APPROCHE INTELLIGENTE À BASE DE RAISONNEMENT À PARTIR DE CAS POUR LE DIAGNOSTIC EN LIGNE DES SYSTÈMES AUTOMATISÉS DE PRODUCTION

N. Ben Rabah

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

ÉCOLE DOCTORALE SCIENCES DU NUMÉRIQUE ET DE L'INGÉNIEUR

THÈSE EN CO-TUTELLE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LA MANOUBA

Discipline : **Informatique**

Et

DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Discipline : ***Automatique, Génie Informatique, Traitement du Signal et Image***

Présentée et soutenue publiquement par

Nourhène BEN RABAH

Le 14 décembre 2018

Cliquez ici pour entrer du texte.

APPROCHE INTELLIGENTE À BASE DE RAISONNEMENT À PARTIR DE CAS POUR LE DIAGNOSTIC EN LIGNE DES SYSTÈMES AUTOMATISÉS DE PRODUCTION

Thèse dirigée par Mme Véronique CARRÉ-MÉNÉTRIER, Professeur, **Université de Reims Champagne-Ardenne**

Et par M. Moncef TAGINA, Professeur, **Université de la Manouba**

JURY

M. Ouajdi KORBAA,	Professeur,	Université de Sousse	**Rapporteur**
M. François DELMOTTE,	Professeur,	Université d'Artois,	**Rapporteur**
Mme Wided CHAARI,	Professeur,	Université de La Manouba ,	**Examinatrice**
M. Armand TOGUYENI,	Professeur,	Université de Lille,	**Examinateur**
M. Bernard RIERA,	Professeur,	Université de Reims Champagne-Ardenne,	**Examinateur**
Mme Ramla SADDEM,	Maitre de conférences,	Université de Reims Champagne-Ardenne,	**Examinateur**
M. Cyril DE RUNZ,	Maitre de conférences, HDR	Université de Reims Champagne-Ardenne,	**Invité**

Remerciements

C'est avec un immense plaisir et une grande émotion que j'aborde l'écriture des remerciements qui marque la fin de la première étape de mon parcours de recherche et certainement le début d'une autre que j'espère toute aussi fructueuse en résultats et riche en rencontres. Je tiens donc à travers ces lignes à témoigner de ma plus grande reconnaissance envers toutes les personnes qui, de près ou de loin, ont contribué à l'achèvement de ce travail.

En premier lieu, je remercie Monsieur François DELMOTTE, Professeur à l'Université d'Artois et Monsieur Ouajdi KORBAA, Professeur à l'Université de Sousse, d'avoir accepté d'évaluer mon travail et d'être les rapporteurs de cette thèse. Je remercie également Madame Wided CHAARI, Professeur à l'Université de la Mannouba, Monsieur Armand TOGUYENI, Professeur à l'Université de Lille et Monsieur Bernard RIERA Professeur à l'Université de Reims Champagne-Ardenne pour avoir accepté d'examiner ma thèse.

Je voudrai aussi adresser mes chaleureux remerciements à ma directrice de thèse Madame Véronique CARRE-MENETRIER, Professeur à l'Université de Reims Champagne-Ardenne, qui m'a accompagnée de près tout au long de mon parcours de recherche. C'est notamment grâce à son cours de grafcet que j'ai appris à commander les systèmes automatisés de production. Je lui dois beaucoup et je tiens à lui exprimer ma reconnaissance de m'avoir généreusement transmis son savoir et savoir-faire. Je la remercie sincèrement pour son suivi implacable de mes travaux, pour sa confiance, et pour tout le temps et l'énergie qu'elle a su me consacrer durant ces dernières années. C'est une grande chance de pouvoir travailler avec une personne ayant ses qualités scientifiques et humaines.

Je remercie également mon co-directeur de thèse Monsieur Moncef TAGINA, Professeur à l'Université de la Mannouba, de m'avoir inconditionnellement soutenue dans toutes

mes démarches. Je le remercie aussi de m'avoir accordé une grande liberté dans le travail tout en assurant un suivi méticuleux.

J'exprime mes remerciements à mon encadrante Madame Ramla Saddem, maître de conférences à l'Université de Reims Champagne-Ardenne pour la qualité de son encadrement, sa disponibilité, son intérêt envers mon travail et surtout pour la confiance qu'elle m'a accordée et ses encouragements continuels. J'espère qu'en regardant ce travail après coup, j'aurai rempli toutes ses attentes.

Je remercie également Cyril de Runz, maître de conférences HDR à l'Université de de Reims Champagne-Ardenne, pour sa participation active et son implication dans le suivi de cette thèse. Je lui suis reconnaissante pour son aide chaque fois que j'ai sollicité ses compétences.

Je ne laisserai pas cette occasion passer sans exprimer ma sympathie envers tous les membres du CReSTIC de l'Université de Reims Champagne-Ardenne. Je les remercie de m'avoir chaleureusement accueillie et d'avoir contribué à rendre mon séjour parmi eux si riche et agréable. Un grand merci en particulier aux secrétaires du laboratoire Ida, Amandine, Danielle et Marielle. Merci aussi à Romain, Mohamed, Khaled, Imane, Kween, Mehdi et tous les autres que j'aurais involontairement oubliés de citer. Je suis également très redevable envers mes collègues de l'ENSI et du laboratoire COSMOS, à mes amies Wiem, Nourchène et Safa pour leur amitié et partage.

Je profite enfin de cet espace pour adresser l'expression de ma profonde gratitude à ma famille. A Maman chérie, jamais je ne pourrai m'acquitter de ma dette envers toi. Je te dois tout. Merci pour ton amour et ton soutien inconditionnel. Merci aussi d'être souvent la personne qui subit mes mauvaises humeurs. J'espère être à la hauteur dans ce nouveau rôle qui m'attend. Et merci pour toute la préparation de mon pot de thèse ! Papa, merci de m'avoir toujours soutenue et encouragée. A mon mari, Mehdi, merci pour ta présence, ta confiance, ton soutien, ta patience, pour avoir su trouver les mots justes dans les moments les plus difficiles, et bien entendu pour ton amour. A mon frère Anwar, ma sœur Nourchène, ma grand-mère Bouka, ma tante Sonia et mes oncles Majdi, Imed et Mourad, merci pour votre affection et merci de m'avoir supportée quand ça allait moins bien. A feu mon oncle Hosni, tu nous as quitté si tôt sans voir l'achèvement de ce travail que tu attendais tant. Que Dieu te comble de son infinie miséricorde. A feu mon grand-père, tu seras à jamais dans mon cœur et l'amour dont tu m'as comblé guidera toujours mes pas dans cette vie. A toute ma famille, je dis merci d'avoir été toujours là et d'avoir cru en moi, je vous dédie ce travail en guise d'amour et de reconnaissance.

Résumé

Les systèmes automatisés de production (SAP) représentent une classe importante de systèmes industriels qui sont devenus complexes et sensibles aux dysfonctionnements avec des conséquences importantes sur la productivité, la qualité de production et la sécurité des biens et des personnes. Un défi majeur est de mettre en œuvre des approches systématiques d'aide ou de diagnostic afin de garantir la sûreté de fonctionnement. Cette thèse s'intéresse donc au diagnostic en ligne des comportements indésirables des SAP ayant des capteurs et des actionneurs délivrant des signaux binaires et dont la dynamique correspond à celle des systèmes à évènements discrets (SED).

L'efficacité d'une approche de diagnostic au service de ce type de systèmes se mesure à travers le taux de bonne détection de fautes, la précision d'isolation, le nombre de fausses alarmes et la complexité de mise en œuvre de l'approche. L'objectif de ce travail est donc de concevoir des solutions de diagnostic intelligentes qui satisfont les critères mentionnés sans requérir la connaissance complète du fonctionnement interne du système, capables de se mettre à jour en temps réel pour améliorer les performances et ayant un faible coût de mise en œuvre.

L'approche proposée dans cette thèse est basée sur le Raisonnement à Partir de Cas (RàPC) qui est à la fois une méthodologie de raisonnement par analogie et une méthodologie d'apprentissage issue du domaine de l'intelligence artificielle. L'originalité des travaux réside en finalité dans les quatre items suivants : (1) l'approche proposée utilise le RàPC pour le diagnostic des SAP ayant une dynamique de type SED, (2) elle propose un format de représentation de cas inspiré du formalisme Signatures Temporelles Causales qui s'adapte à l'aspect dynamique des systèmes à surveiller, (3) elle introduit une phase qui couplent simulation et mise en forme de données pour la transformation des données issues du système simulé après son émulation en mode normal et défaillant, et (4) elle présente une phase de raisonnement et d'apprentissage qui permet non seulement le diagnostic en ligne du système surveillé mais aussi la mise à jour de la base de cas suite à l'apparition de nouveaux comportements inconnus.

Mots-clés : diagnostic, apprentissage automatique, raisonnement à partir de cas, système d'aide à la décision, systèmes à évènements discrets, systèmes automatisés de production, métriques de distance, systèmes à base de données.

