.11.




(a) Résultat de l'algorithme DDA

(b) Résultat de l'algorithme DDA modifié

FIG. III.11 – Apprentissage d'une séquence de dégradation à l'aide des algorithmes DDA et DDA modifié

L'utilisation de l'algorithme DDA crée 14 prototypes (7 pour le bon fonctionnement et 7 pour la dégradation), tandis que l'algorithme DDA modifié crée 5 prototypes seulement (1 pour le bon fonctionnement et 4 pour la dégradation). Donc, l'utilisation de cet

**Répéter**
  *[Initialisation des poids de sortie]*
  **Pour** tout prototype $i$ de classe $k$ $p_i^k$ **faire**
    $A_i^k \leftarrow 0$ ;
  **Fin Pour**
  *[Itération d'apprentissage]*
  **Pour** tout vecteur d'apprentissage $x$ de classe $c$ **faire**
    **Si** $(\exists p_i^k : R_i^k(x) \geq \theta^+)$ **Alors**
      $A_i^k \leftarrow A_i^k + 1$ ;
    **Sinon**
      *[Création d'un nouveau prototype]*
      ajouter un nouveau prototype $p_{m_c+1}^c$ avec :
      $r_{m_c+1}^c \leftarrow x$ ;
      *[A noter pour un premier neurone, le rayon aura une valeur initiale : $\sigma_{\max}$]*
      $\sigma_{m_c+1}^c \leftarrow \min\left\{\sigma_{\max}, \max_{k\neq c \wedge 1\leqslant j\leqslant m_k}\left\{\sigma : R_{m_c+1}^c\left(r_j^k\right) \leqslant \theta^-\right\}\right\}$ ;
      $A_{m_c+1}^c \leftarrow 1$ ;
      $m_c \leftarrow m_c+1$ ; *[Ajustement des zones de conflits]*
      **Pour** tout $k \neq c$, $1 \leqslant j \leqslant m_k$ **faire**
        $\sigma_j^k \leftarrow \min\left\{\sigma_{\max}, \max\left\{\sigma : R_j^k(x) < \theta^-\right\}\right\}$ ;
      **Fin Pour**
    **Fin Si**
  **Fin Pour**
**jusqu'à ce que** (plus de modifications du réseau (ajout de prototype et/ou modification des rayons d'influences))

Alg III.4: Algorithme DDA modifié

algorithme réduit de façon satisfaisante le nombre de prototypes et évite les prototypes de très petites tailles.

Le principal problème de ce nouvel algorithme est l'importance dans l'ordre d'apparition des points d'apprentissage. Ce défaut est dû à la première modification de l'algorithme DDA : la suppression de la vérification des classes. En supprimant cette capacité les premières classes apprises seront « favorisées » par rapport aux suivantes. Il est donc nécessaire de « classer » ces points d'apprentissage. Pour cela nous utilisons une technique qui déterminera les centres des zones de points (les classes) : la technique Fuzzy Min-Max (Simpson, 1992; Simpson, 1993; Chang *et al.*, 2001; Zemouri, 2003).

Cette technique permet de déterminer le nombre de centres et leur valeur initiale d'une manière itérative. Durant cette phase d'initialisation, des hyper-cubes à $n$ dimensions sont créés. Les limites d'un hyper-cube sont définies par les coordonnées maximales

et minimales de chaque dimension des points appartenant à cet hyper-cube. Le degré d'appartenance d'un point à chaque hyper-cube est déterminé par la fonction d'appartenance ci-dessous :

$$H_j\left(x, \gamma_j, \beta_j\right) = \frac{1}{n} \cdot \sum_{i=1}^{n} \left[1 - f\left(x_i - \beta_{ji}\right) - f\left(\gamma_{ji} - x_i\right)\right]$$

avec : $H_j$ le degré d'appartenance d'un point $x$ à l'hyper-cube $j$. Ce degré d'appartenance est compris dans l'intervalle $[0, 1]$ ; $x_i$ la $i^{\text{ème}}$ dimension du vecteur d'entrée $x$ ; $\beta_{ji}$ et $\gamma_{ji}$ la valeur de la $i^{\text{ème}}$ dimension des points maximums et minimums respectivement du $j^{\text{ème}}$ hyper-cube ; $f$ étant la fonction floue définie par :

$$f\left(x\right) = \begin{cases} 1, & \text{si } x > \eta \\ x/\eta, & \text{si } 0 \leqslant x \leqslant \eta \\ 0, & \text{si } x < 0 \end{cases}$$

avec $\eta$ la sensibilité de l'hyper-cube.

La valeur de $\eta$ détermine la pente de la décroissance du degré d'appartenance $H_j$ d'un point en fonction de son éloignement par rapport à l'hyper-cube $j$ (figure III.12). Les auteurs de cet algorithme ne donnent aucune méthode formelle pour initialiser ce paramètre de sensibilité du degré d'appartenance. Par défaut, nous utiliserons $\eta = 0{,}25$, ce paramètre n'a que peu d'importance par rapport au paramètre de gestion de la création d'un hyper-cube que nous verrons dans l'algorithme III.5.

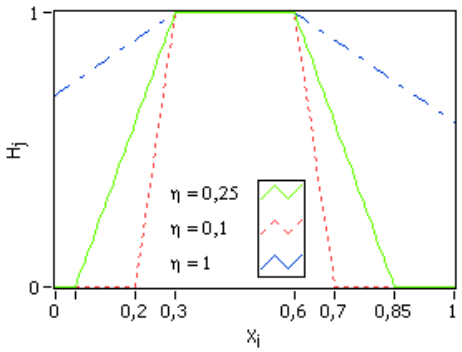



Fig. III.12 – Technique Fuzzy Min-Max - Sensibilité de l'hyper-cube

L'algorithme Fuzzy Min-Max possède trois phases : extension de l'hyper-cube, test de recouvrement et phase de redimensionnement de l'hyper-cube. Pour la phase d'initialisation des centres, nous n'avons utilisé que la partie extension pour former les différents nuages de points. Les différentes étapes de l'algorithme sont présentées dans l'algorithme III.5.

1. Initialisation des valeurs maximales et minimales du premier hyper-cube par le premier point présenté au réseau.
2. Calcul du degré d'appartenance de chaque point d'entrée.
3. Extension de l'hyper-cube ayant la plus grande fonction d'appartenance selon la condition suivante :

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \left(\max\left(u_{ji}, x_i\right) - \min\left(v_{ji}, x_i\right)\right) \leqslant \theta$$

avec $\theta$ le paramètre de l'algorithme contrôlant la création des nouveaux hyper-cubes. $\theta$ représente la norme entre deux points extrêmes de l'hyper-cube, il est directement déduit du rayon d'influence maximal par la relation suivante :

$$\theta = 2 \cdot \sigma_{\max}$$

4. Si aucun hyper-cube ne peut être élargi, un nouvel hyper-cube contenant le nouveau point est créé. Après avoir présenté au réseau l'ensemble des données d'apprentissage, un certain nombre d'hyper-cubes sont créés en fonction de la valeur du paramètre $\theta$.

ALG III.5: Etapes de l'algorithme Fuzzy Min-Max

On calcule alors les centres de chaque hyper-cube. Ces centres seront les premiers points appris par l'algorithme DDA modifié.

Suivant les problèmes rencontrés, nous pouvons nous apercevoir que l'algorithme DDA malgré les points extrêmement positifs que nous lui avons donné au début de cette partie apporte de nombreux problèmes notamment par l'utilisation de deux seuils. En effet ces seuils sont particulièrement utiles lorsque les zones de points sont distinctes, mais nous sommes dans un cas où des points de classes différentes peuvent se chevaucher. Pour retrouver ces zones de conflits qui sont des zones de doutes nécessaires, nous faisons la simplification suivante :

$$\theta^+ = \theta^- = \theta$$

En appliquant ce nouvel algorithme sur l'exemple d'une séquence de dégradation (figure III.10), nous obtenons les résultats de la figure III.13. Ce nouvel algorithme crée 4 prototypes seulement (1 prototype pour le bon fonctionnement et 3 pour la dégradation).

Nous avons testé cet algorithme sur un second benchmark de surveillance de bras de robot[1].

[1]Luis Seabra Lopes and Luis M. Camarinha-Matos, Universidade Nova de Lisboa, Monte da Caparica, Portugal, 23 avril 1999 (Benchmark disponible sur le site de l'Université de Californie à Irvine :

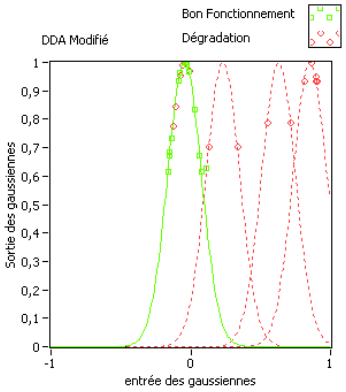



FIG. III.13 – Apprentissage d'une séquence de dégradation avec l'algorithme DDA modifié

Cette base de données contient des mesures de forces et de couples sur un robot après détection de l'erreur. Chaque erreur est caractérisée par 15 relevés sur les capteurs de force et de couple récupérés à intervalles réguliers débutés juste après la détection de l'erreur.

Les auteurs ont proposé 5 bases de données, chacune définissant un problème d'apprentissage différent :

LP1 : Problème dans l'approche du bras pour une prise de pièce ;

LP2 : Problème dans le transfert d'une pièce ;

LP3 : Position de la pièce après un problème durant le transfert ;

LP4 : Problème dans l'approche du bras pour un dépôt de pièce ;

LP5 : Problème dans le mouvement du bras avec la pièce.

Nous nous intéressons au problème LP2, qui concerne la collision du bras de robot lors du transfert de la pièce. Cette base de données propose 5 classes de sortie avec les répartitions suivantes :

- Fonctionnement normal 42,6 %
- Collision avant 12,8 %
- Collision arrière 14,9 %
- Collision à droite 10,6 %
- Collision à gauche 19,1 %

De plus, parmi les informations fournies par les auteurs, nous nous limitons aux informations capteurs de force pour simplifier l'étude des différents réseaux.

http://kdd.ics.uci.edu/databases/robotfailure/robotfailure.html )

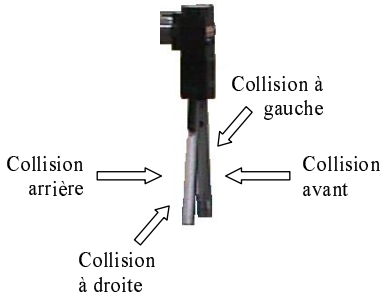



FIG. III.14 – Différents types de collision de la pince

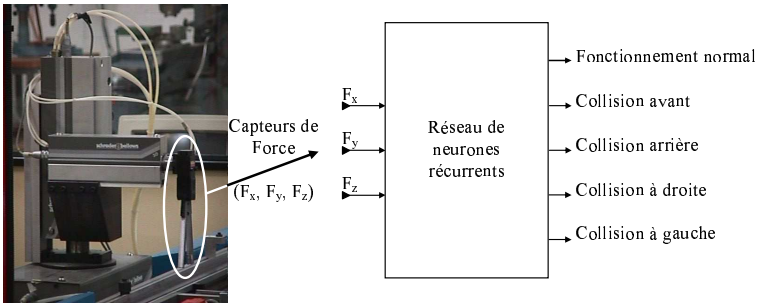



FIG. III.15 – Application de surveillance d'un bras de robot (Manipulateur pneumatique Schrader)

Les figures III.14 et III.15 présentent de façon schématique l'application de surveillance d'un bras de robot et les quatre types de défaillances pouvant se produire (le relevé des valeurs s'effectue toutes les 21 ms).

Nous effectuons l'apprentissage des 47 séquences avec les deux algorithmes. Nous trouvons à l'aide de l'algorithme DDA 239 prototypes sachant que l'algorithme a été arrêté pour cause de conflits entre prototypes. Le conflit crée des prototypes inutiles que nous avons enlevés. Les 239 prototypes sont obtenus dès la $4^{ème}$ itération. L'algorithme DDA modifié donne 28 prototypes.

### III.3.3 Synthèse sur l'outil de détection

Suite à une étude présentée dans (Palluat *et al.*, 2005*b*), nous utilisons pour la phase de détection le réseau RRFR. Son fonctionnement est dynamique et ses entrées sont reliées aux différents capteurs du système à surveiller. En sortie, nous trouvons les modes opératoires de ce système. Son apprentissage s'effectue en deux étapes. Une phase d'apprentissage de la partie dynamique grâce à sa première couche constituée de neurones à fonction d'activation sigmoïdale et retour local de la sortie. A partir du travail présenté

dans (Zemouri, 2003), nous avons développé des algorithmes permettant d'effectuer l'apprentissage de cette couche. L'utilisateur n'a qu'un seul paramètre à indiquer : le seuil d'oubli qui représente la valeur entre 0 et 1 pour laquelle la donnée est considérée comme négligeable. Un second paramètre est nécessaire pour faire fonctionner les algorithmes, mais il est directement déduit de la base d'apprentissage et qui correspond à la durée à mémoriser. La seconde phase d'apprentissage correspond à la partie statique des données constituée de la couche cachée (neurones gaussiens) et de la couche de sortie (neurones linéaires). A partir des travaux de (Berthold *et al.*, 1995) pour l'algorithme DDA et (Simpson, 1992; Simpson, 1993; Chang *et al.*, 2001; Zemouri, 2003) pour l'algorithme Fuzzy Min-Max, nous avons développé les algorithmes d'apprentissage de ces deux couches. La donnée nécessaire à l'apprentissage est un seuil d'apprentissage qui fixe le degré à partir duquel l'information est pertinente.

## III.4 Phase de Diagnostic

Le diagnostic doit permettre à partir de l'observation de symptômes de remonter jusqu'aux causes expliquant ces symptômes. On considère donc qu'il existe une relation causale d'implication entre les causes et les effets observés : causes $\Rightarrow$ effets (symptômes) (Mellouli *et al.*, 2000*a*). Cependant, la logique ne permet pas de fournir une information sur l'impliquant à partir de l'impliqué. L'idée de base développée ici est donc d'émettre une hypothèse explicative sur l'impliquant au regard des observations relatives à l'impliqué (Mellouli *et al.*, 2000*a*; Mellouli *et al.*, 2000*b*). Appliquée au diagnostic, cela revient à émettre une hypothèse sur les causes en fonction de l'observation de symptômes.

La réalisation d'un diagnostic est de plus rendue délicate dans la mesure où les informations disponibles sont le plus souvent incomplètes, imprécises et incertaines. Le système de diagnostic doit donc impérativement prendre en compte ces incertitudes de manière à rendre pertinentes les informations qu'il fournit.

Compte tenu des ces exigences pour le diagnostic, nous orientons notre réflexion vers les méthodes à bases de modèles explicatifs. Celles-ci sont effectivement les mieux adaptées pour la modélisation des relations de cause à effet indispensables au diagnostic. Un modèle de type *modèle de panne* semble donc le plus adapté pour un système de diagnostic. Le chapitre précédent nous a permis de mettre en avant les capacités d'approximation de modélisation des réseaux neuro-flous mais que ceux-ci étaient quasiment non exploités dans les méthodes à base de modèle explicatif. Nous choisirons cet outil pour la construction d'un modèle de pannes.

Étant donné les difficultés rencontrées pour l'acquisition des modèles, les expertises déjà disponibles en entreprise constituent des sources précieuses d'informations, le modèle de panne réalisé avec les réseaux neuro-flous s'appuie sur les outils courants de diagnostic que sont les arbres de défaillances (AdD) et les AMDECs.

L'incertain est pris en compte par la partie floue ce qui permet de donner des degrés de crédibilité flous associés aux états de pannes.

Enfin, nous utilisons une approche abductive qui permet de « remonter aux causes des pannes » à partir des observations. L'algorithme de recherche des causes correspond à une approche descendante dans l'arbre de défaillances.

## III.4.1 AMDEC et Arbres de défaillances

### III.4.1.1 AMDEC

C'est une méthode d'analyse préventive de la sûreté de fonctionnement qui permet une analyse systématique, composant par composant, de tous les modes de défaillance possibles et qui précise leurs effets sur le système global. Elle a pour but d'analyser les conséquences des défaillances et d'identifier les pannes dont les répercussions sur la sécurité sont importantes. L'analyse de criticité permet de classer les risques afin de s'attacher à réduire, en priorité les plus importants qui sont jugés inacceptables.

L'AMDEC présente l'avantage de pouvoir être mis en œuvre tout au long du cycle de vie d'un système. Cependant, elle est principalement utilisée en tant que technique d'analyse préventive pour détecter les défaillances potentielles, évaluer les risques et susciter des actions de prévention. (Pross, 2001)

Trois types d'AMDEC ont été développés :
- L'*AMDEC Produit* a pour but d'assurer la fiabilité d'un produit en améliorant la conception de celui-ci.
- L'*AMDEC Processus* a pour but d'assurer la qualité d'un produit en améliorant les opérations de production de celui-ci.
- L'*AMDEC Machine* a pour but d'assurer la disponibilité et la sécurité d'un moyen de production en améliorant la conception, l'exploitation et/ou la maintenance de celui-ci.

L'AMDEC peut se présenter sous forme de tableaux (tableau III.1).

| Fonction | Élément | Défaillance | Causes | Effets | Détection | Criticité |
|---|---|---|---|---|---|---|
| | | | | | | |

Tab. III.1 – Présentation de l'AMDEC sous forme d'un tableau

En fonction du système et des événements à considérer, d'autres champs peuvent être introduits dans l'AMDEC comme par exemple la fréquence ou la gravité.

L'AMDEC constitue donc un outil comprenant une analyse précise et relativement exhaustive[2] des relations de cause à effet entre les modes de défaillance des composants

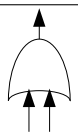
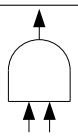
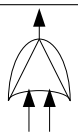
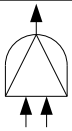
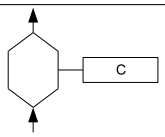
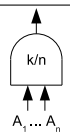
[2]On peut remarquer que c'est justement cette analyse exhaustive qui constitue le principal inconvénient de cette méthode, qui la rend lourde et fastidieuse.

du systèmes et leurs effets observables, avec en plus une évaluation de ces relations.

### III.4.1.2 Les arbres de défaillances

La méthode de l'arbre de défaillances (AdD) est très largement utilisée dans le domaine de la sûreté de fonctionnement. Elle offre un cadre privilégié à l'analyse déductive qui consiste à rechercher les diverses combinaisons possibles d'événements conduisant à la réalisation d'un événement indésirable, et permet de représenter simplement ces combinaisons sous forme graphique au moyen d'une structure arborescente de portes logiques.

