République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

*Ministère de l'E*nseignement Supérieur et de la Recherche Scientifique

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ORAN Mohamed Boudiaf

Faculté des Mathématiques et Informatique

Département d'Informatique

THESE

Présentée par

BOUGHRARA Asmaa

Pour l'obtention du diplôme de Doctorat

Domaine : Mathématiques Informatique

Filière : Informatique

Option : Systèmes Informatiques et Réseaux (S.I.R)

Thème

Tolérance aux Fautes pour les Systèmes Embarqués

SOUTENUE LE : 12 / 03 / 2015

Devant la commission d'examen composée de :

Qualité	Nom et Prénom	Grade	Etablissement d'origine
Président	Mme. BELBACHIR Hafida	Professeur	U.S.T.O MB
Rapporteur	Mr. CHOUARFIA Abdallah	Professeur	U.S.T.O MB
Examinateur	Mr. BENMOHAMMED Mohamed	Professeur	Univ Constantine2
Examinateur	Mr. BENYAMINA Abou El Hassan	MC-A	Univ Oran 1
Examinateur	Mr. OUSLIM Mohamed	Professeur	U.S.T.O MB

Année universitaire : 2014/2015

Thèse réalisée au Département d'informatique
Faculté de Mathématique et d'Informatique
Université des Sciences et de la Technologie d'Oran
- Mohamed Boudiaf (USTO-MB)

BP 1505 El M'Naouar Bir el Djir 31000
Oran, 31000, Algérie

`http://www.univ-usto.dz/mathinfo/presentation.php`

Présentée par BOUGHRARA Asmaa asmaa.boughrara@univ-usto

Sous la direction de MESSABIH Belhadri messabih@univ-usto.dz
CHOUARFIA Abdallah chouarfia@univ-usto.dz

Étude de cas réalisée dans La société Algérienne de la verrerie ALVER, ORAN

Remerciements

Je remercie Dieu le Tout Puissant de m'avoir donné le courage et la patience de mener à bien cette thèse.

Mes premiers remerciements vont à Mr MESSABIH Belhadri qui m'a encadré durant cette thèse. Je le remercie pour m'avoir fait confiance et m'avoir donné l'opportunité de me lancer dans cette aventure, pour son optimisme légendaire et pour avoir su m'encourager et m'apporter son expérience et son soutien scientifique.

Je remercie Mr CHOUARFIA Abdallah pour avoir accepté d'être mon directeur de thèse pour cette dernière année.

Je remercie également les membres du jury : Mme. BELBACHIR Hafida qui a accepté de présider ce jury, ainsi que Mr. OUSLIM Mohamed, Mr. BENMOHAMED Mohamed et Mr. BENYAMINA Abou d'avoir accepter de faire partie du jury.

Je remercie Mr Tetsuji OGAWA de l'Université de Waseda, Tokyo, Japon pour son aide précieuse et ses conseils qui nous ont permis d'effectuer ce travail.

Je remercie Mme HAMEL Djemila et Mr KARA Kada pour m'avoir reçu au sein de la société Algérienne de la verrerie ALVER et pour leur aide inestimable.

Je remercie Mr Juan Antonio DE LA PUENTE et Peter PRADELY de l'Université Polytechnique de Madrid pour m'avoir reçu et consacré leur temps durant mon stage.

Un grand merci à toute l'équipe du groupe STRAST qui m'a accueilli chaleureusement, pour leur soutien et leur attention.

Merci à Mr ILES Amine pour m'avoir donné l'occasion de faire partie d'un club scientifique ou j'ai pu élargir mes connaissances dans la programmation des micro-processeurs.

Un merci à Mlle JELAIB et OUSSAMNIA Mohamed pour m'avoir aidé pour la traduction de mes articles et leurs précieux conseils.

Un très grand merci au groupe de Facebook « échange article et publication scientifiques », sans lui je n'aurai pu avancer dans ma recherche lors de ma première et deuxième année de doctorat.

Je remercie très très particulièrement mes parents : BOUGHRARA Abdelghani et DINARS Oum El Djilali pour leur amour et leur soutien inconditionnel. De m'avoir encouragé et faciliter la vie. Cette thèse est aussi la leur.

Je remercie mes sœurs : Wafaa, Lamia et Maïssa avec qui j'ai partagé toutes mes larmes et mes joies.

Je remercie mes sœurs de cœur : SENHADJI Sara , OUDDEN Wafaa, SAN AHMED Rania et FEZZA BELHADJ Rim , qui ont crue en moi, ont ressentie mes craintes et m'ont incité à continuer avec leurs précieux conseils.

Je remercie tout les membres de ma famille paternelle et maternelle pour leurs encouragements.

Je remercie ZERKOUK Meriem et MENAD Nadia pour ces années de voisinage, de collaboration et parfois de causerie.

Merci (et surtout bon courage) aux prochains sur la liste : BENKERAMA Soumia, NEZEREG Houwaria, GACEM Sara, OUJEDI Sihem , SILARBI Samiya, ZENAK Chahra, JEDID Nadir, BELBACHIR Redouane et KIES Ali, pour leur amitié, leur compagnie durant toutes ces années de thèse et leur solidarité.

Je finis par un merci à mes enseignants qui m'ont marqué dans ma vie et ont apporté quelque chose pour ma personne : Mme KRIBI (de mon école primaire), Mme ZEROUAL (de mon CEM), Mme MERAKCHI, Mme ZIAD et Mlle BELBACHIR (de mon lycée) et tous mes enseignants de mon université.

Que tous ceux et celles que j'ai oublié n'en reçoivent pas moins ma gratitude.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Cadre Général et Objectifs

Aujourd'hui les systèmes embarqués ont pris une part importante dans notre vie quotidienne. Ils couvrent un large spectre allant du micro-contrôleur jusqu'aux systèmes complexes comme le contrôle du trafic aérien. Ainsi, il est important de prendre en compte les risques potentiels pouvant résulter de leur dysfonctionnement : pertes économiques, dommages à l'environnement, atteinte à la santé et pertes de vies humaines. Les progrès techniques ont influencés les formes d'interaction homme-machine où l'opérateur est de moins en moins impliqué dans des tâches manuelles afin d'intervenir lorsque qu'une panne se présente. En conséquence, les chercheurs portent un intérêt particulier pour le développement de mécanismes de sureté de fonctionnement qui puissent être intégrés et adaptés aux systèmes embarqués.

Lorsqu'un ou plusieurs services remplissant les fonctions du système dysfonctionnent, on dit alors, que le système souffre d'une **défaillance**. Une défaillance est définie comme la déviation du service de son comportement correct. Une déviation se produit lorsqu'une partie ou la totalité d'un ou plusieurs composants du système a été affectée par une erreur, dont cette dernière est causée par une faute.[1]

Les défaillances étant causées par des fautes, la sûreté de fonctionnement a pour objectif de les abroger, pour empêcher qu'elles se produisent. Pour cela, des solutions existent : la prévention des fautes, l'élimination des fautes et la tolérance aux fautes.

Cela dit, un système embarqué n'est jamais totalement sûr. La présence ou l'occurrence d'erreurs est inévitable. Ainsi les techniques de tolérance aux fautes représentent la solution optimale permettant de faire confiance aux systèmes embarqués.

La modélisation de la tolérance aux fautes dans un système passe initialement par un mécanisme de détection d'erreur suivi d'un mécanisme de recouvrement d'erreur. L'originalité d'un mécanisme de tolérance aux fautes se distingue dans le type de méthode de détection et/ou de recouvrement d'erreur. L'objectif de notre travail est de proposer un nouveau mécanisme de détection d'erreur dans le cadre de la tolérance aux fautes pour les systèmes embarqués.

Au cours des dernières années un volume grandissant de mécanismes de tolérance aux fautes est apparu et a apporté des résultats significatifs.

Toutefois la complexité des systèmes embarqués n'a pas cessé de croitre, et les mécanismes de tolérance aux fautes traditionnels ne peuvent pas suivre le rythme dynamique des nouvelles architectures et paradigmes. De plus, il est difficile pour ces mécanismes d'analyser, diagnostiquer et réparer une défaillance dans un laps de temps court (les défaillances peuvent se produire de manière arbitraire). Ainsi, le système doit réagir avant même que la défaillance ne survienne. Cela exige que le mécanisme de tolérance aux fautes soit capable de modéliser le comportement du système et d'évaluer son état d'exécution actuel pour pouvoir prendre une décision aussi rapide que possible.

Cox [2] identifie deux grandes catégories de modèles : les modèles empiriques et les modèles substantifs. Les modèles empiriques cherchent à offrir une représentation raisonnable et un calcul docile des caractéristiques des données observées. D'autre part, les modèles substantifs, sont fondés sur la sémantique du sujet et cherchent à expliquer et modéliser le système, ils permettent d'induire un concept de nature statistique à partir seulement de séquences d'apprentissage. C'est dans les modèles substantifs que notre recherche s'est orientée.

Plusieurs mécanismes d'apprentissage automatique par optimisation ont été développés et largement utilisés afin de modéliser le comportement d'un système. Nous citons : les Réseaux de Neurones Artificiels (RNA), les Arbres de Décisions, les Algorithmes Génétiques, les Séparateurs à Vaste Marge (SVM) et les Modèles de Markov Cachés (HMM). Ces mécanismes d'apprentissage sont des outils puissants qui ont fait leur preuve dans le domaine des systèmes embarqués. Cependant, ils présentent des limites en termes de temps d'exécution et de coût des ressources du système. Les HMM arrivent à répondre à ce double objectif. Ils sont l'outil adéquat pour la prédiction des défaillances dans un système embarqué temps réel.

Dans un tel modèle, une séquence est considérée comme une suite d'observations générée par des états. Le principe des HMM décrit le passage d'un état à un autre état à l'aide de probabilités de transitions et comment un état du HMM peut emmètre un élément de la séquence , à l'aide de probabilité d'observation. Initialement introduit en écologie dans [3] sous le nom de « *Probabilistic Functions of Markov* » , les HMM n'ont été popularisés, par la suite, que par le travail de Rabiner dans [4]. Alors que leur application pouvait se résumer à des recherches basées sur la reconnaissance de la parole, ils ont été appliqués avec succès à une variété de problèmes, allant de la classification automatique de l'électrocardiogramme (ECG) [5], la modélisation du trafic des réseaux [6], la reconnaissance faciale [7], la vidéo surveillance intelligente dans l'informatique pervasive [8] et bien d'autres travaux. Cappé *et al.* dans leur livre « *Inference in Hidden Markov Models* » [9] montrent avec plus de 300

références l'utilisation réussie des HMM dans différents domaines.

Bien entendu, comme tout modèle, les HMM ont été restreints pour résoudre certains problèmes. Ce qui a stimulé l'inventivité des chercheurs pour optimiser ce modèle. Plusieurs variantes existent : Modèle Semi Markov Caché (HSMM), Modèle de Markov Caché Hiérarchique (HHMM) ..... Etc [1].

Toutefois, les HMM standards et leurs variantes présentent un inconvénient majeur : ils ne prennent pas en considération les caractéristiques dynamiques de l'état i.e. la probabilité d'observation dépend de l'état actuel du système et de son état précédent ainsi une observation générée a un **impact** sur l'état actuel et l'état futur du système.

Les ***Partly Hidden Markov Models (PHMM),*** proposés par Kobayashi et Haruyama dans [10], sont une variante des HMM qui permettent d'assurer le conditionnement entre l'état actuel et l'observation précédente. Cette particularité procure aux PHMM un atout substantiel pour la prédiction des défaillances.

Afin de valider cette approche, les PHMM sont appliqués sur un cas réel portant sur un four industriel de production de verre de la société Algérienne de la verrerie ALVER filière du groupe Saint-Gobain. L'industrie du verre est une industrie lourde très concentrée qui dépend de la fiabilité de son four de production pour répondre à la demande croissante. Cependant, le four est sujet à des dysfonctionnements (durée de fonctionnement du four sans arrêt permanent est de 6 à 7 ans) ou même d'un arrêt prématuré dont la réparation pourrait prendre jusqu'à cinq ans. Par conséquent, des compétences pluridisciplinaires en termes d'électronique et d'informatique industrielle sont nécessaires pour prévoir sa défaillance.

Le travail de recherche se synthétise en deux principaux axes :

— Présentation d'une taxonomie sur les mécanismes de tolérance aux fautes développés dans le domaine des systèmes embarqués. Ces mécanismes sont classés en trois catégories : Méthodes conventionnelles, Surveillance du système et Mécanismes de modélisation. Les limites des mécanismes sont expliquées.

— Étude détaillée d'une nouvelle extension des Modèles de Markov Cachés : les *Partly Hidden Markov Models*. Le but dans le cadre de ce travail est d'expliquer ce modèle, à travers une présentation détaillée dans un premier temps, puis par l'implémentation de son algorithme d'apprentissage.

1. L'Annexe A détaille les variantes de HMM

Ce document est divisé en deux parties principales :

Partie 1 présente l'état de l'art en trois chapitres.

Le **premier chapitre** présente les concepts généraux d'un système embarqué temps réel.

Le **chapitre 2** présente la terminologie liée à la sureté de fonctionnement et défini le concept de la tolérance aux fautes. Puis, trace les grandes lignes sur les méthodes de tolérance aux fautes réalisées dans le domaine des systèmes embarqués.

Chaque mécanisme de modélisation et d'apprentissage est présenté brièvement dans le **chapitre 3** appuyé par des travaux connexes. Un intérêt particulier est porté aux HMM. Une discussion détaillée des mécanismes d'apprentissages est présentée par la suite où les limites des HMM est clairement expliquée pour montrer le rôle des *Partly Hidden Markov Models* (PHMM).

Partie 2 définie de manière explicite la modélisation et l'implémentation du modèle proposé en trois chapitres aussi.

Le **chapitre 4** présente notre contexte de travail, les PHMM. Les notions fondamentales sont expliquées : paramètres du modèle et les trois problèmes des PHMM. L'approche proposée est clairement exposée et comment les PHMM interviennent dans le cadre de la prédiction des défaillances.

Le **chapitre 5**, présente l'étude de cas. La verrerie ALVER, l'importance du four et les limites auxquelles il fait face sont expliquées.

Le **chapitre 6** présente la partie évaluation du modèle présenté. Plusieurs expérimentations sont réalisées pour montrer les performances du PHMM.

Cette thèse se termine avec une **conclusion générale** qui résume les points essentiels abordés dans ce travail et présente quelques perspectives à notre travail.

Partie 1 : État de l'Art

Chapitre 1

Systèmes Embarqués Temps Réel

Avec les progrès techniques de ces dernières années dans l'utilisation de systèmes à microprocesseurs, une nouvelle forme de l'électronique basée sur des dispositifs informatiques embarqués a vue le jour. Aujourd'hui, les systèmes embarqués sont de plus en plus variés et ils prennent de plus en plus de place dans notre vie quotidienne. Dans ce chapitre, nous commençons par une définition du concept « Système Embarqué » puis nous abordons, en détail, les caractéristiques et les contraintes que peut avoir ce type de système et nous terminons par une présentation de l'aspect temps réel des systèmes embarqués.

1.1 Systèmes Embarqués

Un système embarqué est une association entre des composants logiciels et matériels fixes ou peu modifiables conçus pour un type d'application bien particulier. En anglais on dit « ***Embedded Systems*** »r.[11]

Certains ouvrages qualifient les systèmes embarqués de **produit intelligent**.

Un système embarqué est composé d'un sous système de contrôle (l'application informatique) qui **agit** sur un système contrôlé (l'environnement physique), on dit que l'application est en **interaction** (forte ou faible selon les cas) avec **son environnement**. Le système de contrôle reçoit les **données** depuis le système contrôlé à l'aide de **capteurs** et il interagit sur l'environnement (envoie de commandes) à l'aide d'**actionneurs** (figure 1.1.1).

L'environnement peut être un moteur, un procédé industriel, un ou plusieurs groupes d'individu, de patients, etc. La nature de l'environnement influence le choix d'action prise par l'application.

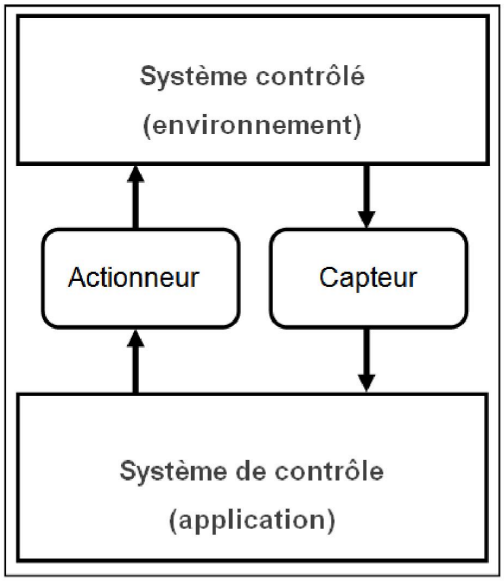


FIGURE 1.1.1 – Architecture Générale d'un Système Embarqué

Une application embarquée est composée d'un ensemble d'actions qui opèrent sur des données en tenant compte des évènements qui apparaissent au niveau de son environnement. Ainsi, les trois concepts (**actions**, **données** et **événements**) sont les composants de base de tout système embarqué. Sachant qu'une **action** peut correspondre à tout traitement ayant une fonctionnalité bien précise au niveau de l'application (ex : une transaction dans les bases de données temps réel, une opération de transmission sur un réseau, un agent dans les systèmes à base de connaissances, etc.). Une **donnée** représente toute information simple ou composée utilisée par une ou plusieurs actions pour s'exécuter. Une mesure de température, un fichier contenant une image ou un paquet véhiculé par un réseau sont des exemples de données. Un **événement** désigne un changement d'état dans le système informatique et/ou dans son environnement. [12]

1.2 Caractéristiques d'un Système Embarqué

Au cours des années qui ont suivi le développement des systèmes embarqués, ces derniers se sont démarqués du système informatique ordinaire par différentes caractéristiques :

— **Dimension limitée** : la taille de ce type de système est une caractéristique majeure. Les fabricants proposent de minimiser l'espace mémoire et/ou la consommation d'énergie pour maximiser la durée d'utilisation.

— **Déterminisme de l'objectif** : la partie logicielle est intégrée dans la structure physique. Elle est conçue apriori pour le système embarqué et possède une fonctionnalité fixe à exécuter. L'application est câblée ou stockée en mémoire en lecture seule (en mémoire morte ROM, FLASH, etc.). On parle alors de *Firmware*. Pour des raisons de fiabilité, elle est testée au moment de la conception. Au début, les systèmes embarqués ont été développés par des ingénieurs qui comprennent l'application mais qui possèdent peu de connaissance en informatique. Pour être rentable en termes de concurrence, les fabricants tentent de minimiser les besoins en ressources au profit de la structure logiciel. Des applications sont intégrées au système pour le rendre plus intelligent. Aujourd'hui, la complexité des applications a conduit à des problèmes de fiabilité et a forcé le concepteur du système d'introduire une architecture logicielle et un système d'exploitation.

— **Environnement** : les systèmes embarqués sont soumis à de nombreuses contraintes causées par l'environnement telles que les chocs, les vibrations, les températures, etc. Cette dimension est la plus étudiée et prise en compte lors de la conception.

— **Communication et Réseaux** : les systèmes embarqués ne sont pas toujours des entités indépendantes, ils sont tenus à s'interconnecter avec un système plus large (communication entre machine et homme-machine). Le protocole de transfert de données doit être simple et robuste.

— **Interface utilisateur** : certains systèmes embarqués ne possèdent pas une interface utilisateur s'ils sont conçus pour une seule tâche. Des systèmes plus simples comportent uniquement des boutons, comme des LED. Toutefois, il existe des systèmes plus complexes qui comportent une interface Homme/Machine très sophistiquée. L'interface est conçue pour l'objectif déclaré et est facile à utiliser (en principe, l'utilisation du produit intelligent devrait être auto-explicatif et ne nécessite aucune formation). Avec l'explosion du web, des fabricants de systèmes embarqués proposent une nouvelle option : une interface au style d'une page web sur une connexion au réseau. Cela permet d'éviter le coût d'un système sophistiqué tout en conservant une interface complète sur un autre ordinateur.[12] [13]

1.3 Domaines d'Applications

Les tendances de l'ingénierie des systèmes embarqués ont présenté des changements majeurs au fil du temps. Initialement, les systèmes embarqués étaient destinés aux applications de contrôle/commande pour l'industrie et aux applications militaires. Aujourd'hui, le développement de l'informatique (la dimension et le coût des processeurs de plus en plus réduit) a permis aux applications

embarquées d'être présentes dans divers domaines [12] :

- **Le secteur de la défense** : le suivi de trajectoires de missiles est un exemple. Ce type de systèmes embarqués doit répondre à de très hautes exigences : la robustesse et la sécurité de l'information transmise.
- **Processus industriel** : il s'appuie sur des systèmes en temps réel. Les systèmes de contrôle utilisés dans le processus de l'industrie sont de bons exemples de grands systèmes embarqués. Des questions telles que la fiabilité et la rapidité d'exécution du support matériel et logiciel sont très importantes. Le contrôleur de température et les automates programmables sont des exemples de tel système.
- **Informatique générale** : il s'agit d'application empaquetée dans un ordinateur personnel : les jeux vidéo, TV Box, ...etc.
- **Systèmes de transport** : le contrôle informatisé est intégré ou embarqué dans pratiquement tous les systèmes de transport : l'automobile [1], l'avion, le train, le navire, les vaisseaux spatiaux, etc. Le but est le contrôle des différents équipements. Le système de transport a généralement des exigences élevées sur la fiabilité. C'est le type de système embarqué le plus répandu.
- **Traitement de signal** : il permet de pré-traiter de grosses quantités de données avant de les manipuler : le radar, suivi de satellites, simulation de vols et pilotage automatique.
- **Secteur médical** : ce secteur comprend les systèmes de suivi des patients, stimulateurs cardiaques, pompes à perfusion et d'autres systèmes pour le diagnostic et le traitement. La majorité de ces systèmes sont très critiques pour la sécurité car ils fonctionnent en contact étroit avec (ou même à l'intérieur) des corps humains. La fiabilité de ces systèmes est vitale.
- **Systèmes de télécommunications** : ils assurent la communication entre individus. Avec la numérisation et l'introduction de techniques de transmission à large bande filaires et sans fil, de nouvelles applications avec des exigences élevées en temps réel sont introduites, par exemple le multimédia. En outre, le système de télécommunication lui-même est un système distribué complexe temps réel avec des exigences élevées en matière de fiabilité et de robustesse. Le PDA, la téléphonie portable et le routeur sont des exemples de ce type de système embarqué.

Quelque soit la diversité de leur domaine d'application, tous les systèmes embarqués doivent répondre à des exigences nécessaires pour leur exécution.

1. Un véhicule peut incorporer de nos jours plus de 50 processeurs, leur coût représente, parfois, plus de 50% du coût de fabrication.

1.4 Contraintes des Systèmes Embarqués

Au cours des dernières années une croissance considérable dans l'utilisation des systèmes embarqués a été observée. En même temps, les attentes sur leur flexibilité à l'exécution sont en constante augmentation. Contrairement aux autres systèmes informatisés, un système embarqué est destiné à réaliser des tâches prédéfinies tout en respectant un cahier des charges et des contraintes dont : le coût, la fiabilité et la contrainte de temps.

1.4.1 Coût

La majorité des systèmes embarqués sont conçus en masse dans des usines d'assemblage. Ceci implique que le coût d'une seule unité de production doit être aussi faible que possible. Ajouter à cela, la concurrence qui existe sur le marché. Aujourd'hui, un client cherche à ne pas payer plus pour une fonctionnalité supplémentaire. Donc, le rapport entre le prix et la performance est scrupuleusement étudié par le fabriquant.

1.4.2 Fiabilité

Un système embarqué est en constante interaction avec l'homme et l'environnement. Les bugs sont inacceptables. Des conséquences désastreuses peuvent résulter (ex : pertes humaines). Ainsi, la fiabilité d'un système embarqué est un point majeur du processus de conception. Ce qui implique :

— L'existence de mécanismes de détection et de traitement de défauts.

— La possibilité de fonctionner en mode dégradé.

— La confidentialité et l'intégrité des données doivent être assurées car des informations à caractère confidentiel peuvent circuler à travers ces systèmes.

— La stratégie d'entretien : de nombreux produits intelligents sont conçus pour être non maintenables, parce que le partitionnement du produit en unités remplaçables est trop cher. Si, toutefois, un produit est conçu pour être maintenu dans le domaine, la fourniture d'une interface de diagnostic et d'une stratégie d'entretien est très importante.

1.4.3 Contrainte de temps

Lorsqu'un système embarqué a une caractéristique liée à l'existence de **contraintes temporelles** dont il faut tenir compte alors nous parlons d'un **Système Embarqué Temps Réel** (***Embedded Real Time System***). Cette contrainte peut être définie comme : une échéance, intervalle de temps ou fenêtre temporelle.

La communauté informatique s'accorde aujourd'hui à ce que la définition du temps réel ne signifie pas nécessairement la rapidité à donner une réponse, mais plutôt, le respect du temps de traitement car le processus d'exécution est considéré comme un phénomène dynamique. Ainsi un système temps réel est un système dont la réponse qu'il donne dépend non seulement de son exactitude logique, mais également du temps. [12]

Aujourd'hui, la majorité des systèmes embarqués sont des systèmes temps réel. Soit 99% des processeurs fabriqués et destinés pour le temps réel, sont utilisés dans les systèmes embarqués. [14]

Un système embarqué temps réel est généralement en interaction avec l'environnement à travers ses capteurs. À l'arrivée de chaque stimuli en provenance de l'extérieur, le système doit répondre en un temps borné en actionnant les ressources concernées (matérielles ou logicielles). Selon l'importance accordée aux contraintes temporelles, on distingue deux types de système embarqué temps réel :

1) Temps Réel Strict ou Dur : aucun dépassement dans les contraintes temporelles est toléré même dans la pire des situations d'exécution possibles. C'est le cas des systèmes critiques comme le pilotage automatique d'avion, le contrôle d'une centrale nucléaire ou une fusée. [13]

Exemples :

— Le système de contrôle d'atterrissage d'avions est considéré comme étant un système à contrainte temporelle stricte. Si l'opération d'atterrissage de l'avion commencée n'est pas achevée au bout d'un délai borné, un accident grave peut survenir.

— Certains systèmes de supervision d'installations industrielles sont considérés comme des systèmes à contraintes strictes. Par exemple, des pièces fabriquées sur une chaîne de production sont déposées sur un convoyeur et passent devant un contrôleur (un expert ou une caméra) pour détecter certaines anomalies de fabrication. La détection des anomalies doit se faire pendant que la pièce soit en mouvement. Si la détection ne s'est pas faite dans les temps, la pièce sera dirigée sur un autre convoyeur pour un recyclage. S'il y a beaucoup de recyclage alors la chaîne de production s'en trouve ralentie. [12]

2)Temps Réel Souple ou Mou : ce type de système peut accepter des exceptions pour arriver à un traitement dégradé. Il tolère un non-respect des échéances avec certaines limites telles qu'un certain pourcentage de résultat ou une certaine fréquence. [13]

Exemple :

Un système de téléconférence est un système à contrainte temporelle souple. Les images doivent être synchronisées mais des dérives de synchronisation, dues à des messages tardifs, sont souvent tolérées. [12]

Remarque :

L'intérêt scientifique et économique liés aux systèmes embarqués temps réel est grand. Ce qui a conduit, depuis des années, à un déluge de mécanismes développés garantissant une sûreté de fonctionnement pour le temps réel et l'embarqué.

1.5 Sûreté de Fonctionnement des Systèmes Embarqués Temps Réel

Les systèmes embarqués temps réel sont dits **Critiques** car le respect des contraintes de temps et la qualité de service rendue est une exigence absolue en toute circonstance y compris les situations de surcharge des ressources du système (matérielles et logicielles). Cette propriété est définie comme la prédictibilité du comportement du système. Certains fabricants et/ou consommateurs exigent une prédictibilité absolue, d'autres tolèrent un certain seuil en dessus duquel ils admettent que la qualité de service rendu est acceptable.

Il convient donc, lors de la mise en œuvre d'un système embarqué temps réel, de s'assurer qu'il satisfera la propriété de sûreté de fonctionnement : « L'absence de défaillance grave ».

Les efforts en matière de sûreté de fonctionnement des systèmes informatiques ont porté sur les fautes logicielles et matérielles. Des solutions potentielles existent [1] :

— **La prévention des fautes** fut le premier mécanisme développé, le principe est empêcher l'occurrence ou l'introduction de fautes.

— **L'élimination des fautes** cherche à éliminer les fautes susceptibles de déclencher la défaillance du système.

— **La tolérance des fautes** empêche qu'une erreur se propage jusqu'à l'état défaillant.

Un système embarqué temps réel dépend fortement de son mécanisme de sûreté de fonctionnement. Notre travail porte sur la tolérance des fautes pour les raisons suivantes :

— La prévention de fautes concerne la mise en œuvre d'un ensemble de processus visant à maitriser la conception du système. Mais il est impossible de concevoir des systèmes parfait sans erreur.

— L'élimination des fautes n'est pas aussi un mécanisme judicieux puisque l'usure des systèmes embarqués est inéluctable à cause de l'environnement qui est souvent agressif.

1.5.1 Exigences requises

Les systèmes embarqués temps réel sont généralement modélisés par des modèles orientés-états qui mettent l'accent sur le contrôle et les aspects réactifs du système en capturant l'effet des évènements externes provenant de l'environnement sur les états du système. Ces modèles attachent une importance particulière aux questions temporelles.

L'industrie des systèmes embarqués temps réel est caractérisée par une demande croissante de composants matériels moins chers et plus puissants. La taille de ces composants est limitée, l'ajout de nouveau composant pourra compromettre, dans l'ensemble, le fonctionnement du système embarqué. Par conséquent, le mécanisme de tolérance aux fautes doit être intégré au sein du système de manière souple en ayant juste un peu d'effet sur l'architecture du système.

L'objectif fixé dans cette thèse est de proposer un nouveau mécanisme de tolérance aux fautes pour un système embarqué temps réel qui doit respecter deux critères importants : le temps et les ressources limitées.

1.6 Conclusion

Nous avons choisi de commencer cette partie de l'état de l'art par une brève présentation des systèmes embarqués temps réel pour montrer leur importance dans notre vie quotidienne. Nous avons abordé les principales définitions propres à ce type de système. Nous avons signalé l'aspect important de leur fiabilité.

Dans le chapitre suivant, nous aborderons en détail les mécanismes de tolérance aux fautes. [12] et [15] approfondissent le domaine des systèmes embarqués.

Chapitre 2

Techniques de Tolérance aux Fautes

Loin de l'exhaustivité, notre objectif est de tracer les grandes lignes des mécanismes de tolérance aux fautes développés jusqu'à l'heure actuelle et qui sont applicables à différents types de système embarqué.

2.1 Chaîne de Causalité

Un système embarqué est **défaillant** lorsque le service qu'il délivre diffère du service attendu [24]. La défaillance est détectée quand un évènement inattendu se produit i.e. une erreur conduisant à un service incorrect. Une erreur est souvent causée par une ou plusieurs fautes. Cette chaîne de causalité est schématisée dans la figure 2.1.1 :

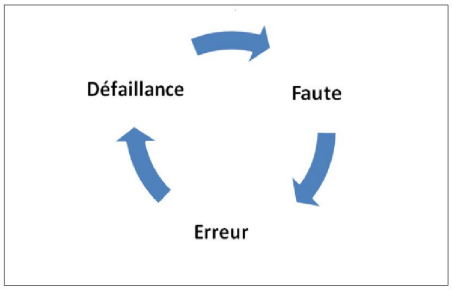


FIGURE 2.1.1 – Chaine de causalité

Ainsi :
— **Une faute** (*Fault*) est un événement ou action pouvant provoquer une erreur.
— **Une erreur** (*Error*) est partie de l'état du système qui est susceptible d'entraîner sa défaillance.

— **Une défaillance (*Failure*)** est une déviation du comportement du système par rapport à son comportement correct ou normal défini à priori pendant sa conception. [1]

Exemple :

Le problème de violation de mémoire arrive à bien expliquer la chaîne de causalité : Si une partie du système d'exploitation est victime d'une fuite de mémoire mais il reste de l'espace, aucune dégradation n'est observée. Cependant, lorsqu'il ne reste plus de mémoire alors l'ordinateur commence à se ralentir et une ou plusieurs erreurs seront observées et ceci conduira à une défaillance du système voire même de l'ordinateur [16].

Chaque système embarqué est unique dans son aspect architectural ainsi il **défaille de manière différente** et appelle un traitement distinctif. La section suivante décrit les différents modes de défaillance.

2.2 Classification des Défaillances

Un système, par rapport à lui-même ou un autre système, ne défaille pas toujours de la même façon, ce qui conduit à la notion de **mode de défaillance** [1]. C'est pourquoi il faut faire appel à une méthode rigoureuse et préventive visant à recenser les défaillances potentielles susceptibles d'affecter le système. La classification des défaillances peut être effectuée suivant différents critères :

— **Critère Gravité** (ou conséquence d'une défaillance) : la gravité d'une défaillance peut prendre plusieurs aspects : la sécurité de l'utilisateur, la perte de fonctionnalité, la dégradation de la qualité, etc. Ce contexte déterminera le niveau à partir duquel on va commencer à observer les effets de la défaillance. Par exemple : une défaillance est considérée comme « inacceptable », si elle conduit à un arrêt complet du système. Une défaillance « acceptable » est une défaillance bénigne si le système peut continuer son service et l'utilisateur le considère comme fiable.

— **Critère Fréquence** : chaque défaillance est classée suivant le nombre de fois de son apparition. L'estimation de la fréquence d'une défaillance n'est pas une tâche aisée. On doit s'appuyer généralement sur des statistiques (si l'on possède un large historique) ou sur des informations apportées par le fournisseur. [17]

— **Critère de Détectabilité** : ce critère est caduc car il faut attendre la signalisation de la défaillance à l'utilisateur. Dans certain cas une défaillance peut être non-détectable et les risques d'accidents humains sont éventuels.[1]

Remarque :

Naturellement, cette classification n'est qu'une proposition de Avizienis *et al.* [1] et de Bazin [17], que chaque groupe de travail peut adapter à son besoin,

son environnement, son objectif. D'autres classifications de défaillance ont été publiées dans Salfner *et al.* [16] et Cristian *et al.* [18].

De ce qui vient d'être présenté, il est fondamental, pour un système embarqué de disposer d'un mécanisme lui garantissant son bon fonctionnement. Appelés mécanismes de sureté de fonctionnement, ils préviennent des défaillances en testant l'état du système soit de manière explicite (exprimer et vérifier des propriétés spécifiques de l'état), soit de manière implicite (via des conséquences visibles : violation d'un délai de garde, violation d'une protection de mémoire, etc.) [16].

La plupart des systèmes embarqués sont critiques et ils doivent pouvoir remplir leurs fonctions malgré la présence de fautes.