Abstract

Automated Production Systems (APS) represent an important class of industrial systems. They have become complex and susceptible to malfunctions, with significant negative consequences on productivity, production quality and security of property and people. The main challenge when using such systems is to implement systemic assistance or diagnosis approaches in order to ensure their operating safety. Within this framework, we focus, in this thesis, on the online diagnosis of the APS equipped with sensors and actuators emitting binary signals. These systems can be considered as 'Discrete Event Systems' (DES).

The effectiveness of an approach for diagnosing such systems is measured through good detection rate, isolation accuracy, false alarms number and the method implementation complexity. The objective of this work is, thus, to propose intelligent diagnosis solutions satisfying the mentioned criteria without complete knowledge about the system's internal functioning. The solutions, whose implementation is not expensive, are able to update themselves in real time in order to improve their performance.

The introduced approach is based on a reasoning and learning methodology derived from 'Artificial Intelligence' (AI) which is the 'Case Based Reasoning' (CBR). The originality of our research work is observed in the following 4 aspects : (1) the developed approach uses the CBR for the diagnosis of the APS with DES dynamics, (2) it suggests a case representation format inspired by the 'Causal Temporal Signatures' (CTS) which is able to adapt to the dynamic aspect of the systems to be monitored, (3) it exploits data acquired from a digital twin after its emulation in both normal and faulty modes to build an empirical knowledge called 'cases' and finally (4) it presents a reasoning and learning phase that allows not only the online diagnosis of the monitored system, but also the updating of the case base following the appearance of the new unknown behaviors.

Keywords : diagnosis, machine learning, case-based reasoning, decision support system, discrete event systems, automated production systems, distance metrics, data driven systems.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Contexte du sujet

Au cours de ces dernières décennies, les **S**ystèmes **A**utomatisés de **P**roduction (**SAP**) [Vogel-Heuser et al. (2015); Bakkari et al. (2015); Araújo et al. (2018)] ont permis d'augmenter la production, d'améliorer la qualité des produits, de réduire les coûts de main-d'œuvre et d'économiser l'énergie et la matière première. En retour, cette automatisation a nécessité de garantir la sécurité et la fiabilité des systèmes tout au long du processus de production. En effet, la réduction de l'intervention humaine et l'automatisation de certaines fonctions ont rendu les SAP plus complexes avec comme conséquence une augmentation des risques de dysfonctionnements.

Jusqu'à présent le diagnostic des SAP est en grande partie une activité humaine, exécutée par les **O**pérateurs **H**umains de **S**urveillance (**OHS**)[Kim and Wysk (2012); Proctor and Van Zandt (2018)]. Ces opérateurs permettent d'analyser les données acquises du système surveillé et de prendre des décisions adéquates suite à l'apparition des comportements indésirables qui traduisent la présence d'un ou de plusieurs dysfonctionnements dans le système. Les OHS sont des «facteurs» de fiabilité en exécutant leurs tâches grâce à leurs connaissances, leur expérience et leur savoir-faire [Longo et al. (2017); Mosier and Skitka (2018); Tao et al. (2018b)]. Cependant, ils peuvent aussi être responsables du manque de fiabilité du système. En effet, gérer une grande quantité de données et agir rapidement peuvent les conduire à prendre des décisions incorrectes qui dégradent la situation [Trentesaux and Millot (2016); Pacaux-Lemoine et al. (2017)]. Par conséquent, la détection et la localisation des dysfonctionnements sont cruciales pour éviter aux systèmes de se diriger vers des situations indésirables voire même catastrophiques sur le plan humain, environnemental et économique. Ceci s'entend sous le vocable « *Diagnostic de défaillances ou de fautes* ». Le diagnostic est l'un des enjeux les plus importants dans

l'industrie [Miljković (2011); Gao et al. (2015); Isermann (2017)], notamment dans l'industrie manufacturière [Vogl et al. (2016); Rajaoarisoa and Sayed-Mouchaweh (2017); He et al. (2018a)]. En effet, chercheurs et industriels ne recherchent pas seulement à détecter et à localiser l'occurrence d'une faute, mais aussi de faire progresser les solutions actuelles de diagnostic dont la mise en œuvre est difficile et coûteuse en temps. Ils proposent pour les améliorer des solutions dites intelligentes. Pourquoi « *intelligentes* » [Aldrich and Auret (2013); Liu et al. (2018); Jia et al. (2018)] parce que ces nouvelles approches ne nécessitent pas d'avoir une connaissance complète du fonctionnement interne du système et parce qu'elles sont capables de se mettre à jour en temps réel afin d'améliorer leurs performances. Elles informent et assistent les OHS dans une solution simple à mettre en œuvre dont l'implémentation n'est pas coûteuse.

Objectifs de la thèse

Dans ce contexte, cette thèse s'intéresse au diagnostic des comportements indésirables dans les SAP. La littérature propose différentes approches traitant de cette problématique. Ces approches se répartissent selon la dynamique des SAP en trois classes : la classe des systèmes continus [Cellier and Kofman (2006)], la classe des systèmes à évènements discrets (SED) [Cassandras and Lafortune (2009)] et la classe des systèmes dynamiques hybrides (SDH)[Zaytoon et al. (2001)]. Dans cette étude, nous nous intéressons au diagnostic des SAP possédant des capteurs et des actionneurs délivrant des signaux binaires (c'est-à-dire Tout ou Rien (TOR)) et qui relèvent des SED. L'objectif de ce travail est donc de proposer des solutions de diagnostic dites intelligentes afin de répondre aux besoins des industriels.

Les approches de diagnostic au service de cette catégorie de systèmes peuvent être vues selon que le diagnostic s'effectue en ligne ou non, selon que le modèle soit spécifié (par automate ou par réseau de Petri) ou non, selon la structure de prise de décision du diagnostic (centralisée, décentralisée ou distribuée), selon la représentation des fautes et leurs reconnaissances [Zaytoon and Lafortune (2013)], etc. En général, les approches se classent en trois grandes familles selon le mode du raisonnement utilisé pour le diagnostic : les approches à base de modèles [Sampath et al. (1995); Debouk et al. (2000); Philippot (2006); Zaytoon and Lafortune (2013)], les approches à base de connaissances [Dousson et al. (1993); Bertrand (2009); Saddem et al. (2012)] et les approches à base de données [Venkatasubramanian et al. (2003); Moosavian et al. (2013); Dou and Zhou (2016); Vazan et al. (2017); Han et al. (2017)]

Les approches à base de modèles sont généralement utilisées dans le cadre d'une connaissance suffisante du fonctionnement interne du système. Elles sont efficaces et ca-

pables de valider la cohérence et la complétude des défauts à diagnostiquer. Cependant, pour fonctionner correctement, ces approches nécessitent des modèles analytiques précis et approfondis du domaine et la difficulté majeure est le coût élevé de mise en œuvre des modèles [Danancher et al. (2011); Saddem and Philippot (2014)]. En effet, la complexité temporelle de mise en œuvre de la plupart des modèles est une complexité exponentielle. Les approches à base de connaissances ont une capacité de diagnostic élevée à cause de la connaissance des symptômes des fautes qu'elles modélisent [Cordier and Dousson (2000)]. Néanmoins, la difficulté majeure réside dans la formalisation des connaissances expertes et dans leurs mises à jour [Guerraz and Dousson (2004); Dousson et al. (2008); Cram et al. (2012); Subias et al. (2014); Saddem and Philippot (2014)]. Ainsi, pour que les approches remplissent leurs missions convenablement, elles nécessitent une base de connaissance complète et cohérente. Les approches à base de données [Venkatasubramanian et al. (2003); Moosavian et al. (2013); Dou and Zhou (2016); Vazan et al. (2017); Han et al. (2017)] ne nécessitent pas de connaitre de manière approfondie le fonctionnement interne du système. Elles n'ont pas besoin d'une modélisation explicite du système à travers un modèle mathématique. Elles utilisent les données d'historiques disponibles et à partir de ces données, elles font des prévisions. Ces approches apprennent de chaque expérience afin d'améliorer leurs performances. Elles s'appuient sur les techniques de l'apprentissage automatique pour atteindre ses objectifs. Toutefois, elles nécessitent une étape de préparation de données afin d'extraire les données les plus pertinentes qui vont être formatées selon la technique d'apprentissage automatique à utiliser.

Contributions de la thèse

Les limitations de certaines de ces approches nous ont conduit à proposer, dans ce travail de thèse, une méthode de diagnostic à base de données qui s'appuie sur une technique d'apprentissage automatique. Parmi les techniques existantes, nous avons choisi d'utiliser le « **R**aisonnement **à P**artir de **C**as (**RàPC**) » [Kolodner (1992); Aamodt and Plaza (1994)] qui se présente comme une méthodologie de raisonnement par analogie, dérivée de l'intelligence artificielle (IA) et appartenant à d'autres disciplines telles que les sciences cognitives, l'apprentissage automatique et les systèmes à base de connaissances [Bergmann (2002)]. Le choix du RàPC est justifié par sa capacité de raisonnement et par sa capacité d'apprentissage incrémental [Aha et al. (1991); Ali et al. (2018)]. En effet, cette technique est capable d'apprendre de chaque expérience et les résultats peuvent être facilement examinés pour découvrir ce qui a été appris et comment.