Les différentes portes logiques utilisables, leurs symboles graphiques ainsi que leurs significations sont donnés dans la table III.2, (Limnios, 1991).

| Symbole graphique | Nom | Signification |
|---|---|---|
|  | OU | La sortie est générée si et seulement si au moins une des entrées existe |
|  | ET | La sortie est générée si et seulement si toutes les entrées existent |
|  | OU Exclusif | La sortie est générée si une entrée et une seule existe |
|  | ET Prioritaire ou Séquentiel | La sortie est générée si et seulement si toutes les entrées existent avec un ordre d'apparition donné |
|  | SI | La sortie est générée si l'entrée existe et si la condition $C$ est vérifiée |
|  | $k$-sur-$n$ Combinaison | La sortie est générée si $k$ parmi les $n$ entrées existent |

Tab. III.2 – Symboles graphiques des portes logiques utilisées dans les AdD

Dans la pratique, on retrouve essentiellement les portes « ET », les portes « OU » et les portes « NON ». Il existe différents types d'Arbres de Défaillance selon les types et la

complexité des systèmes, nous les citerons à titre d'information : les arbres de défaillances cohérents, non-cohérents et à délai qui s'appliquent à des systèmes à variables binaires, ainsi que les arbres de défaillances avec restriction et multiperformants qui s'appliquent quant à eux aux systèmes multiperformants c'est-à-dire pour lesquels les variables ne sont plus binaires mais discrètes ou continues.

Globalement, la construction d'un arbre de défaillances s'organise en trois phases que sont : l'analyse préliminaire, les spécifications et la construction proprement dite. L'analyse préliminaire a pour but de fournir la bonne décomposition du système, d'identifier les différents modes de défaillance des composants. La spécification comprend une étape primordiale qui est la définition de l'événement indésirable qui doit être définie sans ambiguïté et de façon cohérente avec les autres spécifications (modes de fonctionnement du système, conditions initiales, conditions aux limites). La construction de l'arbre se fait en partant de l'événement indésirable. Cet événement-sommet est ensuite décomposé en ses événements-causes immédiats, eux-mêmes décomposés en leurs événements-causes respectifs et ainsi de suite jusqu'à ce que tous les événements-causes non décomposés soient des modes de défaillance des composants du système ou de son environnement. Un exemple d'arbre de défaillances est donné par la figure III.16.

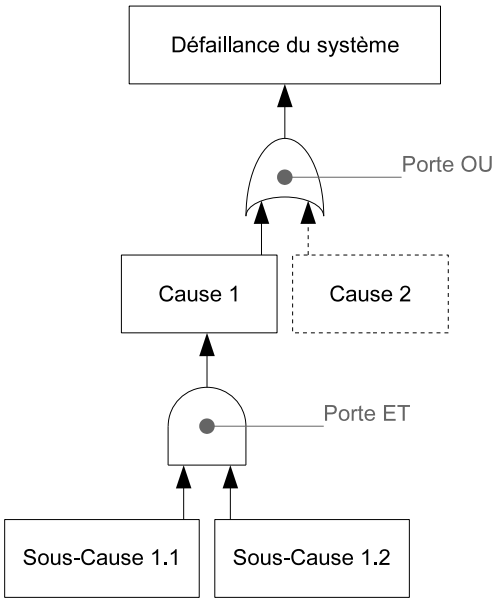



Fig. III.16 – Exemple d'arbre de défaillances

Le principale avantage de cette méthode réside donc dans sa représentation sous forme d'arbre, qui permet d'une part une meilleure lisibilité de la propagation des causes d'une défaillance et d'autre part d'effectuer plus facilement des traitements et calculs.

## III.4.2 Principe du système de diagnostic neuro-flou

Comme nous l'avons vu au chapitre précédent, l'introduction des probabilités au niveau des graphes causaux revient à travailler avec les réseaux bayésiens. Ces derniers sont couramment utilisés pour l'aide à la décision. Le raisonnement est alors basé sur la mise à jour des probabilités des différents nœuds grâce aux différentes probabilités conditionnelles à partir de l'observation de certains nœuds ; les influences entre les nœuds pouvant être bilatérales. Si le réseau correspond à un modèle de panne d'un système, où les influences entre les nœuds correspondent à des relations de cause à effet, on peut obtenir des probabilités relatives à certains états de pannes. Cependant, l'inconvénient majeur de ces réseaux Bayésiens tient à la complexité NP-Hard des calculs et à l'obtention des différentes probabilités conditionnelles.

Dans (Looney *et al.*, 2002), Looney introduit une approche par réseaux bayésiens pour l'aide à la décision. Une suite en a été donnée dans (Looney *et al.*, 2003) pour un développement avec les réseaux de Petri. Les informations relatives à chaque nœud correspondent alors à un marquage flou du réseau donnant un degré de crédibilité pour la place considérée. Comme pour les réseaux bayésiens, il définit une mise à jour des degrés à partir de l'observation de certaines places avec une propagation bilatérale.

Nous avons donc adapté cette approche au problème de diagnostic en considérant des relations de cause à effet en modélisant un arbre de défaillances avec un réseau neuro-flou.

### III.4.2.1 Réseau neuro-flou et diagnostic

Le réseau neuro-flou proposé permet de propager un degré de crédibilité flou à travers des neurones. Ces degrés sont établis à travers des observations et seront propagés jusqu'aux causes. La propagation du degré reflète un raisonnement abductif qui permet de remonter aux causes expliquant l'observation. Les différentes causes possibles de l'événement observé seront donc affectées d'un degré de crédibilité flou.

Considérons une relation de causalité tel qu'un événement $A$ implique un événement $B$, $A \Rightarrow B$. On suppose une observation de $B$ avec un degré de crédibilité $f_B$. Cette relation sera modélisée par le réseau neuro-flou de la figure III.17.

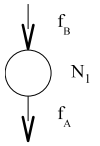



Fig. III.17 – Exemple de la modélisation d'une relation de causalité

Le neurone $N_1$ permet de déterminer le degré de crédibilité $f_A$ en fonction du degré de crédibilité $f_B$. Les neurones utilisés dans le réseau neuro-flou possède les caractéristiques suivantes :

- un « potentiel » ou « activation » égal(e) au maximum des entrées.
- une fonction de transfert qui donne la sortie du neurone en fonction de son « activation ».

Quatre fonctions de transfert seront utilisées dans le réseau neuro-flou :

- linéaire (fonction f0 - figure III.18) : $f(x,\alpha) = x, \forall\, x \in [0,1]$
- linéaire (fonction f1 - figure III.18) : $f(x,\alpha) = 1 - x, \forall\, x \in [0,1]$
- sigmoïdale (fonction f2 - figure III.18) : $f(x,\alpha) = \dfrac{1-\exp(-\alpha \cdot x)}{1+\exp(-\alpha \cdot x)}, \forall\, x \in [0,1]$
- logarithmique (fonction f3 - figure III.18) qui est en partie la fonction inverse de la fonction sigmoïdale précédente : $f(x,\alpha) = \min\left(1, \dfrac{-1}{\alpha} \cdot \ln\left(\dfrac{1-x}{1+x}\right)\right), \forall\, x \in [0,1]$

La fonction de transfert sigmoïdale est une adaptation de la fonction de transfert de Tsukamoto utilisée dans (Looney *et al.*, 2002).

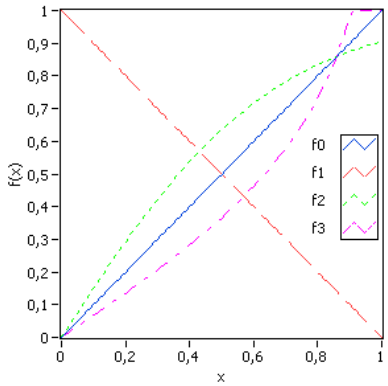



FIG. III.18 – Fonctions de transfert du réseau neuro-flou

### III.4.2.2 Modélisation de l'arbre de défaillances

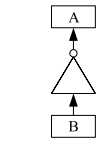
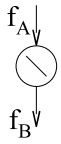
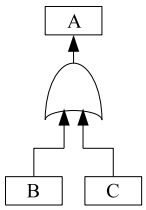
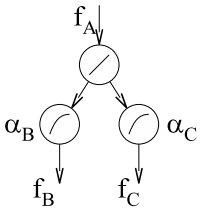
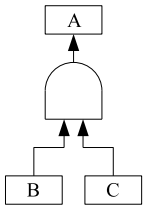
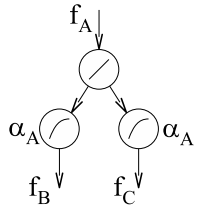
L'arbre de défaillances permet de représenter la combinaison logique de causes conduisant à l'événement indésirable. Nous avons créé les règles permettant la transformation des trois principales portes :

- NON
- ET
- OU

A partir de ces portes, il est possible de construire les autres portes. Par exemple, la porte « OU EXCLUSIF » est une combinaison de ces trois portes :

$$A = B \oplus C = \overline{B} \cdot C + B \cdot \overline{C}$$

Le tableau III.3 indique les trois transformations associées.

| Symbole graphique | Réseau FNN correspondant | Formule mathématique associée |
|---|---|---|
|   |   | $f_B = 1 - f_A$ |
|   |   | $f_B = \frac{1 - \exp(-\alpha_B \cdot f_A)}{1 + \exp(-\alpha_B \cdot f_A)}$ $f_C = \frac{1 - \exp(-\alpha_C \cdot f_A)}{1 + \exp(-\alpha_C \cdot f_A)}$ |
|   |   | $f_B = f_C = \frac{1 - \exp(-\alpha_A \cdot f_A)}{1 + \exp(-\alpha_A \cdot f_A)}$ |

TAB. III.3 – Transformation des portes logiques de l'AdD en réseau neuro-flou

En plus de ces adaptations, nous ajoutons pour chaque événement terminal, un neurone linéaire. Cet ajout est particulièrement utile lorsqu'un même événement conduit à deux portes ce qui permet de diffuser l'information aux deux portes. Un exemple de cet ajout est donné sur la figure III.19. Nous verrons par la suite que l'ajout du neurone linéaire est aussi utile pour la liaison avec le système de détection.

### III.4.2.3 Paramétrage des fonctions

La pertinence du diagnostic dépend des paramètres utilisés dans le réseau. Pour les déterminer, nous utilisons l'AMDEC. En effet, les événements considérés dans l'AdD sont les causes associées aux composants du système considéré. Les fréquences déduites dans l'AMDEC nous permettent de déterminer les coefficients $\alpha$ en partant de l'hypothèse suivante : « *plus une fréquence est élevée, plus les causes associées seront suspectées, et inversement.* »

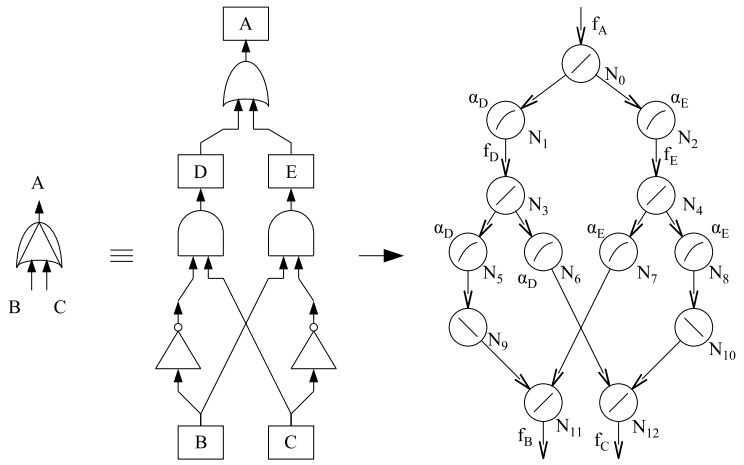



FIG. III.19 – Transformation d'une porte « OU Exclusif »

Seulement tous les $\alpha$ ne peuvent être directement déduits de l'AMDEC, seuls les $\alpha$ liés aux causes primaires peuvent être ainsi déterminés.

Dans un AMDEC, les fréquences sont déduites du MTBF par fuzzyfication. Le nombre de sous-ensembles flous est déterminant pour connaître les coefficients $\alpha$. Ainsi, en posant $s$ le nombre de sous-ensembles flous, nous pourrons obtenir $s$ $\alpha$ $(\alpha_1 \dots \alpha_s)$. Pourtant toutes les causes ne sont pas présentes dans l'AMDEC. Pour les autres causes, nous posons un nouvel $\alpha$ : $\alpha_0$ qui représente le cas où la panne n'est jamais arrivée.

Afin de déterminer les différents $\alpha_i$ liés aux causes, nous posons les conditions suivantes :

- $\alpha_0$ est déterminé tel que : $f\,(0{,}5\,;\alpha_0) = 0{,}2$
- $\alpha_s$ est déterminé tel que : $f\,(0{,}5\,;\alpha_s) = 0{,}8$
- les valeurs de $f\,(0{,}5;\alpha_i)$ sont distribuées linéairement : $f\,(0{,}5\,;\alpha_i) = a \cdot i + b$

  Connaissant $\alpha_0$ et $\alpha_s$, nous avons : $a = \dfrac{0{,}6}{s}\,; b = 0{,}2$

Ainsi, $\alpha_i$ sera déterminé par l'équation suivante :

$$f\,(0{,}5\,;\alpha_i) = \frac{1 - \exp\left(-\alpha_i \cdot 0{,}5\right)}{1 + \exp\left(-\alpha_i \cdot 0{,}5\right)} = \frac{0{,}6}{s} \cdot i + 0{,}2$$

D'où :

$$\alpha_i = 2 \cdot \ln\left(\frac{6 \cdot s + 3 \cdot i}{4 \cdot s - 3 \cdot i}\right) \text{ pour } \forall\, i \in [0, s]$$

Pour les autres $\alpha$, nous effectuons une mise à jour du voisinage des neurones de sortie. Dans (Looney *et al.*, 2002), les auteurs développent cette notion de mise à jour

du voisinage dans le cas d'un réseau bayésien. Les voisins d'un neurone $N$ sont les neurones directement ou indirectement liés à celui-ci. Les voisins se distinguent par leur niveau et par le fait qu'ils soient parents ou enfants. Un voisin de niveau 1 du neurone $N$ est un neurone directement lié au neurone $N$. Un voisin de niveau 2 est un neurone directement lié à un voisin de niveau 1. Un parent est un voisin de niveau 1 dont la connexion se fait vers le neurone $N$, tandis qu'un enfant est un voisin dont la connexion se fait depuis le neurone $N$.

Dans l'exemple de la figure III.19, le neurone $N_5$ a comme voisin de niveau 1 les neurones $N_3$ (parent) et $N_9$ (enfant), comme voisin de niveau 2 les neurones $N_1$, $N_6$ et $N_{11}$, etc...

Afin de diffuser l'information à tous les neurones, nous définissons quelques principes pour la modification du réseau :

1. chaque neurone possède trois états (« *non testé* », « *test en cours* » et « *testé* ») ;
2. chaque neurone n'ayant pas d'enfant est en état « *testé* », tous les autres sont en état « *non testé* » ;
3. pour chaque neurone n'ayant pas d'enfant, le $\alpha$ associé est déterminé en fonction de l'AMDEC grâce à l'équation déterminée précédemment : $\alpha = 2 \cdot \ln\left(\frac{6 \cdot s + 3 \cdot i}{4 \cdot s - 3 \cdot i}\right)$ avec $s$ le nombre de sous-ensembles flous et $i$ la fréquence associée à la cause ($i = 0$ si aucune fréquence) ;
4. tout voisin de niveau 1 « *non testé* » et parent d'un neurone « *testé* » passe en état « *test en cours* » ;
5. pour tout neurone dont l'état est « *test en cours* », le $\alpha$ associé est égal au maximum du $\alpha$ de chacun de ses enfants, le neurone passe ensuite en état « *testé* » ;
6. parmi les neurones « *testés* », si deux neurones ou plus ont un $\alpha$ dont le nom est identique alors la valeur finale du $\alpha$ est la valeur maximale de tous les $\alpha$ possédant le même nom. (Rappel : la transformation d'une porte ET à deux entrées donne deux neurones à fonction sigmoïdale dont les $\alpha$ portent le même nom.)

Dans l'exemple de la figure III.19, les neurones $N_{11}$ et $N_{12}$ n'ont pas d'enfant et donc sont en état « *testés* » (condition 2). Les $\alpha$ sont déterminés par l'AMDEC (condition 3). Les neurones $N_6$, $N_7$, $N_9$ et $N_{10}$ passe en état « *test en cours* » (condition 4). Les neurones $N_6$ et $N_{10}$ ont leur $\alpha$ identique à celui de $N_{12}$, tandis que $N_7$ et $N_9$ ont un $\alpha$ identique à celui de $N_{11}$. Ces neurones passent ensuite en état « *testés* » (condition 5). La condition 6 s'appliquera à l'étape suivante, lorsque $N_5$ et $N_8$ passeront en état « *testés* ». Les neurones $N_5$ et $N_6$ ont un $\alpha$ dont le nom est identique $\alpha_D$. Seul le plus grand des deux sera conservé et sera appliqué au deux neurones. Le même cas se produit avec les neurones $N_7$ et $N_8$ avec $\alpha_E$.

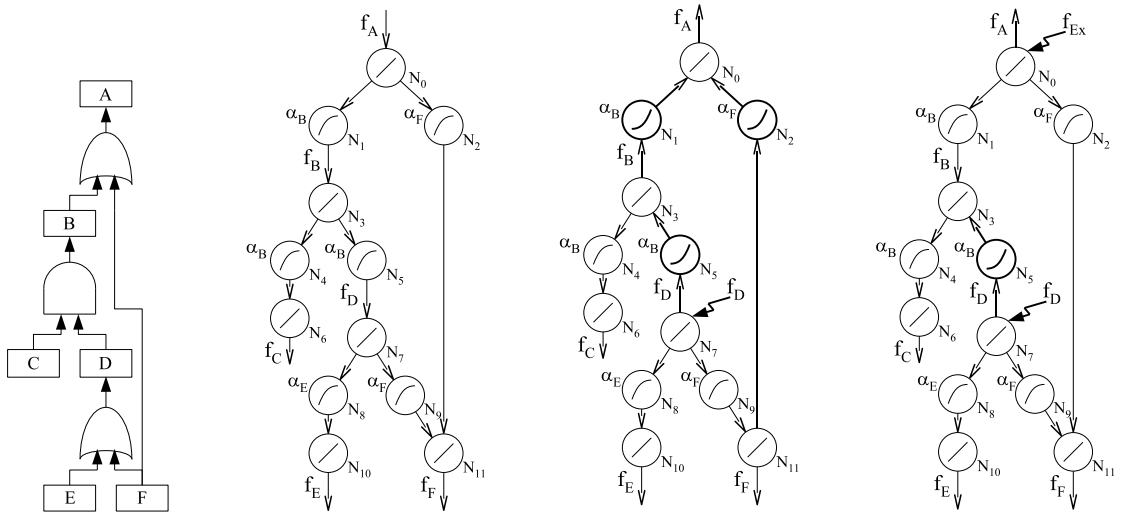
#### III.4.2.4 Liaison avec l'outil de détection

Comme nous l'avons vu dans la partie précédente, l'outil de détection, le réseau RRFR, est utilisé pour classifier les signaux d'entrées. Ces classes sont les modes de fonctionnement que nous pouvons trouver dans l'AMDEC. La liaison avec l'arbre de défaillance s'effectue grâce à l'AMDEC. En effet, les modes de fonctionnement sont liés à des causes et ces causes sont présentes dans l'arbre de défaillances. La liaison s'effectue alors avec l'événement ayant comme « enfant » toutes les causes trouvées dans l'AMDEC. Dans l'exemple III.20(a), si $C$ et $E$ sont les causes du mode $M$ alors l'événement lié à ce mode sera $B$. Chacune des sorties du réseau de neurones RRFR est ainsi reliée à un neurone du réseau neuro-flou. Cette liaison sera considérée comme l'entrée du réseau neuro-flou. Cette entrée pouvant se trouver à différents niveaux dans le réseau neuro-flou, il est nécessaire de modifier le réseau afin de faire parvenir l'information à tous les neurones du réseau et donc donner un degré de crédibilité à chaque cause. Ainsi le sens de propagation peut être inversé dans certains cas et des neurones peuvent voir leur fonction de transfert altérée.