2.3 Tolérance aux Fautes

La tolérance aux fautes est aujourd'hui un domaine informatique relativement mûr, s'appuient sur un imposant corpus de résultats théoriques et expérimentaux. Nous présentons dans cette section, une taxonomie globale des mécanismes de tolérance aux fautes. Notre but est d'une part d'unifier le cadre méthodologique et de présenter de manière cohérente les techniques appliquées dans le domaine des systèmes embarqués et d'autre part de situer notre approche entre ces différentes techniques. Dans [1], la tolérance aux fautes est définie comme « *l'aptitude d'un système informatique à accomplir sa fonction malgré la présence ou l'occurrence de fautes, qu'il s'agisse de dégradations physiques du matériel, de défauts logiciels, d'attaques malveillantes où d'erreurs d'interaction homme-machine* ».

Les méthodes de tolérances aux fautes ne s'adressent pas au système dans son ensemble mais plutôt elles divisent le processus en deux principales étapes (figure 2.4.1) : la détection d'erreurs et le traitement des erreurs [1].

2.3.1 Détection d'erreur

Le but est d'identifier, au moyen d'un mécanisme, un état incorrect du système. Suivant le moment où le mécanisme de détection est invoqué, il existe deux catégories :

1. **La détection préventive :** elle a lieu au moment où l'activité du système est minime comme les phases de démarrage ou de maintenance par exemple. Le principe consiste à vérifier des propriétés à l'aide de test. Cependant, on ne peut pas prévoir toutes les anomalies.
2. **La détection concurrente :** elle a lieu pendant l'exécution normale du système. Elle a pour objectif de détecter la présence d'une erreur afin d'y remédier. [19]

Ceci ne nous dit pas comment le mécanisme arrive à trouver l'état erroné. La section suivante présente une étude détaillée des méthodes de détection d'erreur existantes.

2.3.2 Traitement d'erreur

Le traitement d'erreur est la transition du service incorrect à un service correct [25]. Suivant le type d'architecture du système il existe différents mécanismes traitant les erreurs :

2.3.2.1 Recouvrement (*Error Recovery*)

Le système est dirigé dans un nouvel état, connu à priori, exempt d'erreur à partir duquel il peut continuer son fonctionnement. Suivant le mode de transition, nous identifions deux types de recouvrement :

1)Reprise (*Backward Recovery*)

Le système est ramené dans un état ancien enregistré (figure 2.3.1). La sauvegarde périodique de l'état du système doit s'effectuer au moyen d'un mécanisme de mémorisation dans un support stable qui protège les données contre les conséquences d'une erreur. C'est un mécanisme qui est coûteux en temps et en espace [19] [25].

Les serveurs NonStop de Tandem [20] commercialisés sous la marque HP et qui occupent toujours une très large part du marché des systèmes transactionnels critiques est un exemple de recouvrement par reprise. Dans le cas d'une défaillance, le processeur reprend son activité au dernier point de reprise.

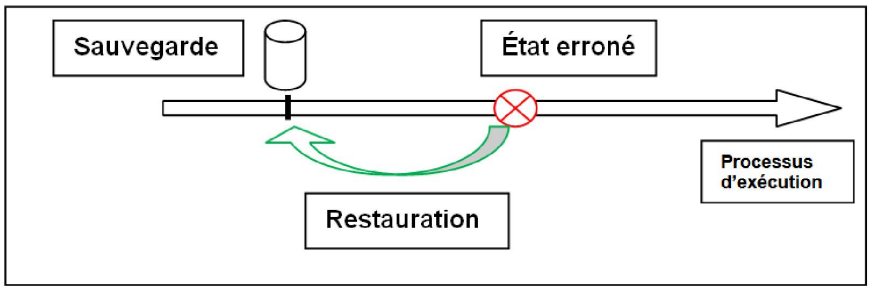


FIGURE 2.3.1 – Recouvrement par reprise

Mejia [21] propose que le recouvrement des erreurs soit un processus en deux étapes : 1) Allocation du temps pour le recouvrement. 2) Ordonnancement des opérations de récupération.

Cependant, ce type de recouvrement est incompatible avec des systèmes temps réel strictes. Également dans le cas ou les points de continuation sont

affectés par une ou plusieurs erreurs [22]. Ainsi une approche alternative existe : le rétablissement par poursuite.

2)Poursuite (*Forward Recovery*)

Le système continue son fonctionnement dans un nouvel état, le plus souvent, de manière dégradée (figure 2.3.2). Le choix de cette technique dépend du type de système et nécessite une analyse préalable des types d'erreurs possibles [1].

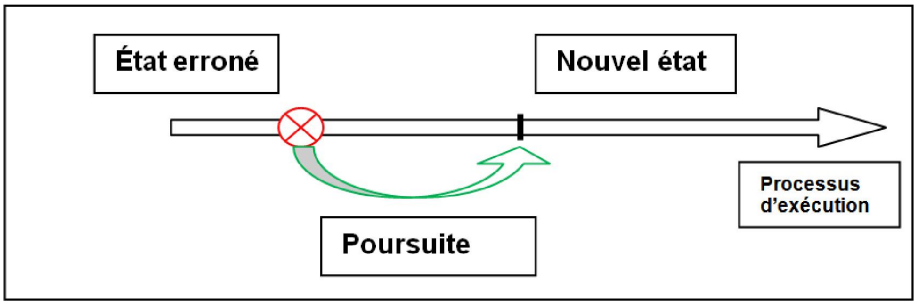


FIGURE 2.3.2 – Recouvrement par poursuite

La communication par paquets est un scénario qui illustre ce type de confinement. Si un paquet est perdu, le capteur peut poursuivre sa tâche en reconstituant totalement ou partiellement l'information manquante en utilisant les paquets suivants et précédents [23].

Cependant, le recouvrement par poursuite peut être inadéquat dans le cas d'un système embarqué qui ne supporte pas la perte de donnée ou dans le cas où le point de poursuite n'a pas été défini [18]. Un dilemme existe, c'est la **compensation**.

2.3.2.2 Compensation (*Error Masking*)

Le système possède une redondance interne suffisante pour masquer (rendre invisibles aux utilisateurs) les conséquences d'une erreur et poursuivre l'exécution. Les composants redondants exécutent le même traitement ; en cas de défaillance de l'un d'entre eux, il est déconnecté et le traitement se poursuit sans interruption sur les autres. La compensation, dans ce cas, se limite à une commutation éventuelle de composants. [25]

L'architecture du système de gestion de commandes de vol des Airbus 320/330 /340, repose sur cette technique pour le traitement d'erreur [25]. De même pour le système de contrôle-commande du trafic ferroviaire de la gare Termini de Rome développé par Ansaldo Segnalamen to Ferroviario [26].

Toutefois, la gestion des redondances dans un système embarqué de manière transparente sans compromettre son architecture reste une tâche ardue malgré la présence de quelques travaux de recherche tel que : Tambe *et al.* [27] qui

proposent le GRAFT (*GeneRative Aspects for Fault Tolerance*) : une approche basée sur les aspects pour modifier des *middlewares* en intégrant une technique de tolérance aux fautes basée sur la redondance des composants. L'approche est réalisée en deux principales étapes : 1) la duplication des composants (2) la duplication de leurs interconnexions afin que les connexions nécessaires peuvent être établies au moment du déploiement de la réplique.

Remarque 1

Dans le cas d'une nouvelle ré-activation d'erreur après un recouvrement (le cas d'une faute persistante), un traitement d'erreur, plus radicale s'impose : **le traitement de faute** [24]. Ce dernier impose au début un diagnostic pour évaluer le degré de propagation de l'erreur, l'importance des dommages causés et identifier la cause de l'erreur suivi d'une des dispositions suivantes [1] :

— **Isolation** : le but est d'empêcher un composant contenant une faute persistante d'être de nouveau réutilisé ainsi la faute devient dormante [19]. Ce type d'agencement a été appliqué par Zajac [28] pour optimiser la fiabilité des puces multi-cœurs massivement défectueuses fabriquées à partir de transistors nanométriques. Le traitement consiste à désactiver les cœurs défectueux.

— **Reconfiguration** : le composant fournissant le service est entièrement remplacé [19]. Généralement, elle accompagne le mécanisme d'isolation.

— **Réinitialisation** : cette approche est simple, consiste à acquérir un nouveau contexte à partir de l'environnement, par exemple : relecture des capteurs dans un système de contrôle-commande. [1]

Remarque 2

Le traitement d'erreur ne peut être efficace que si la détection d'erreur a été correctement exécutée. De nombreux mécanismes de détection d'erreur existent, ils se différencient suivant le type de problème qu'ils traitent. Un des principaux intérêts de ce chapitre est d'énumérer le plus possible ces mécanismes. Nous estimons que la prochaine section est un des points importants abordés dans cette thèse.

2.4 Méthodes de Détection d'Erreur

La tolérance aux fautes est un domaine d'expert mur qui existe depuis plus de vingt-cinq ans. Ainsi les techniques développées jusqu'à l'heure actuelle existent par centaines voir par milliers. Pour capturer la variété de toutes ces approches, surtout concernées par le domaine du système embarqué, une taxonomie est présentée dans cette section. Les différentes approches sont regroupées en trois principaux groupes(figure 2.4.1). Les principaux concepts sont décrits en détail.

2.4.1 Méthodes Conventionnelles

La détection d'erreur est basée sur une redondance, les formes les plus couramment utilisées sont les suivantes :

2.4.1.1 Code détecteur d'erreur (Codage)

Les codes correcteurs d'erreurs sont utilisés dans les transmissions de données informatiques (câbles coaxiaux, fibres optiques, ondes hertziennes, etc) pour éliminer les effets du bruit. Le mécanisme le plus simple est l'utilisation d'un « bit de parité ». Chaque donnée est découpée en bloc de n bits. Un bit supplémentaire est ajouté de telle sorte que le nombre total de bits égaux à 1 dans le bloc soit pair. Si, à la réception, le nombre de bits dans le bloc est impair alors une erreur est détectée. Cependant ce type de mécanisme ne corrige pas les erreurs [29].

Un mécanisme plus compliqué permet de corriger une erreur dans un bloc, c'est le « code de Hamming [1] » [30]. Un bloc de données de k bit est codé en ajoutant un ensemble de m bits de correction. Ces bits rajoutés sont calculés à l'aide de la matrice génératrice et il sont entrelacés avec la donnée initiale de façon à obtenir un mots de code de n bits. Le code envoyé est comparé au code reçu. La « distance de Hamming » entre les deux mots codés est définie par le nombre de positions binaires qui diffèrent entre ces deux mots. La capacité de correction d'un code est définie par la ou les bits erronés qu'il est capable de corriger.

Cependant les codes détecteurs d'erreur tentent de détecter des erreurs causées par des fautes physiques. [1]

2.4.1.2 Duplication et comparaison (Le vote)

Plusieurs répliques (minimum trois) d'un même composant exécutent le même traitement. Une erreur est détectée lorsqu'au moins une des copies fournit un résultat distinct des autres, on parle alors d'un composant auto-testable

1. Cette technique est développée par Richard Wesley Hamming. Il est considéré comme l'un des pionniers de l'informatique dans la discipline de la tolérance aux fautes

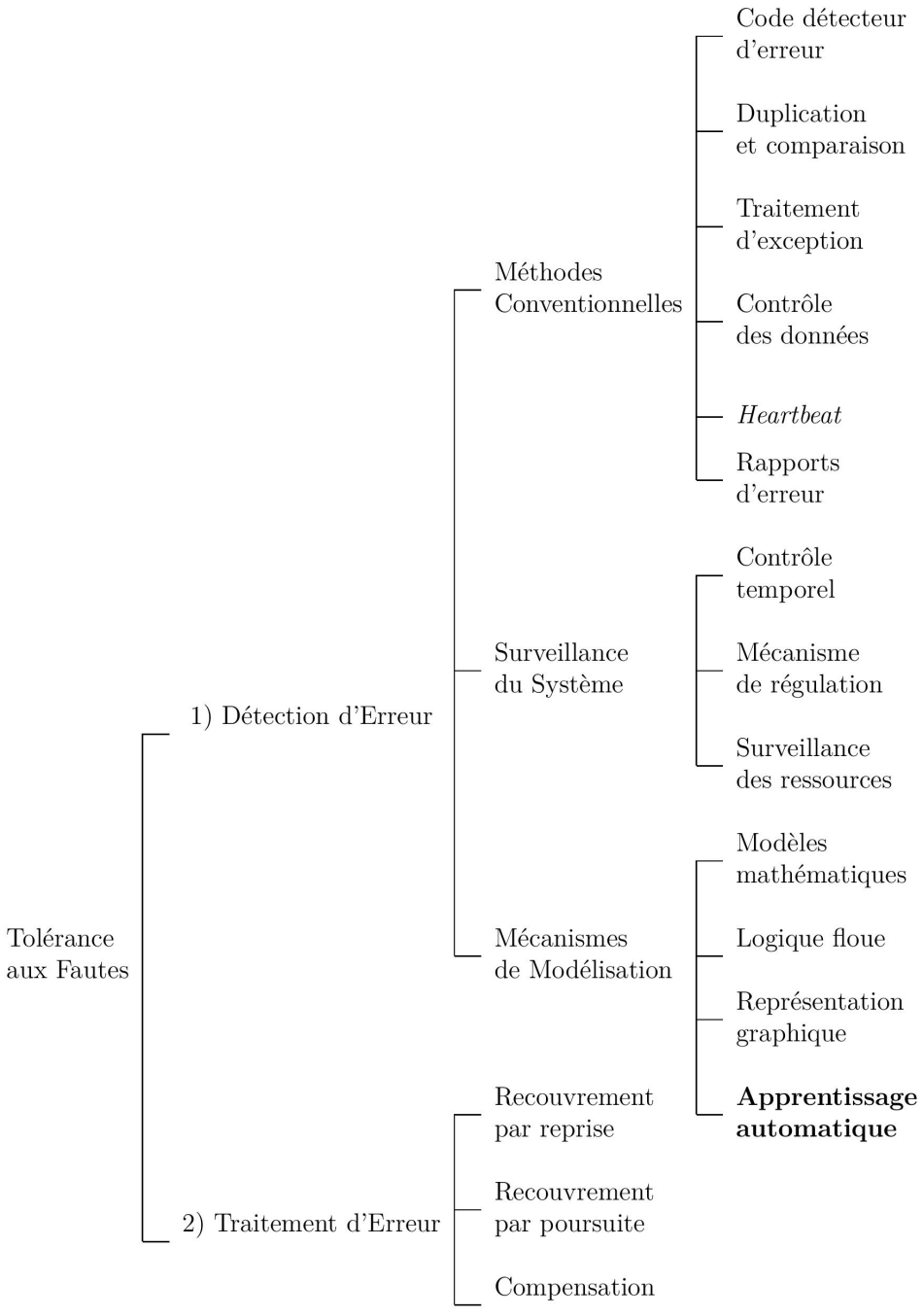


FIGURE 2.4.1 – Techniques de Tolérance aux Fautes dans les Systèmes Embarqués

[1]. Présent dans certains véhicules, un composant auto-testable est obtenu en ajoutant un module de contrôle (*checker*) au composant fonctionnel [42]. En raison de sa simplicité de mise en œuvre, la duplication et la comparaison est un moyen de détection très utilisé dans les systèmes embarqués. Afonso *et al.* [31] proposent d'introduire plusieurs mécanismes de détection d'erreur, dans un Radar, basés sur la duplication. Appelé *N-Version Programming* (NVP), est utilisé pour la détection d'erreur logicielle. Les auteurs soulignent le fait que cette redondance impose un cout supplémentaire au niveau de l'utilisation du CPU, soit plus de 16% par rapport à un système sans mécanisme de détection à base de redondance.

2.4.1.3 Traitement d'exception

Cette technique est plus connue sous le nom de « programmation défensive ». Le but est de protéger le code à l'aide de la notion « d'exception » [1]. Le principe de la gestion d'exceptions a été étudié sérieusement par Goodenough dans [32]. Dès qu'une erreur se produit (exemple : manque de mémoire, calcul impossible, un fichier inexistant, un transtypage non valide,...), le logiciel « déclenche une exception ». Actuellement, les exceptions font partie du langage de programmation. C'est le cas du langage Java qui intègre les exceptions comme une classe particulière : la classe « Exception » [33]. Bouchenak *et al.* [34] proposent que chaque composant logiciel (Servlet Java) traitant une requête sur un serveur J2EE est associée à une exception Java pour la détection d'erreur. Le traitement est réalisé par le bloc catch. Bucchiarone *et al.* [35] proposent de modéliser l'architecture d'un système de contrôle de l'exploitation minière via UML et d'utiliser des exceptions pour le traitement d'erreur. Cependant, les traitements d'exception sont réservés à un ensemble de tâches critiques précédemment déterminées amenant le système à un état où le processus est arrêté.

2.4.1.4 Contrôle des données

Il existe deux types de contrôle : contrôle de vraisemblance et contrôle de données structurées.

Si le mécanisme vérifie la conformité des entrées et/ ou sorties du système pour détecter des erreurs comme une adresse mémoire inexistante ou une violation de protection de segments mémoire, on parle de contrôle de vraisemblance [1]. Il est appliqué dans l'architecture E-GAS pour le contrôle des moteurs essence ou diesel chez BMW, Daimler Chrysler, VolksWagen, Porsche, Audi [36]. Le domaine de la robotique fait aussi appel à ce type de détection d'erreur par exemple le RoboX pour contrôler la vitesse maximale du système et les robots médicaux Hippocrate et SCALPP [37].

Si la détection porte sur l'intégrité sémantique des données ou sur l'intégrité structurelle de la donnée alors elle est nommée : le contrôle de données struc-

turées [1]. Elle est particulièrement applicable à des données structurées dont les différents éléments sont liés par des pointeurs.

2.4.1.5 Heartbeat

Basé sur le principe des battements de cœur. Des messages sont périodiquement envoyés à l'unité de contrôle (à un serveur dans le cas de réseau de communication). Une erreur est détectée si la fréquence de l'envoi change. Ce mécanisme a été initialement développé pour des réseaux informatiques tolérants aux fautes pour une solution orientée entreprise [38]. Duong *et al.* [39] propose le *Heartbeat* afin de détecter le crash du gestionnaire de données. Cet élément est inséré dans l'architecture sous la forme d'une interface « Serveur HeartBeat ». Ce concept est adopté dans le monde de l'automobile : Kim *et al.* [40] proposent d'améliorer la robustesse de l'architecture SAFER (*System-level Architecture for Failure Evasion in Real-time Applications*). Cette architecture a été proposée pour le système de cyber-physiques (CPS) pour la construction de voiture de conduite assistée. L'architecture est composée de plusieurs nœuds de calcul et une bibliothèque de tâches. Cette dernière propage des signaux électriques au niveau de chaque nœud. Quand il y a des signaux consécutives manquants au niveau d'un nœud alors ce dernier est déclaré comme défaillant et il est remplacé par un nœud de secours.

2.4.1.6 Rapports d'erreur

Les rapports d'erreurs peuvent être analysés à l'aide d'un logiciel de test statique. Bien que ce type de mécanisme tarde pour réagir lorsqu'il détecte une ou plusieurs erreurs, il fait partie de plusieurs travaux de recherches. Bauer *et al.* [41] cherchent à optimiser la fiabilité d'une Unité de Commande de Miroir (MCU) d'une voiture. Cette unité a été testée dans différents scénarii de défaillance. Tous les signaux possibles sont envoyés en entrée à la MCU pour obtenir et analyser tous les signaux de sortie possibles. Les auteurs utilisent la bibliothèque JUMBL de classes Java pour l'analyse statique des résultats des tests.

Remarque 1

Pour chaque type d'erreur, il existe un mécanisme qui peut la détecter au mieux. Partant de ce principe, des travaux de recherches proposent d'implémenter plusieurs types de mécanismes de détection d'erreur dans la même architecture du système.

Lu [42] propose une architecture qui repose principalement sur le principe de « réflexivité multi-niveaux ». Différents mécanismes de détection d'erreurs sont implémentés dans le système. Chaque mécanisme traîte un type d'erreurs. Le mécanisme de codage est appliqué pour la détection des fautes logicielles et le contrôle de vraisemblance pour les fautes physiques.

Remarque 2

Nous qualifions, ces mécanismes présentés, comme mécanismes conventionnels car ils représentent des limites que nous résumons en trois principaux points :

— Pour le mécanisme de duplication et comparaison, malgré une mise en œuvre simple, il repose sur l'hypothèse que les unités redondantes sont indépendantes vis-à-vis du processus de création et d'activation des fautes i.e. les fautes sont créées ou activées indépendamment dans les copies. Cependant, dans certains scénarii, un composant redondant peut être affecté par une faute [1].

— La forme la plus sophistiquée de détection d'erreur de ces mécanismes est basée sur une redondance : redondance au niveau des informations, des composants ou algorithmique. Ainsi la détection impose un coût supplémentaire.

— Ces mécanismes ne déclenchent le recouvrement d'erreur que lorsqu'ils détectent une ou plusieurs erreurs alors que certains types d'erreurs affectent le bon fonctionnement du système avant même qu'elles ne soient détectées.

Devant ces limites, d'autres techniques existent, elles sont basées sur un autre concept, celui de l'analyse périodique du système.

2.4.2 Surveillance du Système

Le principe est d'observer l'activité du système en temps réel et de signaler la présence d'une panne. Sachant que la surveillance est effectuée de manière totalement externe sans modification de l'architecture de base du système. Nous identifions trois principales approches :

2.4.2.1 Contrôle temporel

La technique du *watchdog* [1], grâce au faible coût qu'elle impose, est un mécanisme souvent utilisé pour la détection d'erreur en temps réel particulièrement dans la robotique. Le principe est d'ajouter un nouveau composant permettant de surveiller le temps de réponses des autres composants du système. Une erreur est détectée, si au moins un composant dépasse un certain délai (*time-out*). Cette technique permet de détecter une situation de blocage, une boucle infinie ou même l'arrêt du processeur distant. Liu *et al.* [43] proposent une nouvelle architecture ORTGA (*On-demand Real-Time GuArd*) basée sur le contrôle temporel pour optimiser le contrôleur de Feedback.

2.4.2.2 Mécanisme de régulation

Le régulateur (*Feedback*) est un mécanisme qui règle le fonctionnement d'un système physique. Le mécanisme mesure l'état actuel du système, détermine à quel point l'état courant est loin de l'état désiré, puis applique automatiquement un signal de contrôle pour rapprocher le système à l'état désiré. Ce processus est répété de manière itérative pour maintenir le système dans l'état désiré. Le *Feedback* peut être utilisé pour stabiliser l'état d'un système, tout en améliorant sa performance. Il est omniprésent dans l'ingénierie i.e. il fait partie de l'architecture des appareils et des machines électroniques : appareil de climatisation, régulateur de vitesse dans l'automobile, des avions militaires et microscope à force atomique. Les régulateurs sont des appareils électroniques dans lesquels un algorithme est implanté. Un des plus populaires de ces algorithmes est le **proportionnel-intégral-dérivé (PID)** [44].

2.4.2.3 Surveillance des ressources

Crowell *et al.* [45] ont montré que les paramètres des systèmes liés à la mémoire du système présentent des caractéristiques de « vieillissement de logiciel ». Le système S.M.A.R.T [46] (*Self-Monitoring and Reporting Technology*) (Technique d'Auto-surveillance, d'Analyse et de Rapport) est un parfait exemple de mécanisme de surveillance. Il permet de faire un diagnostic de l'état d'un disque dur selon plusieurs indicateurs ,telles que : lecture du taux d'erreur, taux du Seek Error, nombre de secteurs ré-alloués, ... etc., afin d'anticiper les erreurs. Aujourd'hui la plupart des disques durs et des cartes mères fabriqués intègrent ce type de mécanisme.

Remarque

Les mécanismes de surveillance malgré qu'ils aient apportés des résultats signifiants, ils coûtent en temps et en ressources. Ce type de mécanisme se concentre principalement sur l'aspect matériel du système.

Sachant bien que les systèmes embarqués sont caractérisés par leurs ressources limitées. Une nouvelle catégorie de mécanisme propose de construire un modèle qui capte les principaux aspects du comportement du système puis vérifie, lors de l'exécution, si le comportement du système réel ne s'écarte pas du comportement normal.

2.4.3 Mécanismes de Modélisation

Ces dernières années, les travaux dans le domaine de la tolérance aux fautes se sont portés sur l'adaptation du mécanisme de détection d'erreur dans les systèmes embarqués par le biais d'un raisonnement logique fondé sur une évaluation du système. Cette démarche est particulièrement importante dans les systèmes embarqués critiques. Nous distinguons quatre différents mécanismes :

2.4.3.1 Modèles mathématiques

Des équations mathématiques sont utilisées pour modéliser le système. Le but est d'estimer l'état interne du système et de son évolution dans le temps. Les **filtres de Kalman** [47] sont un exemple de cette modélisation mathématique. Holsti et Paakko [48] proposent le mécanisme FDIR « *Fault Detection, Isolation and Recovery* » qui inclut les filtres de Kalman et le Test du χ^2 pour la détection des défaillances dans un satellite d'observation. Walsch [49] propose d'estimer le nombre de composants en état de fonctionner d'un capteur de pression (Rosemount 3051 Pressure Transmitters) avec une certaine probabilité suivant **l'équation de Bernoulli**.

2.4.3.2 Logique floue

L'objectif de la logique floue est de rendre les systèmes informatiques capables de raisonner comme un être humain. Ainsi, l'appliquer lors de la conception d'un système embarqué temps réel permettrai à ce dernier à mieux interagir avec l'utilisateur et comprendre davantage ses besoins [50]. Cheng *et al.* [51] se basent sur la logique floue pour détecter des erreurs dans un système de cluster en estimant le temps moyen de l'épuisement des ressources. Dans le monde de l'industrie, Fonseca *et al.* [52] utilisent la logique floue pour prévoir le dysfonctionnement d'un moteur.

2.4.3.3 Représentation graphique

Il existe des outils qui permettent de modéliser le comportement dynamique des systèmes :

2.4.3.3.1. Réseau de Petri

Pimentel [53] propose une approche basée sur la simulation de Réseaux de Petri Stochastiques pour évaluer la performance (en termes de fiabilité et de sécurité) d'un système embarqué distribué suivant deux types d'architectures. La première architecture propose de travailler avec des composants réparables et la deuxième avec des composants non réparables. L'auteur a conclu que la fiabilité et la sécurité d'un système avec des composants réparables est meilleur que le système correspondant où les composants ne sont pas réparables.

Dans [54] l'auteur propose les Réseaux de Petri Stochastiques pour modéliser le système informatique de contrôle du trafic aérien français (dénommé CAUTRA). Le CAUTRA fournit une assistance automatisée aux contrôleurs et aux régulateurs du trafic afin d'assurer le contrôle du trafic aérien dans des conditions prescrites de sécurité et de régularité. Il est mis en œuvre sur des calculateurs tolérants aux fautes répartis géographiquement dans cinq centres de contrôle de trafic régionaux (CCR) et un centre d'exploitation des systèmes de la navigation aérienne centraux (CESNAC) interconnectés via un réseau de

télécommunication dédié. L'objectif de l'étude est d'évaluer l'impact des défaillances matérielles et logicielles et des procédures de restauration associées sur la sécurité du trafic. À l'étape initiale, l'auteur choisit un composant du système et construit un Réseau de Petri Stochastique décrivant son comportement en supposant que les autres composants sont dans un état de fonctionnement normal. À chaque nouvelle étape, un nouveau composant est ajouté avec les interactions entre les autres composants. Les hypothèses de défaillance et de restauration de ces derniers sont incorporées de façon progressive au niveau de chaque étape. Le modèle final décrit le comportement global du système en tenant compte de toutes les interactions entre les composants. Même si l'approche proposée a répondu à l'objectif fixé (mesurer l'indisponibilité du service et la fréquence d'occurrence en régime stationnaire), l'auteur admet que d'une part la construction d'un Réseau de Petri Stochastique pour l'évaluation de la sûreté de fonctionnement de systèmes réels est une tâche fastidieuse qui nécessite un investissement important de la part des utilisateurs. [54]

Cependant, l'utilisation d'un Réseau de Petri Stochastique conduit souvent à des modèles dont la description graphique est compliqué ainsi inexploitable à cause du grand nombre de transitions instantanées et de tests de marquages essentielles pour décrire les interactions entre les composants.

2.4.3.3.2. Graphes d'interaction des composants

Le modèle reflète la façon dont souvent un composant interagit avec un autre composant. Ce modèle représente le comportement du système sans fautes. Si la proportion de chemins d'exécution entre une instance de composant et d'autres composants s'écarte sensiblement du modèle de référence, l'instance est supposée être défectueuse. Crnkovic [55] explique de manière détaillée le potentiel de cette approche.

2.4.3.3.3. Arbre de défaillance

Un arbre de défaillance (*Fault Tree*) est un modèle graphique inductif (allant des causes vers les effets) permettant de représenter graphiquement un ensemble d'événements susceptibles de déclencher un événement indésirable prédéfini (une défaillance plausible). Cette technique s'appuie sur une analyse préalable du système appelée « Analyse des Modes de Défaillance et de leurs Effets » (AMDE). Les arbres de défaillance sont des outils très sollicités dans l'industrie pour la sureté des systèmes « à risques » (exemple : aérospatial, ferroviaire, nucléaire et naval) et dans l'industrie des produits chimiques [56]. Néanmoins cette méthode tend à être très simpliste, faisant l'impasse sur des détails parfois importants car elle propose la représentation de défaillances « précédemment définies ». La sensibilité aux divers paramètres ne pouvant être connue à priori, les exigences en simplifications sont susceptibles d'engendrer des erreurs importantes sur les résultats finaux. Ainsi, l'obligation de choisir

soigneusement les caractéristiques à représenter est probablement la cause de la difficulté à l'automatisation du processus de ce type de modélisation.

2.4.3.4 Méthodes d'apprentissage automatique

L'apprentissage automatique (*Machine Learning*) est concerné par le développement de mécanismes automatisables permettant à une machine d'apprendre et d'évoluer en tirant parti de son expérience. Ainsi, cette machine devient capable d'effectuer des tâches qui requièrent de l'intelligence. Nous citerons :

- **La reconnaissance de formes** (*Pattern Recognition*) qui permet d'identifier des modèles ou prototype (*Patterns*) par exemple la reconnaissance de la parole ou la vision artificielle. La tâche de reconnaissance permettra à la machine en outre de :
  - **prendre des décisions (Aide à la décision)** : par exemple établir un diagnostic médical .
  - **prédire** : par exemple la prédiction de consommation électrique ou la prédiction de cours boursiers.
- **Le contrôle de processus**, par exemple la conduite de procédés industriels, la conduite de robots.

Dans le domaine de la tolérance aux fautes, les méthodes d'apprentissage automatique permettront à la machine d'appréhender l'apparition future d'une défaillance. La méthode vise à apprendre le maximum du comportement d'une machine. Puis la technique opère sur une ou plusieurs séquences d'observation de l'état du système pour assigner une valeur de classification. Cette classification indique si le système est sujet à des défaillances [57]. Ces méthodes permettent de détecter de manière précoce un état avant qu'il ne conduise à une défaillance.

2.5 Tendances et Défis

Des travaux de recherches ont été proposés et réalisés pendant des années et le défi est toujours d'actualité. Les erreurs dues à l'interaction homme-machine, les fautes matérielles et/ ou logicielles et la complexité de l'installation des systèmes restent inévitables et représentent les principales sources de défaillance.

Comme l'ont montré les sous sections 2.4.1 et 2.4.2, différentes solutions de tolérance aux fautes ont été largement déployées dans la plupart des domaines industriels, télécommunication, services de transport, production et fourniture d'énergie, etc. Toutefois, les besoins en sûreté de fonctionnement et les soucis économiques qui sont étroitement liés rendent ces solutions caduques.

Un système embarqué est devenu un système trop complexe pour qu'un mécanisme assurant sa sûreté de fonctionnement puisse arriver à une description analytique aboutissant à une modélisation déterministe. Dès lors, il est néces-

saire de trouver un compromis qui s'appuie, d'une part sur un déploiement à faible coût et d'autre part qui trace le dynamisme du système embarqué.

Oliner et Sahoo [58] montrent que la prévision des défaillances arrive à améliorer les performances et la fiabilité des systèmes par rapport à des mécanismes traditionnels. De ce fait, le mécanisme proposé dans ce travail est un mécanisme à base d'apprentissage.

2.6 Conclusion

Ce chapitre a survolé les grands principes des mécanismes de tolérance aux fautes dans les systèmes embarqués. Quelques citations des travaux réalisés ont illustré la mise en pratique de ces différents mécanismes et plus particulièrement aux aspects liés aux contraintes temps et ressources limitées.

Cette étude a montré que les méthodes d'apprentissages sont la solution optimale pour optimiser la fiabilité des systèmes embarqués temps réel. Ce type de méthode représente un grand nombre d'avantages : il permet de gagner du temps, il est beaucoup moins coûteux car il s'adapte à la conception du système et il y a une auto amélioration au fil du temps.

Le chapitre suivant est une présentation détaillée des méthodes d'apprentissage automatique. Le choix du mécanisme parmi l'ensemble des méthodes est au cœur de la problématique du chapitre.

Chapitre 3

Mécanismes d'Apprentissage pour la Tolérance aux Fautes

Dans le domaine de l'apprentissage machine, de nombreuses méthodes existent. Ce chapitre présente successivement les algorithmes clés pour la prédiction des défaillances dans les systèmes embarqués temps réel.

Chaque algorithme d'apprentissage est expliqué brièvement, suivi d'une présentation des travaux connexes dans la prédiction des défaillances. Ce chapitre se termine par une discussion sur ces algorithmes.

3.1 Réseaux de Neurones Artificiels

3.1.1 Définition

Introduit par le travail de McCulloch et Pitts dans [59]. Á la différence de la logique floue, les Réseaux de Neurones Artificiels (RNA) (*Artificial Neuronal Network (ANN)*) sont un outil qui permet la classification d'un ensemble de données suivant un *pattern*.

Également appelés « réseaux connexionnistes », ils sont inspirés du fonctionnement élémentaire du système nerveux de l'être humain [60].

Un **Neurone Formel** (Neurone Artificiel) (figure 3.1.1) est un automate doté d'une fonction appelée fonction d'activation du neurone qui transforme un vecteur d'entrées en sortie suivant des règles précises.

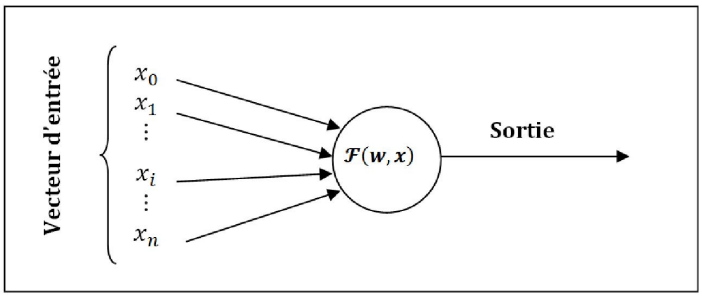


FIGURE 3.1.1 – Architecture d'un Neurone Formel

Chaque entrée x_i est **pondérée** par un poids w_i. Ces poids vont permettre aux neurones d'apprendre et de modifier sa sortie au fur et à mesure de l'apprentissage.