Dans ce cadre, nous proposons dans la première partie de la thèse un système d'aide au diagnostic des SAP qui combine l'utilisation d'un **S**ystème d'Aide à la **D**écision (**SAD**) et le RàPC [Ben Rabah et al. (2017a)]. Ce système s'appuie sur un bloc hors ligne et un

bloc en ligne. Le bloc hors ligne permet de définir un format de représentation de cas, de construire une base de cas normaux (BCN) et une base de cas défaillants (BCD) à partir d'une base de données d'historique. Le bloc en ligne permet d'aider les OHS dans la prise de la décision du diagnostic la plus adéquate. Les résultats des expérimentations sur un système de tri de caisses illustrent l'approche au niveau du format de représentation de cas et au niveau de la base de cas utilisée. Cependant cette première approche ne donne pas entièrement satisfaction et présente certaines limites : (*i*) elle s'avère incomplète puisqu'elle ne modélise pas toutes les fautes des SAP ; (*ii*) elle n'est pas évolutive surtout dans le contexte de l'industrie du futur où les SAP doivent être extensibles, évolutifs et reconfigurables ; (*iii*) elle conduit à un problème d'explosion combinatoire des descripteurs lors de l'implémentation sur des systèmes réels ayant un grand nombre de composants ; et (*iv*) elle est difficilement compréhensible par les utilisateurs du système d'aide au diagnostic.

Pour surmonter ces difficultés et améliorer les résultats, nous avons cherché à étendre cette première approche et nous proposons dans la suite de la thèse une approche de diagnostic en ligne des SAP [Ben Rabah et al. (2017b)] basée sur 3 propositions (voir la figure 1) :

1. la proposition d'un *nouveau format de représentation* de cas qui s'adapte à l'aspect dynamique des SAP, qui est assez expressif et facile à comprendre par les OHS.
2. la proposition d'une phase qui couple simulation et mise en forme de données pour la transformation des signaux des capteurs et des actionneurs à *des cas*. [Ben Rabah et al. (2017c)]. Les signaux sont issus du **système simulé** après son émulation en mode normal et défaillant.
3. la proposition d'une phase de raisonnement et d'apprentissage qui permet le *diagnostic en ligne* du **système réel** surveillé et la *mise à jour de la base de cas* suite à l'apparition de nouveaux comportements inconnus. Cette phase repose sur 5 étapes : *(a)* l'élaboration d'un nouveau cas non résolu, *(b)* la recherche du cas source le plus similaire en utilisant un *nouveau indice de dissimilarité* qui supporte notre représentation de cas, *(c)* la réutilisation de la solution du cas sélectionné *(d)* l'affichage du nouveau cas résolu et *(e)* la sauvegarde et la révision.

Le SAP virtuel utilisé dans l'approche correspond au jumeau numérique d'un SAP réel. Ce concept "Digital Twin" en anglais [Tao et al. (2018a), Zambal et al. (2018), He et al. (2018b)] a récemment été introduit dans le cadre de l'industrie du futur. Il s'agit du double numérique d'un système physique qui repose sur les données des capteurs et des actionneurs. Il offre de nombreux avantages pour les industriels tels que la compréhension et l'anticipation de l'évolution du système réel, l'optimisation de ses performances sans avoir besoin d'embarquer un autre système, la réduction des coûts liés aux réparation, etc.

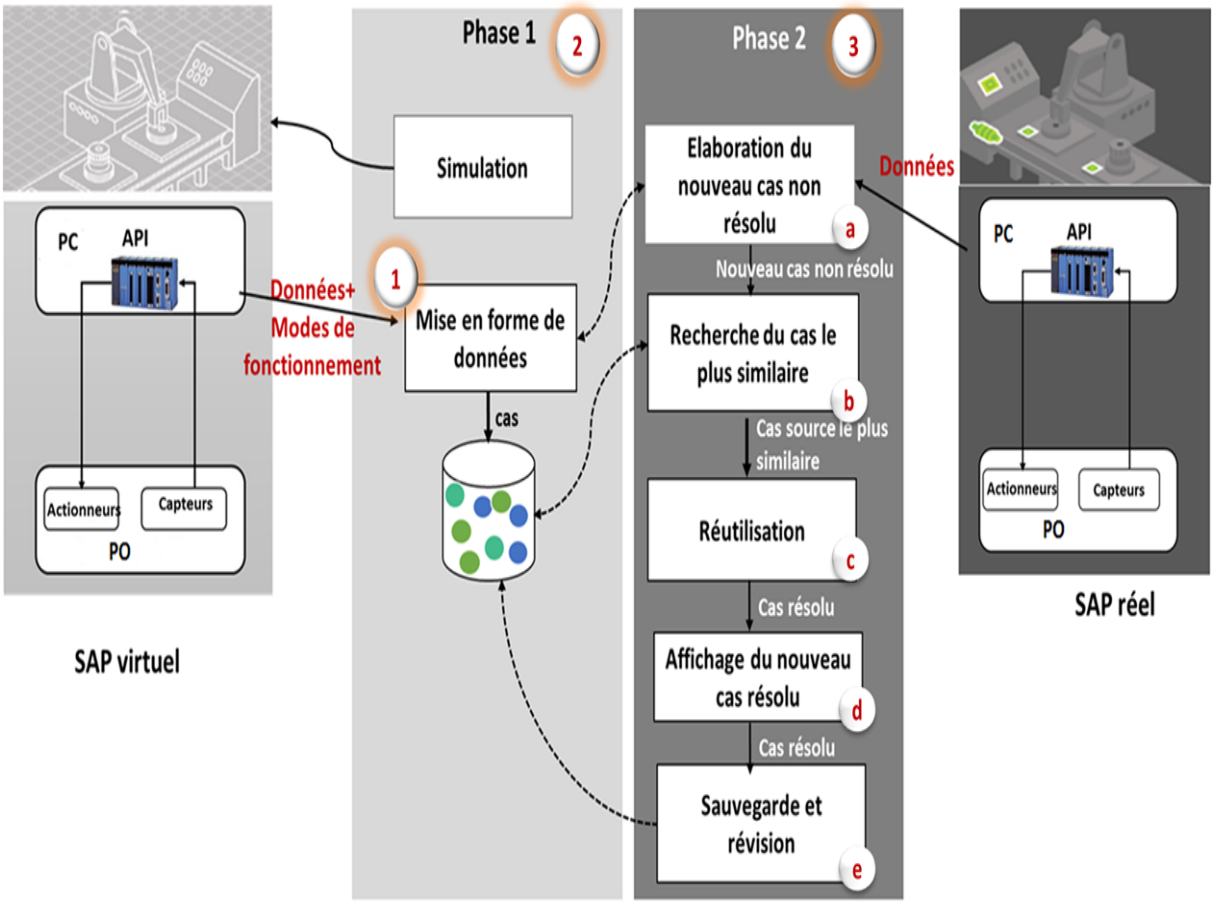


FIGURE 1 – Les différentes étapes de construction de l'approche de diagnostic proposée.

Organisation du mémoire de thèse

Nos travaux de thèse comportent cinq chapitres.

Le premier chapitre est consacré à la présentation de l'état de l'art du diagnostic des fautes dans les SAP. Nous y présentons leurs structures de base et les différents types pour positionner le type de systèmes qui nous intéresse parmi les SAP existants. Nous exposons dans la deuxième partie les méthodes et les approches existantes pour le diagnostic des SAP. Nous présentons à cet effet une classification de ces approches en trois classes : les approches à base de modèles, les approches à base de connaissances et les approches à base de données. Nous présentons ensuite une synthèse de ces approches. Nous concluons le chapitre par un bilan critique de l'existant et une synthèse de la problématique.

Dans le deuxième chapitre nous présentons un état de l'art sur le diagnostic à base d'apprentissage automatique. Dans la première partie du chapitre, nous introduisons l'ap-

prentissage automatique à travers la présentation de quelques notions de base. Nous commençons alors par définir l'apprentissage automatique et les différents avantages de son utilisation. Nous exposons ensuite les différentes méthodes d'apprentissage automatique existantes. Nous nous intéressons spécialement au RàPC. Nous présentons à cet effet son principe, ses différentes étapes et les défis majeurs pour le développement d'un système de diagnostic utilisant le RàPC.

Dans le troisième chapitre, nous détaillons notre première approche d'aide au diagnostic que nous illustrons sur un exemple de tri de caisses proposé dans plusieurs travaux comme benchmark [Philippot et al. (2012); Niguez et al. (2015); Pichard et al. (2016)].

Dans le quatrième chapitre, nous détaillons notre seconde approche que nous illustrons sur un exemple de plateau tournant présenté également comme benchmark dans plusieurs travaux [Riera et al. (2011); Saddem and Philippot (2014); Niguez et al. (2017)].

Dans le dernier chapitre, nous présentons la performance de la seconde approche. Ensuite nous menons une étude comparative sur la performance de cette approche vis-à-vis de certaines méthodes d'apprentissage automatique et également vis-à-vis d'une méthode de diagnostic à base de modèles. Toutes les méthodes sont testées sur le même exemple qui est le plateau tournant.