Comme dans la partie précédente, nous effectuons une mise à jour du voisinage des neurones d'entrée. Nous définissons ainsi quelques principes pour la modification du réseau, afin de diffuser l'information à tous les neurones :

1. chaque neurone possède trois états (« *non testé* », « *test en cours* », et « *testé* ») ;
2. chaque neurone est à l'origine en état « *non testé* » ;
3. chaque neurone lié à l'outil de détection ou à un événement extérieur est un neurone à fonction de transfert linéaire et passe en état « *testé* » ;
4. tout voisin de niveau 1 « *non testé* » d'un neurone « *testé* » devient l'enfant de celui-ci (modification du sens de propagation si nécessaire) et passe en état « *test en cours* » ;
5. pour tout neurone dont l'état est « *test en cours* » : si tous ses parents sont « *testés* » le neurone passe en état « *testé* ». Sinon, le neurone passe en état « *testé* » après avoir eu sa fonction de transfert inversée (aucun changement pour les fonctions linéaires, par contre la fonction sigmoïdale devient logarithmique, et lors d'une mise à jour, une fonction logarithmique devient sigmoïdale).

Dans l'exemple de la figure III.20, l'événement $D$ est lié au système de détection par le degré $f_D$. Le neurone $N_7$ passe alors en état « *testé* » (condition 3). Les neurones $N_5$, $N_8$ et $N_9$ passent en état « *test en cours* » et le sens de propagation est inversé entre $N_7$ et $N_5$ (condition 4). $N_8$ et $N_9$ passent en état « *testés* » sans modification, tandis que $N_5$ voit sa fonction de transfert inversée car il est aussi l'enfant de $N_3$ qui est en état « *non testé* » (condition 5). La modification du réseau se poursuit jusqu'à ce que tous les neurones passent en état « *testé* ». Le résultat de ces modifications est illustré sur la figure III.20(c).




(a) AdD

(b) réseau neuro-flou sans entrée extérieure

(c) réseau neuro-flou lié à l'outil de détection

(d) réseau neuro-flou lié à l'outil de détection et avec entrée extérieure

FIG. III.20 – Exemple de transformation de l'AdD en RNF puis lien avec RRBF et ajout d'une entrée

### III.4.2.5 Informations externes

L'outil de diagnostic doit pouvoir prendre en compte des informations supplémentaires non fournies par le système de détection. Typiquement, ce sont des informations que l'opérateur souhaiterait apporter afin d'améliorer le diagnostic. Ces informations doivent donc être prises en compte lors de l'apprentissage. Elles peuvent être tirées de l'AMDEC dans le cas où aucun capteur présent sur le système ne peut déceler une cause. Vu l'architecture que nous avons établie précédemment il paraît difficile d'intégrer une nouvelle entrée dans le système sans la modifier. C'est pourquoi nous avons fait le choix d'utiliser un réseau spécifique lorsque nous souhaitons utiliser ce type d'entrée. En pratique, l'opérateur pourra choisir entre un diagnostic simple (toutes les informations sont fournies par le système de détection) et un diagnostic avancé (les informations sont à la fois fournies par le système de détection et par l'opérateur). Le réseau utilisé sera conçu exactement de la même manière que précédemment, seulement les neurones d'entrée seront plus nombreux.

Dans l'exemple de la figure III.20, le mode $D$ est toujours lié au système de détection par le degré $f_D$. Nous ajoutons le degré $f_{Ex}$ lié au mode $A$ qui peut représenter l'état général du système. Nous obtenons ainsi le réseau représenté sur la figure III.20(d).

#### III.4.2.6 Illustration de la mise en oeuvre de l'outil de diagnostic sur un exemple industriel

Devant l'inexistence de benchmark de systèmes possédant à la fois un arbre de défaillance et un AMDEC du système, nous avons utilisé le système industriel disponible à l'institut de productique de Besançon. Nous reviendrons plus en détail sur la description du matériel au chapitre IV.3.2. Pour cette application, nous avons extrait une partie de l'AdD (figure III.21) ainsi qu'une partie de l'AMDEC (table III.4).

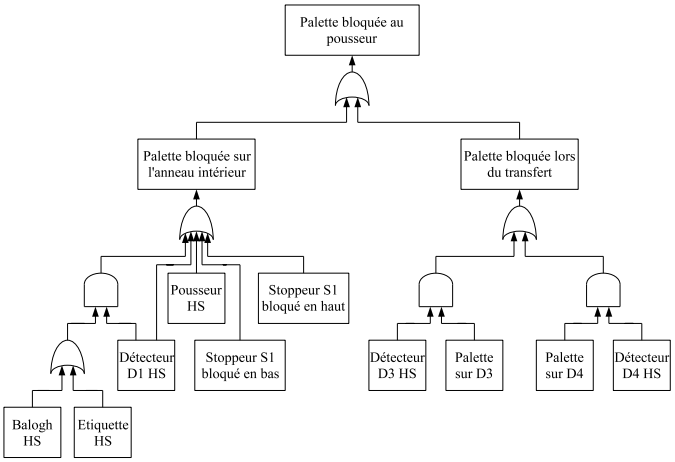



Fig. III.21 – Arbre de défaillance de la zone surveillée

| Modes de défaillances | Cause | Fréquence | Gravité |
|---|---|---|---|
| Palette bloquée sur l'anneau intérieur | Détecteur D1 est en panne | 4 | 2 |
| Palette bloquée sur l'anneau intérieur | Stoppeur S1 bloqué en position basse | 3 | 4 |
| Palette bloquée sur l'anneau intérieur | Le pousseur est en panne | 1 | 2 |

Tab. III.4 – AMDEC de la zone surveillée

A partir de ces informations nous allons appliquer nos techniques de transformation d'arbre de défaillances en un réseau neuro-flou.

Dans un premier temps, nous transformons l'AdD (figure III.21) grâce aux données de la table III.3. Nous obtenons ainsi le réseau de la figure III.22.

Dans un deuxième temps, on paramètre les fonctions en déterminant les différents $\alpha$. A partir de l'AMDEC, nous déterminons les $\alpha$ des causes primaires. Sachant que les fréquences ont été fuzzifiées en 5 fréquences, et en utilisant les règles présentées dans la section III.4.2.3, nous obtenons les $\alpha$ des tables III.5 et III.6.

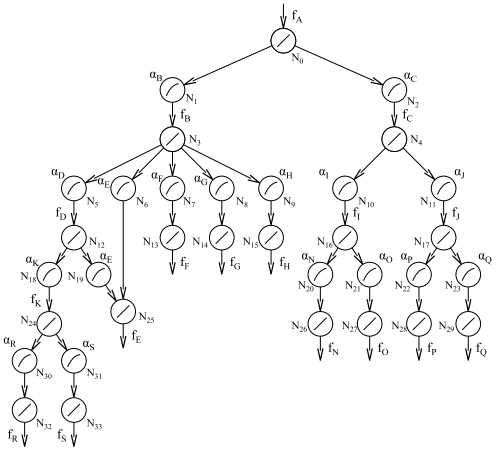


FIG. III.22 – Réseau neuro-flou de la zone surveillée

| Cause | Fréquence | $\alpha$ |
|---|---|---|
| Balogh en panne | 0 | $\alpha_R = \alpha_0 = 0{,}81$ |
| Etiquette en panne | 0 | $\alpha_S = \alpha_0 = 0{,}81$ |
| Détecteur D1 en panne | 4 | $\alpha_E = \alpha_4 = 3{,}31$ |
| Pousseur en panne | 1 | $\alpha_F = \alpha_1 = 1{,}27$ |
| Stoppeur S1 bloqué en position basse | 3 | $\alpha_G = \alpha_3 = 2{,}53$ |
| Stoppeur S1 bloqué en position haute | 0 | $\alpha_H = \alpha_0 = 0{,}81$ |
| Détecteur D3 en panne | 0 | $\alpha_N = \alpha_0 = 0{,}81$ |
| Palette sur D3 | 0 | $\alpha_O = \alpha_0 = 0{,}81$ |
| Détecteur D4 en panne | 0 | $\alpha_P = \alpha_0 = 0{,}81$ |
| Palette sur D4 | 0 | $\alpha_Q = \alpha_0 = 0{,}81$ |

TAB. III.5 – $\alpha$ des causes primaires

Après avoir déterminé les différents $\alpha$, la troisième étape consiste en la liaison avec le système de détection. Nous admettons dans cette application que le système est pleinement fonctionnel et fournit l'information liée à la cause « palette bloquée sur l'anneau intérieur » (neurone $N_3$ - figure III.22) via le degré $f_D$. En utilisant les principes que nous avons définis dans la section III.4.2.4, nous modifions la fonction de transfert du neurone $N_1$ et les neurones $N_1$ et $N_0$ deviennent « enfants » de $N_3$ et $N_1$ respectivement.

Le réseau obtenu est représenté sur la figure III.23.

Pour tester notre réseau, nous faisons varier le degré $f_D$ entre 0 et 1. Le résultat est représenté sur la figure III.24. En plus des causes principales, nous avons ajouté l'événement indésirable de l'arbre de défaillance : $f_A \rightarrow$ Palette bloquée au pousseur.

Les différentes causes sont :
- $f_E \rightarrow$ Détecteur D1 en panne ;
- $f_F \rightarrow$ Pousseur en panne ;

| Cause | $\alpha$ | |
|---|---|---|
| | $\alpha_K = \max(\alpha_R, \alpha_S)$ | $= 0{,}81$ |
| | $\alpha_D = \max(\alpha_K, \alpha_E)$ | $= 3{,}31$ |
| Palette bloquée sur l'anneau intérieur | $\alpha_B = \max(\alpha_D, \alpha_E, \alpha_F, \alpha_G, \alpha_H)$ | $= 3{,}31$ |
| | $\alpha_I = \max(\alpha_N, \alpha_O)$ | $= 0{,}81$ |
| | $\alpha_J = \max(\alpha_P, \alpha_Q)$ | $= 0{,}81$ |
| Palette bloquée sur l'anneau extérieur | $\alpha_C = \max(\alpha_I, \alpha_J)$ | $= 0{,}81$ |
| Palette bloquée au pousseur | $\alpha_A = \max(\alpha_B, \alpha_C)$ | $= 3{,}31$ |

TAB. III.6 – $\alpha$ des causes

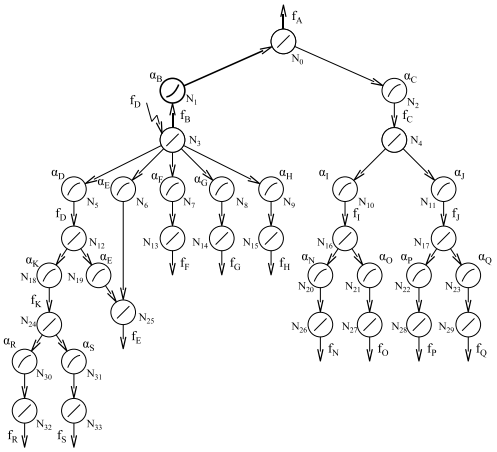


FIG. III.23 – Réseau neuro-flou de la zone surveillée avec liaison avec l'outil de détection

- $f_G \rightarrow$ Stoppeur S1 bloqué en position basse ;
- $f_H \rightarrow$ Stoppeur S1 bloqué en position haute ;
- $f_N \rightarrow$ Détecteur D3 en panne ;
- $f_O \rightarrow$ Palette sur D3 ;
- $f_P \rightarrow$ Détecteur D4 en panne ;
- $f_Q \rightarrow$ Palette sur D4 ;
- $f_R \rightarrow$ Balogh en panne ;
- $f_S \rightarrow$ Etiquette en panne ;

Nous pouvons remarquer que quel que soit la valeur du degré $f_D$, le classement des causes les plus pertinentes est toujours le même :

1. $f_E$ ;
2. $f_G$ ;
3. $f_F$ ;
4. $f_H$ ;
5. $f_R, f_S$ ;
6. $f_N, f_O, f_P, f_Q$ ;

Ce résultat est validé par l'étude de l'AMDEC et de l'arbre de défaillance : il apparaît comme industrialement viable de suspecter en priorité l'élément qui est tombé le plus souvent en panne. C'est notamment le cas pour l'exemple pratique étudié qui met en évidence le défaut effectif du capteur D1.

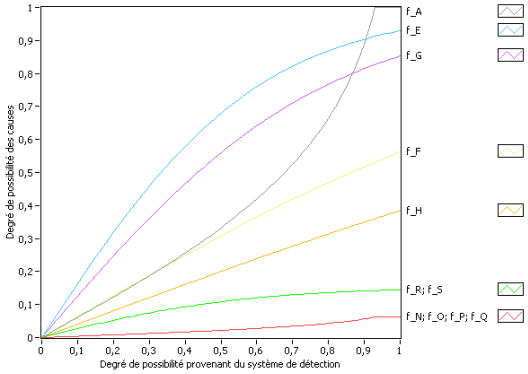



FIG. III.24 – Degré des différentes causes en fonction de $f_D$

### III.4.3 Synthèse sur l'outil de diagnostic

Dans la phase de diagnostic, nous utilisons un réseau neuro-flou. En partant de l'arbre de défaillance, des informations disponibles à la fois par le système de détection et l'opérateur, nous avons établi des règles qui permettent d'obtenir un réseau neuro-flou pour le diagnostic.

Ce réseau est conçu en quatre phases. La première consiste en la transformation de l'AdD en réseau neuro-flou grâce à des règles de transformation que nous avons établies. Ces règles au nombre de trois, permettent la transformation des trois portes principales (ET, OU et NON). Toute autre porte peut être définie à partir de ces trois portes, telle que la porte OU EXCLUSIF que nous avons transformée dans ce chapitre. La seconde phase consiste en l'apprentissage des paramètres des neurones du réseau à l'aide de l'AMDEC. A chaque cause présente dans l'AMDEC est associée une fréquence, nous avons créé des conditions permettant de définir à partir de ces fréquences les paramètres de tous les neurones du réseau. La troisième phase représente la liaison avec l'outil de détection. L'outil de détection possède en sortie des modes de fonctionnement que nous pouvons associer à certaines causes de l'AdD. A partir de cette hypothèse, nous avons présenté diverses règles permettant d'intégrer cette information dans le réseau en le modifiant. Ainsi, à partir des informations, du système de détection, chaque cause possède un degré de pertinence. De même, nous avons intégré la possibilité d'ajouter des informations externes données par l'opérateur. Cette dernière phase nécessite la création d'un nouveau réseau en utilisant les règles utilisées pour la liaison avec l'outil de détection.

## III.5 Conclusion

Nous avons présenté dans ce chapitre les outils utilisés pour effectuer une aide à la surveillance. Ces outils au nombre de deux sont associés à chacune des fonctions principales de la fonction surveillance à savoir la détection de défaillance et le diagnostic de défaut. Nous avons fait le choix d'utiliser un réseau récurrent, le RRBF, pour effectuer l'étape de détection, et un réseau neuro-flou pour l'étape de diagnostic.

Pour la phase de détection, nous avons établi qu'il était nécessaire d'effectuer l'apprentissage du réseau en deux étapes, une étape pour l'apprentissage de la mémoire dynamique et une étape pour l'apprentissage de la mémoire statique. A partir de méthodes existantes, nous les avons modifiées et optimisées pour l'application de détection dynamique.

Pour la phase de diagnostic, nous avons présenté diverses règles que nous avons établies afin de déterminer l'architecture et les caractéristiques du réseau en fonction des informations tirées de l'AMDEC et de l'arbre de défaillances.

Nous allons voir par la suite, l'implantation de ces outils dans un ordinateur industriel et leur utilisation sur une plateforme flexible de production.

# Chapitre IV

# Spécifications de l'outil surveillance intelligent

---

*L'utilisation de l'outil de surveillance intelligent basé sur un réseau récurrent à fonction de base radiale et le réseau neuro-flou peut se décrire en plusieurs cas en suivant une approche UML : Créer un nouvel outil, configurer l'outil, initialiser l'outil, émettre une alarme, demander une aide au diagnostic, mettre à jour la configuration de l'outil et mettre à jour le modèle de l'outil. A partir de ces cas, des scénarii sont établis ainsi que des diagrammes de séquences. Enfin, nous les appliquons sur un cas pratique : une plate-forme flexible de production où nous développerons un outil d'aide à la surveillance sur une zone critique de cette plate-forme.*

---

# IV.1 Introduction

Nous avons présenté dans le chapitre précédent, le système de surveillance dynamique basé sur deux outils : le réseau récurrent à fonction de base radiale, et le réseau neuro-flou. Dans le cadre du projet européen PROTEUS[1], et afin de l'intégrer dans un module d'aide d'une plate-forme globale de e-maintenance, nous avons développé l'outil global d'aide à la surveillance en utilisant la démarche UML (Unified Modelling Language) (Larman, 2002; Rumbaugh *et al.*, 1998). Plusieurs raisons ont conduit à ce choix. La première est sa normalisation par l'OMG[2] (OMG, 2003). L'historique a montré que la profusion des notations est préjudiciable aux entreprises et à leurs fournisseurs. Toute norme doit donc être considérée avec le plus grand sérieux, en particulier les normes du domaine public comme l'est l'UML. Les spécifications d'UML sont accessibles gratuitement. La deuxième raison est l'intérêt montré par les informaticiens pour ce langage de modélisation. Il est intéressant de disposer d'un ensemble de modèles communs. La troisième raison est la possibilité d'utiliser le même atelier de génie logiciel, depuis l'expression des besoins jusqu'à la génération de tout ou partie de l'application. La dernière raison, mais non la moindre, est d'utiliser les principes et concepts objet pour enrichir la démarche de conception de systèmes d'aide à la décision. On en attend des améliorations dans le sens de la richesse mais aussi d'une modularité, d'une cohérence et d'une rigueur accrues. (Morley *et al.*, 2003)

Les caractéristiques du système d'aide à la surveillance sont :

- utilisation simple ;
- interfaçage avec les outils industriels d'acquisition de données (SCADA - Supervisory Control And Data Acquisition - Télésurveillance et Acquisition de Données, Ethernet ...) ;
- utilisation en l'absence d'une base de données et de connaissances complètes, avec la possibilité de prendre en compte les nouvelles expériences (si possible en ligne) ;
- capacité d'identifier les fausses alarmes ;
- interfaçage avec les outils industriels de gestion de la maintenance (GMAO) et compréhension des résultats par un professionnel de la maintenance (paramétrage similaire aux méthodes industrielles de type AMDEC, AdD) ;
- possibilité d'interfaçage avec des interfaces Homme-Machine sur des clients légers (PDA, ordinateurs portables ...) ;
- intégration de l'outil dans un ordinateur industriel.