Un Réseau de Neurones Artificiel est un ensemble de neurones formels reliés en réseau, de sorte que les paramètres sortants d'un neurone deviennent les paramètres entrants d'un autre (figure 3.1.2). Sa représentation architecturale est constitué de neurones en plusieurs couches successives, la première représente la couche de paramètres d'entrée et la dernière est la couche de sortie. Les couches intermédiaires sont considérées comme des couches cachées du réseau car leur fonctionnement ne peut être analysé de l'extérieur du réseau.

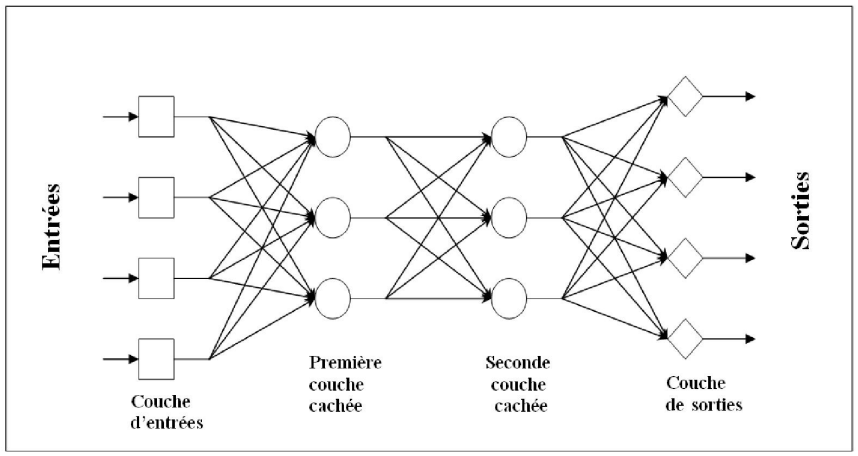


FIGURE 3.1.2 – Architecture d'un Réseau de Neurones Artificiel (avec deux couches cachées)

Les Réseaux de Neurones Artificiels (*Artificial Neural Network*) peuvent être de nature statiques ou dynamiques. Les réseaux dynamiques ont la capacité d'apprendre. L'apprentissage dans un RNA consiste à **ajuster** le poids synaptique sans modifier les entrées. Le **coefficient d'apprentissage** est un autre paramètre fondamental permettant de contrôler la vitesse d'apprentissage. Un petit coefficient évite aux poids du neurone de basculer vers de grandes valeurs à chaque nouvel exemple présenté.

L'apprentissage permet à un RNA d'évoluer. Si l'évolution part dans le sens souhaité alors l'apprentissage est appelé **apprentissage supervisé**. Si la sortie du RNA n'est pas prédéterminée alors c'est un **apprentissage non-supervisé**.

3.1.2 Les Réseaux de Neurones Artificiels et tolérance aux fautes

Troudel *et al.* [61] proposent les réseaux de neurones dans le cadre des Systèmes de Contrôle Intelligents (ICS) pour les moteurs de fusée réutilisables (RRE). Les vibrations des composants mécaniques sont utilisées comme valeurs d'entrées pour estimer, en temps réel, la durée de vie de ces composants. Le RNA est constitué de 15 intrants, deux couches cachées (100 neurones dans la première couche cachée et 50 neurones dans la deuxième couche cachée) et un seul neurone dans la couche de sortie. La performance du calcul du RNA est évaluée par la valeur absolue de l'erreur qui est égale à 0,021 avec un écart type égale à 0,029.

Les Réseaux de Neurones Artificiels ont été appliqués pour prédire les défaillances des rails d'un chemin de fer par Yilboga *et al.* [62]. Les données recueillies par les systèmes d'aiguillage sont utilisées. 75% des données collectées sont utilisées pour l'apprentissage et le reste pour le test. Le taux de bonne classification des niveaux de défaillance était de 99,3%. La moyenne du taux d'erreur était dans les environs de 2,75.

Pawlak et Kowalski [63] développent un système de mesure portable à faible coût pour le diagnostic des défaillances (la détection de défauts du rotor, le déséquilibre de tension et le défaut d'alignement mécanique) d'un moteur d'induction. L'analyse spectrale des courants du stator permet d'obtenir des informations sur les différents défauts du système. Ainsi, le signal courant électrique du stator est considéré comme le vecteur d'entrée. La structure du RNA est de {5-7-3} : 5 neurones en couche d'entrée, 7 en une seul couche cachée et 3 en couche de sortie. La détection des défaillances était efficace à 100%. Cela signifie que les défaillances sont détectés sans aucune erreur.

Fu et Xu [64] utilisent les RNA pour prédire le nombre de défaillances d'une grille de calcul dans les six et huit prochains jours. Le modèle de prédiction est constitué de 4 neurones d'entrée, 3 couches cachées avec 4 neurones dans chaque couche cachée et 1 neurone de sortie. Les données d'une année de fonc-

tionnement de la grille de calcul de l'université de Wayne State sont utilisées pour l'étude de cas. Le pourcentage d'utilisation du CPU, le nombre de paquets transmis par un nœud de calcul et le nombre de demandes d'E/S sur les disques physiques constitues les attributs de cette base de données. Le taux de prédiction donné par cette approche était de 72,7% pour les six prochains jours et 85,3% pour les huit prochains jours.

Szekely [65] propose d'utiliser les RNA pour l'optimisation du fonctionnement d'un circuit *Brushless exciters* qui est utilisé pour stimuler un moteur et synchroniser sa vitesse. Le but de son travail est de proposer un système capable de surveiller le fonctionnement du moteur et de détecter des défaillances. Deux systèmes sont développés : un système sans fil, suffisamment petit pour être embarqué à l'intérieur du moteur. Il permet la transmission des données de surveillance au poste de contrôle pour l'affichage et le stockage. Le second système développé est installé sur un micro-contrôleur et est capable de détecter les défauts de circuit qui apparaissent dans le pont redresseur (*rectifying bridge*) du stimulateur. L'implémentation des RNA pour la détection dans le micro-contrôleur était possible et facile. Les résultats obtenus indiquent une bonne tolérance au bruit, amélioration de la vitesse de détection des défauts par rapport aux systèmes de détection précédents et un bon rapport coût-efficacité.

3.2 Arbres de Décisions

3.2.1 Définition

Un arbre de décision (*Decision tree*) [66] est un outil d'aide à la décision. Il remplace une expertise humaine pour modéliser un raisonnement intellectuel. Il offre une représentation du comportement d'un système sous la forme graphique i.e. un **arbre** (figure 3.2.1). L'extrémité de chaque branche est une réponse que le système peut donner suite à des décisions prisent dans chaque niveau de l'arbre.

La différence par rapport à l'arbre de défaillance (Section 2.4.3.3) est que ce dernier est une évaluation qualitative, destinée à identifier et classer les modes de défaillances. Aussi, il ne prend pas en charge la dépendance entre les évènements. Il est de type binaire, un évènement se produit ou ne se produit pas. Alors qu'un arbre de décision est une évaluation probabiliste. Il prévoit en termes de probabilités les conséquences de l'activation d'une ou plusieurs fautes sur le système. Sa lisibilité et sa rapidité d'exécution justifie son utilisation dans des domaines variés tels que la sécurité, la fouille de données (*Data Mining*), la médecine, etc.

Le principe de l'apprentissage par arbre de décision consiste en deux principales étapes : La construction de l'arbre de décision puis la prise de décision suivant l'arbre construit.

La construction d'un arbre de décision est réalisée par un algorithme qui commence par la segmentation du premier nœud (considérer comme sommet). Il choisit une variables potentielles parmi d'autres suivant un critère donné. Ce dernier caractérise ce qu'on appelle la **pureté** ou le **gain en pureté** lors du passage du sommet vers les feuilles produites par la segmentation. Il existe de nombreux critères, les plus utilisés sont l'**entropie de Shannon** [67] et le **coefficient de Gini.**

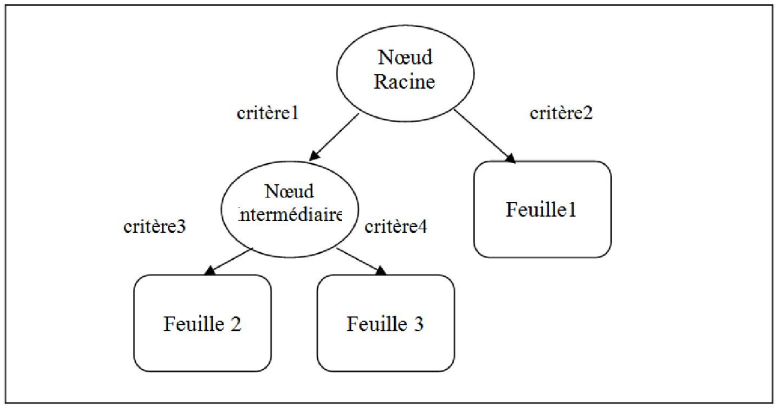


FIGURE 3.2.1 – Exemple d'un Arbre de décision

Il existe différents algorithmes de construction d'arbre de décision, les plus utilisés sont :

— **CART** : cet algorithme a été développé par Breiman *et al.* [68]. L'algorithme construit des arbres de décision binaires (un nœud ne peut avoir que 2 fils). Par la suite, cet algorithme était étendu pour traiter le cas d'attributs continus (à valeurs réelles). Le nombre de tests à explorer dépend de la nature des attributs. Pour la segmentation d'un nœud à un attribut binaire, correspond un test binaire qui est le critère de Gini. Pour un attribut discret ayant n modalités, il lui associe autant de tests qu'il y a de partitions en deux classes, soit 2^{n-1}tests binaires possibles. Enfin, dans le cas d'attributs continus, il y a une infinité de tests envisageables. Dans ce cas, on découpe l'ensemble des valeurs possibles en segments, ce découpage peut être fait par un expert ou de manière automatique [1].

— **ID3** : cet algorithme a été proposé par Quinlan [69]. Il fonctionne avec des attributs binaires et continus. Le test associé à un nœud est réalisé à l'aide de la fonction Gain ou GainRatio basée sur l'entropie de Shannon. L'algorithme permet aussi la construction de règles de décision. Amélio-

1. `www.grappa.univ-lille3.fr/polys/apprentissage/sortie004.html`

rée en 1993 avec une nouvelle version appelée C4.5, l'algorithme permet de travailler avec des attributs ayant des valeurs NULL.

La prise de décision par l'arbre suit le principe suivant : pour chaque nouvel exemple, un chemin est emprunté partant de la racine (**nœud initial**) jusqu'à une feuille de l'arbre. Au niveau de chaque **nœud intermédiaire** un test est exécuté. Ce dernier permet de désigner la branche à suivre dans l'arbre. Ce principe est schématisé dans la figure 3.2.2 :

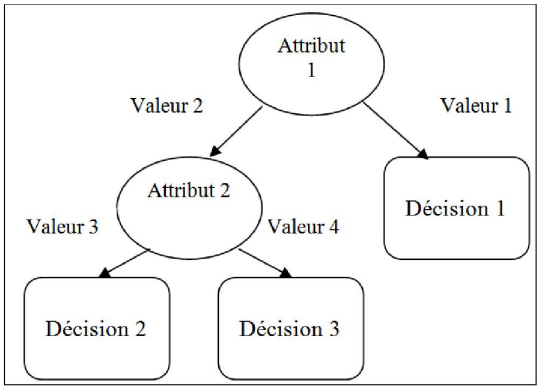


FIGURE 3.2.2 – Prise de décision avec un arbre de décision

Chaque donnée en entrée (**exemple**) est composée d'une liste d'attributs. L'un de ces attributs est l'attribut **cible** ou la **classe**. L'arbre de décision a pour objectif de prédire la valeur de l'attribut cible à partir des valeurs des autres attributs appelés ensemble de **descripteurs**. Bien entendu, la qualité de la prédiction dépend des exemples : plus ils sont variés et nombreux, plus l'apprentissage est optimal, plus une classification précise de nouveaux cas serait faisable.

3.2.2 Arbre de décision et tolérance aux fautes

Khoshgoftaar et Allen [70] proposent l'utilisation de l'algorithme de classification et de régression (CART) pour prédire quel logiciel possède un risque élevé de défaillance pendant sont exécution. L'expérimentation est réalisée par l'étude de quatre versions du logiciel intégrées dans un système de télécommunications. Le taux de bonne classification atteint est important, soit 72.3% pour la version 1, 72.1% pour la version 2, 69.6% pour la version 3 et 62.3% pour la version 4.

Khoshgoftaar et Seliya [71] présentent une étude comparative entre les deux algorithmes CART et C4.5. Le but était d'évaluer leur performance de prédiction dans un système de télécommunication. Le corpus de donnée est divisé en 4 groupes de données. Le premier est utilisé pour l'apprentissage et les trois

autres pour le test. L'algorithme CART a réalisé un taux de bonne classification égale à 68,73% pour le premier groupe de test, 96,90% pour le deuxième et 64,66% pour le troisième. Alors que, l'algorithme C4.5 a réalisé un taux de bonne classification égale à 76,77% pour le premier groupe de test, 74,61% pour le deuxième et 68,73% pour le troisième.

Zhaos *et al.* [72] utilisent l'algorithme ID3 pour la surveillance et le diagnostic des défaillances brusques et des dégradations subtiles d'une imprimante Xerox DC265. Des capteurs sont intégrés dans l'imprimante pour prendre des mesures tel que la vitesse du rouleur de papier, le dégrée d'inclinaison du papier, etc. Ces mesures sont utilisées comme valeurs d'entrée pour la prise de décision. Quatorze types de défaillance doivent être identifiés. L'arbre développé a diagnostiqué 5 défaillances différentes, pour le reste le modèle a confondu entre elles car elles représentent les mêmes symptômes.

3.3 Séparateurs à Vaste Marge

3.3.1 Définition

Proposés en 1995 par Vapnik dans [73], les Séparateurs à Vaste Marge (*Support Vector Machines (SVM)*) sont une méthode de classification binaire par apprentissage supervisé. Ils permettent d'aborder plusieurs problèmes divers et variés comme la régression, la classification, la fusion,... etc.

Cette méthode est basée sur l'utilisation de fonction dite **noyau** (*kernel*) qui permet une séparation optimale des données. Il existe une multitude de fonction noyau : Linéaire, Fonctions à Base Radiale (RBF), Polynomial, Gaussien, Laplacien et Sigmoïde.

Pour deux classes d'exemples données, le but des SVM est de trouver un classificateur qui puisse séparer les données et maximiser la distance entre ces deux classes (figure 3.3.1). Ce classificateur est linéaire appelé **hyperplan**. Les points les plus proches de l'hyperplan, sont appelés **vecteurs de support**. Pour séparer les données, il peut exister plusieurs hyperplans. Les SVM interviennent en cherchant parmi les hyperplans valides, celui qui passe au milieu des points des deux classes. Ceci veut dire : trouver l'hyperplan dont la distance minimale aux vecteurs supports est maximale. Cette distance est appelée **marge** et comme on cherche à maximiser cette marge alors on parle de **séparateurs à vaste marge**. [74]

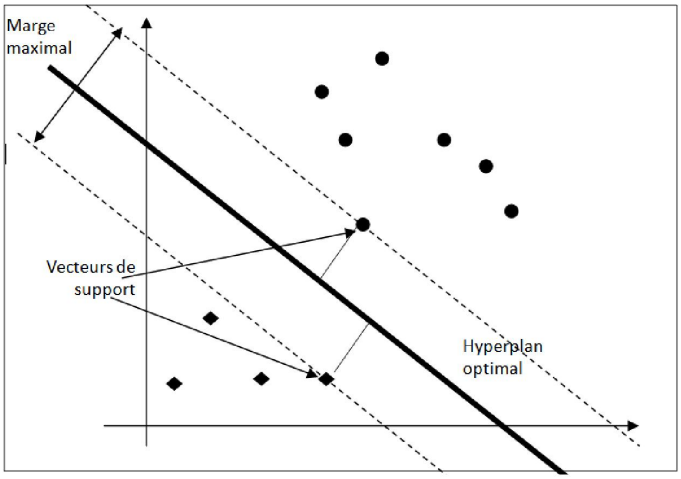


FIGURE 3.3.1 – Hyperplan optimal, marge et vecteurs de support [74]

L'hyperplan optimal est celui qui maximise la marge. La maximisation de la marge a pour objectif d'optimiser la bonne classification d'un nouvel exemple. [74]

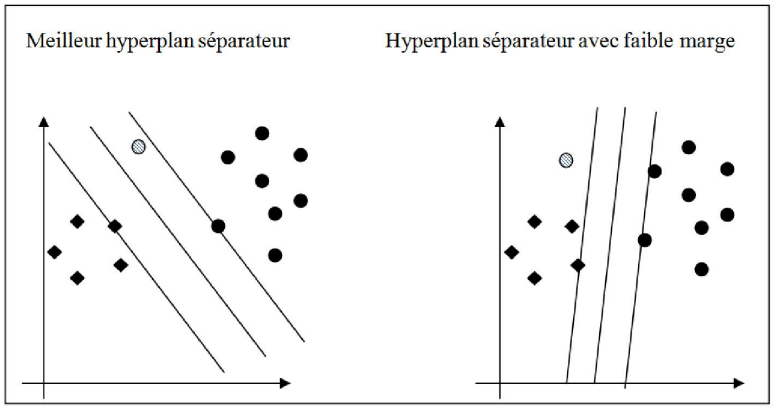


FIGURE 3.3.2 – Maximisation de la marge [74]

La partie gauche de la figure 3.3.2 montre qu'avec une marge grande, un nouvel exemple sera bien classé (hyperplan optimal) alors que sur la partie droite de la figure où la marge est plus petite, l'exemple sera mal classé. Ainsi, une marge plus large est un garant à une bonne classification d'un nouvel exemple.

La classification avec les SVM reposent sur le concept de **plans de décision** qui permet de séparer plusieurs classes différentes. Il existe deux cas de

séparabilité [74] :

1. **Le cas linéairement séparable** : c'est le plus simple, le classificateur linéaire est facilement trouvé.

2. **Le cas non linéairement séparable** : il est plus complexe et dans la plupart des problèmes réels à résoudre il n'existe pas de séparation linéaire possible entre les données (figure 3.3.3) :

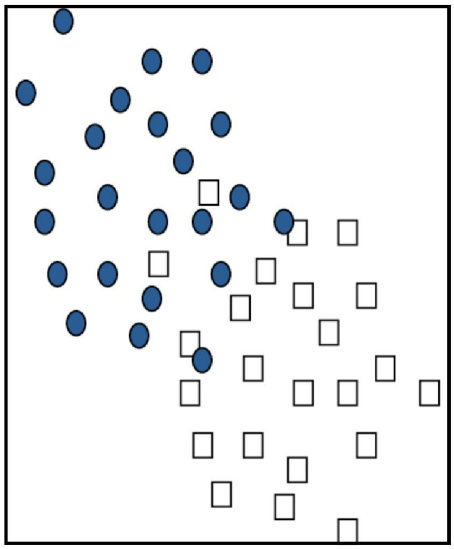


FIGURE 3.3.3 – Exemple d'un cas non linéairement séparable

Au début, les SVM ont été développés pour résoudre des problèmes à 2 classes. Actuellement, ils sont étendus pour des cas de N classes. On parle de **SVM multi-classes** [74].

3.3.2 Séparateurs à Vaste Marge et tolérance aux fautes

Les SVM sont des classificateurs aussi puissants et efficaces que les réseaux de neurones. Murray *et al.* [75] ont appliqué les SVM afin de prévoir les défaillances des disques durs. Les attributs de disques fournis par la technologie SMART sont utilisés pour valider l'approche. Les utilisateurs ont observé que la croissance des valeurs de ces attributs (ou leurs taux de variation) au fil du temps est révélatrice d'une défaillance imminente. Le système SMART permet une surveillance étroite sur un certain nombre d'attributs indiquant l'état et les performances des disques, telles que : lecture du taux d'erreur, taux du Seek Error, nombre de secteurs ré-alloués, ... etc. 25 attributs sont sélectionnés.

Les données de 369 disques sont recueillies. Chaque instance du corpus de données SMART est enregistrée à deux heures d'intervalle. Cependant, les SVM ont donné un taux égale à 17,5% de bonne détection et de 2,3% de fausses alarmes, par rapport à l'approche Rank-Sum qui a fourni une détection de 33,2% et 0,5% de fausses alarmes.

Namburu *et al.* [76] montrent que l'estimation de la gravité des défaillances des moteurs automobiles est faisable avec les SVM. Pour leur étude de cas, les auteurs choisissent le moteur Toyota Camry 544N. Ils utilisent le SIMULINK avec le prototype ECU (*Electronic Control Unit*) pour simuler le modèle du moteur dans plusieurs conditions de fonctionnement (angle de la pédale, la vitesse du moteur, ...etc). Les données de pré- et post-défaillances sont recueillies et huit types de défaillances (F1 - F8) du moteur sont déterminés. L'approche réalise un score de plus de 97 % de précision sur chaque type de défaillance.

Hu *et al.* [77] utilisent les SVM multi-classes avec RBF comme fonction de noyau pour le diagnostic des pannes d'une machine d'extraction de gaz. Un micro-contrôleur ARM doté d'un GPRS sans fil est utilisé pour la surveillance en ligne des mesures d'extraction. Trois différentes valeurs de température, cinq types de densité de gaz et trois rapports de CH_4/H_2, C_2H_4/C_2H_6, C_2H_2/C_2H_4 sont sélectionnés en tant que vecteur d'entrée. Cinq types de défaillance sont définis comme données de sortie : défaillance de surchauffe normale-moyenne-base, défaillance de surchauffe élevée, défaillance de décharge à faible énergie, défaillance de décharge locale et défaillance de décharge à haute énergie. Les résultats expérimentaux de la simulation montrent que le système embarqué développé permet efficacement de surveiller l'état de la machine lors de son fonctionnement et que le modèle proposé est très performant, soit 90 % des défaillance ont été correctement diagnostiqué.

Aravindh *et al.* [78] proposent les SVM multi-classes avec RBF comme fonction de noyau pour détecter les défaillances mécaniques et électriques survenant dans un moteur à induction. Les différents types de défaillance sont : défauts mécaniques de roulement et les défauts électriques qui se produisent dans les rotors. Le signal de vibration du moteur est obtenu à partir de capteurs installés dans le moteur. La méthode de la décomposition de paquets d'ondelettes (*Wavelet Packet Decomposition* ou *WPT*) est utilisée pour extraire les caractéristiques du signal. 50% de l'ensemble de données est utilisé pour l'apprentissage des SVM et le reste pour le test. En combinant WPT et SVM, de très bons résultats sont obtenus soit 97% de bonne classification.

Guo *et al.* [79] proposent d'optimiser les SVM pour diagnostiquer la panne d'un système électronique. L'algorithme Chaos Particle Swarm Optimization (CPSO) est utilisé pour optimiser le choix des paramètres du SVM multi-classe. La fonction RBF est sélectionnée comme fonction de noyau. Pour l'étude de cas, un circuit de conversion du courant au Voltage avec puce CA3140 est utilisé. Quatre modes de défaillances sont identifiés : mode normal, mode

carte de circuit dégradé, mode puce brûlée et mode d'épingles brisées. Pour les données, 7 mesures de tension sont prises depuis 7 points de mesures dans le circuit. 500 valeurs de tension sont enregistrées : 200 pour l'apprentissage et 300 pour le test. 4 classes SVM sont développées, chacune identifie un type de défaillance (SVM1 : mode normal, SVM2 : carte de circuit endommagé, SVM3 : puce brûlée et SVM4 : broches cassées). La faisabilité et l'efficacité de l'approche pour le diagnostic de pannes du système électronique est vérifiée avec un taux de détection de défaillances égale à 98,2 %.

Kim *et al.* [80] proposent les SVM multi-classes avec polynôme comme fonction de noyau pour prédire l'état d'une pompe de Gaz Naturel Liquéfiés (GNL). Les caractéristiques du fonctionnement de la pompe sont utilisées comme données d'apprentissage et de test. Six type de défaillances sont identifiés correspondant aux six étapes de dégradation de la pompe. L'erreur de la classification est dans la moyenne de 18,75% pour les six classes. Le résultat d'estimation de durée de vie restante est proche de la vraie durée de vie restante de la machine. Les auteurs concluent que ces résultats poussent à une étude plus approfondie et à l'application dans l'industrie.

3.4 Algorithmes Génétiques

3.4.1 Définition

Les Algorithmes Génétiques (AG) (*Genetic Algorithm (GA)*) sont des algorithmes d'optimisation inspirés du concept biologique : « la génétique ». Holland [81] s'est passionné des techniques dérivées de la génétique et de l'évolution naturelle (croisements, mutations et sélection) pour proposer cet algorithme. Les AG ont été popularisés grâce au travail de Goldberg [82].

Les AG s'inspirent de la théorie de l'évolution de Charles Darwin pour faire évoluer un ensemble de données, appelé une **population** d'**individus**. Un **individu** représente une solution possible du problème à résoudre. Il est attribué à une fonction appelée **fonction d'adaptation** ou **fitness** qui mesure sa **qualité** (ou son **poids**), elle représente la fonction objectif à optimiser. Les individus potentiels sont sélectionnés, par la suite, pour subir des croisements et des mutations et une nouvelle population de solutions est produite, appelée **individus enfants** ou les **descendants**. Pour la génération suivante, ces enfants vont se substituer, une partie ou la totalité, à la population de la génération précédente pour aussi subir des croisements et des mutations. Chaque nouvelle génération représente une itération jusqu'à ce que le critère d'arrêt soit atteint (figure 3.4.1).

Pour la résolution d'un problème, un Algorithme Génétique doit disposer des cinq éléments suivants [83] :

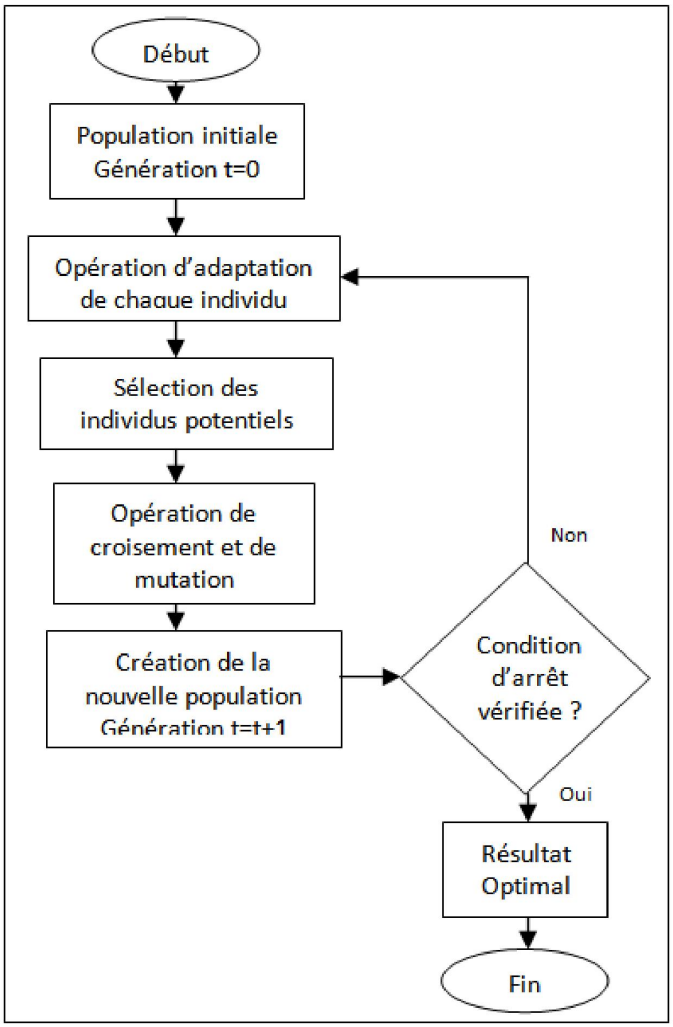


FIGURE 3.4.1 – Organigramme d'un Algorithme Génétique standard [83]

1. Des paramètres de dimensionnement : taille de la population et critère d'arrêt.
   Si la taille de la population est est assez grande, l'évaluation des individus risque d'être très longue et si elle trop petite alors l'AG risque de ne pas converger.
   Le critère d'arrêt définit le nombre maximal de générations à effectuer.

2. Une méthode de codage des éléments de population (les données). Chaque point de l'espace lui est associé une structure de données. La performance des Algorithmes Génétiques dépend de la qualité du codage des données. Le premier codage proposé est le codage binaire par Goldberg [82]. Actuellement c'est le codage réel qui est utilisé. Un autre codage existe sous forme d'un arbre.

3. Un mécanisme de génération de la population qui sert de base pour les générations futures. Dans le cas où le problème n'est pas clairement défini, il est important que la population initiale soit répartie sur tout le domaine de recherche car cette sélection peut rendre plus ou moins rapide la convergence vers un optimum.

4. Des opérateurs (comme le croisement et la mutation) pour modifier les individus d'une population de la génération (t) à la génération (t+1). L'opérateur de croisement combine les gènes des deux individus parents pour générer deux individus enfants. L'opérateur de mutation altère des gènes dans un chromosome.

5. Une fonction d'évaluation de l'individu ou fitness : celle-ci retourne une note qui correspond à l'adaptation de l'élément de population au problème [83].

3.4.2 Algorithmes Génétiques et tolérance aux fautes

Compte tenu de l'importance d'optimiser la fiabilité des systèmes embarqués temps réel, c'est tout naturellement que les Algorithmes Génétiques ont été utilisés pour le développement de nouvelles approches.

Corno *et al.* [84] proposent une nouvelle approche de validation d'un microprocesseur fabriqué, basée sur une simulation. Un Algorithme Génétique aide le concepteur à générer des séquences d'entrée utiles pour être incluses dans le test de qualité du produit. La technique est appliquée à un circuit industriel fabriqué à Centro Ricerche FIAT et a montré que la qualité de la procédure de validation a augmentée. Ceci a amené l'équipe de conception de CRF à réfléchir sur la meilleure façon pour intégrer cette approche dans leur processus de conception standard.

Wattanapongskorn et Coit [85] proposent d'optimiser la fiabilité d'un système embarqué en se basant sur la redondance comme mécanismes de tolérance

aux fautes. Ils considèrent qu'une défaillance peut résulter des fautes dans la spécification du logiciel, des fautes d'un algorithme de vote, et / ou des fautes provenant des composants redondants. Ils suggèrent que cette corrélation peut être résolue avec un modèle d'optimisation. Quatre modèles d'optimisation sont fournis pour la conception d'un système embarqué : pas de redondance (modèle 1), avec une redondance en utilisant l'architecture NVP (modèles 2 et 3) et l'architecture de RB (modèle 4). L'algorithme génétique nécessite que la conception du système (phénotype) doit être codée comme un vecteur de solution (génotype). Ensuite, les opérateurs génétiques (croisement, mutation) sont appliqués pour les générations suivantes, jusqu'à ce que l'AG converge vers une solution, ou un nombre maximal prédéterminé de générations est atteint. Des résultats raisonnables et intéressants sont obtenus et discutés.

Adachi *et al.* [86] proposent d'utiliser les AG pour optimiser une architecture appelée Hip- HOPS. Cette architecture est un outil pour la description du comportement de défaillance au niveau des composants. Il utilise la topologie d'un système à base de composant pour produire automatiquement un arbre de défaillances du système. Dans ce travail, cet outil est étendu (optimisé) pour répondre à un double objectif en termes de fiabilité et de coût. L'outil Hip-HOPS est appliqué à un PCS pour les véhicules.

3.5 Modèle de Markov Caché

Les Modèles de Markov Caché (*Hidden Markov Models (HMM)*) sont un outil d'apprentissage très performant proposés par Rabiner [4]. Cette section est une brève présentation des HMM. [4] [57]

3.5.1 Définition

Un Modèle de Markov Caché est un processus stochastique double : un processus stochastique appelé processus d'émission observable grâce à un processus caché qui est un processus de Markov (figure 3.5.1). Les deux processus sont représentés respectivement par :

- Un ensemble de symboles observés dans chaque état : $V = \{v_1, v_2, v_3, ..., v_M\}$. A l'instant t un symbole observable est désigné par O_t.
- Un ensemble d'états cachés du modèle. : $S = \{s_1, s_2, s_3, ..., s_N\}$. A l'instant t un état est représenté par q_t.

L'émission de chaque état n'est pas déterministe i.e. chaque symbole possède une probabilité d'être émis par chaque état. Ainsi, à un instant t, nous disposons d'une probabilité d'observations sur l'état courant du système : $P(O_t \mid q_t)$

Un HMM est noté λ et se définit par :

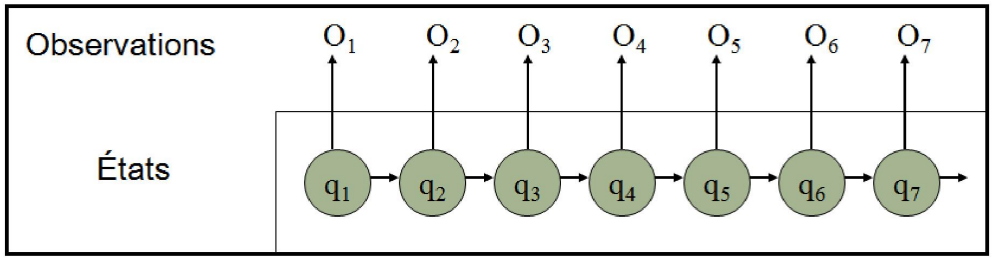


FIGURE 3.5.1 – Structure du Modèle de Markov Caché (HMM)

— N : le nombre d'états cachés du modèle. Chaque état i est représenté par $s_i\,(1 \leq i \leq N)$, noté par $q_t\,(1 \leq t \leq T)$ à l'instant t.

— M : le nombre d'observation qui peut-être observé dans chaque état représenté par $V_i\,(1 \leq i \leq M)$, noté par $O_t\,(1 \leq t \leq T)$ à l'instant t.

— $A = \{a_{ij}\}$: Matrice des probabilités de transition entre les états où a_{ij} représente la probabilité que le modèle évolue de l'état s_i vers l'état s_j (autrement dit : de transition de l'état s_i vers l'état s_j) :

$$a_{ij} = P\left(q_{t+1} = s_j \mid q_t = s_i\right) 1 \leq i \leq N \; 1 \leq j \leq N \; 1 \leq t \leq T \tag{3.5.1}$$

avec :

$$\forall i, j \; a_{ij} \geq 0 \;\text{ et }\; \sum_{j=1}^{N} a_{ij} = 1$$

— $B = b_j(k)$: Matrice des probabilités d'observation de symbole où $b_j(k)$ représente la probabilité que l'on observe le symbole v_k alors que le modèle se trouve dans l'état s_j, soit :

$$b_j(k) = P\left(O_t = v_k \mid q_t = s_j\right) 1 \leq j \leq N, 1 \leq k \leq M \tag{3.5.2}$$

avec :

$$\forall j \; b_j(k) \geq 0 \;\text{ et }\; \sum_{k=1}^{M} b(k) = 1$$

— $\pi = \{\pi_i\}$: Vecteur des probabilités de transition initiale i.e. π_i représente la probabilité que l'état de départ du HMM soit l'état s_i :

$$\pi_i = P\,(q_1 = s_i)\, 1 \leq i \leq N \tag{3.5.3}$$

avec :

$$\forall i\ \pi_i \geq 0 \ \text{ et } \sum_{i=1}^{N} \pi_i = 1$$

Remarque

États finaux : le processus peut s'arrêter dans n'importe quel état donc tout état est final.