Nous terminons ce mémoire de thèse par les conclusions sur nos travaux et les perspectives de recherche associées.

Une annexe, placée en fin de document, complète le mémoire en présentant le principe des méthodes d'apprentissage automatique les plus utilisées pour la résolution des problèmes de classification : machines à vecteurs de support (SVM), les arbres de décision, les forêts aléatoires, les réseaux de neurones et la régression logistique.

Chapitre I

Diagnostic des fautes dans les SAP : État de l'art 1

Sommaire

Introduction

Le diagnostic des **S**ystèmes **A**utomatisés de **P**roduction (**SAP**) est devenu un sujet d'actualité dans la recherche scientifique et industrielle en raison de la complexité et des coûts croissants des interventions de maintenance. Cette complexité et le besoin d'améliorer la sécurité, la fiabilité et la disponibilité de ces systèmes nécessitent des approches de diagnostic afin de détecter et de localiser les dysfonctionnements possibles. Dans ce contexte, nous proposons une synthèse de l'état de l'art sur le diagnostic des SAP.

Le chapitre se compose de trois parties. Dans la première partie nous introduisons les SAP à travers la présentation de quelques notions de base. Nous commençons par définir les SAP et expliquer leurs structures de base. Ensuite, nous présentons leurs différents types pour positionner la classe des systèmes qui nous intéresse parmi les SAP existants. Enfin, nous exposons le principe du fonctionnement des **A**utomates **P**rogrammables **I**ndustriels (**API**) qui sont les principaux composants d'automatisme destinés à la commande des SAP. Dans la deuxième partie de ce chapitre, nous présentons la terminologie adoptée pour désigner les dysfonctionnements présents dans les SAP et nous définissons le module du diagnostic. Dans la troisième partie de ce chapitre, nous exposons les méthodes et les approches réalisées pour le diagnostic des SAP. Nous proposons à cet effet une classification de ces travaux en trois classes selon le mode du raisonnement utilisé pour le diagnostic : à partir de modèles, de connaissances ou de données. Un bilan de l'existant et une synthèse de la problématique seront présentés à la fin de ce chapitre.

1 Les systèmes automatisés de production

Les SAP représentent une classe importante des systèmes industriels qui sont de plus en plus complexes à cause du grand nombre d'interactions et d'interconnexions entre leurs différents composants. De ce fait, ils sont plus sensibles aux dysfonctionnements dont les conséquences peuvent être importantes en termes de productivité, de sécurité et de qualité de production. Par conséquence, les communautés scientifiques des automaticiens et des informaticiens travaillent ensemble pour mettre en œuvre des approches systématiques d'aide au /de diagnostic pour détecter, localiser et identifier les fautes possibles.

1.1 Structure d'un SAP

Un système de production est un ensemble de moyens humains et matériels qui interagissent sur des matières premières, des produits semi-finis ou finis afin de produire d'une manière efficace les produits attendus par les clients. En réduisant l'intervention humaine, ces derniers deviennent des systèmes automatisés de production (SAP). L'automatisation

traduit la capacité des systèmes à gérer seuls les plus grands nombres de situations et à décider de façon optimale les comportements à adopter dans ces situations [Fournier (2002)]. Cette automatisation présente plusieurs avantages tels que [Trentesaux (1996), Perrin et al. (2004)] :

- L'amélioration de la qualité, de la fiabilité et de la sécurité de la production ;
- La diminution du temps de réponse ;
- La gestion efficace de la production ;
- La réduction des coûts de main-d'œuvre, l'économie de l'énergie et des matières premières, etc.

Comme exemples des SAP, nous citons [Groover (2007)] :

- Des machines automatisées qui traitent les pièces ;
- Des lignes de transfert effectuant une série d'opérations d'usinage ;
- Des systèmes de manutention et de stockage automatiques des matériaux ;
- Des systèmes manufacturiers qui utilisent des robots industriels pour effectuer des opérations de traitement ou d'assemblage ;
- Des systèmes d'inspection automatique pour le contrôle de qualité ; etc.

Un SAP se compose de deux parties : la partie opérative (PO) et la partie commande (PC). La partie opérative représente l'ensemble des moyens matériels opérant physiquement sur la matière d'œuvre. La partie commande est l'ensemble des moyens de traitement et d'acquisition de l'information qui assurent le pilotage et la conduite du procédé. Les échanges d'information entre la PC et la PO sont de deux types :

- La PC envoie les ordres vers les actionneurs et les pré-actionneurs de la PO afin d'obtenir les effets souhaités ;
- La PO envoie les comptes rendus vers la PC à travers les capteurs ;

La partie dialogue « Homme Machine (HM) » qui est présentée dans la figure 2 permet la communication entre la PC et les opérateurs humains. En effet, ces opérateurs échangent les informations avec la PC à travers une interface. Ils donnent des consignes à l'aide des boutons marche arrêt ou par un clavier. En parallèle, la PC renvoie des signalisations diverses telles que les voyants lumineux, les sonneries, les messages affichés sur les écrans, etc.

1.2 Les différents types de dynamiques des SAP

Selon leur dynamique de fonctionnement, les SAP peuvent être classés en trois familles [Derbel (2009)] : les systèmes continus, les systèmes à événements discrets (SED) [Cassandras and Lafortune (2009)] et les systèmes hybrides.

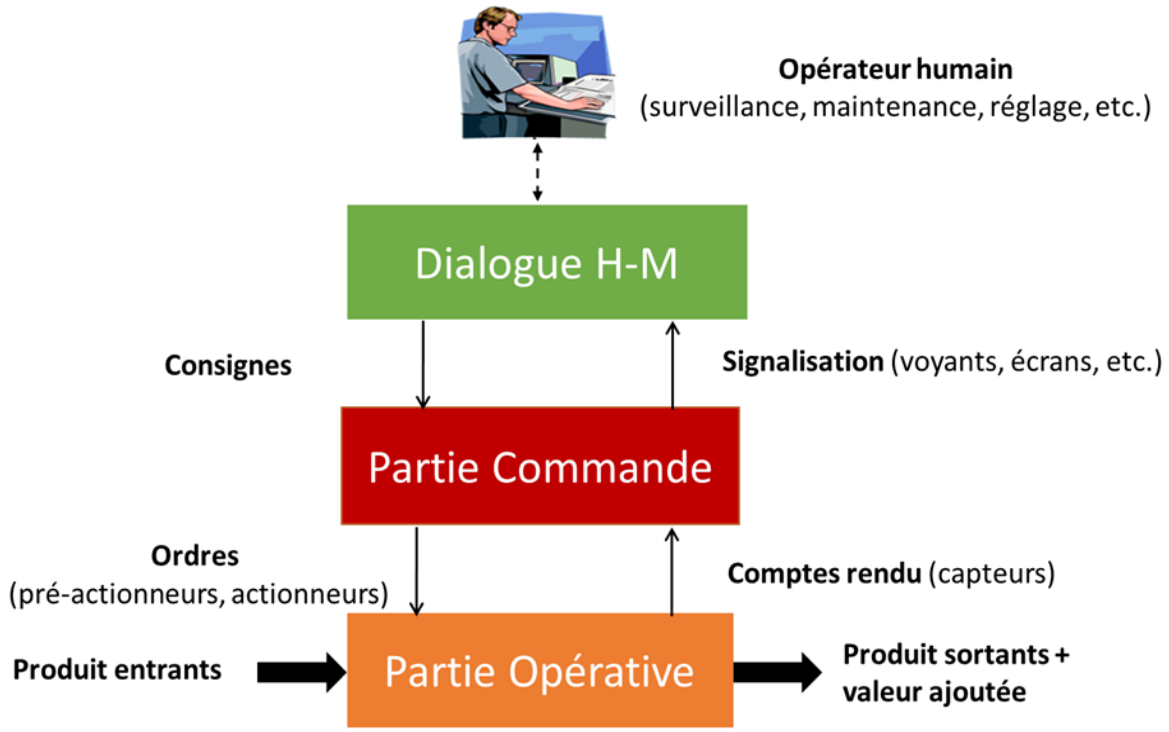


FIGURE 2 – Structure d'un SAP.

a) Systèmes continus

Les systèmes continus sont les systèmes dans lesquels le temps s'écoule de façon continue et où les variables qui décrivent leurs états peuvent changer de valeurs à chaque instant [Erard and Déguénon (1996)].

b) Systèmes à évènements discrets (SED)

Les SED [Cassandras and Lafortune (2009)] sont des systèmes dans lesquels les variables caractérisant leurs états ne changent de valeurs qu'en un nombre fini ou dénombrable de points isolés dans le temps. Ces points représentent des évènements ponctuels qui peuvent être généralement l'arrivée d'un signal ou l'achèvement d'une tâche. A titre d'exemple, les variables décrivant un état dans un système de tri de caisses sont ses différents valeurs de capteurs et actionneurs. Le changement de valeurs de ces variables s'effectue par exemple lors l'arrivée d'une caisse ou lors son déplacement sur les différents convoyeurs. Pour cela, le système de tri de caisses peut être classé comme un SED. La plupart des SED partagent des caractéristiques communes comme [Philippot (2006)] :

- Le parallélisme, qui permet à plusieurs événements d'avoir lieu simultanément et indépendamment dans différentes parties du système ;
- La synchronisation, qui se traduit par l'apparition simultanée de plusieurs événements dans une même période. L'accomplissement de ces évènements nécessite la disponibilité simultanée de plusieurs ressources ou la vérification simultanée de plusieurs conditions.