[1] projet européen ITEA (Information Technology for European Advancement) – PROTEUS – a generic platform for e-maintenance. Site : http ://www.proteus-iteaproject.com/

[2] Site internet de l'OMG : http ://www.omg.org/,
site internet de l'OMG lié à l'UML : http ://www.uml.org/

# IV.2 UML

UML s'articule autour de plusieurs types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Le schéma IV.1 montre comment, en partant des besoins utilisateurs formalisés par des cas d'utilisation et une maquette, et avec l'apport du modèle du domaine, on peut aboutir à des diagrammes de conception qui permettent de dériver du code assez directement. (Roques, 2002)

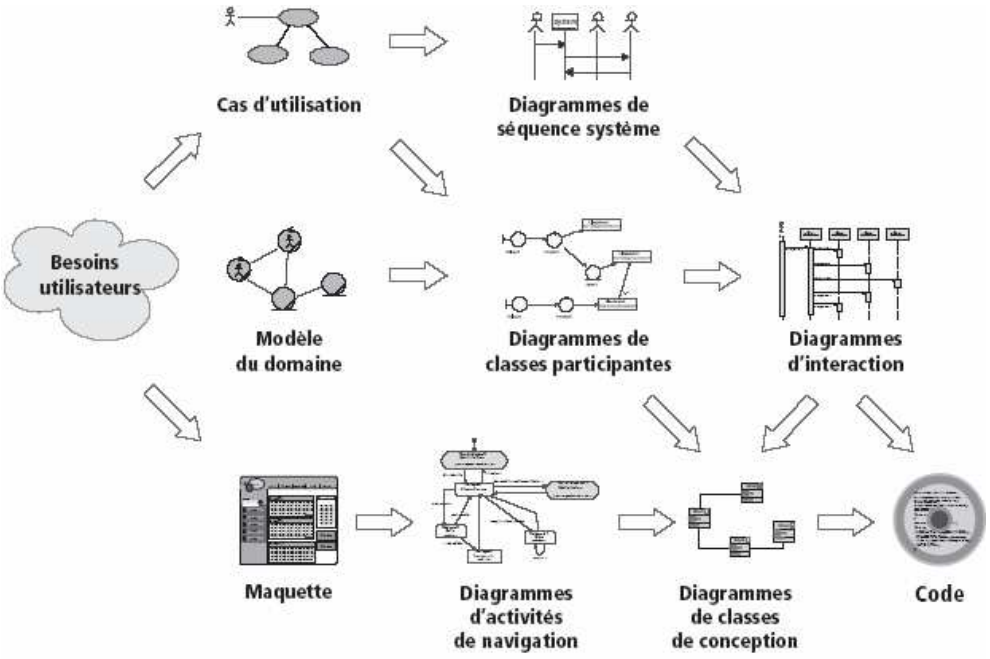



FIG. IV.1 – Schéma du processus de modélisation

Nous nous centrerons sur les diagrammes des cas d'utilisation et d'interaction.

## IV.2.1 Les cas d'utilisation

Ils permettent de définir les limites du système et les relations entre le système et son environnement. Un cas d'utilisation est une manière spécifique d'utiliser le système. Pour aboutir aux cas d'utilisation il est nécessaire de suivre une démarche de modélisation telle que celle présentée dans (Roques, 2002) :

- identifier les acteurs ;
- identifier les cas d'utilisation ;
- structurer les cas d'utilisation en paquetages ;

#### IV.2.1.1 Acteurs

Dans notre cas, les acteurs humains sont :

- Le responsable maintenance
- L'expert de l'outil
- L'opérateur de maintenance

Nous prenons aussi en compte les acteurs non humains :

- SCADA ;
- AdD ;
- GMAO ;
- AMDEC.

#### IV.2.1.2 Cas d'utilisation

Pour chaque acteur identifié précédemment, il convient de rechercher les différentes intentions « métier » selon lesquelles il utilise le système.

En ce qui concerne le Responsable Maintenance, ses cas d'utilisation principaux ont été mis en évidence par l'expression des besoins, à savoir :

- *Créer un nouvel outil*, qui permet d'envoyer à l'expert de l'outil toutes les données nécessaires à la création de l'outil (Configuration et initialisation) ;
- *Mettre à jour la configuration de l'outil*, qui permet au responsable maintenance d'ajouter un capteur à l'outil de détection ;
- *Mettre à jour le modèle de l'outil*, qui permet de mettre à jour les réglages de l'outil lorsqu'une nouvelle information de maintenance provient de la GMAO ;
- *Émettre une alarme*, qui apparaît lorsque le système d'aide à la surveillance détecte ou prédit une défaillance.

En ce qui concerne l'Opérateur de Maintenance, il ne possède qu'un seul cas d'utilisation, à savoir :

- *Demander une aide au diagnostic*, qui permet à l'opérateur de maintenance de demander l'assistance du système d'aide à la surveillance.

Enfin, pour l'acteur Expert de l'Outil, nous pouvons trouver les cas d'utilisation suivant :

- *Créer un nouvel outil*, déjà présenté dans les cas d'utilisation du responsable maintenance ;
- *Configurer l'outil*, qui permet à l'expert de l'outil de configurer le système d'aide à la surveillance à l'aide des données extraites du SCADA et de l'AdD ;

- *Initialiser l'outil*, qui permet à l'expert de l'outil d'initialiser le système d'aide à la surveillance configuré à l'aide des données extraites de l'AMDEC et de la GMAO, et de lancer l'acquisition des données provenant du SCADA ;
- *Mettre à jour la configuration de l'outil*, déjà présenté dans les cas d'utilisation du responsable maintenance.

#### IV.2.1.3 Paquetages

Pour améliorer notre système, nous organisons les cas d'utilisation et les regroupons en ensembles fonctionnels cohérents. Nous créons ainsi trois paquetages :

- le paquetage des acteurs regroupant tous les acteurs
- le paquetage « off-line » regroupant tous les cas d'utilisation qui sont actifs lorsque le système d'aide à la surveillance ne fonctionne pas : Créer un nouveau système, Configurer le système, Initialiser le système.
- le paquetage « on-line » regroupant tous les cas d'utilisation qui sont actifs lorsque le système est en fonctionnement : Mettre à jour la configuration du système, Mettre à jour le modèle du système, Émettre une alerte, et Demander une aide au diagnostic.

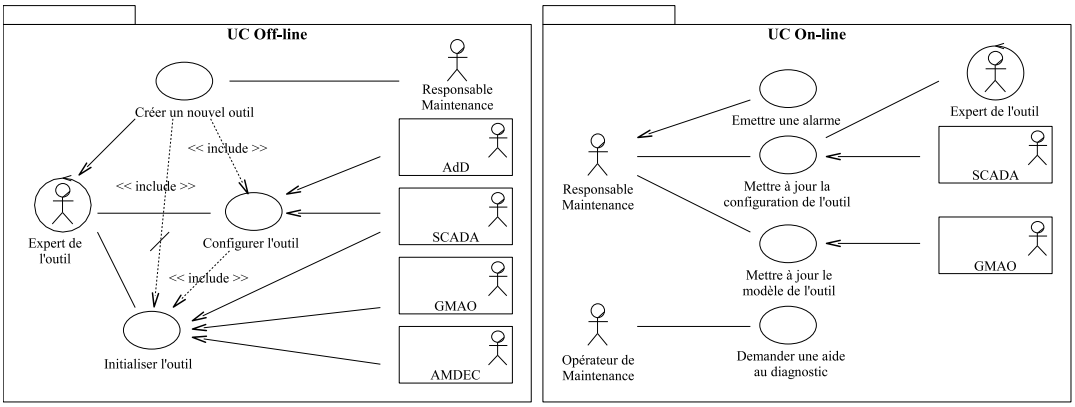
Dans ces paquetages, certains cas d'utilisation sont dépendants d'autres. Ainsi, la création d'un nouveau système nécessite la configuration et l'initialisation de l'outil, qui nécessite elle-même la configuration. Ces liens sont des relations d'inclusion.

Nous regroupons toutes ces informations dans les deux figures IV.2(a) et IV.2(b). Ces figures représentent les deux paquetages « off-line » et « on-line » auxquels nous avons ajouté les liaisons avec les acteurs. Le paquetage des acteurs n'est donc pas représenté afin d'alléger les figures.

### IV.2.2 Spécification détaillée des besoins

A partir des cas d'utilisation que nous avons établis précédemment, nous allons maintenant les décrire de façon détaillée afin d'obtenir une expression des besoins précise. La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML, nous utiliserons une formalisation inspirée de l'ouvrage de Cockburn (2001).

Nous présentons trois cas d'utilisation importants : Configurer l'outil, Initialiser l'outil et Demander une aide au diagnostic. Ces cas d'utilisation définissent les principales caractéristiques de notre système d'aide à la surveillance.




(a) Paquetage Off-line (b) Paquetage On-line

FIG. IV.2 – Paquetages

### IV.2.2.1 Configurer l'outil

Nous supposons que le matériel et les logiciels sont installés et fonctionnent et que le responsable maintenance connaît le système à surveiller. Le scénario nominal de la configuration de l'outil de diagnostic commence par la demande de l'arbre de défaillances. L'expert de l'outil traduit l'arbre en outil de diagnostic. L'architecture de l'outil tiendra compte également de la disponibilité des capteurs du SCADA.

**Acteur principal :** l'expert de l'outil.

**Acteurs secondaires :** les deux systèmes « AdD » et « SCADA ».

**Objectif :** Configuration de l'outil.

**Précondition :** Le Responsable Maintenance a fait appel à l'expert de l'Outil pour créer un nouvel Outil.

**Postcondition :** L'outil est configuré.

**Scénario nominal :**

1. Le système « AdD » fournit l'Arbre de Défaillance à l'Expert de l'Outil ;
2. L'Expert de l'Outil configure l'outil de diagnostic grâce à l'Arbre de Défaillance ;
3. L'Expert de l'Outil configure l'outil de détection afin qu'il soit abonné aux variables utiles du système « SCADA ».

#### IV.2.2.2 Initialiser l'outil

L'initialisation du système d'aide à la surveillance consiste à régler les paramètres à l'aide des données extraites de l'AMDEC et de la GMAO. Ce processus tiendra compte de la criticité (fréquence, gravité) des défauts et du triplet associé Symptôme, Origine, Actions fourni par la GMAO.

**Acteur principal :** L'expert de l'outil.

**Acteurs secondaires :** Les trois systèmes « AMDEC », « GMAO » et « SCADA ».

**Objectif :** Initialisation de l'outil.

**Précondition :** L'outil est configuré.

**Postcondition :** L'outil est initialisé.

**Main scenario :**

1. Le système « AMDEC » fournit l'AMDEC à l'Expert de l'Outil ;
2. L'Expert de l'Outil analyse l'AMDEC et extrait les données utiles (Mode de fonctionnement, cause, fréquence et gravité) ;
3. L'Expert de l'Outil initialise les outils de détection et de diagnostic avec les données extraites ;
4. L'Expert de l'Outil initialise l'outil de diagnostic afin qu'il soit abonné aux événements maintenance du système « GMAO » ;
5. L'outil de détection reçoit les valeurs des capteurs du système « SCADA ».

#### IV.2.2.3 Demander une aide au diagnostic

Un défaut a été détecté ou prédit et/ou l'opérateur de maintenance a besoin d'aide pour effectuer un diagnostic. Le point de départ du diagnostic est la demande d'assistance de l'opérateur de maintenance. Le système suggérera alors un diagnostic et, en conclusion, l'opérateur de diagnostic validera le diagnostic.

**Acteur principal :** L'opérateur de maintenance.

**Objectif :** L'outil fournit une aide au diagnostic.

**Précondition :** L'Outil est en fonctionnement (Configuré et Initialisé). Une panne a été détectée / prédite et / ou l'Opérateur de Maintenance a besoin de l'aide de l'Outil.

**Scénario nominal :**

1. L'Opérateur de Maintenance requiert une aide au diagnostic ;

2. L'Outil fournit un ensemble de causes possibles classées par degré de pertinence et par degré de gravité ;
3. L'Opérateur de Maintenance valide le diagnostic.

**Exceptions :**

1a. L'Opérateur de Maintenance choisit un diagnostic plus précis.

1. L'Opérateur de Maintenance accède à un formulaire spécialisé lui permettant d'ajouter des informations qui ne sont pas données par l'outil de détection (fumée, odeurs ...) et le cas d'utilisation continue à l'étape 2 du scénario nominal.

3a. L'Opérateur de Maintenance n'est pas satisfait par les résultats.

1. L'Opérateur de Maintenance revient à l'étape 1 du scénario nominal pour lancer une nouvelle demande.

1. L'Opérateur de Maintenance abandonne la demande. Le cas d'utilisation se termine (échec).

### IV.2.3 Diagrammes d'interaction

Les diagrammes d'interaction englobe deux types de diagrammes UML spécialisés :

- Les diagrammes de séquence ;
- Les diagrammes de collaboration.

Les deux diagrammes donnent une représentation d'interactions entre les objets. Nous n'utilisons que les diagrammes de séquence pour représenter ces interactions avec trois types de classes d'analyse pour représenter notre système d'aide à la surveillance : les « dialogues », les « contrôles » et les « entités ». Les cas d'utilisation représentés sont :

- Configurer l'outil (figure IV.3) ;
- Initialiser l'outil (figure IV.4) ;
- Demander une aide au diagnostic (figure IV.5).

### IV.2.4 Liens avec les outils

Nous avons présenté dans le chapitre précédant, les outils qui nous paraissent les plus adaptés à cette application. Nous avons aussi présenté les méthodes afin de concevoir ces outils à partir d'une base de données. Nous allons voir maintenant le lien entre les scénarii de configuration et d'initialisation de l'outil avec ces méthodes.

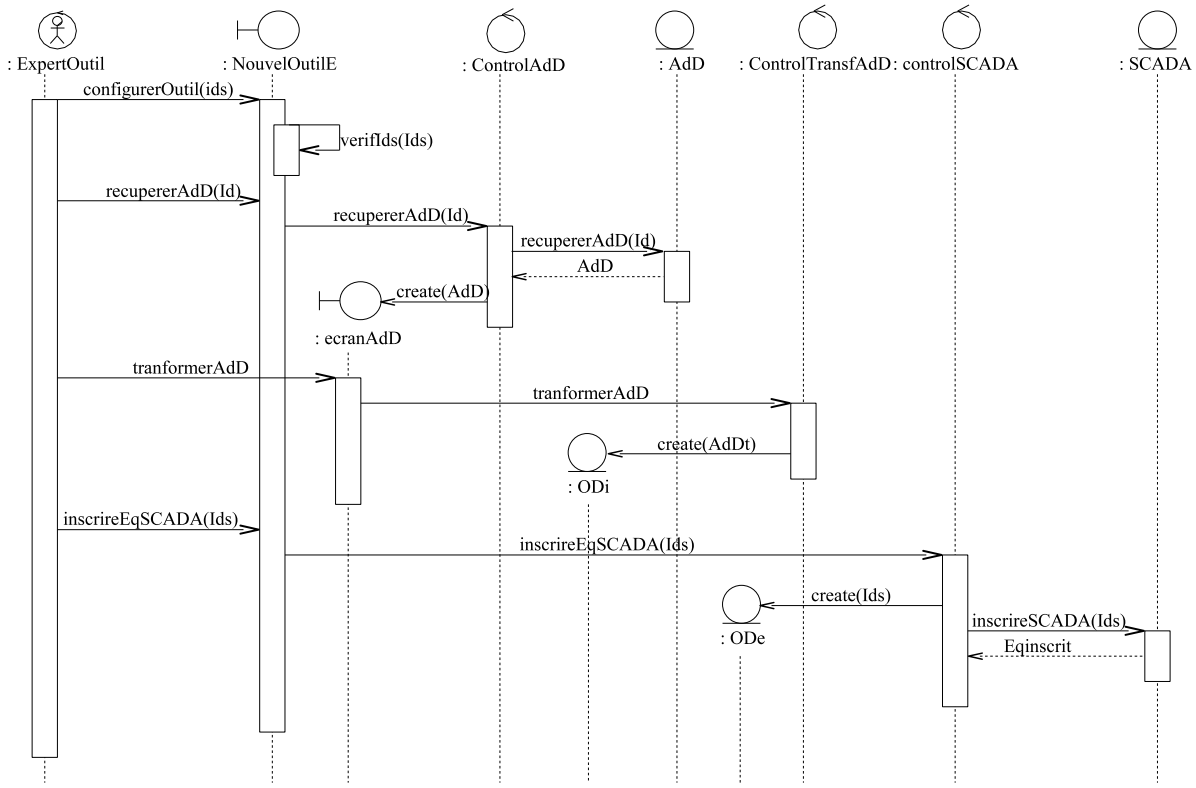



FIG. IV.3 – Diagramme de séquences pour la configuration de l'outil

### IV.2.4.1 Scénario de configuration de l'outil

Nous avons établi précédemment que l'outil de détection est un réseau récurrent à fonction de base radiale. Dans le scénario de configuration, nous avons spécifié au point 3 du scénario nominal : « L'Expert de l'Outil configure l'outil de détection afin qu'il soit abonné aux variables utiles du système ». Cette action permet de déterminer l'architecture de la couche d'entrée du réseau. Chaque neurone d'entrée étant lié à chaque capteur fourni par le SCADA. Nous pouvons aussi établir les paramètres de la couche d'entrée en étudiant les variables d'entrée. Un historique est nécessaire afin d'effectuer la normalisation des données d'entrée. Si cet historique est indisponible, il est nécessaire de connaître les variations considérées comme « normales » pour chaque capteur, ainsi que la longueur de la séquence nécessaire pour une bonne mémorisation des signaux d'entrées. Une seule sortie est ajoutée, c'est la sortie correspondant au bon fonctionnement du système.