Il existe trois problèmes fondamentaux que les HMM peuvent résoudre. Plus connu comme : **Les trois problèmes du HMM**, ils sont définis de la manière suivante :

Problème 1 : Étant donné la séquence d'observations $O = \{O_1, O_2, ..., O_T\}$ et le modèle $\lambda = (A, B, \pi)$, comment évaluer la probabilité d'observation $P\,(O \mid \lambda)$?

Problème 2 : Comment trouver une suite d'états $Q = \{q_1, q_2, ..., q_T\}$, qui maximise la probabilité d'observation de la séquence ?

Problème 3 : Comment déterminer les paramètres (A, B, π) pour maximiser la probabilité d'émission de la séquence d'apprentissage $O = \{O_1, ..., O_T\}$? [4] [57]

Il existe des algorithmes pour résoudre ces trois problèmes, la sous-section suivante les présente de manière abrégée.

3.5.1.1 Évaluation de modèle

Soit le modèle $\lambda = (A, B, \pi)$ et la séquence observée $O = \{O_1, O_2, ..., O_T\}$, la probabilité d'observer cette suite d'observations sachant le λ est définie par $P\,(O \mid \lambda)$. Cette probabilité est calculée suivant l'algorithme *Forward-Backward* [87] qui se compose de deux fonctions :

• La fonction « *Forward* » noté α : qui calcule la probabilité d'émission de la séquence d'observations partant de l'état q_1 jusqu'à l'état q_t à l'instant t ;

• La fonction « *Backward* » noté β : qui calcule la probabilité d'émission de la fin de la séquence d'observations de l'état q_{t+1} à l'état final T. [4] [57]

Ainsi :

$$P\,(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i).\beta_t(i)$$

3.5.1.2 Calcul du chemin optimal

Le deuxième problème de HMM consiste à trouver dans le modèle λ la meilleure suite d'états Q^*, qui maximise la probabilité $P(Q, O \mid \lambda)$ c'est-à dire de trouver le meilleur chemin qui génère la séquence d'observation $O = \{O_1, O_2, ..., O_T\}$ suivant cette probabilité.

L'algorithme de Viterbi permet de formaliser le problème du chemin optimal en fournissant en sortie la valeur P^* : la probabilité de l'émission de la séquence par la meilleure suite d'états $\{q_1^*, q_2^*, ..., q_T^*\}$.[4] [57]

Remarque

Pour pouvoir modéliser les séquences à traiter, les HMM doivent être construits et paramétrés. Cependant, comment les HMM sont construits ?

Les HMM sont construits à l'aide d'un algorithme d'apprentissage appliqué sur un ensemble de séquences représentatives du domaine qu'on souhaite modéliser appelées **séquences d'apprentissage**.[4] [57]

3.5.1.3 Apprentissage du HMM

Le principe de l'apprentissage est d'apprendre ou d'entrainer les HMM à partir d'un ensemble de séquences connu pour estimer les paramètres $\lambda = (A, B, \pi)$ du HMM qui maximisent la probabilité $P(Q, O \mid \lambda)$, à partir d'un ensemble d'observations $O = \{O_1, O_2, ..., O_T\}$.

Pour cela, une procédure de ré-estimation existe, c'est l'algorithme d' *Expectation -Maximisation* (EM) qui a été initialement introduit par Baum et Eagon [3] pour la maximisation de la vraisemblance. Définie dans les ouvrages traitant les HMM comme l'algorithme Baum-Welch, il s'agit d'un algorithme qui affine au fur et à mesure le modèle λ tout en maximisant la probabilité de génération d'une séquence d'observations $O = \{O_1, O_2, ..., O_T\}$ et non celles du chemin le plus probable de Viterbi. [4] [57]

3.5.2 Modèles de Markov Cachés et tolérance aux fautes

L'application des Modèles de Markov Cachés dans la prédiction des défaillances des systèmes embarqués temps réel, a fait l'objet de nombreux travaux de recherche. Les travaux les plus cités sont présentés dans cette section.

Kang *et al.* [88] proposent une approche avec les Modèles de Markov Cachés Discrets (DHMM) [2] pour la reconnaissance des états d'usure d'un outil de coupe YT15 en acier avec une dureté de $HB243$. L'approche est définie comme suit :

1. **Acquisition des données :** Le dynamomètre (YDC-III) est utilisé pour détecter la force de coupe : $Fz(t)$ et l'accéléromètre (B&K4370) est utilisé

2. Ce type de HMM est expliqué dans l'Annexe A.

pour détecter les vibrations de l'outil de coupe : $a(t)$. Le signal $Fz(t)$ et le signal $a(t)$ sont prélevés par l'IPC avec le convertisseur A/D. La figure 3.5.2 montre que l'accélération de la fréquence de vibration augmente avec l'augmentation de l'usure de l'outil.

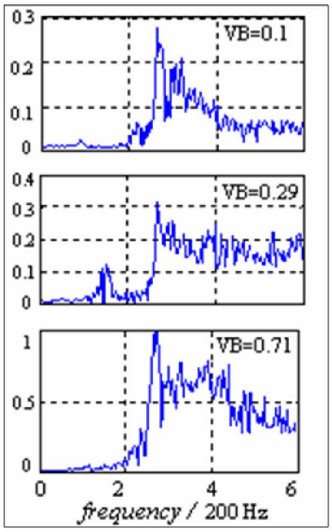


FIGURE 3.5.2 – Fréquence de Fz(t) dans trois différents états d'usure [88] L'axe horizontal représente le temps et l'axe verticale représente l'amplitude de la fréquence de vibration.

2. **Quantification vectorielle :** chaque spectre est normalisé et codé pour former un ensemble de valeurs discrètes.
3. **Modélisation** : trois modèles DHMM sont créés représentant trois état d'usure de trois signaux de vibration différents : $VB = 0,1mm$, $VB = 0,29mm$ et $VB = 0,7mm$, respectivement. Dix échantillons de données sont utilisés pour chaque modèle DHMM. Un apprentissage de 30-50 est exigé pour atteindre une précision optimale.
4. **Reconnaissance de l'outil usé** : une séquence observée est soumise aux différents modèles DHMM pour calculer la probabilité de vraisemblance. La probabilité la plus élevée est sélectionnée et l'état d'usure est identifié. La figure 3.5.3 présente la probabilité de vraisemblance donnée par les trois modèles DHMM. De ces résultats, les auteurs concluent que l'approche est efficace où les trois états d'usure sont reconnus.

DHMM	λ_1	λ_2	λ_3	recognition
0.1	***-16.59***	-21.68	-97.23	initial wear
0.21	-21.46	***-18.32***	-84.59	normal wear
0.29	-34.89	***-19.38***	-75.06	normal wear
0.41	-47.99	***-15.89***	-59.98	normal wear
0.5	-51.78	***-27.53***	-45.57	normal wear
0.62	-67.22	***-32.62***	-38.73	normal wear
0.71	-86.91	-69.62	***-11.38***	severe wear
0.79	-98.47	-85.23	***-15.27***	severe wear

FIGURE 3.5.3 – Résultats de reconnaissance des états d'usures [88]

Pour la reconnaissance de l'état d'usure de l'outil est réalisée par :

$$max[P(O_t|\lambda_i)] \ (i = 1, 2, 3)$$

Prenons l'étape de reconnaissance d'une séquence observée avec le signal $VB = 0,41$, le modèle donne $logP(O_t|\lambda_1) = -47,99$, $logP(O_t|\lambda_2) = -15,89$ et $logP(O_t|\lambda_3) = -59,98$. L'indice maximum $max\, P(O_t|\lambda_i) = 2$. Ainsi l'état d'usure normal est reconnue.

Le processus de forage est l'un des procédés d'usine les plus utilisés dans l'industrie, Jusqu'à 50 % des opérations des usines aux États-Unis impliquent le forage (du forage de la terre jusqu'aux pièces métalliques) et environ 40% des opérations d'extraction de métal dans l'industrie aérospatiale impliquent le processus de forage. La qualité des trous forés est cruciale et dépend de l'état de la mèche de la perceuse. Ainsi, la détection précoce de sa case et/ou son usure excessive est importante.

Ainsi, Baruah *et al.* [89] emploient les HMM pour prédire la défaillance d'une machine de forage vertical. L'approche a pour objectif d'identifier l'état (*health-state*) de la mèche de la perceuse et d'estimer la durée de vie restante (RUL) de la machine de forage. La RUL est une distribution de probabilité. Elle représente le nombre de transitions de l'état actuel à l'état de défaillance correspond au nombre d'essais (*Hole*) à percer avec succès avant la défaillance de la mèche, car une transition d'un état à un autre se produit dans chaque essai. Les auteurs ont procédé comme suit :

1. **Acquisition des données :** les signaux capturés de la mèche de la perceuse : la longitude de la force de perce (*longitudinal thrust-force*) et le signal de couple (*torque signal*) sont les caractéristiques révélatrices d'une future défaillance (figure 3.5.4). Ainsi, c'est sur ces deux signaux que l'estimation de RUL se base.

Le dispositif expérimental est constitué d'une machine CNC-HAAS VF-1, un poste de travail avec le logiciel LabVIEW pour le traitement du signal. Un Kistler 9257B piezo-dynamometer est utilisé pour mesurer le *longitudinal thrust-force* et le *torque signal*. L'acquisition de données est réalisée avec la carte PCI-MIO-16XE-10. La perceuse est constituée de deux mèches twistées avec un taux d'alimentation égale à $4,5\,ipm$ (pouces par minute) et une vitesse de rotation égale à $800\,rpm$ (tours par minute). Quatorze mèches de perceuses sont utilisées dans l'expérimentation. Chaque mèche est une barre en acier inoxydable avec une épaisseur de $0,25\,pouces$. Chacune d'elles est utilisée jusqu'à ce qu'elle atteigne un état de défaillance physique. Soit en raison d'une usure excessive ou en raison d'une déformation plastique brute de la pointe de l'outil due à l'augmentation excessive de la température. Les données sont recueillies à $250Hz$, considérées comme suffisantes pour saisir la dynamique de l'outil de forage en termes de *Thrust force* et de *Torque*. Ces signaux sont par la suite normalisés .

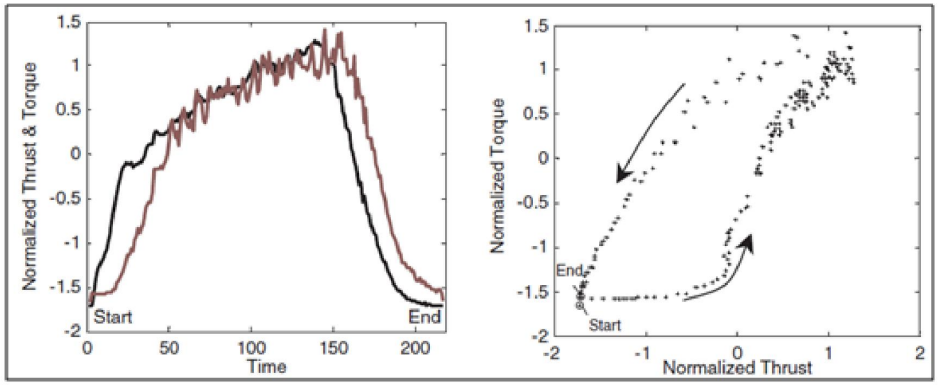


FIGURE 3.5.4 – à gauche : les signaux *Thrust force* et *Torque* normalisés un trou particulier. à droite : diagramme conjoint des signaux *Thrust force* et *Torque* normalisés au cours d'un perçage d'un trou particulier.[89]

2. **Modélisation** : quatre états de la mèche sont identifiés : bon, moyen, mauvais et pire. Quatre HMM sont créés : HMMGood, HMMMedium, HMMBad et HMMWorst. Sur les Quatorze mèches, les données de dix mèches sont utilisées pour l'apprentissage des modèles HMM et les données provenant des quatre mèches restantes sont utilisées pour tester les modèles.

3. **Résultat** : L'exactitude de la classification des quatre HMM est en moyenne égale à 96,9%. La représentation graphique de la probabilité de vraisemblance pour les mèches N°4 et N°5 est illustrée dans la figure 3.5.5 :

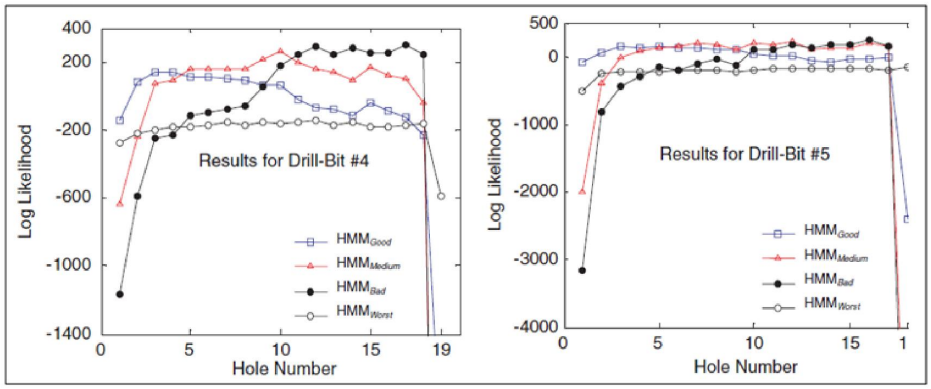


FIGURE 3.5.5 – Trajectoires de la probabilité de vraisemblance pour différents modèles HMM de certaines mèches [89]

Néanmoins, comme tout modèle, les HMM représentent des limites. Plusieurs travaux de recherches pour la prédiction des défaillances basées sur les dérivés des HMM ont été proposés.

Ainsi sur la même étude de cas de Baruah *et al.* [89] que Camci et Chinnam [90] proposent les HMM standards et les **HMM Hiérarchiques** (HHMM) pour estimer en temps réel l'état d'une mèche de perceuse d'une machine de forage CNC.

1. **Acquisition des données :** douze mèches de la perceuses sont utilisées. Les caractéristiques de l'expérimentation sont identiques à celles du travail de Baruah et Chinnam [89]. Le *longitudinal thrust-force* et le *torque signal* (figure 3.5.6) sont utilisés pour l'estimation de l'état compte tenu de leur forte corrélation avec l'état de l'outil de forage.

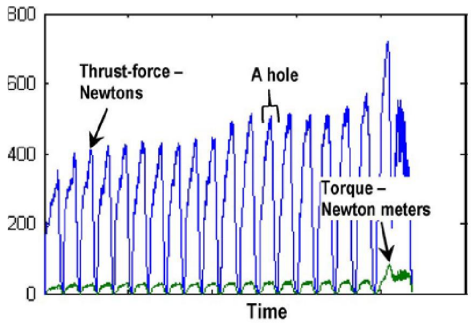


FIGURE 3.5.6 – Données capturés du Trust-force et torque de la mèche N°5 [90]

2. **Modélisation** : deux modélisations différentes sont appliquées, la première approche emploie un ensemble de HMM (chaque HMM représente un état distinct.) tandis que la seconde approche utilise un seul HHMM. Pour la première modélisation, quatre HMM sont créés (chacun représente un état distinct) où chaque HMM est initialisé avec quatre états cachés et la probabilité d'observation est représentée par la distribution de Gauss. Les auteurs ont remarqué que lorsque moins de quatre états cachés sont utilisés, la précision de la classification diminue et que lorsque plus de quatre, deux ou trois HMM arrivent à représenter le même état d'une mèche donnée.
3. **Résultat** : une fois la phase d'apprentissage terminée, la donnée correspondante à un essai est introduite à tous les HMM et celui avec une probabilité de vraisemblance élevée est défini comme le représentant de l'état. Les résultats des estimations de l'état sont donnés dans la figure 3.5.7 pour les 12 mèches de la perceuse. Chaque mèche est représentée dans une ligne, les différents nombres d'essais forés avec succès sont représentés en colonnes. Une cellule vide indique que la mèche correspondant à la ligne a défailli avant d'atteindre le essai correspondant à la $n^{éme}$ colonne. Les nombres dans chaque cellule désignent le numéro du HMM qui représente l'état de la mèche.

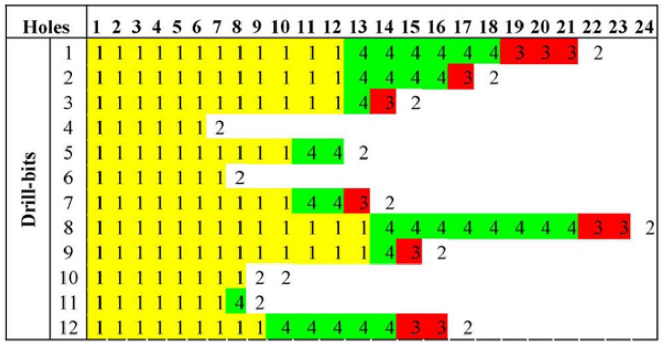


FIGURE 3.5.7 – Estimation de l'état de toutes les mèches en utilisant les HMM [90]
(1 : Tout neuf, 2 : proche de la défaillance, 3 : utilisé de manière excessive ,4 : utilisé)

Avec les HHMM pour la deuxième modélisation, l'évaluation expérimentale montre qu'il existe cinq états cachés pour représenter une mèche.

La figure 3.5.8. présente l'estimation de l'état de l'ensemble des douze mèches de forages par les HHMM. L'état N°4 est représenté seulement dans trois mèches de perceuse. Les auteurs proposent que les états N°4 et N°5 peuvent être combinés dans un seul état.

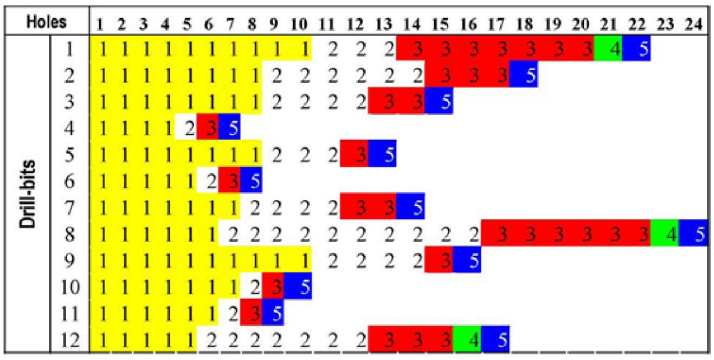


FIGURE 3.5.8 – Estimation de l'état de toutes les mèches en utilisant HHMM [90]
(1 : Tout neuf, 2 : presque neuf, 3 : modérément utilisé, 4 : utilisé de manière excessive, 5 : proche de la défaillance)

Comme on peut le voir dans le tableau de la figure 3.5.9, les résultats des estimations d'état semblent meilleurs avec les HHMM par rapport à un groupe de HMM.

		R Square			RMSE
		average	worst	best	median
	HMM	0.53	0.3	0.79	5.74
Presented method	HHMM	TR 0.91 TST 0.90	0.73	0.996	TR 3.78 TST 4.03

FIGURE 3.5.9 – Comparaison des HMM et HHMM utilisés [90]
(TR : apprentissage, TST : test, RMSE : la moyenne de l'erreur quadratique, R Square : coefficient de détermination)

Avec cette étude, les auteurs concluent que les HHMM représentent certains avantages :

- Les HHMM donnent la possibilité de modéliser tous les états en utilisant un seul modèle. Il suffit de former un HHMM au lieu d'un ensemble de HMM.
- Une meilleure classification : HHMM donnent une meilleure estimation de l'état par rapport à un HMM standard.
- Probabilité de transition : HHMM calcule automatiquement les probabilités de transition entre états. Cette transition n'est pas faisable entre l'ensemble des HMM, alors que les probabilités de transition entre les états sont importantes pour le calcul RUL.
- Le temps d'apprentissage pour HHMM varie entre 174 et 255 secondes en fonction du nombre d'états utilisés dans les niveaux supérieurs et inférieurs alors que les HMM réguliers, prennent environ 212 secondes.

Xing-hui et Jian-she [91] proposent, de même, d'appliquer les HMM et les HHMM pour calculer la durée de vie (RUL) d'une pièce de roulement mécanique (*rolling bearing*) (figure 3.5.10) :

FIGURE 3.5.10 – Pièce de roulement mécanique (*rolling bearing*) [92]

Le composant roulement est un élément important et très utilisé dans les machines tournantes. La défaillance d'un roulement peut provoquer la panne

de la machine. Par conséquent, le pronostic de défaut du composant à roulement est très important pour garantir l'efficacité de la production et de la sécurité de l'installation. La défaillance du roulement implique généralement plusieurs états dégradés. Un petit changement dans l'alignement d'un roulement pourrait causer une petite entaille dans la pièce qui pourrait alors provoquer une encoche dans la bague du roulement, ce qui pourrait ensuite causer des entailles supplémentaires et mener à une défaillance complète de la pièce. Généralement, le roulement possède cinq états : normale, piqures (*pitting*), détachement (*desquamate*) peu profond, détachement profond et défaillant. Mais le temps entre les cinq états est très long et la précision en utilisant les HMM pour le pronostic est très faible. Les auteurs proposent le modèle HHMM pour représenter les cinq états du roulement à travers leur partition en sous états.

1. **Acquisition des données :** les auteurs utilisent le signal de vibration du roulement comme donnée de traitement. L'expérimentation est constituée d'un moteur électrique, un capteur de «*torque*», un testeur de tension. Plusieurs pièces de type : Drive-end $SKF620512KHZ$ avec des vitesses de rotation de $1797r/min$. Elles sont de tailles différentes. Les diamètres varient entre $0.1778mm$, $0.3556mm$, $0.5334mm$ et $0,7112mm$. Le signal est décomposé en parties pour obtenir huit types de fréquences (de faible à élevé). Ensuite, le paquet *Wavelet Packet Analysis* est utilisé pour l'analyse du signal en ondelettes. La méthode Lloyd [3] est utilisée pour la quantification vectorielle.
2. **Modélisation** : quatre HMM sont crées, avec le nombre d'état caché N égale à 2 et M à 4. Chaque échantillon de données de test est utilisé dans les quatre HMM, ensuite, chacun d'eux calcule la probabilité de vraisemblance.
3. **Résultat** : les auteurs ont conclu que l'espérance de vie (RUL) du roulement est en moyenne de 243.1744 heures avec une marge d'erreur de 33.1744 heures. Sachant, qu'en utilisant les HHMM le temps d'apprentissage était de courte durée par rapport au HMM ainsi le RUL est estimé rapidement. La figure 3.5.11 montre le calcul de taux de probabilité de défaillance du roulement avec les HHMM :

Salfner [93] utilise le Modèle Semi-Markov Caché (HSMM) pour prévoir la défaillance d'un système de télécommunication commercial basé sur des rapports d'erreurs. Cette approche de prédiction de défaillances en ligne est effectuée en trois étapes :

1. **Prétraitement des données** : les messages d'erreur survenus dans un laps de temps donné avant le temps présent sont enregistrés. Ils forment une séquence d'erreur. Chaque séquence est pré-traitée : l'assignation

3. C'est un algorithme qui permet de construire un quantificateur scalaire optimal en minimisant la distorsion.

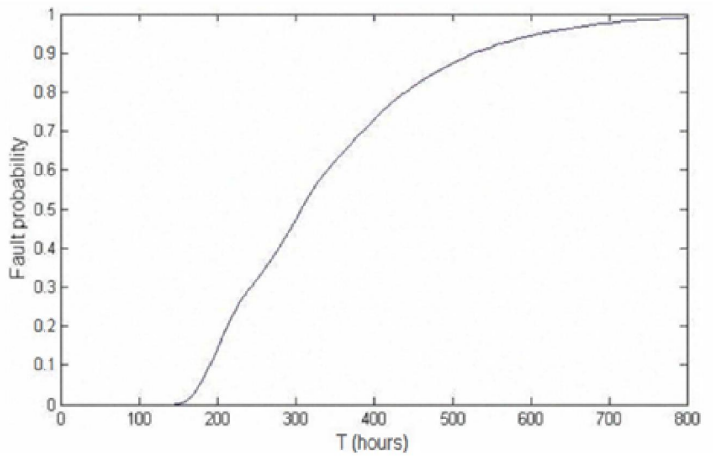


FIGURE 3.5.11 – Valeur de probabilité de défaillance du roulement avec les HHMM [91]

de symboles et filtrage (seuls les symboles qui se produisent plus fréquemment sont choisis comme données pour l'expérimentation). Pour l'apprentissage, les séquences sont regroupées en plusieurs clusters.

2. **Modélisation** : au début les HMM standards sont choisis comme modèle pour la prédiction [94]. Cependant, les auteurs ont réalisé que les données d'entrée sont des séquences temporelles (séquences en temps continu) et les HMM standard ne sont pas conçus pour le temps continu. Pour répondre à cette problématique, les HMM sont combinés avec un processus Semi-Markov résultant les Modèles Semi-Markov Cachés ou HSMM. A l'aide des HSMM, la similitude avec les séquences de défaillance et de non-défaillance est calculée. Cette valeur est utilisée comme mesure de prédiction entre la séquence observée et les séquences de données utilisées pour l'apprentissage en temps réel.

3. **Résultat** : Les meilleurs résultats sont obtenus avec des HSMM à 100 états avec : un F-mesure [4] maximale de 66%, une précision de 70%, un rappel de 62% et le taux de faux positif de 0,016. Les auteurs ont déterminé que 20 minutes est le temps nécessaire pour que le modèle puisse observer le système, enregistrer les données et donner une prédiction plus exacte possible. Ils ont conclu aussi grâce à l'analyse comparative entre HSMM, SVM et la technique DFT que la prédiction de défaillance basée sur les HSMM surpasse les autres approches de prédiction de défaillance de manière significative. Cependant, l'amélioration des prévisions de défaillance se fait au prix d'une complexité de calcul : l'apprentissage du

4. L'Annexe B explique en détails ces métriques d'évaluations

modèle consomme 2,38 fois plus de temps et la prévision en ligne 224,5 fois plus de temps que l'approche comparative la plus lente.

Zhao *et al.* [95] exploraient davantage l'approche de Salfner [93] en combinant les HMM et HSMM pour prédire la défaillance d'un disque dur à partir des données SMART. Les auteurs considèrent que les attributs SMART observés forment des séries chronologiques (existence d'une dépendance temporelle entre données) et emploient les HMM et les HSMM pour la prédiction des défaillances. L'approche est définie comme suit :

1. **Acquisition des données :** les données sont fournies par le *Center for Magnetic Recording Research* de l'université de Californie, San Diego. Les données contiennent 178 lecteurs (drives) «Non-défaillant» et 99 «Défaillant». Pour chaque lecteur, une entrée (instance) de 60 attributs SMART est enregistrée toutes les 2 heures pour un total de 600 heures. Les auteurs admettent que dans cette base de données, il y a très peu d'instance représentant les disques défectueux ce qui la rend déséquilibrée.
2. **Modélisation** : Un HMM et un HSMM sont construits à partir de séquences d'apprentissage positives (disques défectueux). En suite, un autre HMM et un HSMM sont construits avec des séquences d'apprentissage négatifs (disques parfaits). Les données de test (disques sans étiquette) sont évaluées par les quatre modèles. La différence entre les deux valeurs de probabilité de vraisemblance (disques défectueux et disques parfaits) pour chaque modèle est calculée. Si la différence est supérieure à un seuil qui est précédemment déterminé, le disque testé est prédit comme défaillant, dans le cas contraire il est prédit parfait.
3. **Résultat :** Pour une étude comparative, deux autres modèle sont choisis : le test Rank-Sum et les SVM.

   Dans une première série d'expériences, les auteurs ont commencé avec 25 attributs pour distinguer les disques défaillants et non-défaillants. Après plusieurs tests sur les HMM et HSMM, ils ont trouvé 4 attributs qui fournissent une bonne prédiction de défaillance, à savoir : ReadError18, Servo2, Servo10 et FlyHeight7.

   Sur les attributs, ReadError18 et Servo10, les HMM et HSMM ont largement surpassé le test Rank-Sum. Pour l'attribut Servo10, le taux de détection est de 46 % et 30 % pour les HMM et HSMM respectivement, et 0 % de fausse alarme pour les deux modèles alors que pour le test Rank-Sum, il n'y avait pas de détections mesurables et une fausse alarme était moins de 1%. Pour ReadError18, les HMM ont surpassé HSMM avec un faible taux de fausses alarmes. La raison possible pour expliquer ce résultat est que les HSMM ont besoin de plus de données pour l'apprentissage et la base de données utilisée pour ce travail est trop limitée pour donner de bons résultats. Les taux de détection obtenus par le test Rank-Sum

sur les autres attributs Servo2 et FlyHeight7 sont trop faibles pour être mesurés.

Dans la deuxième série d'expériences, les deux modèles HMM et HSMM sont combinés, et atteignent des taux de détection élevés de 52 % et 0 % de fausses alarmes. Ceci montre aussi que la combinaison HMM-HSMM surpasse les modèles Rank-sum test et SVM qui ne capturent pas la relation temporelle entre les valeurs d'attributs au fil du temps. De même pour le temps d'exécution où les HMM et HSMM imposent moins de temps pour l'apprentissage et la détection (tableau 3.1).

Modèle	Apprentissage (en seconde)	Test (en seconde)
HMM (simple)	192,4	0,04
HSMM (simple)	191,8	0,47
HMM-HSMM	3848,4	0,16
Rank-sum	(aucun apprentissage n'est nécessaire)	2,82
SVM	1279,1	1,64

TABLE 3.1 – Temps d'exécution pour HMM, HSMM, Rank-Sum et SVM [95]

Sur le même type de données (SMART) Teoh *et al.* [96] ont modifié la structure des HMM standard. Ils ont sélectionné 24 attributs et ont obtenu une précision de l'ordre de 90 %.

D'un autre côté, Murugesan *et al.* [97] proposent d'optimiser les HMM pour identifier le nombre optimal des états cachés à employer pour prédire une future défaillance à partir d'une séquence d'erreurs. L'optimisation est réalisée à l'aide des Algorithmes Génétiques.

3.6 Discussion des Mécanismes d'Apprentissage

Il a été observé que le processus de défaillance peut être non stationnaire et donc la distribution des probabilités des défaillances varie dans le temps. Les Motifs de non-stationnarité sont multiples : changements dans la configuration, différents modes d'utilisation et variation de l'environnement externe. Dans ces cas, des techniques de prédiction de défaillance à base de reconnaissance de forme (*Pattern Recognition*) pour identifier les symptômes qui indiquent une future défaillance semble l'approche à suivre.

Même si les mécanismes d'apprentissage précédemment présentés sont des outils puissants qui ont fait leur preuve dans le domaine des systèmes embarqués, ils présentent tous des limites.

Les **Réseaux de Neurones Artificiels** sont très utilisés dans de nombreuses applications, par exemple la reconnaissance de la voix dans une voiture, traitement des images dans un téléphone mobile, etc.

Néanmoins, un Réseau de Neurone Artificiel exige à ce que son apprentissage sur la même donnée se répète des dizaines de milliers de fois. Ainsi dans le cas d'un corpus énorme tel qu'une base de données industrielles, le temps d'apprentissage peut être déraisonnable. Ajouter à cela, si la valeur de l'erreur est mal fixée (trop minimisée ou maximisée) alors la totalité de son apprentissage peut être considéré faux car le réseau classera mal les instances. On appelle ce type d'apprentissage l'*over-fitting*.

D'un autre côté, un réseau de neurones est vu comme une boite noire : des valeurs sont insérées en entrées et il retourne un résultat. Pour certaines applications c'est suffisant, mais pour d'autres non. Notamment dans des applications de mécaniques d'automatismes qui demandent ce qui se passe à l'intérieure de cette boite pour comprendre le mode de fonctionnement d'une pièce mécanique et/ou électronique avant son usure. Pour cela, des techniques d'apprentissage comme les arbres de décision sont généralement sollicitées pour pouvoir comprendre le comportement d'un système embarqué.

Cependant la performance d'un **Arbre de Décision** dépend impérativement de la définition de la bonne taille de l'arbre et des données qui ont servi à sa construction. Dans certains cas, le modèle peut être considéré comme restreint et ne représente pas le système dans sa totalité (ne représente pas tout le comportement possible du système) car l'échantillon de données sur lequel il a été construit est insuffisant. Dans d'autres cas, le modèle est considéré comme complexe en raison de sa grande dimension et il se voit incapable d'être extrapolé à de nouvelles données.

Un arbre de décision peut être traduit en termes de règles. Les **règles de décision** sont de la forme suivante :

Si *<condition1>* Alors *<Action 1>*

Si *<condition2>* Alors *<Action 2>*

⋮

Si *<conditionN>* Alors *<Action N>*

L'apparition d'une défaillance est prévue lorsque l'ensemble des conditions est vérifié. Cependant, cette approche peut retourner trop de règles. Avec ce type de mécanismes, il faut trouver un ensemble de règles qui est assez général pour détecter le maximum de défaillances sans générer trop de fausses alertes. [16]

Même si l'implémentation des **Séparateurs à Vaste Marge** est généralement peu coûteuse en terme de temps, la recherche et la sélection des paramètres permettant d'obtenir une performance optimale impose des phases de test souvent très longues. Ainsi, la réalisation d'un programme d'apprentissage par les SVM consiste à résoudre un problème d'optimisation imposant un système de résolution dans un espace de dimension conséquent. [74]

Pour les **Algorithmes Génétiques**, ils restent des outils d'optimisation. Ils doivent faire appel à un autre algorithme d'apprentissage pour résoudre un problème, comme le cas du travail de Murugesan *et al.* [97].

Dans cette thèse, nous estimons qu'un bon mécanisme de tolérance aux fautes à base d'apprentissage doit être capable de répondre à un double objectif : le coût et le temps i.e. n'impose pas beaucoup de charge au niveau matériel et logiciel pour assurer sa tâche et doit agir dans un espace de temps bien déterminé. Néanmoins, nous avons vu que les mécanismes précédemment discutés n'arrivent pas à répondre à ce double objectif. En plus, un mécanisme adéquat doit refléter à la fois le système et la manière dont il se comporte pour permettre au mieux d'anticiper une future défaillance.

Les **Automates Probabilistes** permettent de modéliser l'évolution d'un système embarqué dans le temps par :

— des état : qui décrivent le système à chaque instant t .

— des transitions entre états : qui décrivent la manière dont ses états s'enchainent. [57]]

L'état défaillant d'un système embarqué ne peut pas être mesuré mais il peut être observé à travers les valeurs générées par lui. Mais avec les automates probabilistes, la génération de symboles se produit sur les transitions. Tandis qu'avec les **Modèles de Markov Cachés** la génération de symboles s'effectue sur les états, où chacune lui est associé une probabilité d'observation.

C'est pourquoi les HMM sont l'outil adéquat pour la prédiction des défaillances dans un système embarqué temps réel.

Les raisons sont les suivantes [94] [98] :

— Les HMM Offrent une structure souple et s'adapte à des modifications ;

— Ils ont prouvé leur efficacité dans plusieurs domaines (reconnaissance de la parole, l'écriture, le comportement humain, ...) ;

— Il existe une correspondance entre les éléments du HMM et les entraves de la sûreté de fonctionnement (figure 2.1.1) :

  * Les **fautes** qui affectent l'état d'un système sont non observées donc elles correspondent aux **états cachés**.