– La compétition, qui se manifeste par l'occurrence de certains évènements qui excluent l'apparition d'autres.

c) Systèmes hybrides

Les Systèmes Dynamiques Hybrides (SDH) sont des systèmes qui intègrent à la fois une dynamique continue et une dynamique discrète [Zaytoon et al. (2001); Antsaklis and Koutsoukos (2003)]. Ils sont composés : (a) d'un SED dont l'espace d'état est un ensemble fini et les transitions entre les états sont réalisées suite à des évènements ponctuels, (b) d'un "ensemble de modèles continus" représenté par des équations différentielles contraignant l'évolution de l'état continu et (c) d'une "interface" pour l'échange des informations entre la partie discrète et la partie continue [Hamdi (2010)].

1.3 Les Automates Programmables Industriels

Quel que soit le type de SAP, il est souvent contrôlé par un "Automate Programmable Industriel (API)". Ces API sont très utilisés dans l'industrie grâce à leurs performances de sécurité et à leurs différents langages de programmation [Commission et al. (1993)]. Un API exécute un cycle (T) comprenant trois opérations successives (voir la figure 4) :

– **E** : la lecture des entrées qui consiste à la réalisation d'une image du monde extérieur à travers un enregistrement des états des entrées qui sont des variables non contrôlables (c'est-à-dire les capteurs) ;

– **Pg** : l'exécution du programme de l'API ;

– **S** : la mise à jour des états des variables contrôlables (c'est-à-dire les actionneurs).

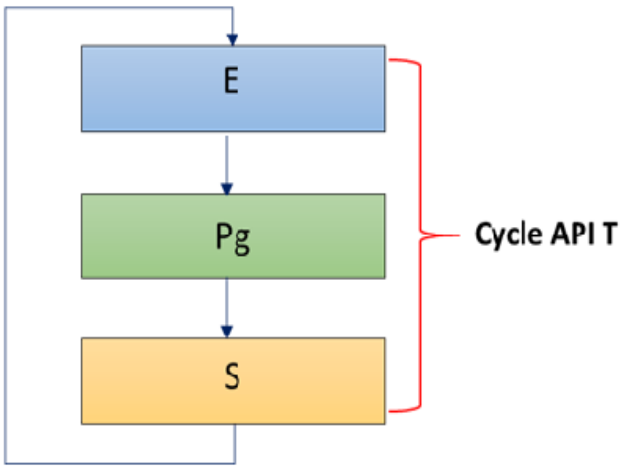


FIGURE 3 – Le cyle d'un API.

Le cycle de l'API peut être exécuté cycliquement ou périodiquement. L'exécution cyclique permet à l'API d'exécuter un nouveau cycle dès que le cycle précédent est terminé

(voir la figure 4) alors que l'exécution périodique exécute un nouveau cycle après avoir reçu une synchronisation par une horloge [Commission et al. (1993)]. Les durées de lecture des entrées et la mise à jour des sorties sont des constantes qui ne dépendent que de l'API utilisé et de sa configuration. Le temps de traitement, lui, dépend de la taille du code que le microprocesseur de l'API doit exécuter [Rohee (2005)].

Par la suite, nos travaux s'articuleront autour des SAP commandés par des API dont l'algorithme d'exécution est cyclique (voir la figure 4).

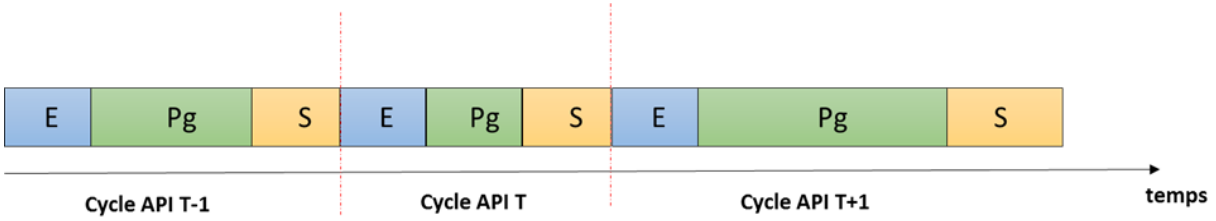


FIGURE 4 – Le fonctionnement cyclique d'un API sur plusieurs cycles au cours du temps.

2 Le problème du diagnostic dans les SAP

Pour traiter le diagnostic des SAP, les deux communautés scientifiques des automaticiens et des informaticiens n'utilisent pas la même terminologie. En effet, il n'existe ni une terminologie consensuelle pour désigner les dysfonctionnements, ni une définition consensuelle du terme diagnostic. Pour cette raison, nous présentons dans ce qui suit la terminologie adoptée pour désigner les dysfonctionnements et la définition utilisée dans ce travail pour le terme diagnostic.

2.1 Terminologie adoptée pour les dysfonctionnements

La communauté scientifique des automaticiens utilise le terme "*faute*" pour désigner un dysfonctionnement dans le système. Pour cela, l'ensemble des méthodes développées pour le traitement des dysfonctionnements des systèmes industriels est connu sous la dénomination de "*Fault Detection and Isolation (FDI)*". En parallèle, la communauté scientifique des informaticiens utilise plutôt le terme de "*défaillance*".

Nous pouvons nous demander si ces termes sont interchangeables ou non. Dans la réalité, il y a une causalité phénoménologique entre "*faute*", "erreur" et "*défaillance*". Cette causalité est définie dans [Boulanger (2013)]. Elle est illustrée à travers la figure 5. Une faute résulte en général d'une défaillance. L'activation d'une faute dans une entité engendre une erreur. La propagation de l'erreur aboutit à une nouvelle défaillance. Cette nouvelle défaillance pourra à son tour engendrer une faute et le processus est itéré. Quelles

que soient les fautes, les erreurs ou les défaillances, toutes constituent des entraves à la sûreté de fonctionnement d'un SAP. Dans cette étude, nous utiliserons le terme "*faute*" et nous supposons que les terme "*faute*" et "*défaillance*" sont interchangeables.

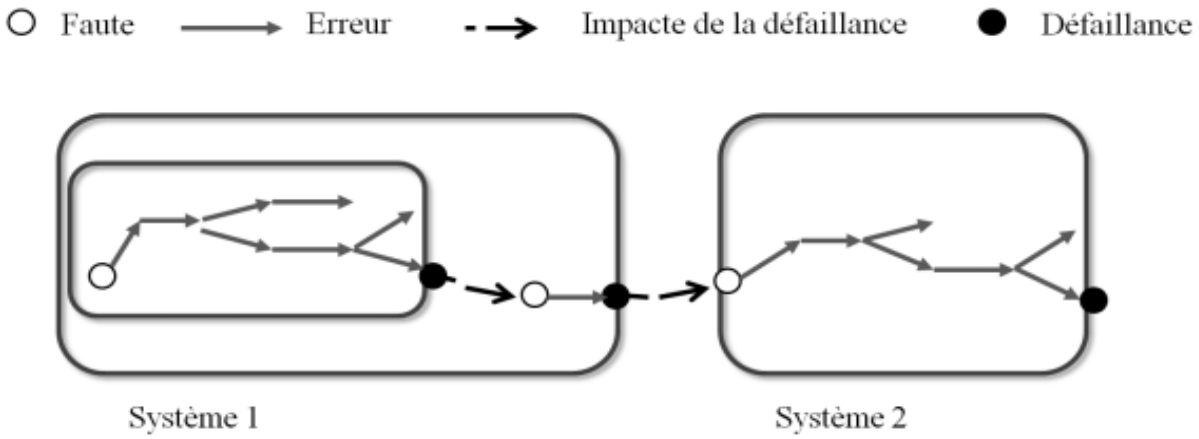


FIGURE 5 – Relation entre "*faute*" et "*défaillance*" selon [Boulanger (2013)].