Pour l'outil de diagnostic, nous avons utilisé un réseau neuro-flou. Au point 2 du scénario principal, nous avons spécifié : « L'Expert de l'Outil configure l'outil de diagnostic

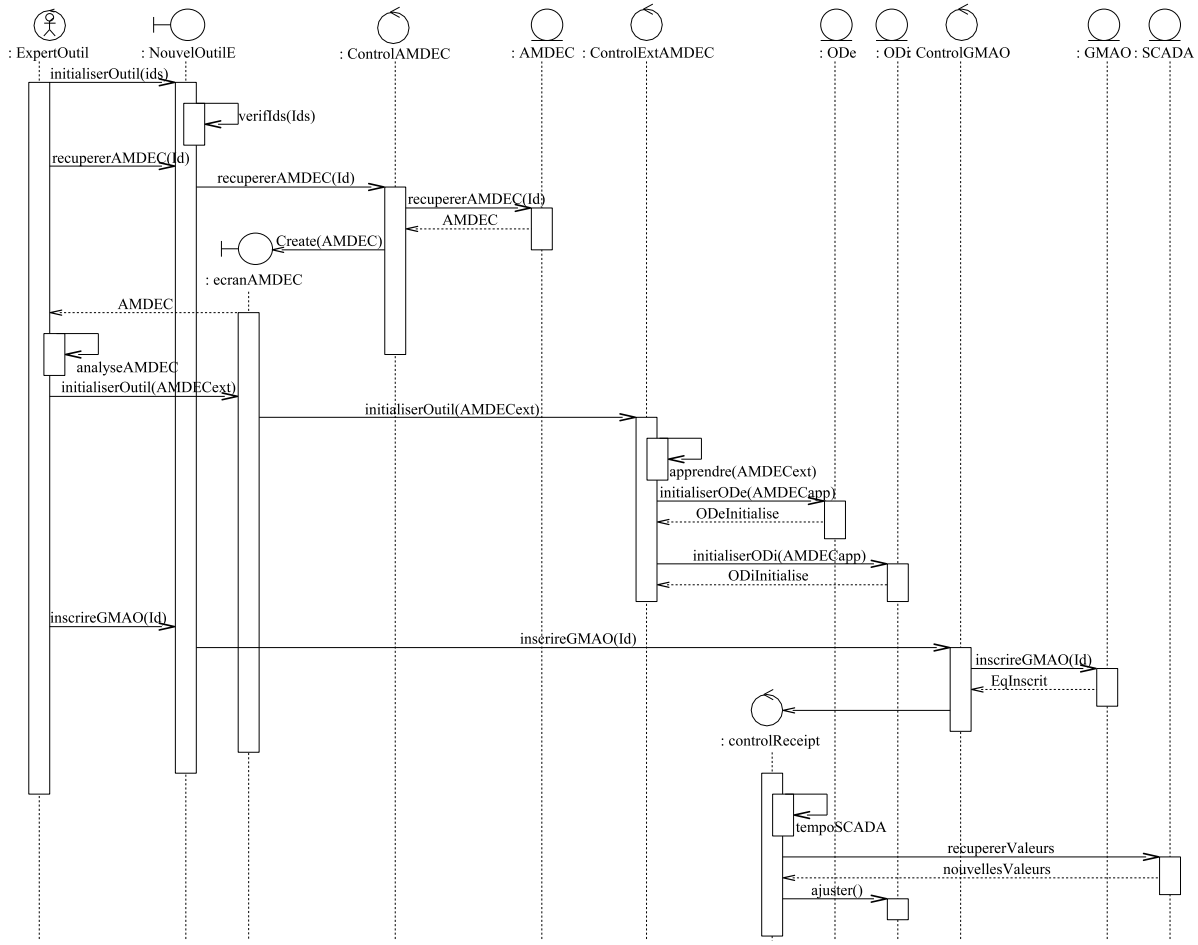



FIG. IV.4 – Diagramme de séquences pour l'initialisation de l'outil

grâce à l'arbre de défaillances ». Pour cela nous utilisons les règles que nous avons défini précédemment pour transformer un arbre de défaillances en un réseau neuro flou.

### IV.2.4.2 Scénario d'initialisation de l'outil

Au point 3 du scénario, « L'Expert de l'Outil initialise les outils de détection et de diagnostic avec les données extraites ». Ces données sont les informations extraites de l'AMDEC. L'initialisation de l'outil de détection consiste à créer les sorties correspondant à chacun des modes de fonctionnement. Lors de l'initialisation, les neurones de la couche cachée sont créés en utilisant l'algorithme DDA modifié présenté au chapitre précédant. Les séquences d'apprentissage sont tirées de l'historique. En cas d'absence d'historique, une simulation du mode normale peut être présentée au système. Si les

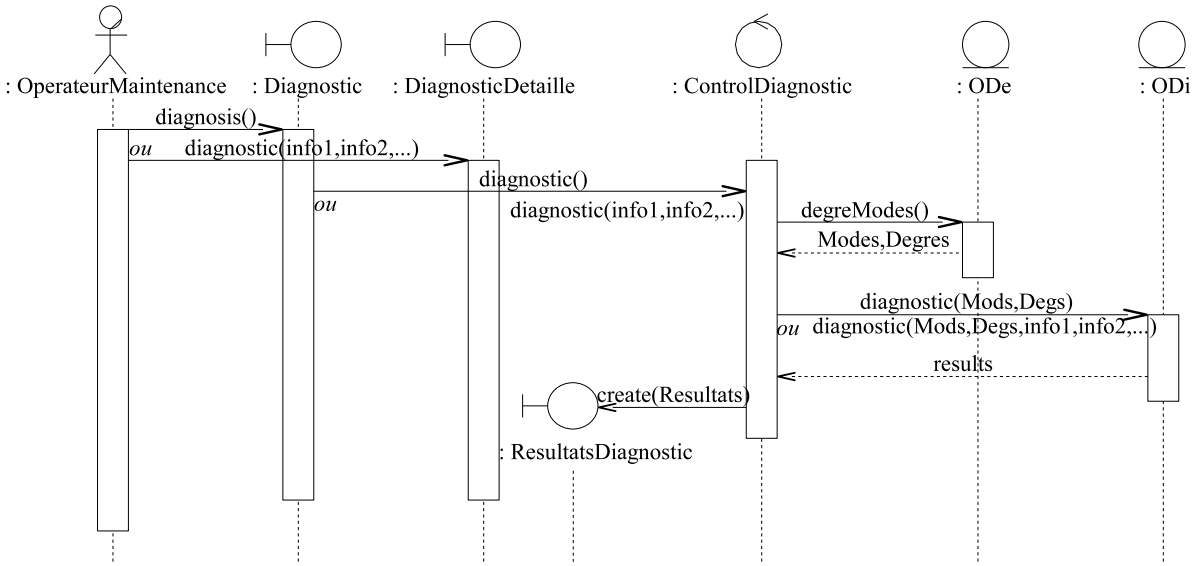



FIG. IV.5 – Diagramme de séquences pour le diagnostic

fonctionnements anormaux ne peuvent être simulés, les neurones de sorties ne seront pas connectés et seront donc retirés des modes identifiables par le système de détection. Ces modes pourront être ajoutés ultérieurement (scénario de la mise à jour du modèle du système). L'initialisation de l'outil de diagnostic s'effectue par les règles que nous avons définies au chapitre précédant en utilisant les modes de fonctionnement (liaison entre les deux outils) et les fréquences et les gravités des causes.

Au point 4, « L'Expert de l'Outil initialise l'outil de diagnostic afin qu'il soit abonné aux événements maintenance du système » ce qui permettra de mettre à jour le réseau neuro-flou au niveau des gravités des causes, d'ajouter de nouvelles causes, ou de créer de nouveaux liens avec le système de détection.

### IV.2.5 Synthèse sur l'UML

Après avoir présenté dans les chapitres précédents les différents outils qui seront utilisés pour la conception de l'outil d'aide à la surveillance. Nous avons vu, via l'utilisation d'une méthodologie UML, les interactions que cet outil aura avec différents systèmes existants. Cette méthodologie nous aura permis de concevoir les différents diagrammes de séquences associés aux cas d'utilisation. Ces diagrammes regroupent toutes les fonctions que nous devons trouver dans l'application. Nous allons voir comment cette application a été conçue avec quel logiciel et sur quelle plate-forme il a été testé.

# IV.3 Application

Dans le cadre d'un projet OSÉO anvar[3] (Palluat *et al.*, 2004*b*), nous avons effectué un prototypage à l'aide du logiciel LabVIEW en mettant au point un système d'aide à la surveillance dans un module temps réel de type PXI (national Instrument).

Notre choix de logiciels s'est porté sur LabVIEW de National Instrument car c'est un environnement permettant de combiner acquisition, traitement et présentation des données, évitant ainsi le stockage intermédiaire de celles-ci, une caractéristique bien utile pour une analyse en temps réel par exemple. De plus, son implantation dans un environnement industriel est facilitée car il permet d'être intégré aisément dans un ordinateur industriel extrêmement compact. Enfin, ce logiciel s'adapte à de nombreuses plates-formes et peut être aisément intégré dans un PDA qui faisait l'objet d'un des objectifs du projet.

## IV.3.1 Présentation de LabView

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est un logiciel de développement d'applications d'instrumentation. Bien que tout à fait utilisable dans un grand nombre de domaines, LabVIEW est plus particulièrement destiné à l'acquisition de données et au traitement du signal. En effet, ce logiciel offre de larges possibilités de communication entre l'ordinateur et le monde physique (par cartes d'acquisitions analogiques ou numériques, cartes GPIB, réseau, liaisons série et parallèles, etc.) ainsi que d'importantes bibliothèques mathématiques permettant de traiter les signaux mesurés. L'idée de LabVIEW est de remplacer les instruments de mesure et d'analyse d'un laboratoire par un ordinateur muni de cartes spécifiques et d'un logiciel approprié, au même titre qu'un ordinateur muni d'une carte son et d'un logiciel de musique peut remplacer n'importe quel instrument de musique ou bien encore une table de mixage. Dans le cadre de la mesure, les cartes permettent de convertir des signaux électriques (provenant de capteurs mesurant des grandeurs physiques) en données numériques. Ainsi, un seul ordinateur muni d'une carte d'acquisition analogique et de LabVIEW est capable de remplacer un voltmètre, un fréquencemètre ou un oscilloscope. De plus, on pourra traiter, analyser et archiver sur disque automatiquement les mesures effectuées.

## IV.3.2 Plate-forme flexible

Ce système a été appliqué sur un système industriel, une plate-forme flexible, disponible à l'Institut de Productique[4] de Besançon. Cette plate-forme est équipée de cinq

[3]Site : http ://www.anvar.fr/agenaccoaideaide.htm
[4]Site : http ://www.institutdeproductique.com/

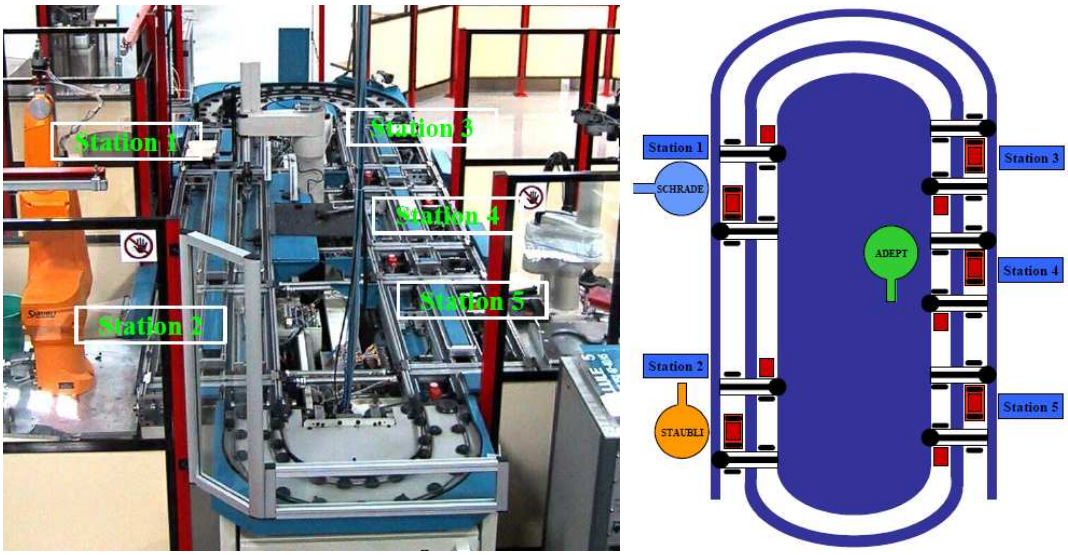
automates industriels communiquant entre eux via un réseau industriel local. Le système flexible permet de déplacer des palettes pouvant recevoir des composants à assembler.

Le réseau permet d'échanger les informations reçues par les automates concernant les changements d'état des capteurs, les séquences de GRAFCET, et le déroulement des opérations réalisées sur les palettes.

La plate-forme est divisée en cinq stations. Chacune de ces stations dispose de son automate. Elles fonctionnent donc indépendamment les unes des autres.

### IV.3.2.1 Vision d'ensemble de la plate-forme

La plate-forme est constituée de deux anneaux (cf figure IV.6). Un anneau intérieur et un extérieur.




(a) Photo de la plate-forme (b) Schéma de la plate-forme

FIG. IV.6 – Vue de la plate-forme

L'anneau intérieur permet de faire circuler les palettes entre toutes les stations. Cet anneau doit donc toujours être libre pour laisser circuler les palettes.

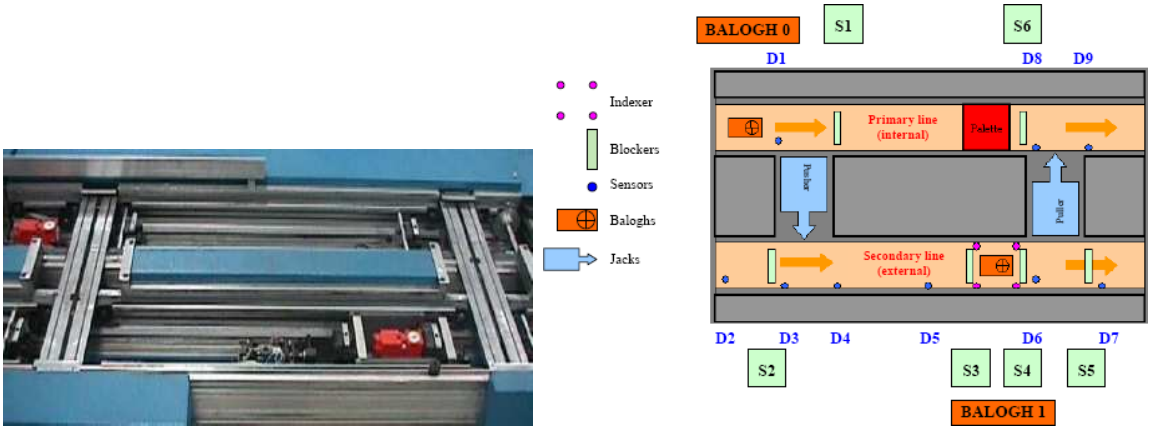
L'anneau extérieur est dédié au traitement des tâches à effectuer sur les palettes. C'est sur cet anneau que sont installés les robots et le manipulateur. L'architecture de l'installation permet de faire transiter une palette d'une station à la suivante sans la faire revenir sur l'anneau intérieur. Ceci est valable pour toutes les stations sauf pour le passage de la station n°2 à la station n°5.

Les palettes circulent sur la plate-forme grâce à des courroies plates. Chaque station dispose d'un moteur électrique, entraînant les courroies.

Les palettes sont munies d'une étiquette magnétique qui leur sert de « mémoire embarquée ». Ces mémoires peuvent être lues dans chaque station grâce à des têtes magnétiques de lecture / écriture (BALOGH). Ces étiquettes permettent de mémoriser la gamme d'assemblage des produits, pour savoir par quel(s) poste(s), les palettes doivent passer.

### IV.3.2.2 Détail d'une station

Toutes les stations sont identiques. Chacune d'elles est composée d'un pousseur, d'un tireur, d'un indexeur, de 9 détecteurs, de 6 stoppeurs et de 2 BALOGH.




(a) Photo d'une station

(b) Schéma d'une station

FIG. IV.7 – Détail d'une station

### IV.3.2.3 Application de notre outil

Nous avons donc développé notre outil à l'aide du logiciel LabVIEW et l'avons intégré dans un ordinateur industriel PXI de National Instrument. Le choix de cet automate est dérivé du choix du logiciel de développement puisque LabVIEW est développé par National Instrument.

La liaison se fait actuellement via un ordinateur fixe mais des développements futurs sont prévus pour une communication par ondes radio grâce à la technologie WiFi. Pour cela, nous avons fait l'acquisition de deux boîtiers convertisseurs Ethernet - WiFi,

l'un permettant de relier le PXI et l'autre l'ordinateur fixe. Enfin, des tests avec un PDA seront effectués afin de faciliter l'utilisation de notre outil pour un opérateur de maintenance. Tous les éléments décrits ci-dessus sont repris sur la figure IV.8.

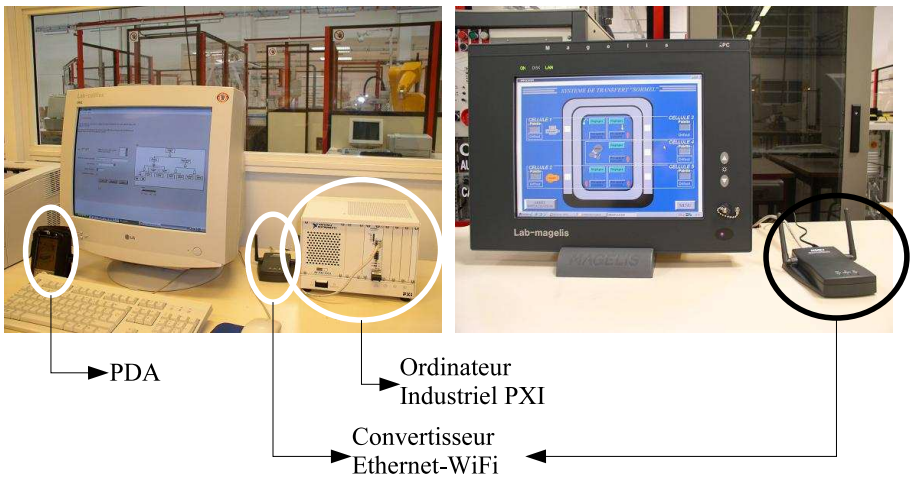



FIG. IV.8 – Vue d'ensemble du prototype de système d'aide à la surveillance

### a) Extraction des données pertinentes

Nous limitons pour cet exemple l'étude sur l'entrée d'une station. Nous trouvons sur la figure IV.9 l'extrait du schéma d'une station représentant la zone surveillée et l'extrait de l'arbre de défaillance.