  * Le comportement d'un système ne peut être observé qu'à travers des signaux (ex : signaux de vibration, débit de transmission, ...etc) représentant les **erreurs**, qui sont la manifestation de fautes, correspondent aux **symboles d'observation** des états cachés.

— De plus, le temps d'exécution des HMM pour l'apprentissage et la prédiction est court. [94] [98]

Un système embarqué est un ensemble d'entités qui interagissent avec d'autres entités. Une entité peut être le composant lui même, un autre composant du

même système ou un autre d'un environnement externe. Ainsi cette interaction représente une forme de causalité : le comportement d'une entité influence le comportement d'une autre.Une fois une des composantes est endommagée, le changement de son état aura un incident sur lui-même et/ou sur un autre composant qui dépend de lui. Comme ce deuxième composant est défectueux, un autre incident aura un impact sur lui et vers un troisième composant. Le même scénario se répètera sur ce dernier et ainsi de suite (figure 3.6.1). Ainsi, une erreur qui ce manifeste dans un composant arrive à affecter l'activité des composants non défaillants.

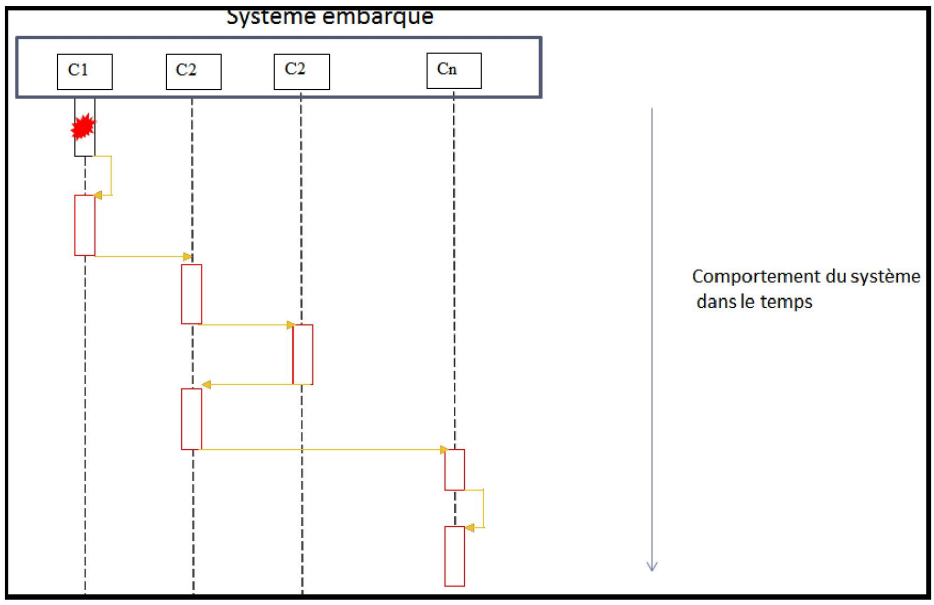


FIGURE 3.6.1 – Causalité du comportement dans un système embarqué

Suivant la définition des HMM, ce problème est défini comme suit : une fois que l'**état** d'un composant change (défaille), il génère une **erreur** qui affectera l'état d'un autre composant dépendant de lui et une autre erreur sera produite et ainsi de suite. Donc une erreur observée a un impact sur l'état futur du système i.e. les paramètre des observations et des transitions d'état dépend de l'observation précédente.

Cependant les HMM et ses variantes HHMM et HSMM ne tiennent pas compte des caractéristiques dynamiques de l'état. Ils définissent que la probabilité d'observation dans chaque état est unique.

En 1997, Kobayashi et Haruyama [10] optimisent les HMM standards, en intégrant le conditionnement entre l'état actuel et l'observation précédente. Une nouvelle variantes des HMM a vue le jour, les ***Partly Hidden Markov Models*** (PHMM). Pour cela, les PHMM sont choisis dans ce travail comme le

modèle de tolérance aux fautes dans les systèmes embarqués temps réel. [99]

3.7 Conclusion

Ce chapitre s'est focalisé sur les algorithmes d'apprentissage et leur rôle dans la tolérance aux fautes. Chaque algorithme était présenté de manière abrégée suivie d'une présentation des travaux connexes réalisés dans le domaine des systèmes embarqués. Il a été déduit que les HMM sont l'outil adéquat.

Cependant, ce dernier présente des limites. Il ne traite pas les dépendances entre l'état actuel du système et son comportement précédemment observé. De ce fait, les PHMM sont proposés comme mécanismes de détection d'erreurs dans un système embarqué temps réel.

La deuxième partie de cette thèse explique en détails le modèle proposé et présente, à travers une évaluation automatique, le potentiel de ce modèle.

Partie 2 : Contribution : Application des Partly Hidden Markov Models

Chapitre 4

Partly Hidden Markov Models

Ce chapitre est une présentation détaillée des *Partly Hidden Markov Models*. Les questions d'inférence pour les PHMM sont discutées, y compris la description de la propriété d'estimation de la probabilité de vraisemblance et la méthode d'estimation des paramètres du PHMM. Ce chapitre s'appuie sur les travaux de Kobayashi *et al.* [10] [100].

4.1 Définition du PHMM

Un HMM standard est composé de deux processus, le processus de Markov caché et le processus observable. Kobayashi et Haruyama dans [10] ont modifié le deuxième processus pour obtenir ce que nous appelons aujourd'hui les *Partly Hidden Markov Models*.

Avant de décrire les PHMM proprement dits, nous commençons par expliquer c'est quoi un processus observable.

4.1.1 Processus observable

Un modèle ou processus stochastique est un processus aléatoire qui peut changer d'état s_i, $i = 1, \ldots, n$ au hasard, aux instants $t = 1, 2, \ldots, T$. Le résultat observé est la suite des états $S = s_1, s_2, \ldots, s_T$ dans lesquels il est passé. On dit que ce processus **émet** des séquences d'états. Une séquence d'état est émise avec une probabilité $P(S) = P(s_1, \ldots, s_T)$ [57].

La loi de probabilité de chaque état, à un instant t, dépend de son évolution antérieure, ainsi la loi d'évolution du modèle stochastique est définie de la manière suivante [57] :

$$P(s_1, \ldots, s_T) = P(s_1, \ldots, s_{T-1}) \times P(s_T \mid s_1, \ldots, s_{T-1}) \tag{4.1.1}$$

Si la dynamique du processus stochastique est entièrement déterminée par une probabilité initiale $P(s_1)$ et des probabilités de changement d'un état à un

autre état alors ce processus est dit **Markovien**[1] ou **chaîne de Markov**. La loi d'évolution du processus devient [57] :

$$P(s_1, \ldots, s_T) = P(s_1) \times P(s_2 \mid s_1) \times \ldots \times P(s_T \mid s_1, \ldots, s_{T-1}) \quad (4.1.2)$$

Un modèle ou chaîne de Markov est un processus qui se compose d'états s_i et de transitions. On note q_t , l'état observé à l'instant t. Ainsi, la probabilité d'être dans un état à l'instant t ne dépend que de l'état à l'instant $(t-1)$ [57] :

$$\forall t \; P(q_t = s_i \mid q_{t-1} = s_j, q_{t-2} = s_k \ldots) = P(q_t = s_i \mid q_{t-1} = s_j) \quad (4.1.3)$$

Le modèle de Markov définit que chaque état correspond à un évènement observable ou physique mais cette « propriété » ne peut représenter certains problèmes tels que la reconnaissance de la parole ou de l'écriture. Rabiner dans [4] s'est débarrassé de cette critique en proposant qu'un état n'est pas associé exclusivement à une observation donnée qu'il émettrait a coup sûr. Ainsi, chaque observation est une fonction probabiliste de chaque état du modèle. En outre, ce ne sont pas les états qui sont observés, mais les observations qu'ils émettent. Par conséquent la suite d'observation est dissociée de la suite d'état qui devient non observable. Ce nouveau modèle est appelé : **Modèle de Markov Caché** ou ***Hidden Markov Model (HMM)***.

La procédure de génération d'une séquence $O_1, O_2, ..., O_T$ de symboles à l'aide d'un HMM consiste à générer un symbole sur chaque état rencontré en utilisant la distribution de probabilité de génération associée à l'état. Ainsi, La probabilité que l'état i émette le symbole y est comme suit [10] :

$$P(O_t \mid O_{t-k}O_{t-k+1}...O_{t-1}) = P(O_t = y \mid q_t = s_i) \quad (4.1.4)$$

4.1.2 Modification du processus observable

D'après l'équation 4.1.4, Rabiner définit que la probabilité d'observation à l'instant t dépend uniquement de l'état à cet instant même. Cependant, Kobayashi *et al.* soulignent que les HMM traitent le processus stochastique stationnaire par morceaux. Ils signalent ainsi que cette propriété est limitée pour résoudre des problèmes tel que la reconnaissance des gestes humains. Ils proposent d'introduire une dépendance entre l'observation précédente et l'état actuel (figure 4.1.1).

Ainsi, sa loi d'évolution est obtenue à l'aide de la probabilité définie comme suit [10] :

1. Les processus de Markov portent le nom de leur découvreur : Andreï Markov.

$$P\left(O_t \mid O_{t-k}O_{t-k+1}...O_{t-1}\right) = P\left(O_t = y \mid q_t = s_i, O_{t-1} = x\right) \tag{4.1.5}$$

La caractérisation du processus se résume donc par l'obtention des probabilités initiales $P\left(s_0\right)$ et des probabilités des états conditionnés par leurs évolutions antérieures.

Pour résumer, on dit alors que le *Partly Hidden Markov Model* dérive du Modèle de Markov Caché standard. Il produit une séquence en utilisant deux suites de variables aléatoires ; l'une cachée et l'autre observable. A la différence des HMM standards, il définit que les paramètres des observations et les transitions d'état sont dépendants de l'observation précédente comme le montre la figure 4.1.1 :

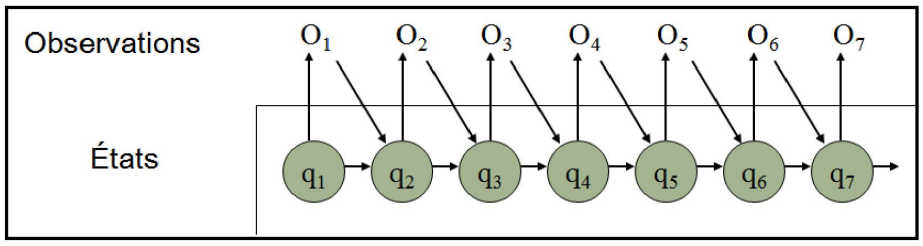


FIGURE 4.1.1 – Structure du *Partly Hidden Markov Model*

4.2 Les Paramètres du PHMM

PHMM comme processus stochastique est fortement vue comme un automate probabilistes, il est défini par une structure composée de [10] :

- $S = \{s_1, s_2, s_3, ..., s_N\}$: l'ensemble d'états,
- $V = \{x, y, z, ..., w\}$: symboles (ou alphabet) d'observation,
- On note :
  - $Q = \{q_1, q_2, ..., q_T\}$ avec $q_i \in S$: la suite des états qui a émis une séquence
  - $O = \{O_1, O_2, ..., O_T\}$ avec $O_t \in V$: et les symboles ou l'alphabet observés
- et par cinq paramètres :
  - a_{ij} : La probabilité qu'à l'instant $t+1$ l'état soit s_j sachant qu'à l'instant t est s_i : $P\left(q_{t+1} = s_j \mid q_t = s_i\right)$
  - $b_i(x)$: La probabilité d'observer le symbole x à l'instant $t-1$ alors qu'on est à l'état s_i à l'instant t : $P\left(O_{t-1} = x \mid q_t = s_i\right)$

— $c_{ij}(y)$: La probabilité d'observer le symbole y à l'instant $t-1$ sachant qu'on est à l'état s_i et qu'on sera à l'instant $t+1$ dans l'état s_j : $P\left(O_{t-1} = y \mid q_t = s_i, q_{t+1} = s_j\right)$

— $d_i\left(x, y\right)$: La probabilité d'observer le symbole x à l'instant t et le symbole y à l'instant $t-1$ sachant qu'on est à l'état s_i à l'instant t : $P\left(O_{t-1} = y, O_t = x \mid q_t = s_i\right)$

— $\pi_i\left(y\right)$: La probabilité que l'état initial est s_i et qu'on a déjà observé le symbole y : $P\left(O_0 = y \mid q_1 = s_1\right)$

Une observation dépend de deux états : l'état caché courant et l'observation précédente, ainsi [10] :

La probabilité de transition de l'état s_i à l'état s_j sachant qu'on a observé le symbole x est définie comme :

$$A_{ij} = P\left(q_{t+1} = s_j \mid q_t = s_i, O_{t-1} = x\right) \tag{4.2.1}$$

$$= \frac{P\left(q_{t+1} = s_j \mid q_t = s_i\right).P\left(O_{t-1} = x \mid q_{t+1} = s_j, q_t = s_i\right)}{P(O_{t-1} = x \mid q_t = s_i)} \tag{4.2.2}$$

$$= \frac{a_{ij}.c_{ij}(x)}{b_i\left(x\right)} \tag{4.2.3}$$

La probabilité d'observer le symbole y à l'état s_j sachant qu'on a observé le symbole x est défini comme :

$$B_j(y) = P\left(O_t = y \mid q_t = s_j, O_{t-1} = x\right) \tag{4.2.4}$$

$$= \frac{P\left(O_t = y, O_{t-1} = x \mid q_t = s_j\right)}{P(O_{t-1} = x \mid q_t = s_j)} \tag{4.2.5}$$

$$= \frac{d_j(y, x)}{b_j\left(x\right)} \tag{4.2.6}$$

Nous utiliserons la notation λ pour caractériser le modèle PHMM.

4.3 Les Trois Problèmes du PHMM

Le PHMM hérite aussi des trois problèmes du HMM où le premier problème est un problème d'évaluation, qui peut également être vu comme un problème d'estimation de la capacité d'un modèle à reconnaitre une séquence d'observations donnée. Le second problème se ramène à l'idée de déterminer la suite

d'états cachés qui maximise la probabilité d'observer cette séquence. Enfin, le troisième problème consiste dans l'entrainement (*training*) du PHMM par des séquences d'observations, en vue d'optimiser ses paramètres pour résoudre des problèmes spécifiques donnés.

4.3.1 Évaluation de la probabilité de l'observation (*Evaluation*)

Soit une séquence d'observations $O = O_1, O_2, ..., O_T$ et le modèle PHMM, quelle est la probabilité $P(O|\lambda)$ que la séquence O soit observée ? La réponse est formulée comme suit :

$$P(O|\lambda) = \sum_{Q} P(O, Q|\lambda) = \sum_{Q} P(O|Q, \lambda) \, . \, P(Q|\lambda) \tag{4.3.1}$$

Cette équation est d'une complexité polynomiale : si le PHMM possède N états et la taille de la séquence d'observation égale à T alors le calcul de la probabilité $P(O|\lambda)$ est de l'ordre de $\Theta(2TN^T)$. Une solution existe : l'évaluation ***Forward-Backward (FB)***.

Cet algorithme se compose de deux fonctions :

— La fonction « ***Forward*** » notée α : qui calcule la probabilité d'émission de la séquence d'observation partant de l'état initial s_1 jusqu'à l'état s_t.

— La fonction « ***Backward*** » notée β : qui calcule la probabilité d'émission de la fin de la séquence d'observation de l'état s_{t+1} en aboutissant à l'état final s_T.

Sachant que le calcul de α se fait avec t croissant alors que le calcul de β se fait avec t décroissant.

Ainsi le calcul de la probabilité d'observer une suite d'observations sachant λ suivant l'algorithme *Forward-Backward* est définie par :

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(j).\beta_t(j) \tag{4.3.2}$$

4.3.1.1 L'algoritthme *Forward*

Cet algorithme est appelé *Forward* car l'induction est réalisée en avant. Soit $\alpha_t(i) = P(O_0O_1 \ldots\ldots O_t, q_t = s_i|\lambda)$ la probabilité d'avoir généré le début de la séquence $O = O_1, O_2, ..., O_t$ et d'être à l'état s_i à l'instant t.

Le calcul de cette probabilité est décrit dans l'algorithme suivant :

Algorithme 4.1 Algorithme *Forward* du PHMM [10]

Initialisation

$\alpha_1(i) = P(q_1 = s_i | O_0 = y) = \pi_i(y)$; $1 \leq i \leq N$

Induction

$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i).A_{ij}.B_j(O_t)$

$= \sum_{i=1}^{N} \alpha_{t-1}(i).\frac{a_{ij}.c_{ij}(O_{t-1})}{b_i(O_{t-1})}.\frac{d_j(O_t,O_{t-1})}{b_j(O_{t-1})}$; $1 < t \leq T$ et $1 \leq j \leq N$

Terminaison

$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

4.3.1.2 L'algorithme *Backward*

La définition de l'algorithme *Backward*, ou β-pass est analogue à la définition de l'algorithme *Forward*, sauf que le processus commence depuis la fin (l'instant T) et fonctionne en arrière pour aboutir dans l'état s_j à l'instant $t+1$. La probabilité de *Backward* est définie comme suit :

$$\beta_t(i) = P(O_{t+1}.\ldots.O_T | \, q_t = s_i, \, \lambda) \tag{4.3.3}$$

Le calcul de cette probabilité est décrit dans l'algorithme suivant :

Algorithme 4.2 Algorithme *Backward* du PHMM

Initialisation

$\beta_T(i) = 1$; $1 \leq i \leq N$

Induction

$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j).A_{ij}.B_j(O_{t+1})$, $1 < t \leq (T-1)$ et $1 \leq i \leq N$

4.3.2 Calcul du chemin optimal (*Decoding*)

Le deuxième problème du PHMM consiste à trouver dans le modèle λ la meilleure suite d'états Q, qui maximise la probabilité $P(Q, O \mid \lambda)$ c'est-à dire de trouver le meilleur chemin qui émet la séquence d'observation O suivant cette probabilité.

De même que le premier problème, l'approche directe est inapplicable puisqu'elle possède une complexité de $O(N^T)$. Elle consiste à calculer la probabilité de génération suivant tous les chemins possibles et de choisir parmi ces chemins celui qui possède la probabilité la plus élevée. C'est pourquoi l'algorithme de

Viterbi existe. C'est un algorithme très similaire à celui de *Forward* et permet de trouver le chemin dit **chemin de *Viterbi***.

L'algorithme de Viterbi permet de formaliser le problème du chemin optimal en fournissant en sortie la valeur $P*$: la probabilité de l'émission de la séquence par la meilleure suite d'états $(q_1^*, \ldots, q_T^*)$.

L'approche que nous proposons dans cette thèse n'impose pas l'utilisation de l'algorithme de *Viterbi*, ainsi nous n'allons pas le détailler dans cette sous section.

4.3.3 Apprentissage du modèle (*Learning*)

Le principe de l'apprentissage automatique est de faire en sorte, à l'aide d'une procédure itérative, d'affiner le modèle.

Suivant le principe du maximum de vraisemblance, l'objectif est d'estimer les paramètres du PHMM qui maximisent la probabilité $P(O \mid \lambda)$, à partir d'un ensemble de séquences représentatives du domaine qu'on souhaite modéliser appelées séquences d'apprentissage.

Deux procédures de ré-estimation existent :

4.3.3.1 Algorithme Segmental K-means

C'est un apprentissage basé sur l'algorithme de *Viterbi* pour réajuster les paramètres du modèle qui maximisent la probabilité $P(O|\lambda, Q^*)$: la probabilité de génération suivant les chemins les plus probables.

Le principe est que la probabilité de génération d'une séquence d'observations suivant son chemin de *Viterbi* est supérieure par rapport à la probabilité de génération suivant n'importe quel autre chemin.

Cependant, cet algorithme n'est exécutable que dans le cas où les chemins de *Viterbi* sont connus. Un autre algorithme efficace est disponible, qui est une instance de l'algorithme ***Expectation-Maximisation « EM »*** : l'algorithme ***Baum-Welch***.

4.3.3.2 Algorithme *Baum-Welch*

L'algorithme *Baum-Welch* est un algorithme qui affine au fur et à mesure le modèle λ tout en maximisant la probabilité de génération d'une séquence d'observations $O = O_1, ..., O_T$.

L'algorithme cherche à ré-estimer les paramètres du modèle λ du PHMM de manière itérative :

— Choisir le modèle initial λ_0

— Calculer les paramètres du modèle λ_1 à partir de λ_0, puis λ_2 à partir de λ_1

— Répéter la procédure, tant que « la condition est vérifiée »

La procédure se réitère, jusqu'à un certain point limite atteint. Ce point est une condition qui doit être vérifiée. Cette propriété est définie avec la **fonction auxiliaire** $Q\left(\bar{\lambda}, \lambda\right)$:

$$Q\left(\bar{\lambda}, \lambda\right) = \sum_{Q} \left[P\left(Q \mid O, \lambda\right) . logP\left(O, Q \mid \bar{\lambda}\right)\right] \tag{4.3.4}$$

$\bar{\lambda}$ représente le modèle ré-estimé.

Si la valeur de $Q\left(\bar{\lambda}, \lambda\right)$ augmente alors la valeur de $P\left(O \mid \lambda\right)$ augmente, à savoir :

$$Q\left(\bar{\lambda}, \lambda\right) > Q\left(\lambda, \lambda\right) \Rightarrow P\left(O \mid \bar{\lambda}\right) > P\left(O \mid \lambda\right) \tag{4.3.5}$$

$\bar{\lambda}$ doit donc améliorer la probabilité de l'émission des observations de l'ensemble d'apprentissage. Donc la procédure se réitère tant que $P(O|\lambda_{n+1}) > P(O|\lambda_n)$

Formule de ré-estimation des paramètres du PHMM

Dans le modèle HMM standard et suivant le mode d'émission des observations, nous distinguons deux modèle de HMM : HMM Discret et HMM Continue (voir Annexe A).

Ainsi, dans les PHMM, si les valeurs du processus observé sont de type réel i.e. vecteur évalué dans un espace euclidien $O \subset \mathbb{R}$, le PHMM est qualifié de PHMM continu. Les variables b_i, c_{ij}, d_i sont représentées par la Distribution Gaussienne :$\aleph\left(O_t; \mu; \sigma\right)$. Chaque variable est définie comme suit [10] :

$$\aleph\left(O_t; \mu; \sigma\right) = \frac{1}{\sigma.\sqrt{(2\pi)}} * exp(-\frac{(O_t - \mu)^2}{2.\sigma^2}) \tag{4.3.6}$$

4.4 Domaine d'Application du PHMM

Les PHMM ont été appliqués de manière efficace dans divers domaines dont les suivants :

4.4.1 Reconnaissance du mouvement humain

Le problème de la reconnaissance des mouvements corporels humains et de la main avec le traitement d'image est une des approches les plus intéressantes dans le domaine de la reconnaissance des formes. Kobayashi et Haruyama [10] proposaient pour la première fois les PHMM pour la reconnaissance du langage des signes japonais.

6 mots de ce langage ont été choisis : "Non", "Douteux", "Au revoir", "Venez ici", "Parlez", "J'aime". Seulement les gestes de la main droite ont été utilisés

pour exprimer ces mots. Vingt sujets ont interprété les six mots du langage des signes, quatre fois chacun.

Le score du PHMM était meilleur par rapport au HMM soit 73% de réduction d'erreur.

4.4.2 Reconnaissance de la parole

Kobayashi *et al.* [100] affirmaient que si les caractéristiques dynamiques des modèles de la parole (*speech patterns*) sont prises en considération,ils devraient contribuer à améliorer la performance (augmenter la fiabilité) de reconnaissance.

Le corpus pour l'apprentissage et le test est obtenu de la base de données de parole JNAS. Il est constitué de 216 jeux de mot prononcés par 7 parleurs de sexe masculin. Les modèles HMM et PHMM sont construits avec un apprentissage de 10000 phrases. Le PHMM a donné un meilleur score, soit 93,4% de taux de détection et une réduction du taux d'erreur égale à 39% .

4.4.3 Hybridation de PHMM avec HMM pour la reconnaissance de la parole

Ogawa et Kobayashi [101] proposent une hybridation entre les PHMM et les HMM pour la reconnaissance de la parole.

Cent phrases de la base de données de parole ASJ-JNAS (chaque cinq phrases sont prononcées par un sujet différent) sont utilisés pour l'expérimentation. Le modèle hybride donne le meilleur résultat de prédiction soit 96,2%. Ce résultat réduit les erreurs de reconnaissance de mots à 25% par rapport au HMM qui donne un score égale à 94.9%. Le modèle hybride réduit aussi le taux d'erreur par 12% en le comparent au PHMM dont ce dernier donne un score de 95,5%.

4.5 Les PHMM et la Prédiction des Défaillances

L'application des Modèles de Markov Cachés et ses variantes peut être classé en deux catégories. La première traite des problèmes de reconnaissance, la seconde traite des problèmes de segmentation de séquences. Dans ce travail les PHMM sont utilisés pour traiter un problème de la première catégorie.

La reconnaissance ou l'anticipation des défaillances à base des PHMM est une solution de tolérance aux fautes attrayante pour les systèmes embarqués temps réel que nous déployons en deux étapes :

1. **Identification des caractéristiques du système révélatrices d'une future défaillance**

L'objectif de cette première étape est de déterminer quelles sont les caractéristiques qui doivent être prises en considération pour prédire la défaillance du système étudié.

1.1. Étude de cas

Le comportement défaillant d'un système embarqué temps réel est identifié par des symptômes d'erreurs.

De ce fait, il est indispensable de comprendre le système à étudier pour déterminer quelles sont les caractéristiques révélatrices d'une future défaillance. Le chapitre suivant est entièrement consacré à l'étude de cas.

Une fois cette tâche réalisée, l'étape suivante est la collecte des données puisque l'apprentissage ne peut pas se faire sur un corpus vide.

1.2. Collecte des données

Après une étude approfondie du système à étudier, on peut avoir une idée sur les données qui doivent être utilisées. Ces données ne sont pas toujours facile à obtenir. Dans cette étape, on s'intéresse à comment générer et collecter les données. L'apprentissage automatique ne demande pas toujours l'utilisation de toutes ces données. Ainsi une réduction des données s'impose. Dans certains cas des attributs du corpus seront éliminés.

1.3. Prétraitement

Les données collectées doivent être « préparées » ou « nettoyées ». Des données peuvent être omises à cause des erreurs de frappe ou à causes des erreurs dues au système de surveillance.

1.4. Structure du corpus à utiliser

Le composant de base d'un processus d'apprentissage est l'ensemble d'exemples représentants le corpus de données à étudier. Chaque exemple est représenté sous forme d'une ligne (un enregistrement) caractérisée par un ensemble d'attributs. En final, les enregistrements sont regroupés dans une table ou matrice.

Plus le nombre d'exemples est important, meilleure est la précision du résultat donné.

Il existe deux types d'apprentissage automatique : apprentissage supervisé et apprentissage non supervisé. Notre approche est de la première catégorie. Dans l'apprentissage supervisé, les classes sont prédéterminées et les exemples connus, le système apprend à classer selon un modèle de classement (prédire la classe). Un expert doit préalablement étiqueter les exemples.

2. Application du modèle proposé

Une fois les données sont prêtes, le modèle proposé (PHMM) intervient. Cette procédure se déroule en trois phases : phase d'initialisation, phase de *training* et phase de prédiction.

La dernière phase consiste à prédire la classe d'un enregistrement (type de défaillance que représente la séquence de l'observation) avec le modèle préalablement appris. L'enregistrement n'est pas associé à une classe de défaillance

unique mais à une probabilité d'appartenance à chacune des classes prédéterminées.

2.1. Phase d'initialisation

Avant de lancer l'apprentissage, des modèles (objets) λ_i sont créés à partir de la classe PHMM (figure 4.5.1 et 4.5.2), chaque modèle représente un type (classe de défaillance). Pour chaque modèle, des paramètres dits « paramètres d'entrée » doivent être initialisés : le nombre d'états cachés et le type de transitions entre états.

PHMM
+N : entier +M : entier +T : entier +séquence [] : réel #A[][] : réel #B[][] : réel #C[][][] : réel #D[][][] : réel #Pi [] : réel
+réel DistGauss() +GetParam() +ReadObs() +InitParam() #Forward(O [] : réel) #Backward(O [] : réel) #réel CalProba(O [] : réel) #Estimate(O [] : réel)

FIGURE 4.5.1 – Classe PHMM

DistGauss : Fonction qui calcule la distribution gaussienne.
GetParam : Fonction qui lit le nombre d'états (N), la taille de la séquence (T), définit si le modèle est Ergodique ou Gauche-Droit
ReadObs : Fonction qui lit une séquence d'observation.
InitPram : Fonction qui initialise les cinq paramètres du PHMM.
Forward : Fonction Forward.
Backward : Fonction Backward.
CalProba : Fonction qui calcule la probabilité de vraisemblance pour une séquence d'observation O.
Estimate : Fonction qui réajuste les paramètres du modèle suivant une séquence d'observation O. C'est l'algorithme Baum-Welch.

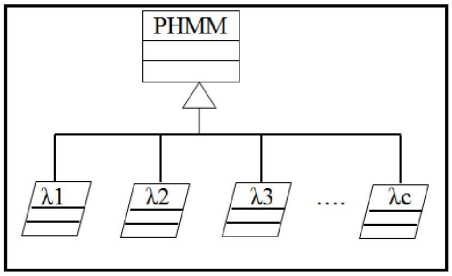


FIGURE 4.5.2 – Diagramme d'héritage du PHMM

2.2. Phase de *Training*

Pour chaque classe de défaillance, il existe un modèle de PHMM (objet de la classe) qui la représente. Ainsi, les paramètres de chaque modèle sont affinés à partir d'un ensemble d'enregistrements avec l'algorithme de *Baum-Welch.* Pour cette raison que la phase « étude de cas » est très importante. Elle permet d'identifier avec exactitude le type d'observation révélatrices d'une défaillance utilisée pour le *training* et la prédiction.

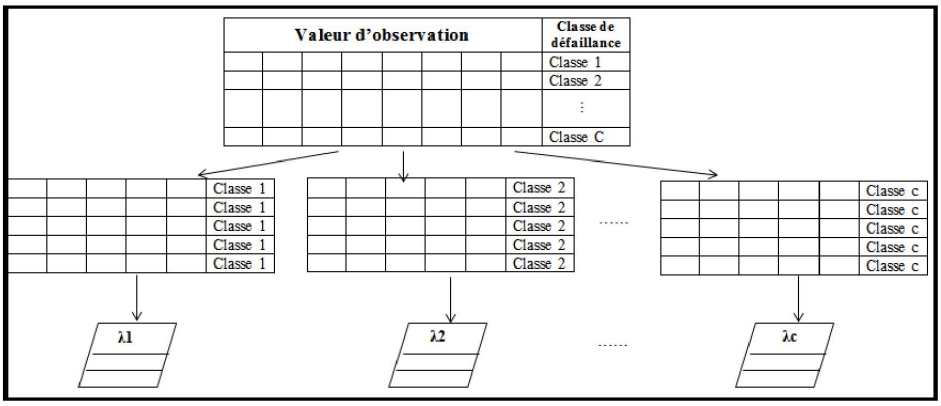


FIGURE 4.5.3 – Modélisation de la phase de *Training*

λ_i *: objet de la classe PHMM représentant le type de défaillance* i*.*

2.3. Phase de prédiction

Une fois les modèles obtenus, des séquences non étiquetées (non attribués à une classe de défaillance) sont évaluées. Pour chaque séquence, une probabilité d'appartenance à une classe de défaillance est calculée avec l'algorithme *Forward-Backward.* La séquence est attribuée à la classe de défaillance dont elle possède la probabilité de vraisemblance la plus élevée.

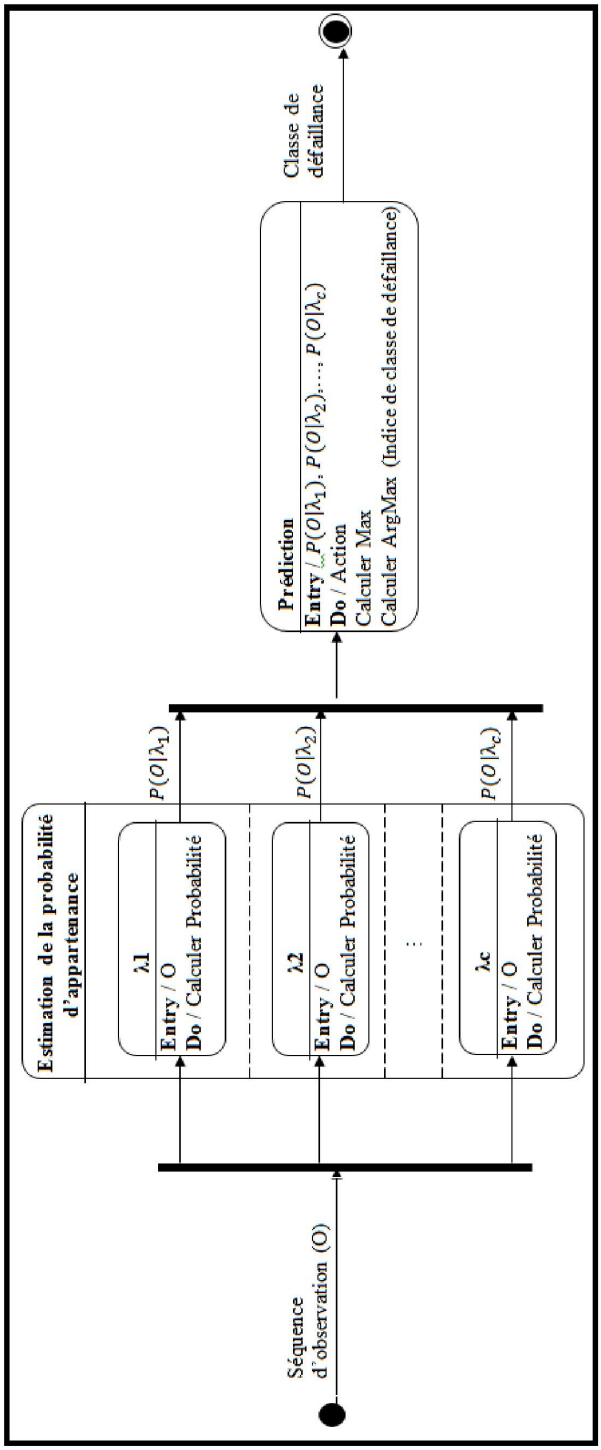


FIGURE 4.5.4 – Diagramme d'états de la phase de prédiction

Remarque

Pour déterminer les besoins de l'algorithme proposé (PHMM) en temps d'exécution lors de la prédiction une évaluation de sa complexité calculatoire s'impose.

4.6 Étude de la complexité

Le temps requis par un algorithme est proportionnel au « nombre d'opérations de base » qu'il exécute. La complexité calculatoire ne s'intéresse pas au nombre exact des opérations qui sont effectuées mais de la façon dont le nombre d'opérations est lié à la taille du corpus de données. Elle cherche à déterminer si dans le cas la taille du problème double, est ce que le nombre d'opérations reste le même ? peut doubler ? ou augmenter d'une autre manière ?