Les fautes présentes dans un SAP peuvent être classifiées selon plusieurs critères tels que la vitesse d'apparition, l'instant d'apparition, le degré d'importance et les conséquences et l'origine [Saddem (2012)] :

- Vitesse d'apparition : L'évolution d'une faute dans le temps peut être une évolution soudaine due à une évolution quasi-instantanée de certaines caractéristiques du SAP ou une évolution graduelle causée par un changement progressif des caractéristiques du SAP.
- L'instant d'apparition : Une faute peut se produire pendant le fonctionnement d'un SAP (faute en fonctionnement) ou à son arrêt (faute à l'arrêt).
- Degré d'importance : Une faute entraine l'inaptitude d'une entité à accomplir certaines ou toutes les fonctions requises. Dans le premier cas, il s'agit d'une faute partielle et dans le deuxième cas, il s'agit d'une faute complète d'un SAP.
- Les conséquences : Si la faute cause des dommages corporels, des dégâts matériels importants ou si elle conduit à d'autres conséquences jugées inacceptable alors elle est critique et majeure, sinon elle est non critique et mineure.
- L'origine : L'origine d'une faute peut être attribuée à l'entité elle-même (la faute est interne) ou à des facteurs externes à l'entité elle-même (la faute est externe). Les fautes internes [Derbel (2009)] peuvent être à titre d'exemple le blocage d'un capteur ou d'un actionneur à 0 ou à 1, le passage inattendu de 0 à 1 d'un capteur ou d'un actionneur, etc. Les fautes externes peuvent être par exemple un court-circuit causant un changement dans la spécification du produit, une coupure de l'alimentation électrique d'un SAP, la rupture du stock d'une matière première, etc.

2.2 Définition du diagnostic

La littérature scientifique présente plusieurs définitions du diagnostic industriel. Nous nous intéressons aux définitions de la communauté SED. Nous pouvons classer ces définitions [Toguyeni (1992); Combacau et al. (2000); Darkhovski and Staroswiecki (2003); Philippot (2006)] en deux classes en fonction de l'emplacement de la fonction de détection. En effet, cette fonction représente très souvent un sujet de débat concernant sa place.

- La **première classe**, considère la détection comme une entité de diagnostic. Cette classe regroupe les méthodes appelées "Fault Detection and Isolation (FDI)" [Darkhovski and Staroswiecki (2003); Philippot (2006)] qui génèrent des indicateurs de défauts appelés "résidus" contenant des informations sur les anomalies ou les dysfonctionnements du système à diagnostiquer. A partir de ces indicateurs, elle :
  - *Détecte* tout écart entre l'état réel de la PO et celui estimé par le modèle ;
  - *Détermine* l'origine de l'anomalie et localise le ou les composants défectueux ;
  - *Identifie* l'instant d'apparition de la faute, sa durée et son importance.
- La **deuxième classe**, ne considère pas la détection comme une entité de diagnostic. Elle la considère plutôt comme une entité de surveillance [Toguyeni (1992); Combacau et al. (2000); Boufaied (2003). Selon [Toguyeni (1992)] la détection est une tâche réactive positionnée entre la PC et la PO. Elle joue un rôle filtre des comptes rendus émis par la PO. Si le compte rendu reçu correspond à un compte rendu attendu alors il est émis vers la PC, sinon un "*symptôme*" est détecté. Si ce symptôme est jugé dangereux (c'est-à-dire un symptôme qui révèle une faute critique et majeure) alors il est émis vers le module de recouvrement, sinon il est émis vers le module de diagnostic.
  - Le module de diagnostic comporte trois sous fonctions : (a) la localisation qui permet de déterminer le sous-ensemble défaillant, (b) l'identification qui détermine le composant fautif et (c) le pronostic qui prédit la panne induite par la faute. C'est une fonction de post-recouvrement utilisée afin d'éviter les fautes.
  - Le module de recouvrement permet de gérer l'arrêt complet du système, la reprise et la reconfiguration dynamique.

Par la suite, nos travaux s'articuleront autour de la définition du diagnostic proposée par les méthodes FDI qui consiste à détecter l'occurrence d'une faute et identifier son type et son emplacement. Aussi, nous diagnostiquerons des fautes internes en particulier celles causées par la PO comme "le passage inattendu d'un capteur ou d'un actionneur de 0 à 1 ou de 1 à 0" et "le blocage d'un capteur ou d'un actionneur à 0 ou à 1".

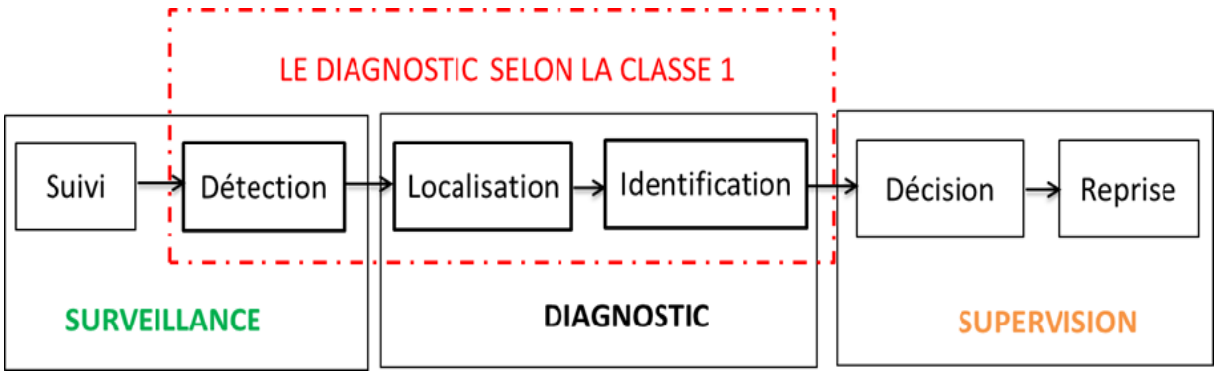


FIGURE 6 – La définition du diagnostic selon la classe 1.

3 Les méthodes de diagnostic existantes

La littérature présente diverses approches et méthodes pour le diagnostic des SED. Ces méthodes peuvent être classées selon plusieurs critères tels que l'implémentation du diagnostic (en ligne ou hors ligne), l'outil de description du modèle (par automate ou par réseau de Petri), la structure de prise de décision du diagnostic (centralisée, décentralisée ou modulaire) et la représentation des fautes et leurs reconnaissances [Zaytoon and Lafortune (2013)], etc.

En général, nous distinguons trois familles d'approches : les approches à base de modèles [Sampath et al. (1995); Debouk et al. (2000); Philippot (2006)], les approches à base de connaissances [Dousson et al. (1993); Bertrand (2009); Saddem et al. (2012)] et les approches à base de données [Venkatasubramanian et al. (2003); Moosavian et al. (2013)]. Nous présentons dans la suite une synthèse de ces différentes approches. En effet, l'objectif de cette section est de positionner la classe des méthodes qui nous intéresse parmi les méthodes existantes.

3.1 Approches à base de modèles

Le diagnostic à base de modèles a été initié par [Reiter (1987)] et [De Kleer and Williams (1987)]. Il consiste à générer des indicateurs de défauts, appelés résidus qui déterminent tout écart entre le comportement réel et le comportement du modèle de référence du système. Cet écart résulte d'un dysfonctionnement dans le système réel. Une description précise du principe général est donnée dans la figure 7.

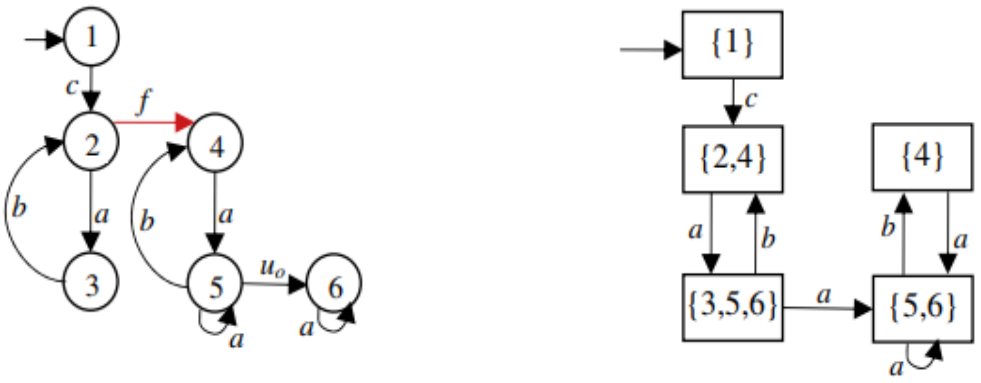
FIGURE 7 – Le principe d'un système de diagnostic à base de modèles.

Les méthodes de diagnostic à base de modèles sont nombreuses. Zaytoon et Lafortune présentent dans [Zaytoon and Lafortune (2013)] un état de l'art des différentes méthodes à base de modèles pour le diagnostic des SED. Nous retiendrons dans la suite une approche très connue par sa grande efficacité, elle est dite approche par "*diagnostiqueur*".

La méthode du diagnostiqueur a été initiée par [Sampath et al. (1995, 1996)]. Elle se base sur un modèle d'automate classique noté $G = (X, E, \delta, x0)$ avec :

- X est l'ensemble des états normaux et défaillants du système ;
- E est l'ensemble des évènements observables et inobservables. Les évènements inobservables sont généralement des fautes ou des évènements générés suite à l'indisponibilité d'un ou de plusieurs capteurs. Bien que les fautes soient des événements inobservables, leurs comportements fautifs sont décrits par des événements observables.
- δ est la fonction de transition ;
- $x0$ est l'état initial.