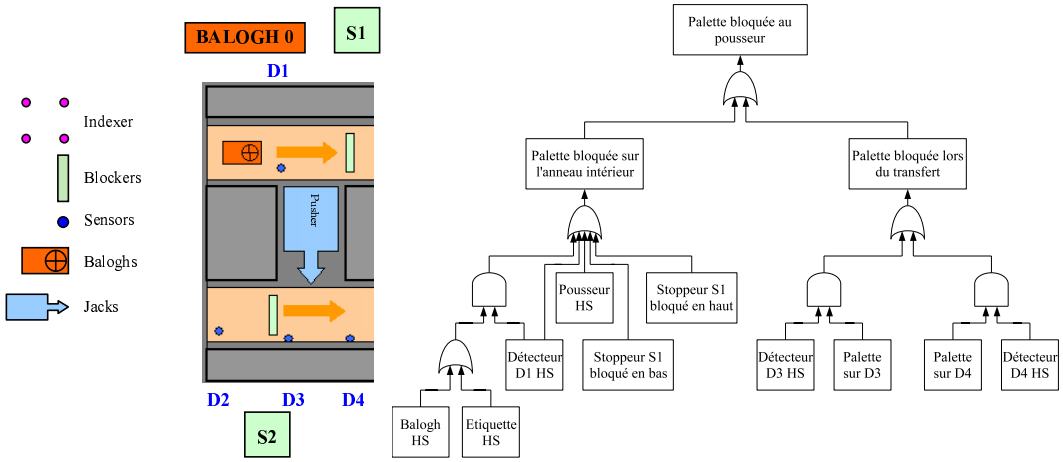
Nous avons aussi extrait de l'AMDEC les données liées à l'entrée du poste (table IV.1).

| Modes de défaillances | Cause | Fréquence | Gravité |
|---|---|---|---|
| Palette bloquée sur l'anneau intérieur | Détecteur D1 est en panne | 4 | 2 |
| Palette bloquée sur l'anneau intérieur | Stoppeur S1 bloqué en position basse | 3 | 4 |
| Palette bloquée sur l'anneau intérieur | Le pousseur est en panne | 1 | 2 |

TAB. IV.1 – AMDEC de la zone surveillée

Il est nécessaire aussi d'établir la liste des variables utiles du SCADA nécessaire à la phase de configuration. Ces variables sont liées aux différents éléments présents sur la figure IV.9(a). Nous retrouvons ces différentes valeurs sur le tableau IV.2.

A partir de ces informations, nous allons établir la configuration et l'initialisation des outils de détection et de diagnostic. Reprenons chaque scénario et voyons comment nous l'avons adapté dans un programme LabView.




(a) Vue extraite de l'entrée de poste

(b) Arbre de défaillance de la zone surveillée

FIG. IV.9 – Données extraites de la zone à surveiller

| Composant | Mnémonique | Commentaires |
|---|---|---|
| Balogh 0 | | Plot de lecture à la volée |
| Balogh 0 | Crlectv | Lecture à la volée validée |
| Balogh 0 | Crpostc | Poste concerné par l'opération |
| Balogh 0 | Pretiqu0 | Présence étiquette sur le balogh 0 |
| D1 | Prpouss | Présence palette au pousseur |
| D2 | Prpstop2 | Présence palette au stoppeur S2 |
| D3 | Prextere | Palette engagée sur l'anneau extérieur au niveau du pousseur |
| D4 | Prexterd | Palette dégagée sur l'anneau extérieur au niveau du pousseur |
| Pousseur | Cdpoussa | Commande avancer le pousseur |
| Pousseur | Cdpoussr | Commande reculer le pousseur |
| Pousseur | Crpoussa | Pousseur avancé |
| Pousseur | Crpoussr | Pousseur reculé |
| S1 | Cdstop1 | Commande baisser le stoppeur S1 |
| S2 | Cdstop2 | Commande baisser le stoppeur S2 |

TAB. IV.2 – Liste des variables utiles

### b) Configuration de l'outil

Dans ce scénario, nous avons 3 étapes :

- Récupération de l'arbre de défaillances ;
- Transformation de l'arbre de défaillances en réseau neuro-flou ;
- Abonnement du réseau RRBF au variables du SCADA.

La première étape consiste à récupérer l'arbre de défaillances. Nous avons fait le choix que cet arbre de défaillances se trouvait sous la forme d'un fichier image. Pour simplifier les choses, nous avons spécifié que le format de ce fichier image serait du JPEG. La première étape consiste donc en l'affichage de l'image représentant l'arbre de défaillances (figure IV.10).

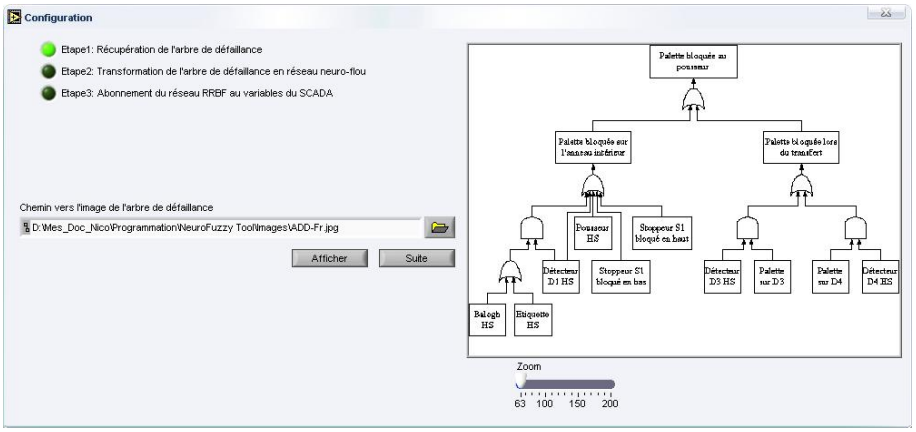



FIG. IV.10 – Capture de LabVIEW : récupération de l'arbre de défaillance

La deuxième étape consiste en la transformation de l'arbre de défaillance en réseau neuro-flou. Pour cela, nous demandons à l'expert de transformer l'arbre de défaillance en un ensemble de clusters composés :

- de la référence du cluster,
- du type de cluster (Porte ET, Porte OU, Porte Non, événement élémentaire),
- du nom de l'évènement indésirable en sortie de la porte,
- de la référence de la (des) porte(s) amont,
- du nombre de clusters en aval.

Pour simplifier le travail de l'expert, la référence du cluster est établie automatiquement en fonction de sa position dans l'arbre de défaillance. En effet, nous avons fait le choix de définir l'arbre de défaillances comme un tableau à deux dimensions dont chaque élément correspondrait à une porte et chaque ligne correspondrait à un « étage » dans l'arbre de défaillance.

Ainsi l'arbre de défaillances de notre exemple (figure IV.9(b)) peut être représenté par le tableau de la figure IV.11.

Par exemple le cluster (1,1) aura comme information :

- référence : (1,1),
- type : Porte OU,
- événement : Palette bloquée lors du transfert,

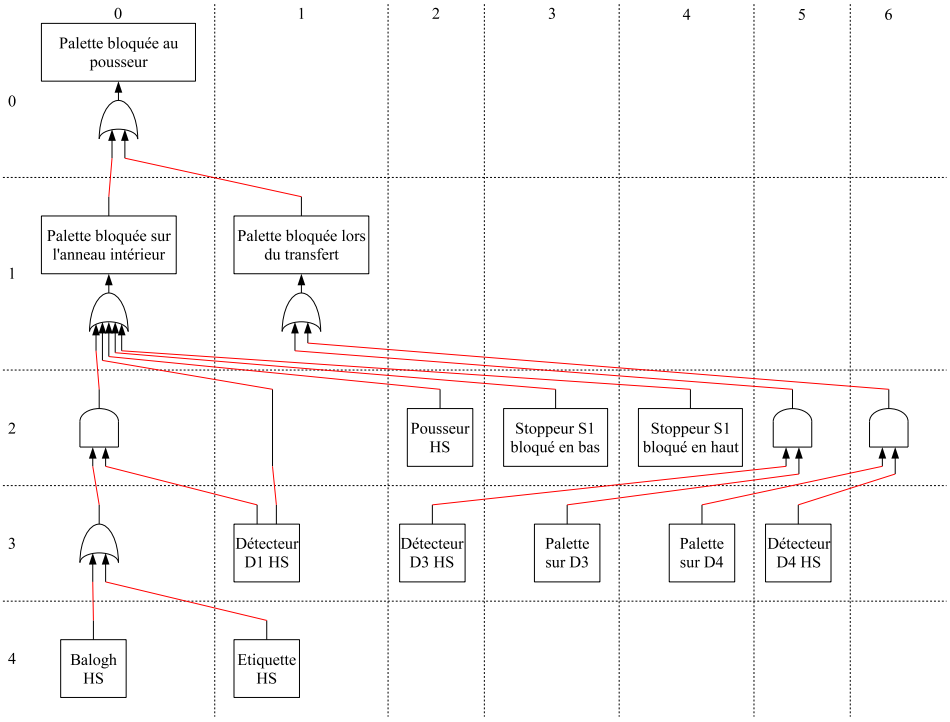



Fig. IV.11 – Modification de l'arbre de défaillance

– référence(s) amont : 0 (et non (0,0) car l'étage sera forcément celui directement inférieur),
– nombre de clusters en aval : 2.

A noter que nous ajoutons un type de cluster supplémentaire, le cluster « ligne ». Ce cluster est utilisé dans le cas où nous n'avons ni une porte ni un événement élémentaire. Dans l'exemple, le cluster (2,1) est un cluster « ligne ». Sur la figure IV.12, nous pouvons voir la seconde phase de la configuration avec la création d'un cluster.

Afin de vérifier les informations entrées, nous ajoutons un tableau récapitulatif avec le nombre de clusters par « étage » et par « type » (figure IV.13).

Enfin, dans la dernière étape, l'expert de l'outil doit abonner l'outil de détection aux variables du SCADA. Pour cela, nous faisons l'hypothèse que ces données sont accessibles via un serveur OPC. Pour cela l'expert indiquera l'adresse IP du serveur ainsi que le port sur lequel les données sont accessibles. Enfin, il donnera les références des capteurs qu'il souhaite observer (colonne « Mnémonique » du tableau IV.2). Pour simplifier l'intégration de ces références, l'expert devra entrer l'adresse du serveur et le port sur lequel ces données sont accessibles. La liste des capteurs utilisables est ainsi référencé automatiquement. Il ne restera plus qu'à cocher les références désirées. Un compteur indiquera en temps réel, le nombre de variables abonnées (figure IV.14).

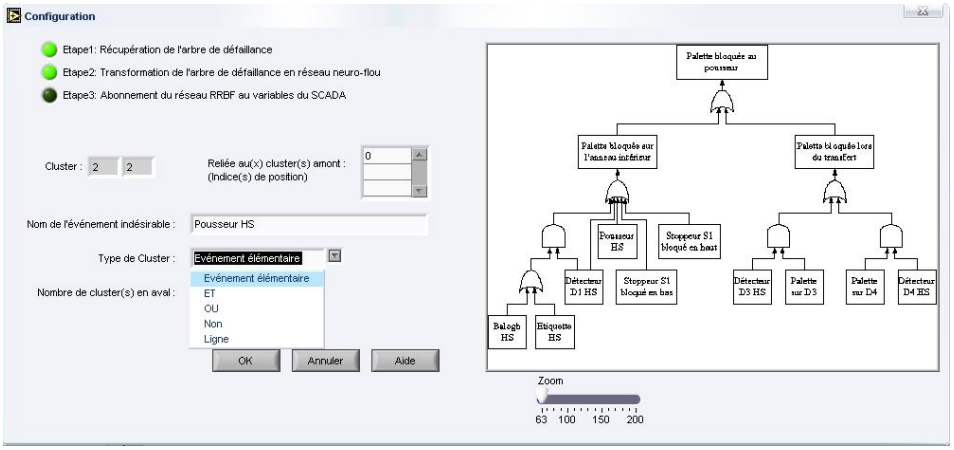



FIG. IV.12 – Capture de LabVIEW : transformation de l'arbre de défaillance en réseau neuro-flou

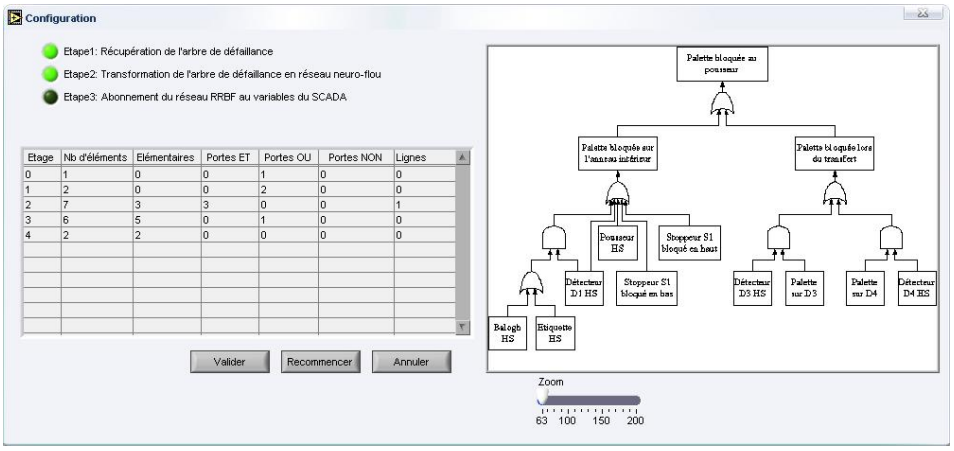



FIG. IV.13 – Capture de LabVIEW : Récapitulatif transformation de l'ADD en RNF

### c) initialisation de l'outil

Dans ce scénario, nous avons 5 étapes :

- Récupération de l'AMDEC ;
- Extraction des données utiles de l'AMDEC ;
- Apprentissage de ces données par le RRBF et le réseau neuro-flou ;
- Abonnement du réseau NF aux événements maintenance de la GMAO ;
- Démarrage de l'acquisition par le RRBF.

La première étape consiste en la récupération de l'AMDEC. L'AMDEC pouvant se trouver sous différentes formes, nous supposons que l'expert de l'outil possède le programme permettant la lecture de l'AMDEC. Nous passons ainsi directement à la seconde étape : l'extraction des données utiles de l'AMDEC. L'expert de l'outil doit


FIG. IV.14 – Capture de LabVIEW : Abonnement du réseau RRBF aux variables du SCADA

retranscrire les différents modes de fonctionnement, les causes, les fréquences et gravités de l'AMDEC. Pour cela, à chaque ajout, l'expert de l'outil aura le choix sur différentes causes définies par les événements primaires de l'arbre de défaillances. Si ceux-ci ne correspondent pas à ce qu'il désire alors tous les événements du reste de l'arbre de défaillances seront proposés. Pour le mode de fonctionnement, seuls les modes déjà entrés par l'expert seront présentés. Si aucun ne correspond l'expert peut entrer un nouveau mode de fonctionnement. (figure IV.15)

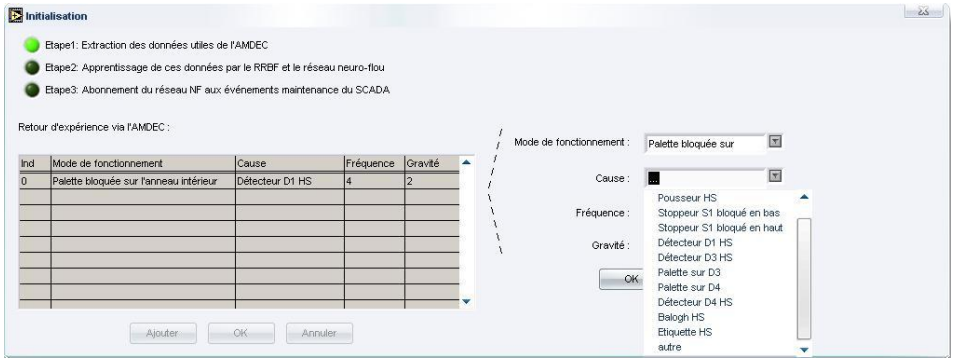



FIG. IV.15 – Capture de LabVIEW : Extraction de l'AMDEC

La troisième étape consiste en l'apprentissage des données. L'expert peut visionner les différents modes de fonctionnement que pourra reconnaître le système de détection. S'il possède une base d'apprentissage, l'expert peut indiquer le fichier correspondant. Sinon, l'apprentissage sera effectué lors de la mise à jour du modèle de l'outil. (figure IV.16)

Dans la quatrième étape, l'expert de l'outil doit abonner le réseau neuro-flou aux événements maintenance. Pour cela nous utilisons les web-services afin d'obtenir les


FIG. IV.16 – Capture de LabVIEW : Apprentissage des réseaux

informations désirées. L'expert de l'outil doit donc spécifier le fichier contenant ces informations. (figure IV.17)

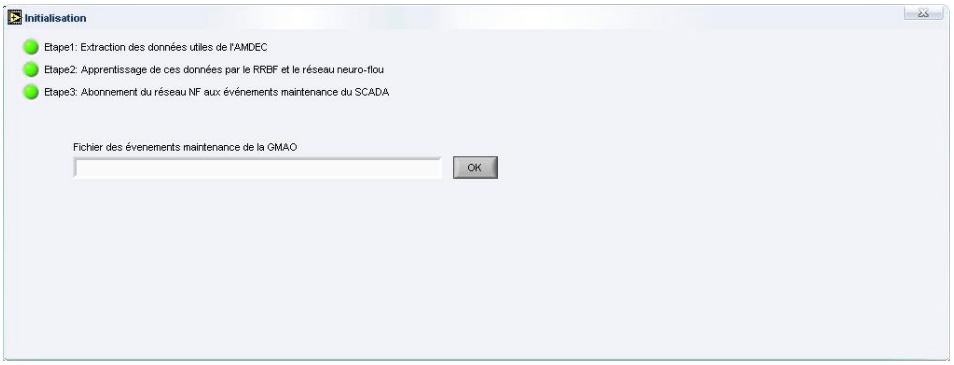



FIG. IV.17 – Capture de LabVIEW : Abonnement du réseau NF aux événements maintenance de la GMAO

Enfin, dans le dernier point, l'expert de l'outil n'a aucune action à exécuter. Le programme lance simplement l'acquisition des valeurs grâce aux paramètres fournis lors de la dernière étape de la configuration. Après cette étape, l'outil d'aide à la surveillance est opérationnel. Nous allons maintenant voir le cas où l'opérateur de maintenance demande une aide au diagnostic.

### d) Demander une aide au diagnostic

Dans ce scénario, l'opérateur décide de demander une aide au diagnostic suite au blocage d'une palette sur le système de transfert (figure IV.18).

Il lance alors une demande d'aide au diagnostic, la réponse doit comporter les différentes causes possibles classées par degré de pertinence et par gravité. Nous avons fait le choix de représenter le résultat sous forme d'un tableau regroupant ces informations.

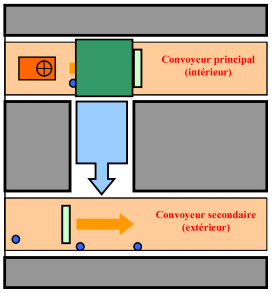



FIG. IV.18 – Exemple du blocage d'une palette

Pour améliorer le confort visuel, nous avons associé un code couleur à chaque gravité (du vert – gravité nulle, au rouge – gravité maximale). Le résultat de la demande de diagnostic est représenté sur la figure IV.19.