Dans cette étude, nous nous intéressons au « pire des cas » « *worst case* » : quel est le plus grand nombres d'opération qui doivent être effectuées pour une taille de problème donné ?

La prédiction des défaillance à base des PHMM fait appel à deux algorithmes :
- l'algorithme de *Baum-Welch* pour la phase de *traning.*
- l'algorithme de *Forward-Backward* pour la phase de prédiction.

a) La complexité de l'algorithme *Forward* est de l'ordre $O(N^2T)$. Pour chaque séquence d'observations de taille T et pour chacun des N états cachés, une sommation de N termes est calculée . Ce calcule est réalisé dans un modèle ergodique (entièrement connectés) mais cette complexité devient de l'ordre de $O(NT)$ pour le modèle gauche-droite (*left-to-right).*

b) L'exécution de l'algorithme *Baum-Welch* dépends du nombre des paramètres du modèle, le nombre de réitération et le nombre d'instances utilisées pour le *traning.*

Le PHMM possède cinq paramètres.
- π : possède $N\,M$
- A : possède N^2 transitions (dans le cas d'un modèle ergodique)
- B : possède $N\,M$
- C : possède $N^2\,M$ transitions (dans le cas d'un modèle ergodique)
- D : possède $N\,M^2$

Puisque M est déterminé par l'application donc elle est supposée être constante. Par conséquent, l'estimation de chaque paramètres est :
- $O(N)$ pour π
- $O(N^2T)$ pour A
- $O(NT)$ pour B
- $O(N^2T)$ pour C

— $O(NT)$ pour D

Ainsi la complexité calculatoire des PHMM est de l'ordre de $O(N^2T)$. Si nous avons n instances appelées pour le *traning* alors la complexité du PHMM est de l'ordre de : $n * O(N^2T) = O(N^2 T\, n)$ dans le cas d'un modèle ergodique et de l'ordre de : $n * O(NT) = O(N\, T\, n)$ dans le cas d'un modèle gauche-droite.

Finalement, en ajoutant le critère du nombre de réitération, la complexité des PHMM est aussi de l'ordre de $O(N^2\, T\, n\, I)$ pour le modèle ergodique et de $O(N\, T\, n\, I)$ pour le modèle gauche-droite.

La complexité des PHMM est de même ordre que les HMM, sauf qu'il faut prendre en considération que les PHMM possèdent deux paramètres supplémentaires par rapport au HMM. C'est un facteur à prendre en considération pour le temps du *traning* du PHMM. C'est à dire que le temps pour le PHMM sera supérieur au HMM standard.

Sachant que la taille de la séquence T et le nombre d'états cachés N sont déterminés précédemment, lors de l'étude de cas, et ne risquent pas de changer lors de l'apprentissage et de la prédiction, le facteur qui influera sur le temps d'exécution est bien sur le nombre d'instances utilisées pour le *traning*.

4.7 Conclusion

Dans ce chapitre le principe des *Partly Hidden Markov Models* a été expliqué. De manière détaillée les paramètres du modèle ont été présentés.

Par la suite, les étapes de l'approche proposée a été expliquée et comment les algorithmes *Baum-Welch* et *Forward-Backward* interviennent pour prédire la défaillance d'un système embarqué temps réel.

Cette approche sera mise en œuvre sur un four de production de verre de la société ALVER. En raison de son importance, une usine de verrerie dépend essentiellement de la fiabilité de son four de production pour répondre à la demande du marché.

Chapitre 5

Étude de cas : Verrerie d'Oran

Après avoir exposé la partie tolérance aux fautes de la thématique, ce chapitre présente la deuxième partie soit le système embarqué.

En raison de son importance, le secteur de la verrerie industrielle est choisi, plus particulièrement, le four de production. Ce chapitre s'appui sur une étude de terrain afin d'identifier le type d'observation à utiliser pour l'expérimentation.

5.1 Industrie du Verre Mécanique en Algérie

L'industrie du verre mécanique compte en Algérie **une** seule usine dont son chiffre d'affaires hors taxe en 2013 a été plus de 6 milliard de Dinars Algérien pour une production totale supérieure à 38,000 tonnes dont plus du quart est exporté. Ils existent différents secteurs de production : le verre plat, les fibres, le verre creux et le verre technique, ALVER (l'usine en question) est une manufacture spécialisée dans l'industrie du **verre creux** en tant qu'emballage (bouteilles [1] , flacons, pots, bocaux).

5.2 Processus de Production

Le processus de fabrication du verre comporte cinq principales étapes :

1. Les matières premières (calcin, silice et soude) sont malaxées dans l'atelier de composition (figure 5.2.1) avec des proportions soigneusement pesées. Ce mélange est transporté à l'aide de tapis roulant jusqu'au four pour être fusionné.
2. La fusion : le mélange vitrifiable (matières premières) est introduit en continu dans le four pour être transformé. La composition est chauffée progressivement jusqu'à 1450°C-1600°C dans le **bassin de fusion**. Un

1. La bouteillerie représente en tonnage la part essentielle de la production du verre creux de l'usine.

refroidissement contrôlé (la température baisse jusqu'à 1000°C - 1200°C) est réalisé dans le **bassin de travail** (appelé aussi l'avant-bassin ou distributeur). Cette étape permettra ainsi la transformation du mélange en une masse liquide de verre homogène de **viscosité adéquate** pour la mise en forme.

3. Moulage ou formage : La coulée de verre est transformée, à l'aide de ciseaux automatiques, en gouttes de verre, appelées « paraissons », nécessaire à la confection de la bouteille.
4. Recuit et refroidissement : tous les types de verre subissent un refroidissement, c'est la recuisson. Cette opération est réalisée à l'aide d'un four spécial appelé arche de recuisson. Le produit devient froid et solide.
5. Triage : cette tâche est réalisée par un service de contrôle de qualité. Les articles de mauvaise qualité sont détectés et rejetés pour un éventuel recyclage. La procédure de contrôle permet d'éliminer les bouteilles qui ne répondent pas aux exigences de la commercialisation. Il existe plusieurs catégories de contrôles :

   a) Contrôle dimensionnel : vérification de la longueur totale, les diamètres intérieurs et extérieurs et la planéité du goulot.

   b) Contrôle d'aspect : détecter la présence d'imperfection comme les bulles, les grains non-fondus.

   c) Contrôle de capacité : vérification du poids de la bouteille.

   d) Contrôle de propriétés mécaniques : résistance à la pression interne, à l'écrasement vertical dû au bouchage. Ce test n'est effectué que sur un échantillon de la production, en raison de sa relative lenteur et parce qu'ils impliquent parfois la destruction de l'article.

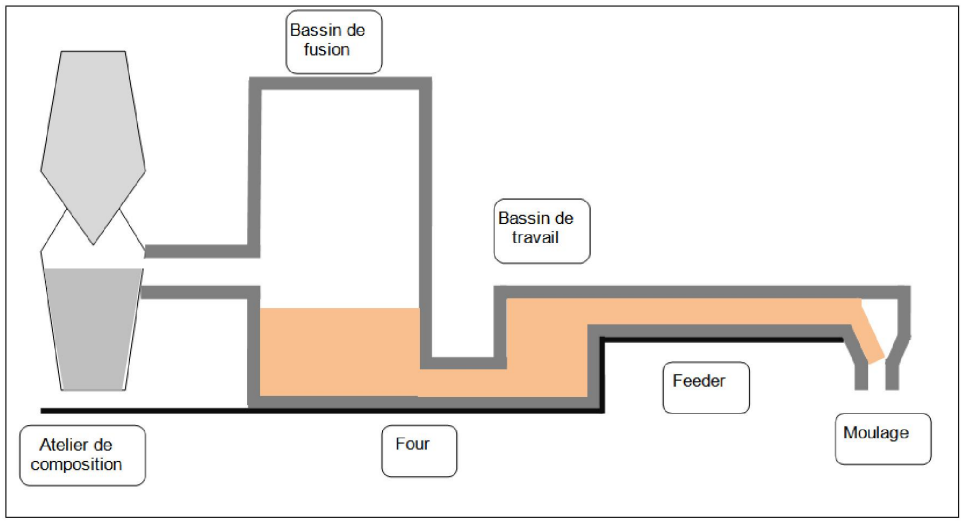


FIGURE 5.2.1 – Processus de fabrication du verre creux

Dans le processus de production, le facteur le plus important est la viscosité. En effet, le procédé de formage est alimenté par le liquide de verre (la composition fusionnée) qui doit avoir une viscosité convenable. Ce facteur est très dépendant de la température et cette dépendance varie selon une loi exponentielle [102]. La figure 5.2.2 montre la viscosité du verre en fonction de sa température.

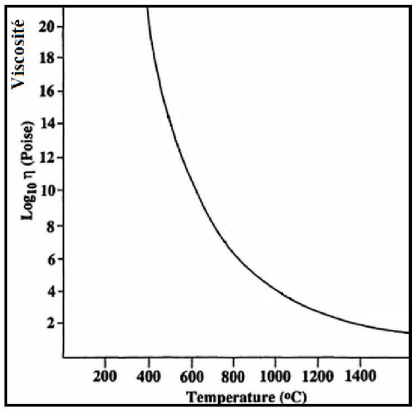


FIGURE 5.2.2 – La viscosité du verre en fonction de sa température [102]

C'est pourquoi la propriété thermique est très importante. Elle influe sur la déformation du verre i.e. des variations inattendues dans la température

engendrent des défauts dans le produit final.

Décidément la fabrication du verre est un travail qui nécessite la fiabilité du four. Cette fiabilité impose aussi à ce que la machine ne soit jamais éteinte (pour une réparation en cas de défaillance grave) car ceci peut provoquer des pertes monétaires énormes à l'entreprise. [103]

5.3 Four de Production

Tout dysfonctionnement du four a des conséquences sur la rentabilité de la production. L'usine se voit, dans l'obligation, de surveiller et de contrôler ses équipements en temps réel afin d'anticiper tout défaut. Pour une surveillance continue du fonctionnement du four, des interfaces sont installées dans l'ensemble de la chaîne de production (figure 5.3.1). Deux types d'interfaces existent :

— **Interface distant**e : il s'agit d'un système informatique installé dans un poste de travail distant. Il permet d'observer et diagnostiquer l'état de l'équipement machine à travers des capteurs et d'agir sur lui grâce à des actionneurs (vannes, vérins, moteurs, etc.).

— **Interface locale** : il s'agit d'interface installée sur l'équipement machine. Elle offre une interaction directe entre l'homme et la machine. Un contrôle manuel peut être exécuté sans avoir recours au système de contrôle distant (exemple : ouverture d'une vanne à la main).

Pour réduire l'intervention humaine dans le processus de production, il existe des composants électroniques intégrés dans des machines industrielles. Au niveau du fours industriel, la température est une grandeur très importante qui doit être surveillée et contrôlée. Elle influe sur les matériaux à fabriquer et sur le four lui-même. Le capteur utilisé pour mesurer la température est un thermocouple. Lorsque la valeur de la température subit des perturbations ou des variations, un composant électronique intervient pour régler la température dans le four, appelé **régulation automatique**.

5.4 Régulation Automatique

5.4.1 Définition

Aujourd'hui dans les procédés industriels, le contrôle d'un certain nombre de grandeurs physiques comme : la température, la pression, le débit, le pH , la température, etc est nécessaire (figure 5.4.1). Ainsi, des composants sont intégrés ou embarqués dans les machines. Le but est de maintenir ses grandeurs à un niveau prédéterminé appelé **consigne de régulation**.

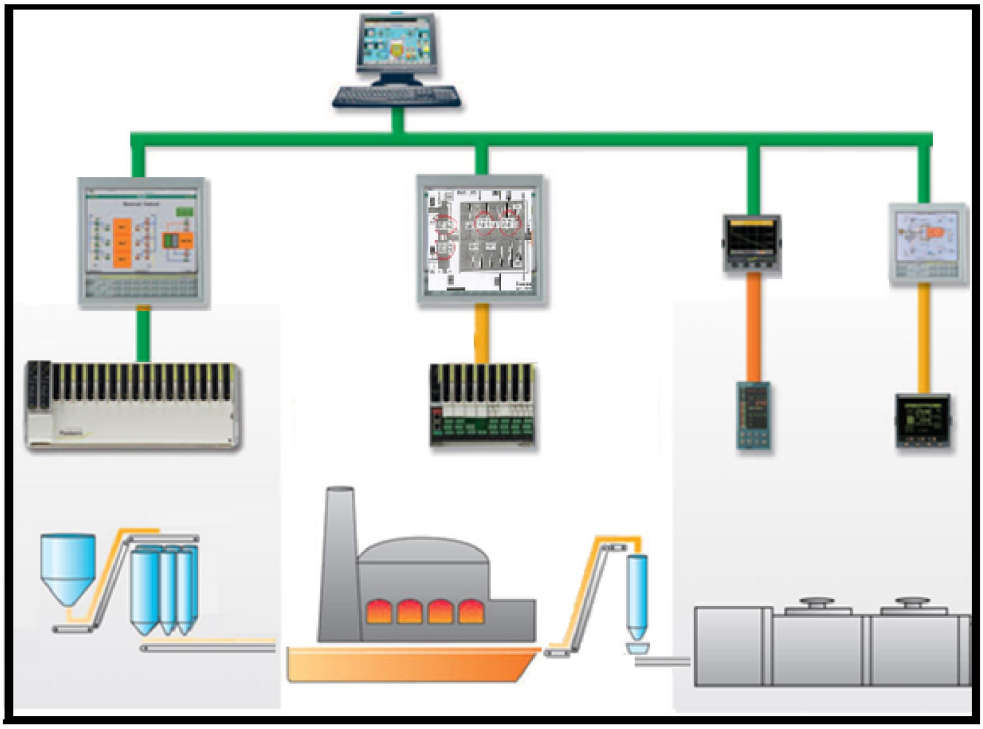


FIGURE 5.3.1 – Surveillance de l'installation de production du verre

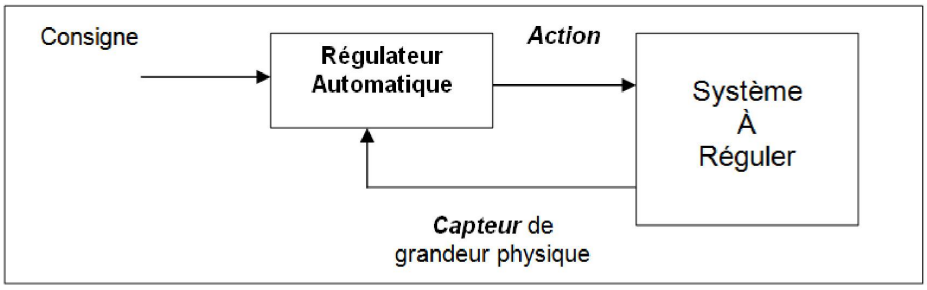


FIGURE 5.4.1 – Schéma général de la régulation automatique

La régulation automatique comprend trois éléments indispensables :

1. **Capteur** : les grandeurs physiques utilisées doivent être mesurées pour pouvoir agir sur le **système à réguler**. Cette opération est réalisée à l'aide des capteurs. Le capteur est un dispositif qui permet la transformation d'une grandeur physique en une grandeur électrique. Le but est de rendre exploitable ces grandeurs physiques pour un traitement ultérieur. Le signal électrique résultant peut être de type analogique ou numérique (figure 5.4.2). Un capteur fiable est caractérisé par la précision et la rapidité des valeurs qu'il donne.

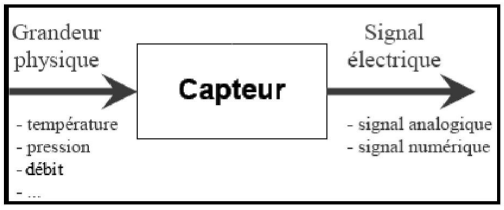


FIGURE 5.4.2 – Structure générale d'un capteur

2. **Régulateur** : l'organe de régulation est composé de deux entités : le comparateur et le correcteur (figure 5.4.3). La première entité compare la grandeur capturée appelée **grandeur à contrôler** à la **valeur de consigne.** Dans le cas d'une non concordance, le correcteur envoie un signal commande à l'unité de contrôle (vanne, moteur, etc.), pour que cette dernière agisse sur le système à réguler. La grandeur qui subit un réglage est appelée **grandeur réglante.**

3. **Actionneur** : c'est l'organe de contrôle. [44]

La présence d'une boucle dans le processus de régulation est obligatoire. Le but est de maintenir constamment la grandeur contrôlée conformément à la consigne.

5.4.2 Régulation automatique dans le four industriel

Dans le four de verrerie, la température (**la grandeur réglée**) est mesurée en permanence grâce aux thermocouples (figure 5.4.4). En suite la grandeur est comparée à la consigne de régulation. En fonction de l'écart mesure/consigne (M,C), le régulateur donne une sortie qui agit sur une vanne automatique (actionneur) qui règle le débit air/gaz.

Le régulateur est l'entité « intelligente » dans la boucle de régulation car la partie correcteur est dotée d'une loi de commande (algorithme). Dans notre étude de cas, le régulateur utilisé dans le four est un régulateur électronique *« Honeywell 2300 »* possédant un algorithme **PID** « Proportionnel Intégral Dérivé »**.** C'est le régulateur le plus utilisé dans l'industrie [104].

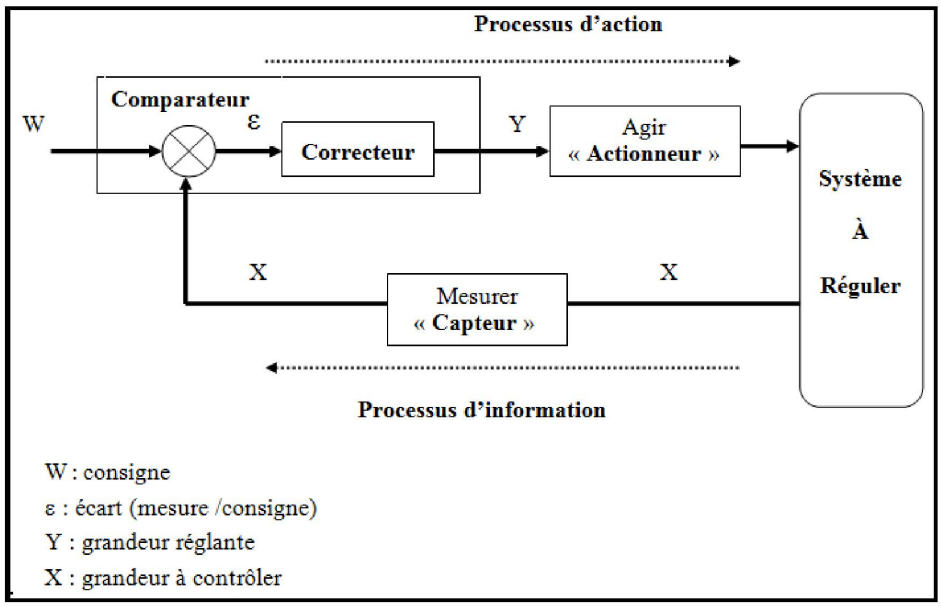


FIGURE 5.4.3 – Structure de base dans un système régulé

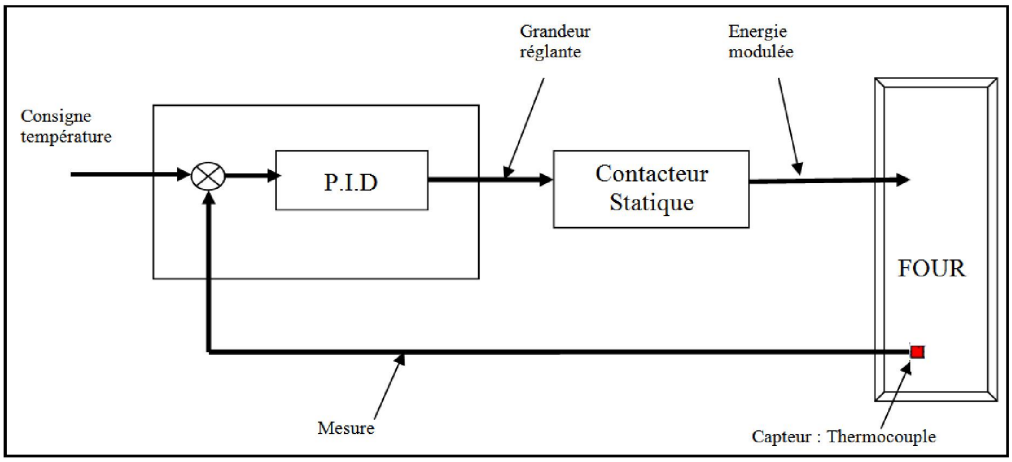


FIGURE 5.4.4 – Régulation de la température dans un four

L'entreprise emploie un nombre important de cadres, techniciens et employés, chargés des tâches administratives, commerciales, des travaux d'analyse et de contrôle physicochimique. Cependant, un groupe de recherche spécialisé dans le domaine informatique lui y est indispensable pour le développement des techniques qui optimisent la production de l'usine même dans la présence

d'une dégradation physique du matériel. Le but est de minimiser l'impact négatif du dysfonctionnement en proposant de nouvelles approches applicables .

5.4.3 Thermocouple

Un thermocouple ou couple thermoélectrique (CTE) transforme une température en une tension électrique mesurable. Il est constitué de deux conducteurs, de nature différente, soudés bout à bout en circuit fermé. Une force électromotrice (f.e.m.) exprimée en micro ou millivolts (μV ou mV) est générée lorsque la soudure est chauffée. Cette mesure en μV est ensuite traduite en degrés centigrades, c'est l'élévation de température.

5.4.3.1 Types de thermocouples

Suivant la nature des deux conducteurs, il existe différent types de thermocouples. Dans notre étude de cas, nous nous intéressons à seulement trois types installés tout au long de la chaîne de production :

Type B : Installer dans le bassin de travail et de fusion.

Composition : platine-rhodium (30 %) / platine-rhodium (6 %).

Usage de 0 °C à 1 600 °C.

Précision est de ± 0,5% de 870°C à 1600°C. La mauvaise précision est largement observée en dessous de 600 °C.

Type K : Installer au niveau des récupérateurs et canalisation.

Composition : Chromel (alliage nickel + chrome) / Alumel (alliage nickel + aluminium (5 %) + silicium).

Usage continu de 0 °C à 900 °C.

Stabilité moins satisfaisante que d'autres thermocouples : Précision est de ± 3°C (entre 0°C et 400°C) et ± 0.75% (400°C - 1250°C).

Type S : Installer dans le Feeder.

Composition : platine-rhodium (10 %) / platine.

Usage de 0 °C à 1 350 °C.

Résistance élevée à la corrosion et à l'oxydation.

Se pollue facilement.

Précision est de ± 2.5°C (0°C - 600°C) et de ± 0.4% (600°C -1600°C).

5.5 Modèle prédictif

Malgré la présence du régulateur PID dans le four, pour maintenir la température à un niveau souhaité, cela n'empêche pas que des variations de température légères auront des conséquences sur le four et sur la qualité du produit

fabriqué. Le régulateur PID ne réagit que lorsque l'erreur a été observée donc il ne prédit pas une erreur. En raison de la présence d'un retard est peut être considéré comme important, les performances d'un PID deviennent insuffisantes posant des problèmes de fiabilité. Ainsi, il faut faire appel à un algorithme d'apprentissage et de prédiction. Le but n'est pas de remplacer le régulateur PID mais de lui associer un autre mécanisme. Cette assemblage permettra d'optimiser encore plus la fiabilité du four de production. Le but est de permettre une meilleure maîtrise du procédé de production en réduisant le coût supplémentaire imposé par la mauvaise production. Sachant que les produits ayant des défauts sont envoyés pour un recyclage [2].

La température est un symptôme révélateur de l'état du four. Ainsi, le PHMM s'appuiera sur la signature thermique pour l'apprentissage et la prédiction des défaillances du four. La figure 5.5.5 montre les échanges de messages entre acteurs et le modèle de manière chronologique :

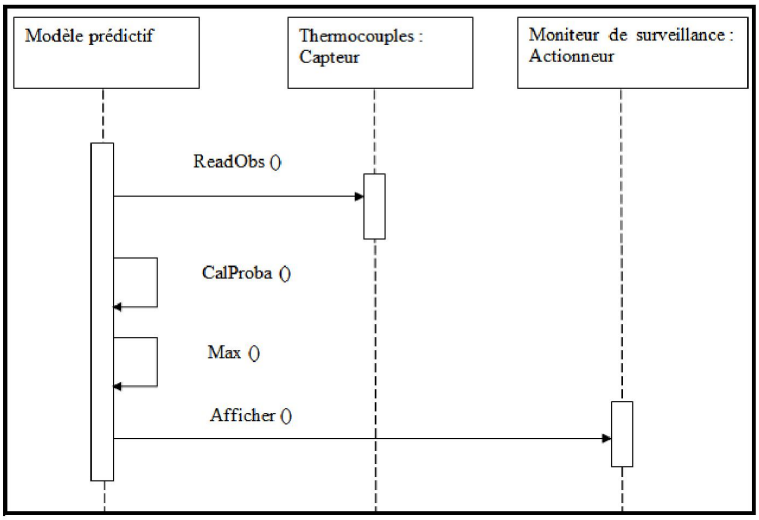


FIGURE 5.5.1 – Diagramme de séquence du mécanisme proposé

Le modèle prédictif lit une séquences de température qui a été mesurée par les thermocouples. Cette séquence est soumise aux différents modèles PHMM pour calculer la probabilité de vraisemblance. La probabilité la plus élevée est sélectionnée. Finalement un message est afficher dans le moniteur de surveillance du four indiquant le type de défaillance.

2. En 2013, le taux de recyclage s'élevait à 35 %.

5.6 Conclusion

L'industrie du verre est une industrie lourde qui dispose d'installations de production de gros tonnage et les usines ne peuvent pas se permettre de s'offrir de nouveaux équipements à chaque dysfonctionnement. En conséquence, la compétition actuelle dans le monde industrielle se base sur l'innovation de nouvelles techniques leurs permettant de préserver le même matériel de production tout en respectant les exigences du marché.

Dans ce chapitre, l'étude de cas sur un four industriel a été clairement expliquée et l'importance de l'approche proposée a été montrée. La section suivante est la partie expérimentation où les étapes suivies pour la collecte des données thermiques seront détaillées.

Chapitre 6

Expérimentation

L'évaluation des performances du modèle proposé dans cette thèse est présenté dans ce dernier chapitre. Nous commençons par détailler la phase de collecte et prétraitement des données. Une étude comparative entre les PHMM et les algorithmes d'apprentissage a été effectuée, en particulier avec les PHMM.

6.1 Collecte des Données

Le four subit des fautes de type matériel due à l'environnement agressive et l'ancienneté du matériel (le four date de 1960). Les variations de température provoquées par l'état dégradé du four engendrent des viscosités dans le produit.

Suivant la terminologie des entraves de la sûreté de fonctionnement : une faute n'est observée qu'à travers une erreur, donc, une faute survenue dans un four est observée grâce à la température de ce four. Ainsi, les signaux thermiques représentent le comportement du four à l'échelle du temps.

L'objectif de la prédiction de défaillance est de déduire un futur défaut suivant des symptômes observés dans le four. Pour ces symptômes, il s'agit d'un ensemble de valeurs de températures (erreurs observées) car une seule erreur observée n'est pas suffisante pour en déduire si une défaillance est imminente ou non.

Pour l'acquisition des températures, le four est directement relié à un ordinateur muni d'une application de surveillance (figure 6.1.1). Les mesures de températures sont prises à partir de quatre capteurs (thermocouples) différents : Zone1 et Zone2 dans le basin de fusion et point de haut et point de bas dans le bassin de travail [103] :

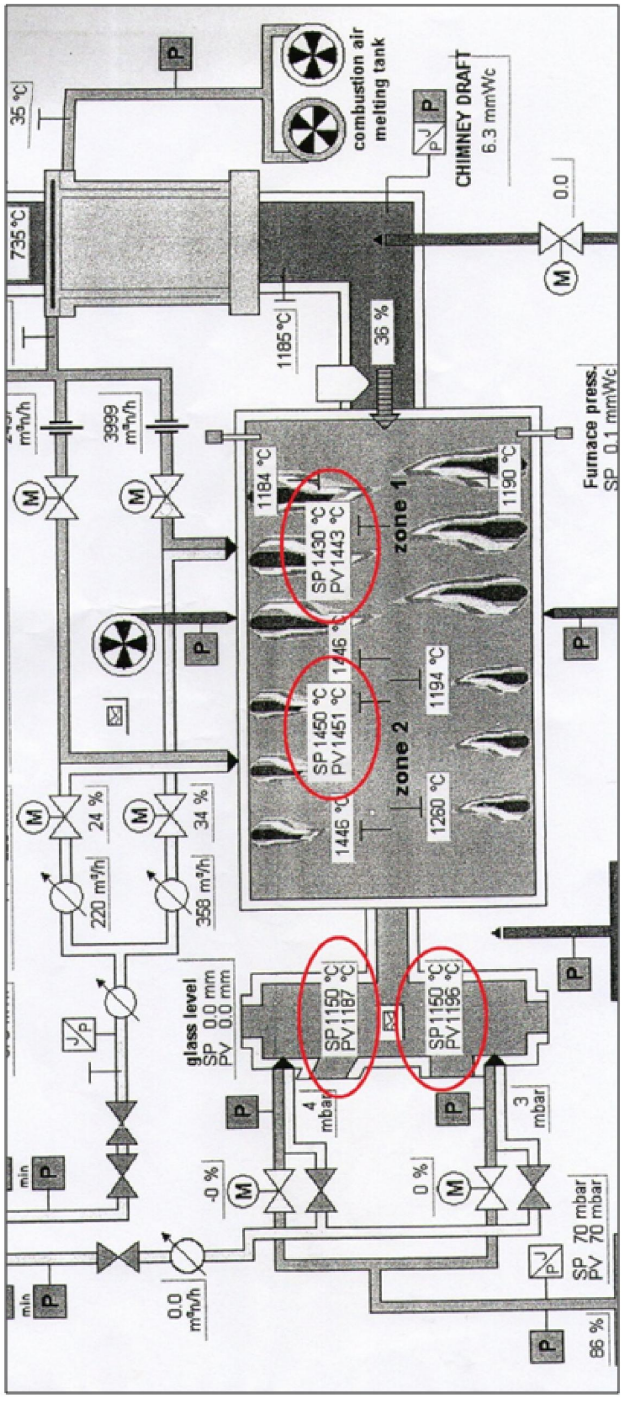


FIGURE 6.1.1 – Conception du Four
En face au milieu basin de fusion et à gauche de la figure bassin de travail. Deux thermocouples Type B installé dans chacun des deux basins.

Le système de surveillance ne possède pas d'application qui permet l'enregistrement des températures sur un support physique, ce qui nous a obligé à effectuer une acquisition manuelle.

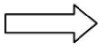
Évaluer un algorithme de prédiction automatique est une tâche délicate car elle dépend fortement de l'exactitude des données d'apprentissage. Un temps non négligeable a été consacré à la phase de collecte et d'enregistrement.

Les mesures de température sont recueillies chaque heure sur une période de 333 jours (figure 6.2.1). Au total, 31 968 valeurs de température ont été enregistrées. Ces valeurs sont de type numérique. Ainsi, elles ne nécessitent aucun type de pré-traitement tel que l'écart où la normalisation. [103]

6.2 Agrégation des Données en Classes de Défaillance

Pour l'organisation des horaires de travail, l'usine applique le 3 x 8[1]. La quantité et la qualité de la production de chaque équipe n'est mesurée qu'à la fin de la journée. Par conséquent, nous avons obtenu pour chaque jour, trois matrices de 8 lignes et 5 colonnes correspondant aux températures prises au niveau de chaque capteur, où chaque matrice (figure 6.2.1) lui est associée le taux de bon tonnage.

Heure	BF.1	BF.2	BT.G	BT.D
08h00	1567	1527	1151	1160
09h00	1504	1533	1154	1162
10h00	1510	1532	1152	1161
11h00	1512	1529	1149	1159
12h00	1505	1533	1151	1159
13h00	1505	1531	1150	1159
14h00	1498	1531	1156	1164
15h00	1508	1531	1161	1170



Bon Tonnage
61%

FIGURE 6.2.1 – Valeurs de température enregistrées associée au taux de bon tonnage produit
(BF.1 : Basin de fusion-Zone1, BF.2 : Basin de fusion-Zone2, BT.G : bassin de travail-Gauche, BT.D : bassin de travail-Droit)

1. Trois équipes tournent par roulement de huit heures consécutives sur un même poste, pour assurer un fonctionnement continu sur les 24 h d'une journée.

Remarque

Le calcule du taux de bon tonnage est réalisé après la phase de tirage, du processus de production (voir section 5.2). Il représente le nombre d'articles de bonne qualité (conformes aux exigences de la commercialisation) divisé par le nombre totale d'articles fabriqués, comme le montre la formule suivante :

$$taux\,de\,bon\,tonnage = \frac{nombre\,d'articles\,de\,bonne\,qualité}{nombre\,totale\,d'articles\,fabriqués} * 100$$

Pour la classe de défaillance, nous avons présenté dans la section 2.2 du chapitre 2 que les défaillances sont classées suivant un des critères : le degré de gravité, la fréquence de leurs apparitions et leur détectabilité. Dans notre cas d'étude et selon l'équipe d'évaluation, les défaillances sont classées suivant leur niveau de gravité, comme indiqué dans le tableau suivant [103] :

Niveau de gravité	**Impact sur l'équipement**
Mineure	Matériel intact
Moyenne	Équipement rapidement réparable
Majeure	Réparation longue
Catastrophique	Dégâts matériels importants, l'environnement autour de la machine est affecté
Inacceptable	Outil de production hors usage, (nécessité de remplacement)

TABLE 6.1 – Classification des défaillances

Remarque

Notons que contrairement aux approches citées dans [90] [93] et [95], la classe de défaillance « non-défaillant » est omise du fait que le four étudié est un matériel ancien qui présente certaines déficiences.

En raison de l'absence d'application intégrée au système de contrôle nous permettant de prendre une telle mesure, nous étions contraint de solliciter les connaissances d'un expert de l'entreprise pour trouver une alternative.

Il ressort de cette discussion que si le four est usé, cela aura un impact sur la qualité de la production. Plus la défaillance est importante plus les conséquences seront désastreuses sur la production de verre. Ainsi, le niveau de gravité de la défaillance est mesuré en termes de taux du mauvais tonnage comme le montre le tableau 6.2 :

Niveau de gravité	Taux du mauvais tonnage
Mineure	0%-20%
Moyenne	20%-30%
Majeure	30%-50%
Catastrophique	50%-60%
Inacceptable	60%-100%

TABLE 6.2 – Classification des défaillances
Taux du mauvais tonnage =100% - Taux de bon tonnage

L'étape de collecte des données étant achevée, nous essayons maintenant de valider de manière automatique la qualité de l'approche proposée par le biais d'une expérimentation. Pour évaluer un algorithme d'apprentissage il existe deux méthodes :

1. **En terme de valeurs prédictives** : Plusieurs métriques sont utilisées pour évaluer les performance du modèle comme : la précision, le rappel ....etc.
2. **En terme de probabilité de reconnaissance** : Le principe est d'observer en temps réel si le modèle arrive à anticiper une future défaillance.