Le diagnostic des fautes consiste à construire un automate déterministe appelé "observateur" dont les transitions sont des événements observables du système et les états sont des estimations du vrai état du système. Cet observateur est construit à partir du modèle global du système qui est modélisé à travers un automate à états finis dont l'alphabet est composé d'événements observables et d'événements inobservables. Les fautes sont représentées par des événements inobservables. La construction de l'observateur se fait par réduction des transitions correspondants aux événements inobservables.



(a) L'automate. (b) L'observateur.

FIGURE 8 – L'automate (a) et son observateur (b).

La figure 8b présente un exemple d'observateur construit à partir du modèle global du système qui est présenté dans la figure 8a. Cet observateur est obtenu en supprimant les transitions correspondant aux événements inobservables (c'est-à-dire l'événement f qui relie l'état *2* et l'état *4* et qui modélise l'occurrence d'une faute et l'événement u_0 qui relie les deux états *5* et *6* et qui représente un évènement inobservable normal) et

en fusionnant les états auxquels conduisaient ces transitions avec les états dans lesquels arrivent l'événement observable (c'est-à-dire fusionner les états *2* et *4* dans un seul état, fusionner les états *3*, *5* et *6* dans le même état et fusionner aussi les états *5* et *6* dans un même état). Dans [Cassandras and Lafortune (2009)], l'auteur présente un algorithme de construction d'observateurs discrets.

La construction de l'observateur peut conduire à un problème d'explosion d'états. En effet, dans le pire des cas, la taille de l'observateur est exponentielle par rapport à la taille de l'automate. Ceci est dû au fait que les états d'observateur sont des sous-ensembles d'états d'automates [Zaytoon and Lafortune (2013)].

L'automate de l'observateur peut être raffiné pour construire un automate appelé "diagnostiqueur" en associant les étiquettes "F_i" et/ou "N" aux états de l'observateur. Chaque état du diagnostiqueur est représenté sous forme de couples (*état*, *étiquette*) où l'*étiquette* est la liste de fautes $\{F_1,F_2,F_i, ...\}$ sur le chemin menant dans cet *état*. L'état du diagnostiqueur peut être qualifié "$Normal$", "$F_i - certain$" ou "$F_i - incertain$". Il est :

- "$Normal$" si l'état du diagnostiqueur est étiqueté par N,
- "$F_i - certain$" si tous les couples de l'état du diagnostiqueur ont F_i dans l'étiquette ;
- "$F_i - incertain$" s'il ne coïncide à aucun des deux cas précédents.

Afin de décrire le fonctionnement du diagnostiqueur, nous considérons de nouveau l'exemple de la figure 8a. La figure 9 représente le diagnostiqueur de cet automate. Ses états sont étiquetés par F et N. N pour désigner normal et F pour signifier fautif. L'état qui est représenté en vert est un état normal. Les états représentés en rouges sont des états $F - certain$ et ceux qui sont en orange représentent des états $F - incertain$.

Cette approche de diagnostiqueur autour d'une structure centralisée fournit les meilleures performances en terme de diagnostic, cependant la principale limite de l'approche réside dans la gestion de l'explosion combinatoire liée au formalisme des automates [Saddem and Philippot (2014)]. En effet, l'approche classique consiste à décomposer le système en plusieurs composants. Ensuite, un modèle automate est associé à chaque composant. Le diagnostiqueur global du système dans une approche centralisée est alors obtenu par le produit synchrone des automates de ses composants. C'est ce produit synchrone qui engendre souvent une explosion combinatoire quand le système est complexe.

Pour combler cette limite, des diagnostiqueurs décentralisés [Debouk et al. (2000); Philippot (2006)] et distribués ont été proposés [Boufaied (2003); Su (2004); Fanti and Seatzu (2008)]. A titre d'exemple, [Philippot (2006)] a proposé un diagnostiqueur décentralisé basé sur un ensemble de diagnostiqueurs locaux à partir des modèles élémentaires de la PO et d'un modèle de spécifications en intégrant l'information temporelle des ac-

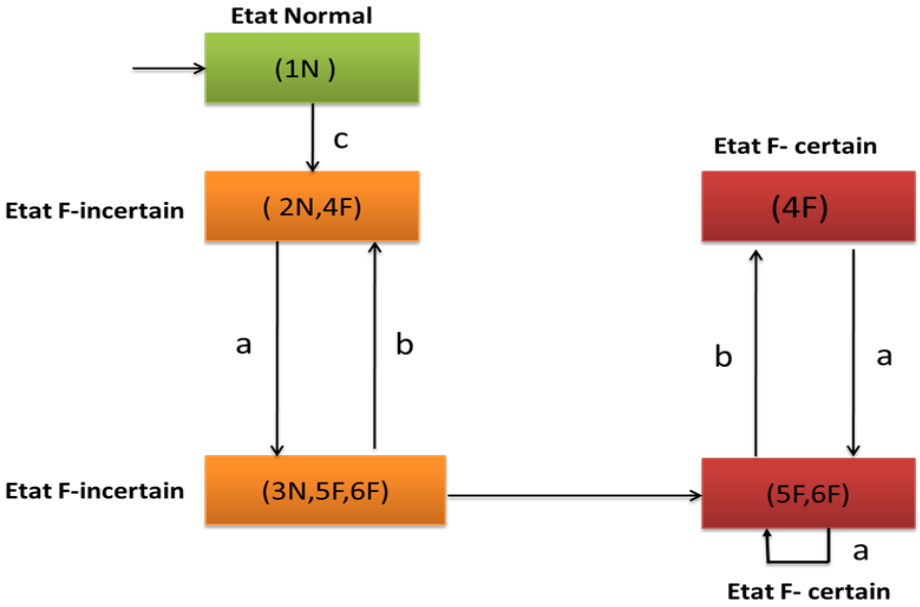


FIGURE 9 – Le diagnostiqueur de Sampath.

tionneurs. L'occurrence de toute faute doit être détectée et isolée par au moins un diagnostiqueur local en utilisant sa propre observation locale sur l'exécution du système. Les diagnostiqueurs locaux sont construits par un expert qui doit analyser pour chaque état de la PO la possibilité de survenance des fautes et surtout la possibilité de détecter et d'isoler ces fautes.

3.2 Approches à base de connaissances

Les experts humains sont capables d'effectuer un niveau élevé de raisonnement à cause de leurs grandes expériences et connaissances sur leurs domaines d'expertise. Les systèmes à base de connaissances modélisent ces expériences et connaissances pratiques de l'expert sous forme de règles ou d'autres formalismes de description de comportements pour le diagnostic des fautes. La performance des systèmes à base de connaissances doit être comparable à celle de l'expert humain. Pour le diagnostic des SED, la littérature distingue plusieurs méthodes à base de connaissances telles que les chroniques avec ces différentes formalisations [Dousson et al. (1993); Boufaied (2003); Bertrand (2009); Carle et al. (2011)] et les Signatures Temporelles Causales (STC) [Saddem et al. (2012)]. La conception de ces méthodes est divisée en deux étapes (voir la figure 10) :

- La construction d'une base de connaissance qui décrit les comportements normaux et/ou défaillants en se basant sur l'expérience et la connaissance experte ;
- La reconnaissance de ces comportements dans un flux d'événements.

Les chroniques selon [Dousson et al. (1993)] sont un ensemble de motifs d'évènements

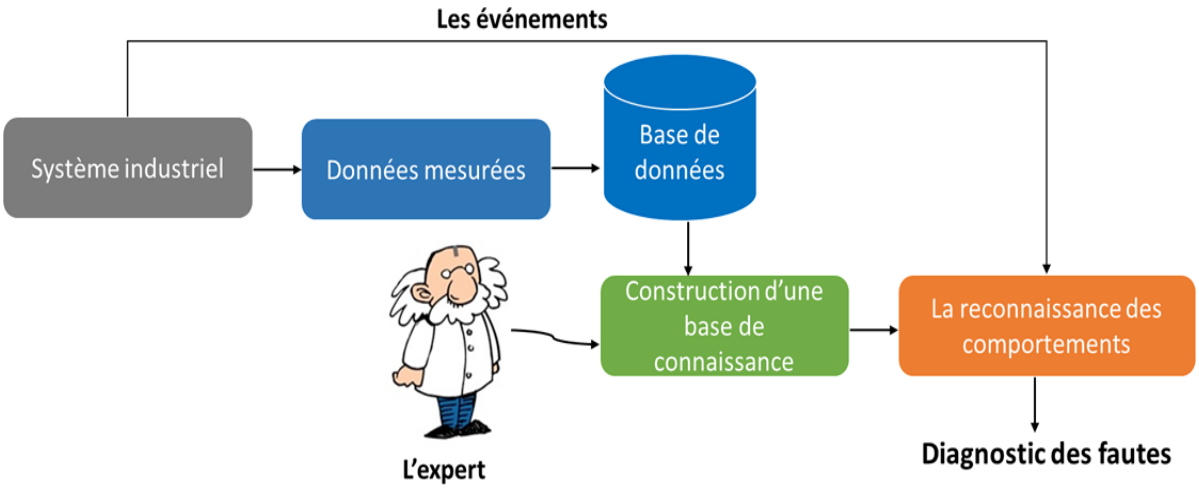


FIGURE 10 – Le principe d'un système de diagnostic à base de connaissances.

qui représente les évolutions possibles du monde observé. Ces motifs sont exprimés par des prédicats. Ils sont partiellement ordonnés, contraints par des relations temporelles et reliées par une conjonction explicite. Chaque chronique est étiquetée par une situation qui peut être normale ou défaillante. L'ensemble des chroniques forme la base de chroniques (appelé en anglais « *Chronicle Base* »). Les événements d'une chronique représentent les alarmes et les contraintes temporelles représentent leurs temps d'apparition. Le système de reconnaissance de chronique (appelé en anglais « Chronicle Recognition System : CRS ») est chargé d'analyser les flux d'entrées d'alarmes et d'identifier tout motif représentant une situation décrite par une chronique sans aucune historisation (c'est-à-dire sans garder une traçabilité sur les flux d'alarmes qui ont mené à chaque reconnaissance d'un comportement détecté). Le modèle de temps utilisé est linéaire et discret.