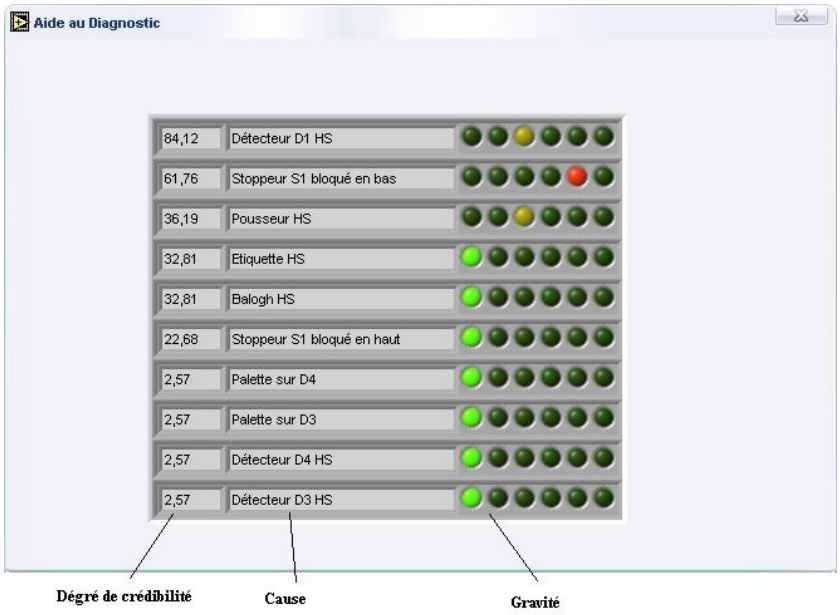



FIG. IV.19 – Capture de LabVIEW : Exemple du blocage d'une palette

Grâce à l'outil, l'opérateur de l'outil devra suspecter en priorité le détecteur D1 mais devra faire attention stoppeur S1 qui possède à la fois un fort degré de pertinence et une forte gravité. A l'aide de cet outil, l'opérateur pourra cibler plus facilement cibler les éléments à regarder.

## IV.4 Conclusion

Nous avons présenté dans ce chapitre l'intégration de l'outil d'aide à la surveillance. En partant de besoins simples (interfaçage avec outils et méthodes existants, intégration dans un ordinateur industriel ...), nous avons conçu un outil permettant de mettre en œuvre une surveillance dynamique d'un système industriel, capable de prévenir le responsable de maintenance en cas de défaillances et d'aider l'opérateur lors d'un diagnostic. Nous avons définis les outils dans les chapitres précédants. Dans ce chapitre, nous avons spécifié comment ces outils fonctionnaient, comment ils interagissaient entre eux mais aussi avec le reste du système ainsi qu'avec les opérateurs. Pour cela, nous avons suivi une démarche UML. Nous avons ainsi défini les différents cas d'utilisation de l'outil ainsi que ses diagrammes d'interaction. Par la suite, nous avons développé les programmes dans un langage pouvant aisément être intégré dans un ordinateur industriel. Notre choix s'est porté sur LabVIEW avec une intégration dans un ordinateur industriel PXI. Nous avons enfin présenté une application pratique d'un système flexible de production par la création d'un outil d'aide à la surveillance sur une zone ciblée du système.

# Conclusion générale

Les travaux de recherche présentés dans ce mémoire portent sur l'étude d'un système d'aide à la surveillance avec des techniques de l'intelligence artificielle. Dans le cadre de la surveillance, notre étude concerne la détection de défaillances et le diagnostic de défauts. Nous avons ainsi proposé de nouveaux algorithmes d'apprentissage pour la partie détection qui sera un réseau récurrent à fonctions de base radiales (RRFR) ainsi qu'une nouvelle architecture de réseau neuro-flou pour la partie diagnostic. Le réseau RRFR gère le temps via sa première couche constituée de neurones à fonction de base sigmoïdale et à retour local de la sortie. Elle agit en tant que mémoire dynamique. La couche cachée constituée de neurones gaussiens représente la mémoire statique du réseau. Les algorithmes d'apprentissage que nous avons établis permettent de configurer et d'initialiser le réseau à l'aide de deux paramètres (un degré d'oubli et un degré de pertinence) et d'une base d'apprentissage. Cette base d'apprentissage contient l'ensemble des modes de fonctionnement du système à surveiller ainsi que les séquences de mesures des différents capteurs permettant d'obtenir les modes de fonctionnement. L'apprentissage du réseau neuro-flou utilise quand à lui l'arbre de défaillance pour déterminer son architecture et l'AMDEC du système pour déterminer les paramètres de ses fonctions d'activation.

Les principales contributions de cette étude sont regroupées en trois parties. Une première partie regroupe l'état de l'art sur les différentes méthodologies de surveillance des systèmes de production. La deuxième partie de notre travail correspond à l'essentiel de notre contribution scientifique. Nous avons ainsi proposé une nouvelle technique d'apprentissage pour le réseau RRBF lorsqu'il est utilisé comme outil de détection. Cette nouvelle technique est basée sur deux algorithmes existants : le DDA et le Fuzzy Min-Max. Concernant le diagnostic, une nouvelle architecture de réseaux neuro-flous a été proposée, à partir de l'arbre de défaillance et de l'AMDEC du système à surveiller. La troisième partie concerne l'exploitation industrielle de cette étude. Cette dernière s'est effectuée grâce au logiciel de programmation LabVIEW dans le cadre d'un projet OSÉO anvar, sur une base élaborée dans le cadre du projet européen PROTEUS, en suivant une démarche UML.

Ce mémoire débute par un état de l'art exhaustif sur les techniques utilisées dans le domaine de la surveillance industrielle. La première étape a consisté en la définition des

principaux concepts de ce domaine et à les placer dans des domaines plus génériques que sont l'automatique et la sûreté de fonctionnement. Cet état de l'art nous a amené à faire des choix dans certaines définitions devant le nombre important de points de vue divergeants. La seconde étape a consisté à établir un état de l'art sur les travaux de surveillance.

Nous avons ainsi effectué une classification de ces méthodes à partir de deux notions simples, l'existence ou non d'un modèle formel de l'équipement à surveiller. Nous avons donc présenté, d'une part, les méthodes sans modèle : les outils statistiques et les techniques de l'intelligence artificielle, et d'autre part, les méthodes avec modèle, à savoir les méthodes de modélisation fonctionnelle et matérielle et les méthodes de modélisation physique. Les techniques avec modèles ont pour principe de comparer l'état théorique du système fourni par le modèle avec son état courant donné par les observations. Elles sont toutefois difficiles à mettre en œuvre et se montre peu robuste par rapport aux d'incertitudes liées à la complexité des systèmes et à l'influence de l'environnement extérieur. Les techniques de l'Intelligence Artificielle ne se basent pas sur le modèle de l'équipement et prennent en compte les perturbations ainsi que les bruits de mesure, d'une manière implicite. La surveillance à base de modèle est souvent opérée hors ligne, empêchant ainsi des traitements temps réel. En revanche, l'Intelligence Artificielle offre des outils totalement découplés de la structure du système, permettant un suivi temps réel de l'évolution de celui-ci. Le raisonnement en ligne fait que l'approche de l'Intelligence Artificielle est plus robuste à des changements de modes opératoires comme pour les systèmes ayant plusieurs configurations. Elle est donc évolutive. Les systèmes de surveillance par outils de l'Intelligence Artificielle peuvent donc représenter d'excellents systèmes d'aide à la décision pour l'expert humain.

La conclusion qui ressort de cette étude met en évidence quatre points essentiels à prendre en compte lors de la conception d'un outil d'aide à la surveillance. Ces quatre points clés permettent de mettre en avant les avantages et les inconvénients des différentes méthodes de surveillance. Ils peuvent être énoncés de la manière suivante :

1. les difficultés liées à l'acquisition des informations nécessaires et en particulier des modèles,
2. la capacité à prendre en compte l'incertain et l'imprécision,
3. la généricité des outils et leur capacité à évoluer avec le système,
4. la validation (ou l'évaluation) des résultats obtenus.

Dans ce contexte, nous avons pu constater qu'un seul outil ne pouvait à la fois effectuer la partie détection et la partie diagnostic dans un système de surveillance. Ainsi nous avons fait le choix d'un outil différent adapté à chacune des parties.

Pour la partie détection, où le temps est très important, nous nous sommes intéressés aux réseaux de neurones temporels pour leur capacité d'apprentissage, leur parallélisme

dans le traitement, leur capacité de faire face à des problèmes inhérents à la non-linéarité des systèmes, et leur rapidité de traitement quand ils sont implémentés en circuit intégré.

Pour la partie diagnostic, deux outils semblent être particulièrement intéressants. Les réseaux de neurones permettent de conserver les capacités d'apprentissage. Par ailleurs la logique floue semble indispensable pour fournir une aide au diagnostic pertinente, par ses capacités à formaliser les connaissances nécessaires aux systèmes de diagnostic et à prendre en compte l'imprécision et l'incertitude.

La classification des réseaux de neurones temporels se base sur deux concepts : l'intégration de la notion temporelle et la nature de la réponse. D'un point de vue temporel, nous retrouvons une représentation externe et interne (implicite ou explicite) du temps. Concernant la nature de la réponse, le choix se situe par rapport à son aspect local et global. Nous avons ainsi montré que les architectures à représentation interne implicite du temps, appelés communément réseaux de neurones récurrents, sont les mieux adaptés pour la surveillance et le diagnostic. L'apport de la nature de la réponse nous a permis de limiter le nombre de réseaux puisque la phase de détection correspond à un problème d'extrapolation. Nous avons donc montré à travers des applications de classification et de prédiction, que le réseau récurrent à fonction de base radiale est le plus adapté. Les algorithmes de base existants ne sont pas optimisés pour une application de détection dynamique. Nous avons ainsi proposé une démarche permettant la construction de l'outil de détection. La première étape consiste en l'apprentissage de la mémoire dynamique du réseau RRFR. La seconde étape consiste en l'apprentissage de la mémoire statique. Nous nous sommes ainsi basés sur l'algorithme DDA que nous avons modifié sur quatre points : suppression d'un point de vérification, ajout d'un rayon d'influence initial, ajustement des zones de conflits ne s'effectuant qu'en certaines conditions, et utilisation d'un seul seuil de vérification. Ces modifications ont induit l'utilisation d'un second algorithme permettant une initialisation du réseau. Nous avons utilisé pour cela une technique de type Fuzzy Min-Max.

Les réseaux neuro-flous permettent d'exploiter les capacités d'apprentissage des réseaux de neurones d'une part et les capacités de raisonnement de la logique floue d'autre part. Différentes combinaisons de ces deux techniques existent et mettent en avant des propriétés différentes :

- réseau flou neuronal,
- système neuronal / flou simultanément,
- modèles neuro-flous coopératifs,
- modèles neuro-flous hybrides.

Dans des applications de diagnostic, on trouve principalement des modèles neuro-flous hybrides, pour lesquels réseau de neurones et système flou sont combinés de manière homogène.

En partant de l'arbre de défaillance et des informations fournies à la fois par le

système de détection et l'opérateur, nous avons établi des règles qui permettent d'obtenir un réseau neuro-flou pour le diagnostic.

Ce réseau est conçu en quatre phases. La première consiste en la transformation de l'AdD en réseau neuro-flou grâce à des règles de transformation que nous avons établies. La seconde phase consiste en l'apprentissage des paramètres des neurones du réseau à l'aide de l'AMDEC. A chaque cause présente dans l'AMDEC est associée une fréquence, nous avons créé des conditions permettant de définir à partir de ces fréquences les paramètres de tous les neurones du réseau. La troisième phase représente la liaison avec l'outil de détection. L'outil de détection possède en sortie des modes de fonctionnement que nous pouvons associer à certaines causes de l'AdD. A partir de cette hypothèse, nous avons présenté diverses règles permettant d'intégrer cette information dans le réseau, en le modifiant. Ainsi, à partir des informations du système de détection, chaque cause possède un degré de pertinence. De même, nous avons intégré la possibilité d'ajouter des informations externes données par l'opérateur. Cette dernière phase nécessite la création d'un nouveau réseau en utilisant les règles utilisées pour la liaison avec l'outil de détection.

Enfin, nous avons présenté la partie exploitation industrielle qui s'est déroulée en deux étapes.

La première a été de formaliser les éléments exposés dans les parties précédentes. Cette étape a été réalisée dans le cadre du projet européen PROTEUS à l'aide du formalisme UML. Nous en avons extrait sept cas d'utilisations :

1. Créer un nouvel outil ;
2. Configurer l'outil ;
3. Initialiser l'outil ;
4. Émettre une alarme ;
5. Demander une aide au diagnostic ;
6. Mettre à jour la configuration de l'outil ;
7. Mettre à jour le modèle de l'outil.

Les trois premiers cas se font hors ligne et représentent la mise en place de l'outil dans son environnement. C'est dans ces cas d'utilisation que nous trouvons les algorithmes que nous avons développés. Les quatre cas d'utilisations qui suivent se présentent lors du fonctionnement de l'outil. Le cas d'utilisation « Émettre une alarme » est directement relié à l'outil de détection permettant de prévenir en temps réel l'opérateur d'un problème sur l'équipement. Cet outil dynamique est capable d'effectuer une prédiction. En cas de panne, l'opérateur pourra faire appel à l'outil pour lui demander une aide au diagnostic. Dans ce cas, ce sera à l'outil de diagnostic d'agir. Les deux cas de mise à jour permettent la prise en compte de l'intégration d'un nouvel élément dans l'équipement à surveiller

ainsi que la prise en compte des différents diagnostics et réparations effectués depuis la mise en fonctionnement de l'outil.

Ces cas d'utilisation nous ont conduits à créer des diagrammes d'interaction qui permettent de représenter les liens qui associent les acteurs aux outils. Dans ce mémoire, nous avons développé trois cas d'utilisation : Configurer l'outil, Initialiser l'outil et Demander une aide au diagnostic.

La deuxième étape s'inscrit dans le cadre du projet OSÉO anvar avec le prototypage de l'outil. Il s'est effectué en LabVIEW qui est un environnement permettant de combiner acquisition, traitement et présentation des données, évitant ainsi le stockage intermédiaire de celles-ci, caractéristique très utile pour une analyse en temps réel. De plus, son implantation dans un environnement industriel est facilitée car il permet d'être intégré aisément dans un ordinateur industriel extrêmement compact.

Nous avons ainsi mis au point un système d'aide à la surveillance embarqué dans un module temps réel de type PXI (National Instrument). Les outils et leur résultat sont accessibles via une connexion ethernet permettant l'accès à distance (à l'intérieur ou à l'extérieur de l'entreprise). Cette application ouvre des perspectives très intéressantes pour l'externalisation de la maintenance, en permettant aux opérateurs de se focaliser de manière plus efficace sur leur métier de base.

Cet outil ouvre la voie à de nombreuses perspectives. En effet, nous avons pu montrer que nous pouvions intégrer les outils industriels tels que l'AMDEC et l'AdD pour la conception d'un outil flexible permettant d'effectuer une aide au diagnostic. Nous avons aussi montré que grâce à cet outil nous pouvons prendre en compte des informations qu'actuellement seul l'opérateur peut donner (bruit anormal de la machine, odeur de brûlé ...) De nombreux développements complémentaires sont ainsi à envisager, notamment les cas d'utilisation de mise à jour qui n'ont pas été développés.

Un autre point devant être abordé se situe au niveau du traitement dynamique. En effet, nous avons supposé que les données évoluaient dans des gammes de temps communes. A l'avenir, il faudra donc prévoir la gestion de variables évoluant à des rythmes différents comme par exemple, une température qui n'aura pas le même taux d'échantillonage qu'une vitesse de rotation d'un arbre. Enfin, la mémoire dynamique du réseau RRBF n'étant pas infinie il faudra prendre en compte la gestion de très longues séquences. Au niveau logiciel, le développement est à compléter au niveau de la configuration du réseau : l'acquisition de l'arbre de défaillance est relativement laborieuse ; un système qui permettrait la reconnaissance des portes à partir du fichier image est envisageable. Après avoir intégré les derniers cas d'utilisation, il faudrait développer l'outil au format serveur-client ou le serveur contiendrait l'outil et serait intégré au PXI et où les clients pourraient être de petits logiciels à intégrer dans un PDA par exemple.

Cet outil ouvre la porte à de futurs outils d'aide à la surveillance où chaque action entreprise par un opérateur de maintenance serait apprise afin de préserver et d'exploiter

tout le savoir faire et l'expérience de ces opérateurs. Nous pouvons penser aussi que les recherches doivent s'orienter non pas sur une seule technique de l'IA par problème mais un ensemble de techniques complémentaires. Ainsi un travail collaboratif entre un réseau neuro-flou tel que nous l'avons spécifié dans ce rapport, associé à un raisonnement à base de cas pourrait améliorer le résultat d'un diagnostic.

# Bibliographie


Aamodt, Agnar et Enric Plaza (1994). Cased based reasoning : Foundational issues,methodological variations, and system approaches. *AI Communications* **7**(1), 39–59.

Aghasaryan, Armen (1998). Formalisme HMM pour les réseaux de Petri partiellement stochastiques : Application au diagnostic de pannes dans les systèmes répartis. Thèse de Doctorat. Université de Rennes I.

Aghasaryan, Armen, Renée Boubour, Eric Fabre, Claude Jard et Albert Benveniste (1997). A petri net approach to fault detection and diagnosis in distributed systems. Technical Report 1117. INRIA. Rennes, France.

Amit, Daniel J. (1988). Neural network counting chimes. *Proceeding National Academy of Science USA* **85**, 2141–2145.

Anglano, Cosimo et Luigi Portinale (1994). B-W analysis : a backward reachability analysis for diagnostic problem solving suitable to parallel implementation. Dans : *Lecture Notes in Computer Science ; Application and Theory of Petri Nets 1994, Proceedings 15th International Conference, Zaragoza, Spain* (Robert Valette, Ed.). Vol. 815. pp. 39–58.

Åström, Karl Johan, John J. Anton et Karl-Erik Årzén (1986). Expert control. *Automatica* **22**(3), 227–286.

Barreto, Guilherme de Alencar et Aluizio Fausto Ribeiro Araújo (2001). Time in self-organizing maps : An overview of models. *International Journal of Computer Research* **10**(2), 139–179.

Basseville, Michèle (1988). Detecting changes in signals and systems - a survey. *Automatica* **24**(3), 309–326.

Basseville, Michèle et Marie-Odile Cordier (1996). Surveillance et diagnostic de systèmes dynamiques : approches complémentaires du traitement de signal et de l'intelligence artificielle. Technical Report 2861. INRIA. Rennes, France.

Benveniste, Albert, Eric Fabre, Claude Jard et Stefan Haar (2001). Diagnosis of asynchronous discrete event systems, a net unfolding approach. Technical Report 4181. INRIA. Rennes, France.

Bergmann, Ralph (1998-2000). *Introduction to Case-Based Reasoning*. University of Kaiserlautern.

Béroule, Dominique (1987). Guided propagation inside a topographic memory. *IEEE : 1st international conference on neural networks, San Diego, CA* **4**(4), 469–476.