Un système informatique ne peut être intégré dans un système électronique tant qu'il n'a pas été testé en *off-line* et prouvé son efficacité. Pour cela, nous évaluerons l'algorithme proposé en terme de valeurs prédictives afin de montrer son apport par rapport au HMM standard et par rapport à d'autres algorithmes d'apprentissage pour la prédiction des défaillances.

6.3 Implémentation

L'expérimentation a été effectuée sous WEKA [2]. Cet outil est une librairie d'algorithmes d'apprentissage et de Data Mining conçue entièrement en Java. Il est diffusé sous licence publique GNU et considéré à la pointe de la technologie de l'application des techniques d'apprentissage. WEKA prend en charge plusieurs tâches d'exploration de données différentes telles que le prétraitement des données, la classification, le regroupement, la régression et la visualisation [105].

6.3.1 Intégration du PHMM dans WEKA

WEKA en version native n'inclut pas les PHMM, cela nous a conduit à modifier le code source du module HMM afin d'intégrer le modèle PHMM.

2. Le weka est disponible gratuitement à l'adresse `www.cs.waikato.ac.nz/ml/weka`, dans des versions pour Unix et Windows.

Le module HMM est une classe Java (weka.classifiers.bays.HMM) qui n'est pas incluse par défaut dans le WEKA. Le paquet HMMWeka doit être téléchargé puis intégré dans le logiciel. Nous avons étendu le code source du HMM pour la prise en compte du modèle PHMM. La figure 6.3.1 montre une partie du diagramme de classe de WEKA. Seul les classes avec lesquelles la classe PHMM interagit sont représentées. Pour éviter toute exhaustivité dans le diagramme, les méthodes les plus importantes de chaque classe sont affichées :

6.3.2 Prétraitement des données

WEKA utilise le format de fichier **arff** (*Attribute-Relation File Format*) pour traiter les données. Un fichier arff est composé d'une liste de vecteurs (instances) définis par leurs valeurs d'attributs. Les deux principaux types d'attributs utilisés pour notre corpus de données sont : `REAL` (pour les valeurs de températures) et `Nominal`(pour la classe défaillance) : `@ATTRIBUTE Failure {Unacceptable, Catastrophic, Major, Mean, Minor}` signifie que la classe ***Failure*** peut avoir comme valeur : *Unacceptable*, *Catastrophic*, *Major*, *Mean* ou *Minor*.

Le fichier d'entrées, doit être composé d'un ensemble de vecteurs, chaque vecteur est un ensemble de valeurs de température associé à une valeur de la classe de défaillance. Ainsi, chaque matrice obtenue lors de la phase de collecte de données (figure 6.2.1) doit être réduite en un vecteur. Une simple moyenne des valeurs de chaque colonne de la matrice peut fausser le traitement. La solution est la concaténation des lignes de chaque matrice pour éviter toute perte d'information.

L'attribut « Heures » a été omis car il n'a aucun effet sur le résultat. Ainsi, chaque matrice de 8 lignes et 5 colonnes devient un vecteur de 32 valeurs (8 * (5-1) = 32).

En final, notre fichier d'entrée est composé de 999 instances avec 33 valeurs (32 valeurs de température + la classe de défaillance).

6.3.3 Validation croisée

Pour déterminer la performance du modèle proposé, la validation croisée reste depuis plus de 40 ans la solution adéquate. Le principe de cette technique est que le corpus de données (échantillon) soit divisé en deux, une partie pour le *Training* et l'autre partie pour la Prédiction.

Un ensemble de données de taille trop petite, réservé au *Training*, est susceptible de produire des résultats erronés. La question est comment déterminer la taille minimale de l'ensemble de données qui servira pour le *Training*.

La méthode *« testset validation »* permet de diviser le corpus en échantillon de *Training* ($> 60\%$ du corpus) et échantillon de Prédiction. Mais, des instances qui font partie de l'échantillon de test ne seront pas pris en considération

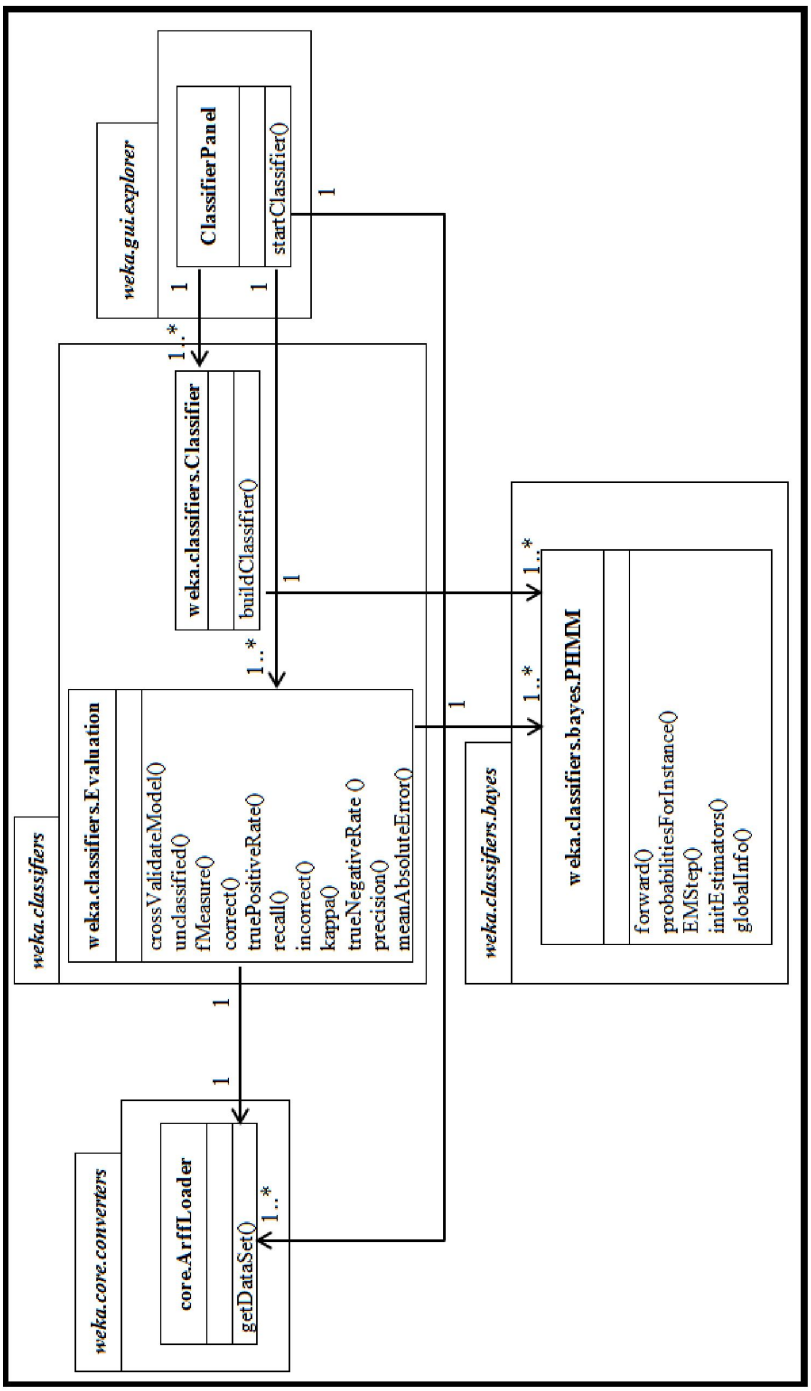


FIGURE 6.3.1 – Partie du diagramme de classe de WEKA

par le processus de *Training* pour déterminer les paramètres du modèle. Pour cela une autre méthode de validation croisée existe ***M**-fold cross-validation.*

L'ensemble de données est divisé en M groupes d'échantillons ou clusters. Un des M clusters est sélectionné pour la Prédiction et les $(M-1)$ autres clusters pour le *Training*. Par la suite, le cluster de Prédiction est intégré dans le groupe de cluster de *Training* et un autre cluster du groupe de *Training* servira pour la Prédiction. L'opération se répète ainsi M fois pour qu'à la fin chaque instance de la base de données a servi au moins une fois pour le *Training*. Dans ce travail, la méthode *M-fold cross-validation* est choisie.

Les algorithmes d'apprentissage dépendent de leurs paramètres, s'ils sont mal ajustés alors tout le résultat donné sera remis en question.

6.3.4 Paramètres d'ajustement

Tous les algorithmes d'apprentissage supervisé et non supervisé dans WEKA, possèdent des paramètres qui doivent être ré-ajustés avant l'exécution de l'algorithme. Naturellement, avant de lancer la prédiction avec les PHMM, les paramètres impliqués dans la modélisation : le type de validation, type de transition d'états et le nombre d'états cachés, sont ajustés de telle sorte que la performance de prédiction de défaillance optimale soit atteinte.

6.3.4.1 Transition d'état

Grâce à l'algorithme *Forward- Backward*, l'évaluation de la valeur de vraisemblance $P(O \mid \lambda)$ est calculée suivant deux variables α et β :

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i).\beta_t(i)$$

Dans le jargon technique, les HMM (de même pour les PHMM) ergodiques sont utilisés pour modéliser les systèmes stationnaires, tandis que les modèles gauche-droite sont utilisés pour des comportements transitoires de modèle [3].

Dans le temps, l'état de four peut s'aggraver ou rester le même, mais ne peut pas s'améliorer automatiquement sans intervention humaine ou automatique. Ainsi, l'option "*left-to-right*" est sélectionnée et la probabilité de vraisemblance est calculée comme suit :

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

6.3.4.2 Nombre d'états

Le nombre d'états est déterminé par un processus de sélection manuelle, dans lequel les PHMM ont été exécutés avec différents nombre états cachés et différents nombres de cluster pour la méthode *M-fold cross-validation.*

3. L'Annexe A explique ces deux type d'HMM.

Nombre de cluster	Nombre d'états	Taux de bonne classification %	Erreur absolue
2	5	70,7946	0,1202
	7	70,9091	0,1178
	8	70,9072	0,1158
	10	71,3043	0,1118
	12	71,9048	0,1147
	14	71,9231	0,1186
	16	72,2222	0,1192
	18	72,8000	0,1175
3	5	72,5000	0,1191
	7	72,5000	0,1148
	8	73,7643	0,1134
	10	73,6842	0,1138
	12	73,8622	0,1141
	14	73,8836	0,1134
	16	73,9653	0,1135
	18	74,0448	0,1141
4	5	73,6518	0,1139
	7	73,9069	0,1134
	8	73,5336	0,1146
	10	73,8752	0,1133
	12	73,9323	0,1131
	14	73,8416	0,1136
	16	73,7481	0,1134
	18	74,0741	0,1132
5	5	73,8654	0,1122
	7	73,7679	0,1131
	8	73,9903	0,1128
	10	74,0558	0,1129
	12	74,0558	0,1131
	14	74,0933	0,1146
	16	74,0933	0,1134
	18	74,1235	0,1136
6	5	73,1242	0,1151
	7	73,4062	0,114
	8	73,2839	0,1133
	10	73,3459	0,1139
	12	73,4104	0,1136
	14	73,0845	0,1148
	16	73,3467	0,1145
	18	73,2653	0,115

Nombre de cluster	**Nombre d'états**	**Taux de bonne classification %**	**Erreur absolue**
7	5	73,125	0,1162
	7	73,1915	0,1154
	8	73,4783	0,1151
	10	73,1111	0,1159
	12	73,5000	0,1166
	14	73,5897	0,1168
	16	73,6842	0,1171
	18	74,0541	0,116
8	5	72,9032	0,117
	7	73,4286	0,1196
	8	73,8889	0,1183
	10	73,0000	0,1195
	12	73,1034	0,1191
	14	73,9445	0,1142
	16	73,8706	0,113
	18	73,7948	0,1136
9	5	73,8423	0,1134
	7	73,7643	0,1134
	8	73,9694	0,1133
	10	73,9694	0,1142
	12	74,156	0,114
	14	74,156	0,1138
	16	74,1359	0,1137
	18	74,3048	0,1137
10	5	74,2574	0,1124
	7	74,4887	0,1128
	8	74,4887	0,1128
	10	**74,5746**	**0,1134**
	12	74,5746	0,1134
	14	74,5746	0,1134
	16	74,5746	0,1134
	18	74,5746	0,1134
11	5	74,5165	0,1144
	7	74,5165	0,1144
	8	74,5165	0,1144
	10	74,5746	0,1136
	12	74,5746	0,1138
	14	74,5746	0,1144
	16	74,3383	0,1132
	18	74,2698	0,1134

TABLE 6.3 – Sélection du nombre de clusters et nombre d'états cachés

A partir du tableau 6.3, la valeur de M (*M-fold cross-validation)* est fixée à 10 et N à 10 états, ils maximisent le taux de bonne classification avec un taux d'erreur faible.

6.3.5 Résultat et Interprétation

6.3.5.1 Validité prédictive

La notion de validité prédictive est très important. Car, dans le cas du temps réel, c'est le résultat donné par l'algorithme qui déterminera si le système embarqué (dans notre cas le four) doit continuer à fonctionner normalement ou des initiatives doivent être prisent pour améliorer le fonctionnement du four.

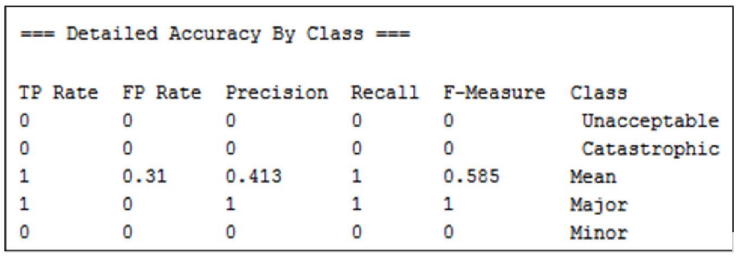
Métrique	**PHMM**
TP Rate	74,6%
FP Rate	5,6%
Précision	64,1%
Rappel	74,6%
F-measure	0,671

TABLE 6.4 – Les valeur prédictives du PHMM

Le taux de Faux Positive est très faible soit 5,6%, ceci montre que le PHMM produit très peut de fausses alarmes.

La précision mesure la qualité des résultats trouvés, soit la capacité du modèle à ne trouver que la bonne classe et le rappel consiste à évaluer la performance du modèle à trouver toutes les classes de défaillance. Ainsi, avec un rappel élevé, le PHMM a correctement prédit 74,6% de toute les classes de défaillances.

La précision et le rappel sont intimement liés. Pour tenir compte de cette relation, un compromis existe, le F-mesure. Ce dernier est la moyenne harmonique de la précision et de rappel (voir Annexe B), plus la valeur de F-mesure est grande plus est la qualité de la prédiction (c'est la performance du système). Avec un F-mesure egale à 0,671 , la qualité de prédiction du PHMM est acceptable.



=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0	0	0	0	0	Unacceptable
0	0	0	0	0	Catastrophic
1	0.31	0.413	1	0.585	Mean
1	0	1	1	1	Major
0	0	0	0	0	Minor

FIGURE 6.3.2 – Mesures d'exactitude par classe du PHMM
(Unacceptable : Inacceptable, Catastrophic : Catastrophique, Major :Majeure, Mean :Moyenne, Minor :Mineure)

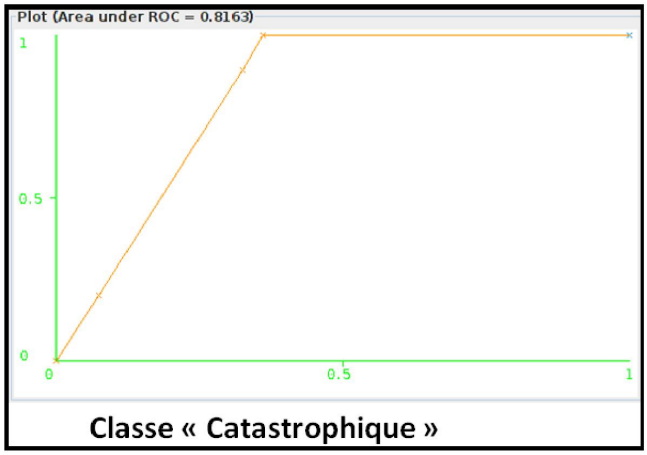
Les valeurs prédictives dépendent du taux de prédiction positif d'un côté, et du nombre d'instances qui représente une classe. Ainsi, si une classe ne possède pas suffisamment d'instances qui la représentent il sera difficile pour le modèle de donner des résultats de prédiction satisfaisants. Dans notre expérimentation, la base de données ne possède pas beaucoup d'instances pour la classe « ***Minor*** » ce qui explique les valeurs nuls dans la table de la figure 6.3.2. On dit que la défaillance « ***Minor*** » est moins fréquente.

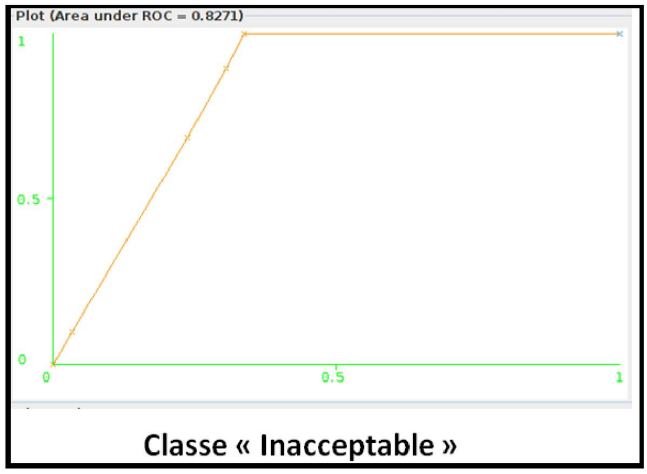
Remarque

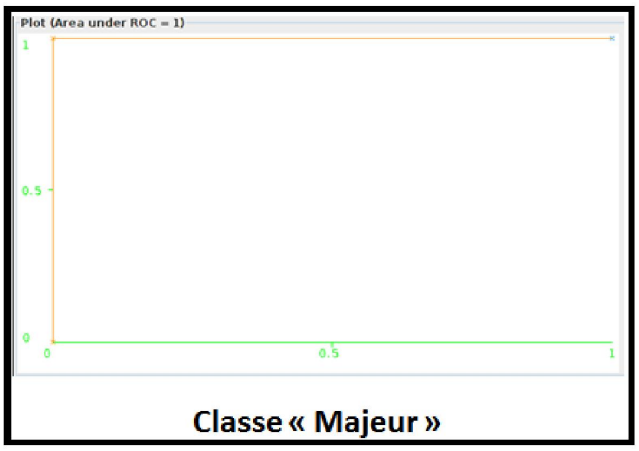
La prédiction d'une future défaillance est la classification d'une séquence d'observation avec une certaine probabilité dans une classe. Ainsi un bon modèle de prédiction est un modèle avec un taux de classification optimal.

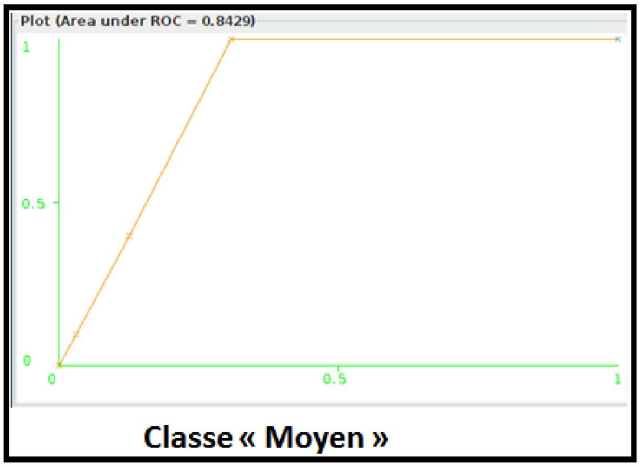
6.3.5.2 Courbe ROC

La courbe ROC est une représentation graphique du taux de vrais positifs par rapport au taux de faux positifs. Elle permet d'afficher un indicateur qui est l'aire sous la courbe (AUC). Plus cette dernière se rapproche de 1, plus le modèle de prédiction est performant. La figure 6.3.3 montre la courbe ROC du modèle de prédiction pour chaque classe de défaillance :


Plot (Area under ROC = 0.8163)
1
0.5
0
0
0.5
1
Classe « Catastrophique »


Plot (Area under ROC = 0.8271)
1
0.5
0
0
0.5
1
Classe « Inacceptable »


Plot (Area under ROC = 1)
1
0.5
0
0
0.5
1
Classe « Majeur »



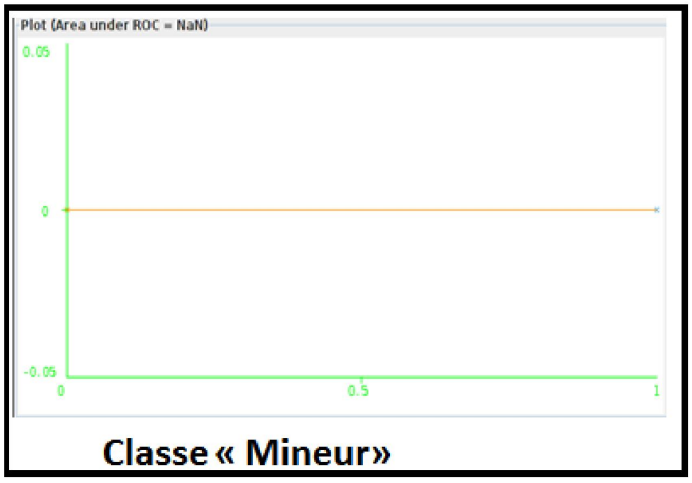


FIGURE 6.3.3 – Courbe ROC

6.3.5.3 Performances de classification

Métriques d'évaluation	PHMM
Instances correctement classées	745 (74,57 %)
Instances incorrectement classées	254 (25,43 %)
Erreur moyenne	0,1134
Kappa	0,57

TABLE 6.5 – Performances de classification du PHMM

Le coefficient *Kappa* mesure le degré de concordance de plusieurs juges. Dans WEKA, il y a deux juges : le modèle de prédiction et la classe réelle. Fleiss dans [106] propose une échelle de degré d'accord suivant la valeur du kappa :

Valeur de Kappa	**Accord**
>0.75	Excellent
0.40-0.75	Bon
<0.40	Mauvais

TABLE 6.6 – Degré d'accord selon le kappa

Nous rapportons pour le PHMM, depuis les tableaux 6.5 et 6.6, un *Kappa* excellent donc l'accord donné par le PHMM est considéré comme « bon ».

6.3.5.4 Comparaisons avec d'autres algorithmes

1) Validité prédictive

Afin d'être en mesure de juger la validité du PHMM, une étude comparative s'impose avec d'autre algorithmes d'apprentissages : les HMM, les RNA, les SVM et le C4.5. Le résultat de la comparaison est résumé dans le tableau 6.7.

1. **HMM :** les même paramètres du PHMM sont appliqués pour le HMM, soit : nombre d'états cachés =10, transition d'état *: left-to-right.*

2. **RNA :** les RNA sont implémentés sous le non de *MultiLayerPerceptron.* Le meilleur résultat est obtenu si le nombre de neurones de la couche cachée est égal au nombre de valeurs. L'erreur absolue est égale à 0,0019.

3. **SVM :** LibSVM propose quatre fonctions de noyau : Linéaire, Polynomial, Fonctions à Base Radiale (RBF) et Sigmoïde. Le SVM donne un résultat optimal avec les trois premières fonctions de noyau, soit 100% de bonne classification avec une erreur absolue égale à 0. Cependant avec la fonction Sigmoïde, le modèle donne 56,66 % de bonne classification avec une erreur absolue égale à 0,1734.

4. **C4.5** : L'algorithme C4.5 est implémenté sous le nom de J48. Cette algorithme n'impose aucun ajustement des paramètres d'entrée avant l'exécution.

La validation croisée est la même pour les quatre algorithmes : *10-fold cross-validation.*

Métriques d'évaluation	RNA	C4.5	SVM	HMM	PHMM
TP Rate	100%	100%	100%	56,7%	74,6%
FP Rate	0%	0%	0%	56,7%	5,6%
Précision	100%	100%	100%	32,1%	64,1%
Rappel	100%	100%	100%	56,7%	74,6%
F-measure	1	1	1	0,41	0,671
Instances correctement classés	999	999	999	567 (56,66%)	745 (74,57 %)
Instances incorrectement classés	0	0	0	433 (43,34%)	254 (25,43 %)
Erreur moyenne	0,0019	0	0	0,246	0,1134
Kappa	1	1	1	0	0,57

TABLE 6.7 – Comparaison du PHMM avec RNA, C4.5, SVM et HMM

Le PHMM atteint un taux de Vrai Positif (TP Rate) supérieur au HMM et un taux de Faux Positif (FP Rate) inférieur, cela signifie que le PHMM arrive à mieux prédire les classes de défaillance tout en déclenchant moins de fausses alarmes par rapport au HMM. Selon la Précision et le Rappel, la performance du PHMM est meilleure par rapport au HMM (tableau 6.7). Ainsi, l'acquisition de nouvelles données améliora certainement plus les performances du PHMM.

Malgré que les résultats données par les algorithmes RNA, C4.5 et SVM sont meilleurs par rapport aux résultats du PHMM, il n'en demeure pas moins qu'ils sont des modèles non-adéquats pour le deuxième critère que nous avons fixé à savoir :le temps.

2) Temps d'exécution

	RNA	C4.5	SVM			HMM	PHMM
Temps d'exécution (seconds)	8,69	0,10	Linéaire	Polynomial	RBF	0,06	0,07
			0,11	0,12	0,13		

TABLE 6.8 – Temps d'exécution du RNA, C4.5, SVM et PHMM

Le temps d'exécution des PHMM est supérieur à celui des HMM, la raison est que le PHMM possède plus de paramètres par rapport au HMM. L'affinement de ces paramètres lors de l'apprentissage nécessite un temps supplémentaire par rapport à l'affinement des paramètres du HMM.

Remarque

Toute l'expérimentation a été réalisée sur un ordinateur de bureau avec processeur Core ™ I5 2410M CPU 2,30 GHz et 4 Go de mémoire.

6.4 Conclusion

Un bon outil de prédiction de défaillance est un outil qui couvre autant de défaillances que possible tout en générant le moins de fausses alarmes que possible. Ce chapitre a présenté une méthode d'évaluation automatique, permettant de se passer d'une évaluation couteuse et incommode imposée par un expert. Avec un F-mesure égale à 67,1% et un temps d'exécution plus court que les algorithmes d'apprentissage (RNA, C4.5, SVM), le PHMM semble être un modèle potentiel pour la prédiction des défaillances dans un système embarqué temps réel.

Suivant le résultat dans le tableau 6.2, nous remarquons que le PHMM donne des résultat optimales avec 10 états cachés. Ainsi, le modèle a distingué 10 états différents représentant l'état (*Health-state*) du four.

Conclusion Générale et Perspectives

Les défaillances du système embarqué sont le fruit d'une ou plusieurs fautes se manifestant dans le système sous forme d'erreurs. Parallèlement, la plupart des systèmes embarqués sont critiques puisqu'ils doivent remplir leurs fonctions malgré la présence d'erreurs. Dans cette thèse, un nouveau mécanisme de tolérance aux fautes pour les systèmes embarqués temps réel est proposé. Ce mécanisme est basé sur les *Partly Hidden Markov Models* (PHMM). Il permet de prédire la défaillance sur une étude de cas réel : le four industriel de la verrerie d'Oran.

Sachant que la température représente le caractère furtif d'erreur i.e. le comportement du système, les PHMM utilisent la signature thermique pour l'apprentissage automatique.

La première étape de toute production scientifique, et de tout projet d'ingénierie est la déclaration appropriée du problème à résoudre à travers un état de l'art approfondi. Une telle étude nous a demandé du temps, soit deux ans pour pouvoir synthétiser et rapporter cet état de l'art. Les différents types de mécanismes de tolérance aux fautes et leurs caractéristiques ont été présentés et expliqués. En se basant sur deux critères importants : le coût et le temps, les mécanismes d'apprentissage semblent être les plus appropriés pour la tolérance aux fautes dans les systèmes embarqués temps réel. Ils permettent d'anticiper une future défaillance en offrant la possibilité de gagner du temps sans imposer un coût supplémentaire et important à l'architecture du système.

Les *Partly Hidden Markov Model*s (PHMM) sont proposés comme mécanisme pour la prédiction des défaillances. Ce modèle permet de conditionner la probabilité d'observation de l'état actuel du système à l'observation de l'état précédent.

Le choix du four n'est pas fortuit. L'industrie du verre dans notre pays fait face à une demande croissante et dépend essentiellement de la fiabilité de son four de production pour répondre à la demande du marché. Pour cette raison, l'étude de cas s'est portée sur un four industriel. Cette approche proposée permettra à l'usine de préserver le même matériel de production tout en respectant les exigences du marché en terme de qualité et de quantité.

La collecte des données nécessaires à l'expérimentation n'était pas une tâche aisée. Il a fallu procéder à une saisie manuelle imposant ainsi un temps non négligeable.

L'application du modèle PHMM sur un cas industriel a été réalisée avec succès et les résultats d'expérimentations (comparaison avec les Arbres de décisions, les Réseau de Neurones Artificiels, les Séparateurs à Vaste Marge et les HMM standards) obtenus montrent l'efficacité des PHMM en termes de taux de prédiction correcte et de temps de réponse.

Bien que le problème ait été posé depuis longtemps, la fiabilité posera toujours un défi pour les systèmes embarqués temps réel. À partir des résultats de cette thèse, plusieurs pistes sont envisageables pour exploiter davantage les PHMM pour la prédiction des défaillances dans les systèmes embarqués temps réel. Comme perspectives, nous pouvons noter :

1) Intégration du mécanisme proposé et Apprentissage en ligne

La prédiction de défaillance n'améliore pas la fiabilité d'un système si elle n'est pas accompagnée d'une action active, ce qui est au-delà de la portée de cette thèse. Par conséquent, nous chercherons dans un travail ultérieur à intégrer le mécanisme proposé dans le système étudié.

De plus, la comportement de tout système temps réel change fréquemment ainsi son mode de défaillance peut être modifiée. La conséquence est que le modèle PHMM obtenu à partir de l'apprentissage hors ligne peut devenir obsolète. La solution est l'apprentissage en ligne. Avec ce type d'apprentissage, le modèle est mis à jour en permanence de sorte qu'il s'adapte aux changements dans le système. Cette solution permet au PHMM de recueillir de nouvelles séquences d'observation pour affiner au mieux ses paramètres.

Un système embarqué est caractérisé par sa structure statique. Il serait difficile d'intégrer une nouvelle fonctionnalité dans le système sans compromettre son bon fonctionnement. Le paradigme Orientée Aspect est la solution à ce problème. Cette technique de programmation permet d'intégrer des exigences non-fonctionnelles dans le système indépendamment de son code métier. Il traite séparément les préoccupations transversales (*cross-cutting concerns*) des préoccupations métier. Ainsi, le modèle proposé est intégré dans le système de contrôle du four avec la Programmation Orientée Aspect.

2) Recherche du chemin optimal

Pour une nouvelle séquence d'observations, il serait intéressant de connaitre la suite d'états par laquelle le système est passé pour générer cette séquence.

Si le PHMM ne présente aucune probabilité d'émission nulle alors n'importe quelle séquence d'états de longueur T peut générer n'importe quelle séquence

d'observations de même longueur. Ainsi, il existe plusieurs séquences d'états possibles pour une séquence d'observations donnée. Pour déterminer avec certitude laquelle a généré cet séquence d'observations, il suffit de choisir la plus probable.

Annexes

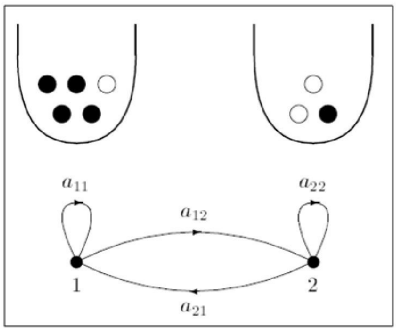
Annexe A

Modèles de Markov Cachés

Modèles de HMM

Suivant le mode d'émission des observations, nous distinguons deux modèle de HMM : HMM Discret et HMM Continu [4].

1. **HMM Discret (DHMM)**
   Lorsque les variables du processus observable prennent leurs valeurs dans un espace discret, la séquence d'observations résultant obéit à un HMM discret.

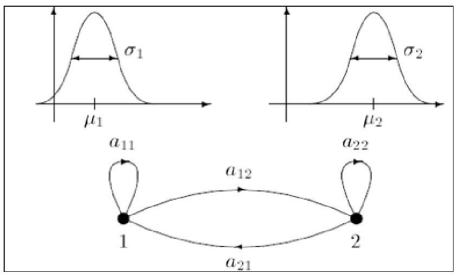


HMM Discret [108]

Avec HMM de type discret, l'ensemble des distributions conditionnelles de la matrice B peut être réduit à :

$$b_i(j) = P\left(O_t = j \mid q_t = s_i\right) = b_{ij} \quad 1 \leq i \leq N, 1 \leq j \leq M$$

2. **HMM Continu (CHMM)**
   Si les valeurs du processus observé sont de type réel i.e. vecteur évalué dans un espace euclidien ($O \subset \mathbb{R}$), le HMM est qualifié de HMM continu.



HMM continue [108]

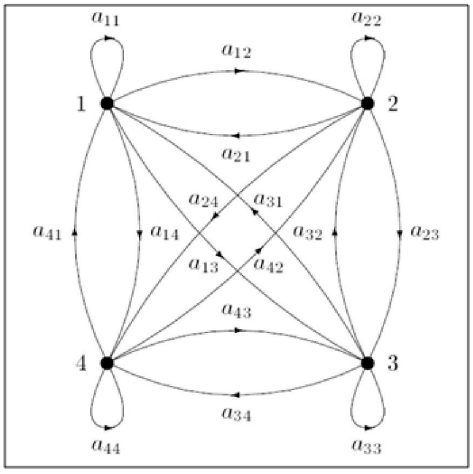
Dans un HMM continu, la matrice B correspond à une famille de fonctions paramétrique et la probabilité d'émission est appelée : la densité d'émission de l'état. La probabilité d'observation à l'état i est représentée par la Distribution Gaussienne :

$$\aleph\left(O_t;\mu;\sigma\right) = \frac{1}{\sigma.\sqrt{(2\pi)}} * exp(-\frac{(O_t-\mu)^2}{2.\sigma^2})$$

Types de HMM (Architecture des HMM)

Suivant le mode de transition des états dans le processus de Markov caché, les HMM sont classés en deux types particuliers : les modèles de Markov cachés ergodiques et des modèles de Markov cachés gauche-droite (*Left-to-right*).

1. **Modèle Ergodique**
   Dans le modèle ergodique, chaque état peut être atteint à partir de n'importe quel autre état de la chaine de Markov. Ce modèle est dit être « entièrement connecté » :



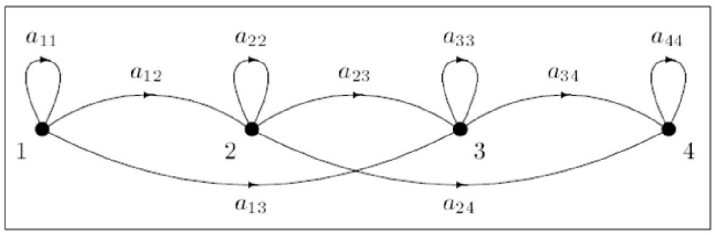
HMM ergodique [108]

Ainsi toutes les transitions d'un état vers un autre sont possibles et la matrice A est caractérisée par des coefficients strictement positif : Soit $a_{ij} > 0, \forall i, j \in S$.

2. **Modèle gauche-droite**

   Également appelé « Modèle de Bakis », ce modèle n'autorise aucune transition d'un état vers un autre état d'indice inférieur : Soit : $a_{ij} > 0$, si $i < j, \forall i, j \in S$.

   La chaîne de Markov évolue dans l'ordre croissant et un état restant ne peut pas être réexaminé plus tard.



HMM gauche-droite [108]

Variantes de HMM

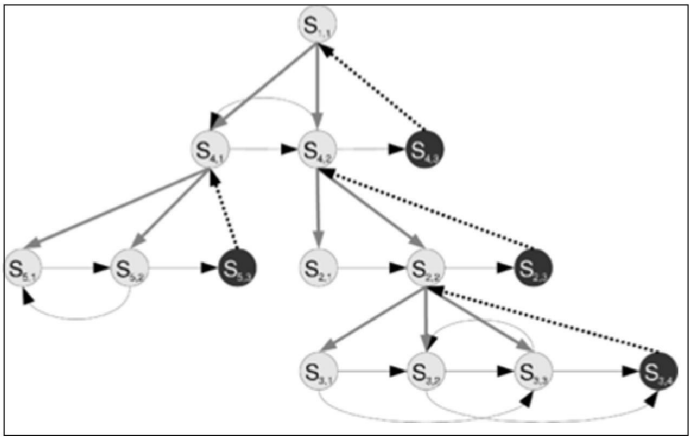
Les HMM sont limités pour résoudre certains problèmes. Ce qui a mené au développement de différentes variantes du HMM.

1. **Modèle Semi-Markov Caché**

En anglais *Hidden Semi Markov Model (HSMM)* est un modèle statistique dont la structure est analogue à la structure du HMM sauf que le processus de Markov Caché est remplacé par un processus semi-Markov. La probabilité de transition entre état caché dépend de la quantité de temps qui s'est écoulée depuis l'entrée dans l'état actuel. L'implémentation des Modèles Semi-Markov Cachés est plus difficile que dans les HMM standards. [87]

2. **Modèle de Markov Caché Hiérarchique**

Introduit par Fine *et al.* [109], les HHMM (*Hierarchical HMM*) définissent que chaque état est un modèle probabiliste autonome soit un HMM en lui-même. Ainsi chaque état n'émette pas un symbole (une observation) comme c'est le cas pour les états du HMM standard mais plutôt des séquences d'observations. La figure suivante schématise très bien les HHMM :



Structure des HHMM [109]

Quand un état appelé « états internes » est activé, il activera son propre HMM qui à son tour activera son HMM sous-jacente et ainsi de suite. Les Processus se répète jusqu'à arriver à un état appelé « état de production ». Une fois l'état de la production a émis un symbole, le contrôle revient à l'état qui la activé. Ce passage d'un état interne à un autre état (qui l'a activé) est appelé une « transition verticale ». La transition entre deux états du même niveau est appelée « transition horizontale ». Quand une transition horizontale conduit à une "terminaison" le contrôle est retourné à un état de niveau supérieur qui a produit la dernière transition verticale.

Annexe B

Métriques d'Évaluation

Les métriques d'évaluation sont des mesures utilisées pour évaluer la qualité du résultat donné par un algorithme d'apprentissage, dans notre cas de prédiction.

1) Prédiction positive et prédiction négative

Un algorithme est dit positif s'il détecte ce pour quoi il est conçu (dans cette thèse une défaillance) et négatif s'il ne le détecte pas. On parle de :

1.1) Vrai Positif (VP) (*True Positive* ou *TP*)

Un résultat est dit vrai positif lorsque la classe d'une instance est correctement prédit par l'algorithme.

1.2) Faux Positif (FP) (*False Positive*)

Un résultat est dit faux positif ou qu'il s'agit d'une fausse alarme lorsque l'instance n'appartient pas à une classe mais l'algorithme l'attribue à cette classe.

1.3) Vrai Négatif (VN) (*True Negative* ou *TN*)

L'instance n'appartient pas à une classe et l'algorithme ne l'attribue pas à cette classe.

1.4) Faux Négatif (FN) (*False Negative*)

Un résultat est dit faux négatif lorsque l'instance appartient à une classe mais l'algorithme ne l'attribue pas à cette classe.

Ces terminologies peuvent être formulés dans une table appelée table de confusion (en anglais *Contingency Table* ou *Confusion Matrix*) :

		Classe	
Prédiction		Positive	Négative
ou	Positive	**VP** ou correct alarme	**FP** ou fausse alarme
Classification	Négative	**FN**	**VN**

Table de confusion

2) Rappel (*Recall*)

Il est donné par :

$$\frac{VP}{VP + FN}$$

Cette mesure peut être expliquée aussi par :

$$Rappel = \frac{\sum_{i=1}^{N} Rappel_i}{N}$$

N : le nombre de classes à prédire.

$$Rappel_i = \frac{Nombre\ d'instances\ correctement\ attribués\ à\ la\ classe\ i}{Nombre\ d'instances\ appartenant\ à\ la\ classe\ i}$$

4) Précision

Cette mesure est donnée par :

$$\frac{VP}{VP + FP}$$

Elle peut peut être mesurée par :

$$Précision = \frac{\sum_{i=1}^{N} Précision_i}{N}$$

N : le nombre de classes à prédire.

$$Précision_i = \frac{Nombre\ d'instances\ correctement\ attribués\ à\ la\ classe\ i}{Nombre\ d'instances\ attribués\ à\ la\ classe\ i}$$

5) F_measure

C'est la moyenne harmonique de la précision et de rappel. Elle mesure la capacité du système à donner toutes les solutions pertinentes et à refuser les autres :

$$\frac{2 * Précision * Rappel}{Précision + Rappel} \quad \in [0, 1]$$

Annexe C

Code Source

Calcule de probabilité de vraisemblance

```
Cal-Proba():réel
Var
  i,j,t:entier;
  Somme: réel;
  alpha: matrice de réel;
Début
  Pour i <- 1 a N Faire
  Alpha=Forward();
  Pour i <- 1 a N Faire Proba = Proba + Alpha[i][T];
retourner Proba
```

Algorithme *Forward* du PHMM

```
Forward(O : tableau de réel): matrice de réel
Var
   i,j,t:entier;
   Somme, Proba:réel;
   alpha:Matrice de réel;
Début
   Pour i <- 1 a N Faire alpha[i][1]=Val_Pi[i][O[0]];
   Pour j <- 1 à N
          for (t=2; t<=T; t++)
            Somme=0;
             Pour i <- 1 a N Faire
               Somme = Somme + (alpha[i][t-1]*(a[i][j]*C[i][j][O[t-1]]
                           /b[i][O[t-1]]));
          Fpour
          alpha[j][t]= Somme*(D[j][O[t]][O[t-1]]/b[j][O[t-1]]);
    Fpour
```

```
     Fpour
    Pour i <- 1 a N Faire Proba = Proba+alpha[i][T];
  retourner alpha;
```

Ré-estimation des paramètres du PHMM

```
  estimer(O : tableau de réel)
  Var
    i,j,t:entier;
    m,n,o,k:entier;
    Last_Proba,Proba:réel;
    Sigma,Mu,Somme,Xi_Sum,Gamma_Sum,Xi_Sum_2,Gamma_Sum2:réel;
    /**********appel de la procédure lire la séquence*********/;
  Début
    Alpha=Forward(O);
    Beta=Backward(O);
  Proba=Cal-Proba();
    Repeter{
       Last_Proba=Proba;
       /*** Ré estimation de Pi ***/
            Pour i <- 1 a N Faire
                Pi[i]= (Alpha[i][1]*Beta[i][1])/Proba;
       /*** Ré estimation de a ***/
            Pour i <- 1 a N Faire
              Pour j <- 1 à N Faire
                    Xi_Sum=0;
                    Gamma_Sum=0;
                    Pour t <- 1 à (T-1) Faire
                          Xi_Sum=Xi_Sum+(Alpha[i][t]*(a[i][j]*C[i][j][O[t-1]]
                                        /b[i][O[t-1]])*(D[j][O[t]][O[t+1]]/
                                        b[j][O[t+1]])*Beta[j][t+1]);
                          Gamma_Sum=Gamma_Sum+(Alpha[i][t]*Beta[i][t]);
                     Fpour
                      Si(Gamma_Sum <> 0) alors a[i][j]=Xi_Sum/Gamma_Sum;
                      Sinon a[i][j]=0;
               Fpour
            Fpour
       /*** Reestimation of B ***/
       Pour i <- 1 a N Faire
              Mu=0;
              Sigma=0;
              /*** Reestimation of Mu_B ***/
              /*** Reestimation of Sigma_B ***/
              Pour t <- 1 à T Faire b[i][O[t]]=Gauss_Dist(O[t], Sigma,
Mu); Fpour
        Fpour
```

```
       /*** Reestimation of C ***/
       Pour i <- 1 a N Faire
             Pour j <- 1 à N Faire
                    Mu=0;
                    Sigma=0;
                   /*** Reestimation of Mu_C ***/
                   /*** Reestimation of Sigma_C ***/
                   Pour t <- 1 à T Faire C[i][j][O[t]]=Gauss_Dist(O[t],
Sigma, Mu);Fpour
             Fpour
        Fpour
       /*** Reestimation of D ***/
       Pour i <- 1 a N Faire
               Mu=0;
               Sigma=0;
               /*** Reestimation of Mu_D ***/
               /*** Reestimation of Sigma_D ***/
               Pour t <- 1 à T Faire D[i][k-1][O[t]]=Gauss_Dist(O[t], Sigma,
Mu);
        Fpour
       /***calculer la nouvelle probabilité****/
       Last_Proba=Cal-Proba();
      jusqu'à ce que (Last_Proba<Proba)
  Fin.
```

Bibliographie

[1] Avizienis,A., Laprie,J.-C., Randell,B., Landwehr,C. E. : Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. On Dependable and Secure Computing, Volume1, N° 1, pp. 11-33, 2004.

[2] Cox, D. R. : Role of models in statistical analysis. Journal Statistical Science, Volume 5, Issue 2, pp. 169-174, 1990.

[3] Baum, L. E., Eagon, J. A. : An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. Bulletin of the American Mathematical Society, Volume 73, N° 3, pp. 360-363, 1967.

[4] Rabiner, L. R. : A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, Volume 77, N° 2, pp. 257–286, 1989.

[5] Thoraval, L., Carrault, G., Bellanger, J. : Heart signal recognition by hidden Markov models : the ECG case. Methods of Information in Medicine Journal, Volume 33, N° 1, pp. 10-14, 1994.

[6] Rossi, S., Romano, P., Palmieri, G., Iannello, G. : A hidden Markov model for Internet channels. Dans (ISSPIT 2003) : IEEE Proceedings of the 3rd International Symposium on Signal Processing and Information Technology, pp. 50–53, 2003.

[7] Wen, C., Zhan, Y. Z. : Facial expression recognition based on combined HMM. Dans International Journal of Computer Applications in Technology (IJCAT), Volume 38, N° 1-3, pp. 172-176, 2010.

[8] Guo, R.T. : A two-stage multimodal speaker location-aware approach in pervasive computing. Published in International Journal of Computer Applications in Technology (IJCAT), Volume 38, N° 1-3, pp. 118-123, 2010.

[9] Cappé, O., Moulines, E., Ryden, T : Inference in Hidden Markov Models. Edition Springer, Springer Series in Statistics, 2005.

[10] Kobayashi, T., Haruyama, S. : Partly-hidden markov model and its application to gesture recognition. Acoustics-Speech and Signal Processing (ICASSP-97), Volume 4, pp. 3081–3084, Avril 1997.

[11] Ganssle, J., Barr, M. : Embedded Systems Dictionary. Edition CMP-Books, 1ére édition, San Francisco et New York, pp. 90-91, USA, 2003.

[12] Z. Mammeri. Systèmes temps réel et embarqués Concepts de base, expression des contraintes temporelles. Support de cours. Universsité de Toulouse 3, 2011.

[13] Kopetz, H. : Real-Time Systems : Design Principles for Distributed Embedded Applications. Edition kluwer academic publishers, Springer, 2ème édition, 378 pages. 2011.

[14] Hansson, H. : ARTES - A Network for Real-Time Research and Graduate Education in Sweden 1997-2006. Publier par Uppsala University, 799 pages, 2006.

[15] Edwards, L. : So You Wanna Be an Embedded Engineer : The Guide to Embedded Engineering. Edition Newnes, 1 ére édition, 256 pages, 2006.

[16] Salfner, F., Lenk, M., Malek, M. : A Survey of Online Failure Prediction Methods. Journal ACM Computing Surveys (CSUR), Volume 42, N°3, Article 10, New York, USA, 2010.

[17] Bazin, H. Amdec. http ://www.bazin-conseil.fr/amdec.html/, Dernier accès en ligne 27-08-2014.

[18] Cristian, F., Dancey, B., Dehn, J. : Fault-tolerance in the Advanced Automation System. Dans (FTCS-20) : IEEE Proceedings of 20th International Symposium on Fault-Tolerant Computing, pp. 6–17. 1990.

[19] Paraud, T. : Adaptation en ligne de mécanismes de tolérance aux fautes par une approche par composant. Thèse de doctorat, Université polytechnique de Toulouse, pages 166, Janvier 2009.

[20] Bartlett, W. and Spainhower, L. : Commercial Fault Tolerance : A Tale of Two Systems. IEEE transactions on dependable and secure computing, Volume 1, N° 1, pp. 87-96, Octobre 2004.

[21] Mejia-Alvarez, P. : Scheduling Fault Recovery Opertions in Real Time Systems. Dans le journal Computación y Sistemas, Volume 6, N° 1, pp. 51-61, Septembre 2002.

[22] Silva, L.M., Silva, J.G. : Avoiding checkpoint contamination in parallel systems. Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, pp. 364 – 369, Munich, Germany, June 1998.

[23] Wang, L.F. : Fault handling in embedded industrial measurement and control systems : issues and a case study. Dans le Proceeding de IEEE Systems Readiness Technology Conference, AUTOTESTCON 2003, pp. 713–719, Septembre 2003.

[24] Taïani, F., Killijian, M.O., Fabre, J. C. : Intergiciels pour la tolérance aux fautes. Revue des sciences et technologies de l'information, série TSI, Éditions Hermès Lavoisier, Volume 25, N° 5, pp. 599-630, 2006.

[25] Arlat, J., Crouzet, Y., Deswarte, Y., Laprie, J.-C., Powell, D., David, P., Dega, J. L., Rabéjac,C., Schindler, H., Soucailles,J.-F. : Fault Tolerant Computing. Dans le livre : Wiley Encyclopedia of Electrical and Electronics Engineering, Edition John Wiley and Sons, Inc, 2001.

[26] Mongardi., G. : Dependable Computing for Railway Control Systems. Dependable Computing and Fault-Tolerant Systems, Springer, Volume 8, pp. 255-277, 1993.

[27] Tambe, S., Dabholkar, A., Gokhale, A. : Fault-tolerance for Component-based Systems - An Automated Middleware Specialization Approach. Proceedings of International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009), pp. 47-54, Vanderbilt University, Nashville, USA, Mars 2009.

[28] Zając, P. : Fault tolerance through self-configuration in the future nanoscale multiprocessors. Thèse de doctorat, Institut National des Sciences Appliquées de Toulouse, pages 167, Juin 2008.

[29] http ://www.enseignement.polytechnique.fr/profs/informatique/Georges.Gonthier/pi97/perrin/codes.html, Dernier accès en ligne 16-12-2014.

[30] Hamming, R. W. : Error Detecting and Error Correcting Code. Dans the bell system technical journal, Volume 26, N°2, pp. 147-160, Avril 1950.

[31] Afonso, F., Silva,C., Brito, N., Montenegro, S., TavaresA. : Aspect Oriented Fault Tolerance for Real-Time - Embedded Systems. Dans (ACP4IS' 08) : Proceedings of the 2008 AOSD workshop on Aspects, components, and patterns for infrastructure software Brussels, publier par ACM, N°2, pp. 1-8, Belgique, Mars 2008.

[32] Goodenough, J.B. : Exception Handling : Issues and a Proposed Notation. Journal of Communications of the ACM, Volume 18, N° 12, pp. 683-696, USA , Décembre 1975.

[33] Lafosse, J. : Java EE - Guide de développement d'applications web en Java. Edition : Eni, 1ère édition, pages 611, pp-280-283, 2009.

[34] Bouchenak, S., Krakowiak, S., De Palma, N. : Tolérance aux fautes dans les grappes d'applications Internet. Rapport technique, INRIA Projet Sardes – France, 5 au 8 avril 2005.

[35] Bucchiarone, A., Muccini, H., Pelliccione, P. : Architecting Fault-tolerant Component-based Systems : from requirements to testing. Journal Electronic Notes in Theoretical Computer Science (ENTCS), Elsevier Science Publishers B. V, Volume 168, pp. 77-90, Pays-bas, Février 2007.

[36] Standardized e-gas monitoring concept for engine management systems of gasoline and diesel engines. Rapport technique de BMW, Daimler-Chrysler, Volkswagen, Porche, Audi, 2004.

[37] Lussier, B. : Tolérance aux fautes dans les systèmes autonomes. Thèse de doctorat, Centre National de la Recherche Scientifique (LAASCNRS), institut national polytechnique de Toulouse, pages 133, pages 32-33, Avril 2007.

[38] OpenEnterprise Fault ToleranceNetwork Reference Guide (V2.83), Rapport technique, Volume 2, N° 83, Avril 2012.

[39] Duong, P-Q., Pérez Cortés, E., Collet, C. : La tolérance aux fautes adaptable pour les systèmes à composants : application à un gestionnaire de données. Dans (BDA '02) : proceeding de la 18ème Journées Bases de Donnèes Avancées, pp. 161-174, Octobre 2002.

[40] Kim, J., Puraniky, P., Rajkumar, R. : Realizing a Fault-tolerant Embedded Controller on Distributed Real-Time Systems. Journal SIGBED Rev, publier par ACM, Volume 10, N° 4, pp. 33-36,USA, 2013.

[41] Bauer, T., Bohr, F., Landmann, D., Beletski, T., Eschbach R., Poore, J. : From Requirements to Statistical Testing of Embedded Systems. Dans (SEAS '07) Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems, IEEE Computer Society Washington, DC, pp. 1-3, USA, 2007.

[42] Lu, C. : Robustesse du logiciel embarqué multicouche par une approche réflexive : application à l'automobile. Thèse de doctorat, Institut National Polytechnique de Toulouse, pages 142, Décembre 2009.

[43] Liu, X., Ding, H., Lee, K., Sha, L., Caccamo, M. : Feedback fault tolerance of real-time embedded systems : issues and possible solutions. ACM SIGBED Review Homepage archive, Volume 3, N°2, pp. 23-28, Avril 2006.

[44] Aström, K.J., Murray, R.M. : Feedback Systems : An Introduction for Scientists and Engineers. Edition Princeton University Press, pp. 1-25,293-312, 2010.

[45] Crowell, J., Shereshevsky, M., Cukic, B. : Using fractal analysis to model software aging. Rapport Technique, Universsité West Virginia, Department d'CSEE, 2002.

[46] http ://www.hdsentinel.com/smart/index.php. Dernier accès en ligne 04/03/2015.

[47] Abuhadrous,I. : Système embarqué temps réel de localisation et de modélisation 3D par fusion multi-capteur. Thèse de doctorat, Ecole des Mines de Paris, pages 228, 2005.

[48] Holsti, N., Paakko, M. : Towards advanced fdir components. Dans DASIA 2001 : DAta Systems In Aerospace, Nice, France, Juin 2001.

[49] Walsch., A. : Industrial Embedded Systems - Design for Harsh Environment. Support de cours. 2012.

[50] Ibrahim, A.M. : Fuzzy Logic for Embedded System Application. Edition Newnes, 1 ére edition, pages 312, 2004.

[51] Cheng, F., Wu, S., Tsai, P., Chung, Y., Yang, H. : Application cluster service scheme for near-zero-downtime services. IEEE Proceedings of the International Conference on Robotics and Automation, Volume 4, pp. 4062–4067, Avril 2005.

[52] Fonseca, J., L Afonso, J., Martins, J.S., Couto, C. : Fuzzy logic speed control of an induction motor. Dans le Journal Microprocessors and Microsystems, Volume 22, N° 9, pp. 523–534, Mars 1999.

[53] Pimentel, J. R. : Safety-Reliability of Distributed Embedded System Fault Tolerant Units. 29th Annual Conference of the IEEE Industrial Electronics Society-IECON '03, Kettering University, Flint, Michigan Volume 1, pp. 945-950, USA, 2003.

[54] Kaâniche , M. : Évaluation de la sûreté de fonctionnement informatique — fautes physiques, fautes de conception, malveillances. HDR, Institut National Polytechnique de Toulouse, N° 129, pp.7-24, 1999.

[55] Crnkovic, I. : Component-based software engineering for embedded systems. Dans ICSE '05 Proceedings of the 27th international conference on Software engineering, ACM, pp.712-713, USA, 2005.

[56] Bedford, T., Cooke, R. : Probabilistic Risk Analysis : Foundations and Methods. Edition Cambridge University Press, 1ére edition, Royaume-Uni, pages 414, Avril 2001.

[57] Cornuéjols, A., Miclet, L., Kodratoff, Y. : Apprentissage artificiel Concepts et algorithmes. Edition EYROLLES, 2ème édition, pages 803, Juin 2010.

[58] Oliner, A. J., Sahoo, R. K. : Evaluating cooperative checkpointing for supercomputing systems. Dans le IEEE Proceedings of 20th International Parallel and Distributed Processing Symposium (IPDPS 2006). Rhodes Island, 2006.

[59] McCulloch, W. S., Pitts, W. S. : A logical calculus of ideas imminent in nervous activity. Dans le Bulletin of Mathematical Biophysics, Volume 5, N° 4, pp. 115-133, 1943.

[60] Jodouin, J. F. : Les réseaux de neurones : principes et définitions. Edition Hermès, 1ére édition, pages 124, France, 1994.

[61] Troudet, T., Merrill, W., Center, N., Cleveland, O. : A real time neural net estimator of fatigue life. Dans IEEE Proceedings of International Joint Conference on Neural Networks (IJCNN 90), San Diego, CA, Volume 2, pp. 59–64, USA, Juin 1990.

[62] Yilboga, H., Eker, O. F., Güçlü, A., Camci, F. : Failure Prediction on Railway Turnouts Using TimeDelay Neural Networks. Dans (CIMSA) IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 134 – 137, Italy, 2010.

[63] Pawlak, M., Kowalski, C.T. : Low-cost embedded system for the IM fault detection using neural networks. (ICEM) International Conference on Electrical Machines, pp. 1–5, Italy , Séptembre 2010.

[64] Fu, S., Xu, C-Z. : Quantifying event correlations for proactive failure management in networked computing systems. Journal of Parallel and Distributed Computing, Volume 70, N° 11, pp. 1100–1109, November 2010.

[65] Szekely, Z. : Embedded system for the detection of brushless exciter failure. Thèse de doctorat, universsité de Purdue, Indiana, USA, Novembre 2012.

[66] Quinlan, J. R. : C4.5 : Programs for Machine Learning. Edition Morgan Kaufmann, pages 302, USA, 1993.

[67] Shannon, C. E. : A mathematical theory of communication. Dans Bell System Technical Journal, Volume 27, pp. 379-423,623-656, 1948.

[68] Breiman, L., Friedman, J., Stone, C-J. , Olshen, R.A. : Classification and Regression Trees (The Wadsworth Statistics/Probability Series). Edition Wadsworth Publishing, pages 368, 1984.

[69] Quinlan, J. R. : Induction of Decision Trees. Machine Learning Journal, Kluwer Academic Publishers, Volume 1, N° 1, pp. 81-106, USA, 1986.

[70] Khoshgoftaar, T.M., Allen, E.B. : Predicting fault-prone software modules in embedded systems with classification trees. Dans Proceedings 4th IEEE International Symposium on High-Assurance Systems Engineering, pp. 105–112, Washington, DC, USA , 1999.

[71] Khoshgoftaar, T. M., Seliya, N. : Comparative Assessment of Software Quality Classification Techniques : An Empirical Case Study. Dans le journal Empirical Software Engineering, Volume 9, pp. 229–257, 2004.

[72] Zhao, F., Koutsoukos, X., Haussecker, H., Reich, J., Cheung, P. : Monitoring and Fault Diagnosis of Hybrid Systems. IEEE transactions on systems, man, and cybernetics—part b : cybernetics, Volume 35, N° 6, pp. 1225–1240, Décembre 2005.

[73] Vapnik, V. : The nature of statistical learning theory. Editer par Springer, 2ème édition, pages 314, 1999.

[74] Mohamadally H., Fomani B. : SVM : Machines à Vecteurs de Support ou Séparateurs à Vastes Marges. BD Web, ISTY3. Versailles : Université de Versailles St Quentin, 20 pages, 2006.

[75] Murray, J., Hughes, G., Kreutz-Delgado, K. : Hard drive failure prediction using non-parametric statistical methods. Proceedings of ICANN/ ICONIP, Turkey, 2003.

[76] Namburu, S. M., Chigusa, S., Prokhorov, D., Qiao, L. , Choi, K., Pattipati, K. : Application of an Effective Data-Driven Approach to Real-time Fault Diagnosis in Automotive Engines. IEEE Aerospace Conference, pp. 1–9, Mars 2007.

[77] Hu, H., Qian, S., Cao, J. : Monitoring and Fault Diagnosing System Design for Power Transformer Based on Temperature Field Model and DGA Feature Extraction. Dans (WCICA 2008) 7th World Congress on Intelligent Control and Automation, pp. 1800–1805, Juin 2008.

[78] Aravindh, K.B., Saranya, G., Selvakumar, R., Swetha Shree, R., Saranya, M., Sumesh, E.P. : Fault detection in Induction motor using WPT and Multiples SVM. International Journal of Control and Automation, Volume 3, N° 2, pp. 9-20, 2010.

[79] Guo, Y., Ma, J., Xiao, F., Tian, T. : SVM with Optimized Parameters and Its Application to Electronic System Fault Diagnosis. Dans IEEE Conference on Prognostics and Health Management (PHM), pp. 1–6, Denver, USA, 2012.

[80] Kim, H-E., Tan, A.C.C., Mathew, J., Kim, E. Y. H., Choi, B-K. : Machine prognostics based on health state estimation using SVM. Third World Congress on Engineering Asset Management and Intelligent Maintenance Systems Conference, Beijing, China, Octobre 2008.

[81] Holland, J. : Outline for a logical theory of adaptive systems. Journal of the Association of Computing Machinery, ACM, Volume 9, N° 3, pp. 297-314, 1962.

[82] Goldberg, D.E. : Genetic Algorithms in Search, Optimization and Machine Learning. Edition Addison-Wesley Professional, 1ére édition, 432 pages, 1989.

[83] Taouche, R. : Prévision du comportement mecanique d'alliages biphases par algorithmes genetiques et reseaux de neurones. application aux systemes wc-co. Thèse de doctorat, Universsité de Mentouri Constantine, pp. 3-11, 82 pages, 2010.

[84] Corno, F., Reorda, M.S., Squillero, G., Manzone, A. : Automatic test bench generation for validation of RT-level descriptions : an industrial experience. Design, Automation and Test in Europe Conference and Exhibition, pp. 385–389, Paris, 2000.

[85] Wattanapongskorn, N., Coit,D. W. : Fault-tolerant embedded system design and optimization considering reliability estimation uncertainty. Reliability Engineering and System Safety journal, Volume 92, Issue 4, pp. 395–407 Thailand, 2007.

[86] Adachi, M., Papadopoulos, Y., Sharvia, S., Parker, D. : An approach to optimization of fault tolerant architectures using HiP-HOPS. Journal Software—Practice & Experience, Volume 41 Issue 11, pp. 1303-1327, Octobre 2011.

[87] Shun-Zheng, Y. : Hidden Semi-Markov Models. Publier dans Artificial Intelligence, Elsevier, Volume 174, N°2, pp. 215–243, 2010.

[88] Kang, J., Kang, N., Feng, C-J., Hu, H-Y. : Research on tool failure prediction and wear monitoring based HMM pattern recognition theory. ICWAPR '07 : International Conference on Wavelet Analysis and Pattern Recognition, Volume 3, pp. 1167 – 1172, Beijing, Novembre 2007.

[89] Baruah, P., Chinnam, R. B. : HMM for diagnostics and prognostics in machining processes. International Journal of Production Research, Volume 43, N°. 6, pp. 1275–1293, Mars 2005.

[90] Camci, F., Chinnam, R.B. : Health-state estimation and prognostics in machining processes. IEEE Transactions on Automation Science and Engineering. Volume 7, N° 3, pp.581-597, 2010.

[91] Xing-Hui, Z., Jian-She, K. : Hidden Markov models in bearing fault diagnosis and prognosis. Published in : IEEE Second International Conference on Computational Intelligence and Natural Computing Proceedings (CINC), Volume 2, pp. 364 – 367, Wuhan, China, 2010.

[92] Tapered Roller Bearings. http ://www.generalbearing.com/products/roller-bearings.html , Dernier accès en ligne 23-10-2014.

[93] Salfner, F. : Event-based Failure Prediction : An Extended Hidden Markov Model Approach. Thèse de doctorat , Université de Humboldt, pp.10–11, Berlin, 2008.

[94] Salfner, F., Malek, M. : Using Hidden Semi-Markov Models for Effective Online Failure Prediction. In proceeding 26th IEEE International Symposium on Reliable Distributed Systems. Beijing, China, pp.161-174, 2007.

[95] Zhao, Y. , Liu, X. , Gan, S. , Zheng, W. : Predicting Disk Failures with HMM- and HSMM Based Approaches. Advances in Data Mining Applications and Theoretical Aspects : 10th Industrial Conference (ICDM 2010), Volume 6171, pp.390–344, Berlin-Germany, 2010.

[96] Teoh, TT., Cho, SY., Nguwi, YY. : Hidden Markov Model for Hard-Drive Failure Detection. Proceeding of The 7th International Conference on Computer Science & Education (ICCSE 2012), pp.3-8, Melbourne, Australia, 2012.

[97] Murugesan, N., Suguna, P., Muthumani, N., Thanamani, A-S. : Sequence Similarity Using Gaines Hidden Markov Model for Failure Prediction. International Journal of Recent Trends in Engineering, Volume 2, N°. 2, November 2009.

[98] Boughrara, A., Belhadri, M. : Predict a Failure in Embedded Real-Time System. JDLIO'2011 : 1ère journée doctorale, Université d'Es Senia, Oran, Algérie, 31 Mai et 01 Juin 2011.

[99] Boughrara, A., Belhadri, M. : Forecast failure based on PID in embedded real-time system. AWERProcedia - Information Technology and Computer Science, 2nd World Conférence on Innovation and Computer Sciences (INSODE 2012). Volume 2, pp. 198- 205, Izmir, Turquie, 2012.

[100] Kobayashi, T., Furuyama, J., Masumitsu, K. : Partly-hidden markov model and its application to speech recognition. Proceedings of the Acoustics-Speech and Signal Processing (ICASSP '99), pp. 121-124, 1999.

[101] Ogawa, T., Kobayashi, T. : Hybrid modeling of PHMM and HMM for speech recognition. Proceedings of Acoustics, Speech, and Signal Processing, (ICASSP '03) IEEE International Conference on, Volume 1, N°8, pp. 140-143, 2003.

[102] Varshneya, A.K. : Fundamentals of Inorganic Glasses : Arun K. Varshneya. Edition Academic Press, pp. 183-232, 1994.

[103] Boughrara, A., Belhadri, M. : Use partly hidden Markov model to evaluate a future failure. Dans International Journal of Computer Applications in Technology (IJCAT), Volume 39, N°3-4, pp 372-377, 2014.

[104] O'Dwyer., A. : PI and PID controller tuning rules. Publier par Dublin Institute of Technology, Volume 9, pages 624, Ireland, 2009.

[105] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, IH. : The WEKA Data Mining Software : An Update. Publier dans ACM SIGKDD Explorations Newsletter, Volume 11, N° 1, pp. 10-18, USA, Juin 2009.

[106] Fleiss, J.L. : Statistical methods for rates and proportions. Edition John Wiley, 2ème edition, USA, 1981.

[107] Bottou, L., Lin., C-J. : Support Vector Machine Solvers. Chapitre dans le livre :Large Scale Kernel Machines. Publier par MIT Press, pp. 301-320, Cambridge, MA., 2007.

[108] Christophe, C. : Hidden Markov Models and Their Mixtures. DEA-report en mathématiques, Université catholique de Louvain- Belgique, 1996.

[109] Fine, S., Singer, Y., Tishby, N. : The Hierarchical Hidden Markov Model : Analysis and Applications. Machine learning journal, Volume 32, N°6, pp. 41-62, 1998.

Résumé

Les Modèles de Markov cachés (HMM) représentent une solution potentielle pour la prédiction des défaillances des systèmes embarqués temps réel. Ils permettent d'estimer la probabilité d'une future défaillance en se basant sur des observations du système. Pour garantir que le système soit suffisamment fiable et tolérant aux fautes, le mécanisme de prédiction doit envisager le fait qu'une erreur puisse influencer l'état futur du système. Cependant, les solutions étudiées standard ne prennent pas en considération les caractéristiques dynamiques de l'état contrairement aux Partly Hidden Markov Models (PHMM) qui conditionnent la probabilité de transition entre états à l'observation de l'état précédent.

Le présent travail propose d'utiliser les PHMM comme un mécanisme permettant de prédire une future défaillance en se basant sur des observations du système. Pour valider ce travail, une étude de cas est présentée portant sur un four industriel de production de verre. Les résultats obtenus montrent que l'utilisation des PHMM est particulièrement efficace.

Mots clés : Système embarqué, Temps réel, Tolérance aux fautes, Algorithme d'apprentissage, Prédiction des défaillances, Modèles de Markov Cachés, HMM, Partly Hidden Markov Models, PHMM, Four industriel.

Abstract

The Hidden Markov Models (HMM) are a potential solution for failures prediction of embedded real-time systems. They can estimate the probability of future failure based on observation of the system. To ensure that the system is sufficiently reliable and fault tolerant, the prediction mechanism should consider the fact that an error can influence the future state of the system. However, standard solutions studied does not meet the dynamic characteristics of the state unlike Partly Hidden Markov Models (PHMM) which combines the power of conditioning the state transition probability to the previous observation.

This paper proposes to use the PHMM as a mechanism to predict future failures based on observations of the system. To validate this work, a case study is presented on an industrial oven glass production. The results show that the use of PHMM is particularly effective.

Keywords : Embedded system, Real time, Fault tolerance, Machine learning algorithm, Failure prediction, Hidden Markov Models, HMM, Partly Hidden Markov Models, PHMM, Industrial oven,