Berthold, Michael R. (1994). The tdrbf : A shift invariant radial basis function network. Dans : *Fourth Irish Neural Networks Conference - INNC'94*. Dublin. pp. 7–12.

Berthold, Michael R. et Jay Diamond (1995). Boosting the performance of rbf networks with dynamic decay adjustment.. Dans : *Advances in Neural Information Processing Systems* (Gerald Tesauro, David S. Touretzky et Todd K. Leen, Eds.). Vol. 7. Chap. 5, pp. 521–528. MIT Press. Cambridge, MA.

Bouchon-Meunier, Bernadette (1995). La logique floue et ses applications. Dans : *Vie Artificielle*. 272 p. Addison-Wesley, France.

Bouchon-Meunier, Bernadette et Christophe Marsala (2003). *Logique floue, principes, aide à la décision*. Traité IC2, série informatique et systèmes d'information. Lavoisier.

Boullart, Lucas, Krijgsman, Arie et Vingerhoeds, Rob A., Eds.) (1992). *Application of Artificial Intelligence in Process Control*. Pergamon Press. Oxford.

Brézillon, Patrick (2003). Context-based modeling of operators' practices by contextual graphs. Dans : *Human Centered Processes : 14th Mini Euro Conference, Luxembourg*.

Brusoni, Vittorio, Luca Console, Paolo Terenziani et Daniele Theseider Dupré (1995). Characterizing temporal abductive diagnosis. Dans : *Sixth International Workshop on Principles of Diagnosis (DX'95)*. pp. 34–40.

Brusoni, Vittorio, Luca Console, Paolo Terenziani et Daniele Theseider Dupré (1997). An efficient algorithm for temporal abduction. Dans : *Advances in Artificial Intelligence, 5th Congress of the Italian Association for Artificial Intelligence (AI*IA 97), Rome, Italy*. pp. 195–206.

Cayrac, Didier, Didier Dubois et Henri Prade (1996). Handling uncertainty with possibility theory and fuzzy sets in a satellite fault diagnosis application. *IEEE Transactions on Fuzzy Systems* **4**(3), 251–269.

Chang, Fi-John, Jin-Ming Liang et Yen-Chang Chen (2001). Flood forecasting using radial basis function neural networks. *IEEE Transactions on Systems, Man and Cybernetics - Part C : Applications and Reviews* **31**(4), 530–535.

Chappelier, Jean-Cédric (1996). RST : une architecture connexionniste pour la prise en compte de relations spatiales et temporelles. Thèse de Doctorat. Ecole Nationale Supérieure des Télécommunications.

Chappelier, Jean-Cédric, Marco Gori et Alain Grumbach (2001). Time in connectionist models. *Lecture Notes in Artificial Intelligence Serie* **1828**, 105–134.

Cheng, Yue-Lung (2000). Uncertainties in fault tree analysis. *Tamkang Journal of Science and Engineering* **3**(1), 23–29.

Cockburn, Alistair (2001). *Rédiger des cas d'utilisation efficaces*. Eyrolles.

Dash, Sourabh et Venkat Venkatasubramanian (2000). Challenges in the industrial applications of fault diagnostic systems. Dans : *Process Systems Engineering (PSE 2000)*. Vol. 24 de *Computers and Chemical Engineering*. pp. 785–791.

Deledalle, Gérard (1990). Abduction. Dans : *Encyclopédie philosophique universelle* (Sylvain Auroux, Ed.). Vol. 2 : les notions philosophiques. Presse Universitaire de France.

Dourado, António, Alberto Cardoso, Paulo Gil, Jorge Henriques, Rui Pedro Paiva, Carlos Pereira, Amâncio Santos et José Victor (1999). Intelligent control and supervision at centro de informtica e sistemas da universidade de coimbra. Dans : *XX Jornadas de Automática, Salamanca, Espagne*. pp. 423–426.

Dubois, Didier et Henri Prade (1995). Fuzzy relation equations and causal reasoning. Dans : *Special Issue on "Equations and Relations on Ordered Structures : Mathematical Aspects and Applications (A. Di Nola, W. Pedrycz, S. Sessa, eds.), Fuzzy Sets and Systems*. Vol. 75. pp. 119–134.

Dubois, Didier et Henri Prade (2000). *Fundaments of Fuzzy Sets*. Handbook of Fuzzy Sets Series. Kluwer Academic.

Dubuisson, Bernard (1990). *Diagnostic et reconnaissance des formes*. Traité des nouvelles technologies, série Diagnostic et Maintenance. Lavoisier.

Dubuisson, Bernard (2001). *Diagnostic, intelligence artificielle et reconnaissance des formes*. Traité IC2, série productique. Lavoisier.

Elman, Jeffrey L. (1990). Finding structure in time. *Cognitive Science* **14**(2), 179–211.

Ferariu, Lavinia et Teodor Marcu (2002). Evolutionary design of dynamic neural networks applied to system identification. Dans : *15th Triennal World Congress, IFAC*. Barcelona, Spain.

Frank, Paul M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey and some new results. *Automatica* **26**(3), 459–474.

Grosclaude, Irène (2001). Diagnostic abductif temporel - scénarios de pannes, modèles causaux et traitement de l'information. Thèse de Doctorat. Université de Rennes I.

Grosclaude, Irène et René Quiniou (2000). Dealing with interacting faults in temporal abductive diagnosis. Dans : *International workshop on Principles of diagnosis (DX'2000)*.

Grosclaude, Irène, Marie-Odile Cordier et René Quiniou (2001). Causal interaction : from a high-level representation to an operational event based representation. Dans : *International Joint Conference on Artificial Intelligence (IJCAI), Seattle, Washington, USA*. pp. 565–572.

Jacquemin, Christian (1994). A temporal connectionist approach to natural language. *SIGART Bulletin* **5**(3), 12–22.

Jordan, Michael I. (1986). Serial order : A parallel distributed processing approach. Technical Report ICS Report No. 8604. Institute for Cognitive Science, University of California. San Diego.

Keller, Paul E., R.T. Kouzes et L.J. Kangas (1994). Three neural network based sensor system for environemental monitoring. Dans : *Proceedings IEEE Electro94 Conference*. Boston, MA, USA. pp. 378–382.

Lamontagne, Luc et Guy Lapalme (2002). Raisonnement à base de cas textuels - état de l'art et perspectives. *Revue d'Intelligence Artificielle* **16**(3), 339–366.

Larman, Craig (2002). *Applying UML and patterns : An Introduction to Object-Oriented Analysis and Design and the Unified Process.*. Prentice Hall.

Lefebvre, Dimitri (2000). Contribution à la modélisation des systèmes dynamiques à événements discrets pour la commande et la surveillance. Habilitation à Diriger des Recherches. Université de Franche-Comté/ IUT Belfort.

Limnios, Nikolaos (1991). *Arbres de Défaillance*.

Looney, Carl G. et Lily R. Liang (2002). Inference via fuzzy belief networks. Dans : *the ISCA 15th International Conference on Computer Applications in Industry and Engineering (CAINE'2002)*. San Diego, CA, USA. pp. 25–28.

Looney, Carl G. et Lily R. Liang (2003). Inference via fuzzy belief petri nets. Dans : *the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'2003)*. Sacramento, CA, USA. pp. 510–514.

Malek, Maria (1996). Un modèle hybride de mémoire pour le raisonnement à partir de cas. Thèse de Doctorat. Université J. Fourier.

Meador, Jack, Angus Wu, C.T. Tseng et T.S. Lin (1991). Fast diagnosis of integrated circuit faults using feedforward neural network. Dans : *IEEE International Joint Conference on Neural Networks*. Vol. 1. Seattle, USA. pp. 269–273.

Mellouli, Nédra et Bernadette Bouchon-Meunier (2000*a*). Fuzzy approaches to abductive inference. Dans : *Conference on Non-Monotonic reasoning, Berckenridge, Colorado, USA*.

Mellouli, Nédra et Bernadette Bouchon-Meunier (2000*b*). Modélisation de l'inférence abductive par inversion du modus ponens generalisé. Dans : *LFA'2000, La Rochelle, France* (Cépaudes Edition, Ed.). pp. 79–86.

Mellouli, Nédra et Bernadette Bouchon-Meunier (2003). Abductive reasoning and measures of similitude in the presence of fuzzy rules. *Fuzzy Sets and Systems* **137**(1), 177–188.

Minca, Eugenia (2004). Contribution à la supervision des systèmes de production à l'aide des réseaux de Petri Flous : Application à la e-maintenance.. Thèse de Doctorat. Université de Valahia, Targoviste, Roumanie.

Minca, Eugenia, Daniel Racoceanu et Noureddine Zerhouni (2002). Monitoring systems modeling and analysis using fuzzy petri nets. *Studies in Informatics and Control* **11**(4), 331–338.

Monnin, Maxime (2004). Surveillance et aide au diagnostic en utilisant des techniques de l'intelligence artificielle. Utilisation des réseaux de Petri flous. Mémoire de D.E.A. Université de Franche-Comté.

Monnin, Maxime, Nicolas Palluat, Daniel Racoceanu et Noureddine Zerhouni (2004). Diagnosis methods using artificial intelligence. application of fuzzy petri nets and neuro-fuzzy systems. Dans : *Third Conference on Management and Control of Production and Logistics, MCPL2004*. Santiago de Chile.

Morley, Chantal, Jean Hugues et Bernard Leblanc (2003). *UML pour l'analyse d'un système d'information, 2ème édition*. Dunod.

Nauck, Detlef et Rudolf Kruse (1998). A neuro-fuzzy approach to obtain interpretable fuzzy systems for function approximation. Dans : *IEEE International Conference on Fuzzy Systems, Anchorage*. Vol. 2. pp. 1106–1111.

OMG (2003). *Unified Modeling Language Specification. Version 1.5*. Object Management Group.

Ould Abdeslam, Djaffar (2002). Utilisation des Réseaux Neuro-Flous pour le Pronostic et le Diagnostic : Application à la Classification. Mémoire de D.E.A. Université de Franche-Comté.

Palade, Vasile, Ron J. Patton, Faisel J. Uppal, Joseba Quevedo et S. Daley (2002). Fault diagnosis of an industrial gas turbine using neuro-fuzzy methods. Dans : *15th IFAC World Congress, Barcelona, Espagne*. pp. 2477–2482.

Palluat, Nicolas (2002). Méthodologies Neuronales de Diagnostic des Dégradations. Mémoire de D.E.A. Université de Franche-Comté.

Palluat, Nicolas, Daniel Racoceanu et Noureddine Zerhouni (2004*a*). Diagnosis aid system using a neuro-fuzzy approach. Dans : *Advances in Maintenance and Modeling, Simulation and Intelligent Monitoring of Degradation, IMS'2004*. Arles, France. CDROM.

Palluat, Nicolas, Daniel Racoceanu et Noureddine Zerhouni (2005*a*). An UML modelling of a neuro-fuzzy monitoring system. Dans : *16th World Congress of the International Federation of Automatic Control, IFAC'2005*. Prague, République Tchèque. CDROM.

Palluat, Nicolas, Daniel Racoceanu et Noureddine Zerhouni (2005*b*). Utilisation des réseaux de neurones temporels pour le pronostic et la surveillance dynamique :

Etude comparative de trois réseaux de neurones récurrents. *Revue d'Intelligence Artificielle, RSTI série RIA,* **19**(6), 913–950.

Palluat, Nicolas et Daniel Racoceanu (2004*b*). Conception et prototypage d'un système de surveillance dynamique embarqué et d'aide au diagnostic d'un centre d'usinage à grande vitesse. Technical report. OSÉO anvar - Aide aux Jeunes pour l'Innovation Technologique.

Parizeau, Marc (2004). Le perceptron multicouche et son algorithme de rétropropagation des erreurs. Département de Génie Electrique et de Génie Informatique, Université de Laval.

Pasquier, Laurent (2002). Modélisation de raisonnements tenus en contexte : Application à la gestion d'incidents sur une ligne de métro. Thèse de Doctorat. Université Pierre et Marie Curie.

Peirce, Charles Sanders (1978). Ecrits sur le signe. Dans : *L'Ordre philosophique* (Seuil, Ed.). 262 p. Traduction : Gérard Deledalle. Paris.

Pencolé, Yannick (2002). Diagnostic décentralisé de systèmes à événements discrets : application aux réseaux de télécommunications. Thèse de Doctorat. Université de Rennes I.

Peng, Yun et James A. Reggia (1990). *Abductive inference models for diagnostic problem-solving*. Symbolic Computation. Springer-Verlag New York, Inc.

Pross, Stéphanie (2001). Analyse des Défaillances des Systèmes Industriels : Application au Système de transfert SORMEL du Laboratoire d'Automatique de Besançon. Mémoire de D.E.A. Université de Franche-Comté.

Rahnamai, Kourosh, Payman Arabshahi, Tsun-Yee Yan, Thuan Pham et S. G. Finley (1998). An intelligent fault detection and isolation architecture for antenna arrays. JPL TMO Progress Report 42-132.

Reilly, Doug L., Leon N. Cooper et Charles Elbaum (1982). A neural model for category learning. *Biological Cybernetics* **45**, 35–41.

Roques, Pascal (2002). *UML - Modéliser un site e-commerce*. Les cahiers du programmeur. Eyrolles.

Rumbaugh, James, Ivar Jacobson et Grady Booch (1998). *The unified modeling language reference manual*. Addison-Wesley.

Sampath, Meera, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen et Demosthenis C. Teneketzis (1996). Failure diagnosis using discrete-event models. *IEEE Transactions On Control Systems Technology* **4**(2), 105–124.

Sejnowski, Terry et Charles Rosenberg (1986). Nettalk : A parallel network that learns to read aloud. Technical Report JHU/EECS-86/01. Johns Hopkins University, Electrical Engineering and Computer Science.

Simpson, Patrick K. (1992). Fuzzy min-max neural networks - part 1 : Classification. *IEEE Transactions on neural networks* **3**(5), 776–786.

Simpson, Patrick K. (1993). Fuzzy min-max neural networks - part 2 : Clustering. *IEEE Transactions on Fuzzy Systems* **1**(1), 32–45.

Tromp, Laurent (2000). Surveillance et diagnostic de systèmes industriels complexes : une approche hybride numérique/symbolique. Thèse de Doctorat. Université de Rennes I.

Uppal, Faisel J, Ron J Patton et Vasile Palade (2002). Neuro-fuzzy based fault diagnosis applied to an electro-pneumatic valve. Dans : *15th IFAC World Congress, Barcelona, Espagne*. pp. 2477–2482.

Valette, Robert et Luis Allan Künzle (1994). Réseaux de petri pour la détection et le diagnostic. Dans : *Journées nationales : Sureté, Surveillance, Supervision : détection et localisation des défaillances, GDR Automatique*.

Villemeur, Alain (1988). *Sûreté de fonctionnement des systèmes industriels*. n° 67 Dans : *Collection de la Direction des Etudes et Recherches d'Electricité de France*.

Wan, Yat How, Khalid Marzuki et Ahmad Fuad Syed Zain Syed (1999). Transformer fault diagnosis using fuzzy logic interpretations. Dans : *Instrument Asia Technical Symposium'99*. Singapore.

Wang, Jeen-Shing et C. S. George Lee (2002). Self-adaptive recurrent neuro-fuzzy control for an autonomous underwater vehicle. *IEEE International Conference on Robotics and Automation, 2002* **19**(2), 283–295.

Willsky, Alan S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica* **12**(6), 601–611.

Wu, Angus et Jack Meador (1994). A measurement selection for parametric ic fault diagnosis. *Journal of Electronic Testing : Theory and Applications* **5**(1), 9–18.

Zemouri, Ryad (2003). Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : Application à la e-maintenance. Thèse de Doctorat. Université de Franche-Comté.

Zemouri, Ryad, Daniel Racoceanu et Noureddine Zerhouni (2001). The rrbf : Dynamic representation of time in radial basis function network. Dans : *8th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA'2001*. Vol. 2. Antibes, Juan-les-Pins. pp. 737–740.

Zemouri, Ryad, Daniel Racoceanu et Noureddine Zerhouni (2002). Réseaux de neurones récurrents à fonctions de base radiales : RRFR, application au pronostic. *Revue d'Intelligence Artificielle, RSTI série RIA* **16**(3), 307–338.

Zhang, Qinghua (1999). Identification et surveillance de systèmes dynamiques. Habilitation à Diriger des Recherches. Université de Rennes1.

Zwingelstein, Gilles (1995). *Diagnostic des défaillances - Théorie et pratique pour les systèmes industriels*. Traité des Nouvelles Technologies, série Diagnostic et Maintenance.

*Thèse de Monsieur Nicolas PALLUAT*

*Méthodologie de surveillance dynamique à l'aide des réseaux neuro-flous temporels*

**Résumé :** Notre travail porte sur la surveillance industrielle, processus couramment décomposé en deux phases : la détection et le diagnostic. Nous proposons ainsi un système dynamique d'aide à la surveillance, sous la forme de deux outils exploitant les techniques de l'intelligence artificielle. Le premier réalise une détection dynamique intelligente à l'aide des réseaux de neurones récurrents à fonction de base radiale. Le second, basé sur un réseau neuro-flou, effectue une aide au diagnostic.
A partir de l'observation de données capteurs, l'outil de détection détermine l'état du système en associant un degré de possibilité à chacun des modes de fonctionnement. A partir de ces informations, l'outil de diagnostic recherche les causes les plus probables (diagnostic abductif) pondérées par un degré de confiance. En complément et dans une optique d'aide à la décision, nous avons veillé à ce que l'opérateur puisse ajouter des informations supplémentaires. Notons que la configuration et l'initialisation des outils implique de connaître l'historique et les données de maintenance du système. Nous exploitons pour cela les AMDEC et Arbres de Défaillance des équipements surveillés.
La partie applicative de cette thèse se décompose en deux points : l'intégration logicielle de l'ensemble du travail sur un ordinateur industriel (démarche UML + implémentation) ainsi que l'application sur un système de transfert flexible de production.

**Mots-clés :** réseau de neurone dynamique, réseau neuro-flou, diagnostic, surveillance, maintenance, SCADA, GMAO, AMDEC, Arbre de défaillance, UML.

*Methodology of dynamic monitoring using temporal neuro-fuzzy networks*

**Abstract:** Our work concerns the industrial monitoring, process usually parse into two phases: the detection and the diagnosis. We thus propose a dynamic monitoring aid system based on two tools of Artificial Intelligence technics. The first one used for the dynamic detection is a recurrent radial basis function network. The second one, based on a neuro-fuzzy network, perform the diagnosis aid.
With the sensor data, the detection tool determine the degree of possibility of each operating mode of the system. Given these information, the diagnosis tool searchs causes by an abductive approach, and classify them by a degree of credibility. For the configuration and the initialization of the tools, we used some historic and maintenance data of the system like AMDEC and Fault Tree.
The development part of this thesis is divided into two points: a software integration on an industrial computer (UML approach + implementation) and the application on a flexible production system.

**Keywords:** dynamic neural network, neuro-fuzzy network, diagnosis, monitoring, maintenance, SCADA, CMMS, FMECA, Fault tree, UML.