Ce formalisme a été appliqué et étendu par plusieurs chercheurs. [Boufaied (2003)] a proposé une architecture de détection de fautes dans des SED physiquement distribués. Une chronique est définie sous forme d'un ensemble de sous-chroniques. En effet, le système industriel à diagnostiquer est divisé en un ensemble de sous-systèmes communiquant et la chronique qui modélise le modèle temporel du système industriel est représentée par plusieurs sous chroniques associées chacune à un superviseur (voir la figure 11). Afin de rendre le formalisme plus expressif, plusieurs types de contraintes temporelles ont été pris en compte tels que les contraintes de type intervalle, précédence et fenêtre d'admissibilité [Boufaied (2003)].

Les chroniques développées par [Bertrand et al. (2007); Bertrand (2009)] au sein de l'ONERA sont bien distinctes des chroniques développées par [Dousson et al. (1993)]. En effet, une chronique est décrite à travers des relations qui séparent des évènements dans une séquence ordonnée dans le temps. Ces relations peuvent être logiques (relations de

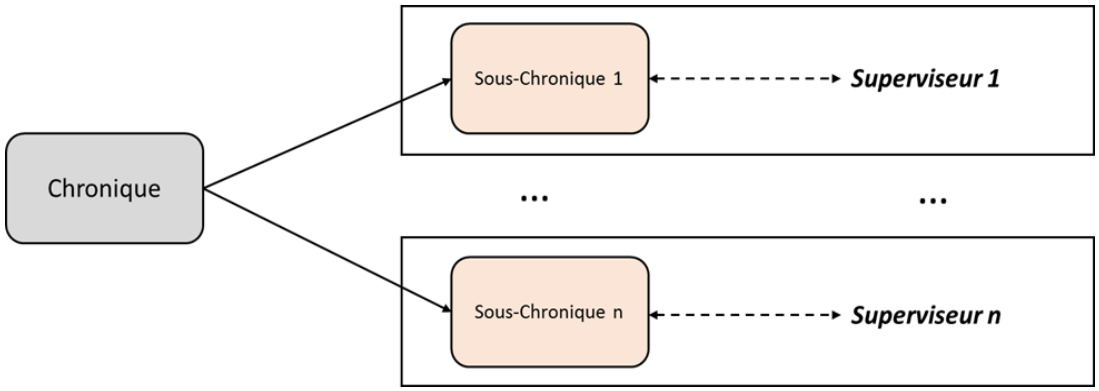


FIGURE 11 – La distribution d'une chronique en n sous chroniques selon [Boufaied (2003)].

conjonction et de disjonction) ou temporelles (séquence ou absence). La reconnaissance d'une chronique est définie par la mise en correspondance entre un flux d'évènements observé et les évènements de la description d'une chronique tout en gardant une traçabilité sur les flux d'événements qui ont mené à chaque reconnaissance d'un comportement détecté. Bertrand a présenté [Bertrand et al. (2007); Bertrand (2009)] différents outils pour leur modélisation tels que à travers les automates à états finis, les automates à compteurs, les automates dupliqués et les réseaux de Petri colorés ainsi il a proposé un algorithme pour leurs interprétations à l'aide de la technique des automates dupliqués. Dans [Carle et al. (2011)] les auteurs ont présenté une nouvelle sémantique pour le langage des chroniques, les évènements qui lient les chroniques ainsi leurs reconnaissances.

La signature temporelle causale (STC) est un concept assez proche des chroniques proposés au début des années 90 par [Toguyeni et al. (1990, 1997)] et ensuite développés par [Saddem et al. (2012)]. Une STC est un formalisme de description et de reconnaissance des comportements d'un système qui est utilisé essentiellement pour le diagnostic des SED. Elle se présente comme un sous ensemble d'événements observables partiellement ordonnés qui caractérise un comportement défaillant d'un système. Ces événements sont contraints par un ensemble de contraintes temporelles portant sur leurs occurrences. Les relations entre les événements peuvent être logiques (les opérateurs de conjonction) ou temporelles (les opérateurs de séquence, absence).

Une STC est une règle composée d'une partie "*condition*" et d'une partie "*conséquence*". La partie "*condition*" est composée d'un ou plusieurs triplets. Soit (X, Y, ct) un triplet. X et Y sont deux événements avec X représente l'événement de référence et Y est l'événement contraint, attendu par rapport à X. ct est une contrainte temporelle qui permet de modéliser une date, un délai ou une durée. S'il n'y a pas de contrainte temporelle entre X et Y, ct devient nct, ce qui signifie aucune contrainte de temps.

Soit la STC décrite par I.1. Cette STC se compose de six triplets.

$$(In, A, nct) * (In, C, nct) * (A, B, ct1) * (A, D, ct2) * (C, D, ct3) * !(D, E, ct4) -> G \quad \text{(I.1)}$$

A, B, C, D et E sont des événements ; In est l'événement toujours occurrent, que nous introduisons de manière théorique pour définir la référence temporelle des événements qui ne sont pas contraints ; G est une faute ou un événement inobservable qui est déduit ; cti est la $i^{éme}$ contrainte temporelle ; nct signifie pas de contrainte temporelle ; le " $!$" est un opérateur de négation ; le triplet (In, A, nct) signifie donc que A est un événement sans contrainte ; le triplet $!(D, E, ct4)$ signifie que E ne doit pas survenir après D tant que la contrainte temporelle $ct4$ est vraie et " $*$" est un opérateur qui sépare chaque triplet. Il est utilisé comme un opérateur logique "Et" qui lie temporellement la reconnaissance de chaque triplet. Le sens de l'équation I.1 est le suivant. Si on a l'occurrence de l'événement A (respectivement, l'occurrence de l'événement C), ensuite si on a les occurrences de l'événement B satisfaisant la contrainte $ct1$ par rapport à A et l'occurrence de l'événement D satisfaisant la contrainte $ct2$ par rapport à A et la contrainte $ct3$ par rapport à C, et si l'on n'a pas l'événement E après l'occurrence de D tant que la contrainte $ct4$ est vraie, on peut en déduire l'événement inobservable G.

Le diagnostic à l'aide des STC consiste alors à interpréter en ligne l'occurrence d'événements afin d'instancier les motifs qui seront reconnus. Une STC est reconnue lorsque tous ses événements se produisent en respectant leurs contraintes. Deux algorithmes de reconnaissance de STC ont été proposés dans la littérature. Le premier est similaire à la technique de l'automate dupliqué [Bertrand et al. (2007)]. Il peut conduire à des situations de sur-diagnostic [Saddem et al. (2011c)]. Le deuxième algorithme est basé sur le concept du monde [Saddem et al. (2012)] qui résout le problème évoqué. Le principal avantage de ce formalisme est sa lisibilité et son expressivité. En effet, il est capable de décrire clairement tous les comportements souhaités. Il est également compréhensible et lisible par les experts.

3.3 Approches à base de données

Les méthodes à base de données utilisent les données d'historiques issues du système [Venkatasubramanian et al. (2003); Moosavian et al. (2013); Dou and Zhou (2016); Vazan et al. (2017); Han et al. (2017)], qui traduisent l'évolution des états de fonctionnement du système. Ces méthodes font directement des prédictions ou généralisent des modèles à partir des données. Elles n'ont pas besoin d'une modélisation explicite du système en utilisant un modèle mathématique. Elles apprennent de l'expérience et améliorent leurs performances. Ces méthodes utilisent les techniques d'apprentissage automatique qui considèrent le problème de diagnostic comme un problème de classification [Moreno

et al. (2013); Chu et al. (2018)], de regroupement [Kiss et al. (2014)] ou de reconnaissance de formes [Wu et al. (2015)].

- La reconnaissance des formes (appelée en anglais "Pattern recognition")[Nasrabadi (2007); Duda et al. (2012)] est une méthode qui se base sur plusieurs algorithmes de classification permettant de reconnaitre les formes. Une forme est représentée par un point représentatif du fonctionnement du système.
- La classification consiste à utiliser un ensembles d'observations labélisées par un expert afin de construire un classifieur. Ce classifieur attribuera une classe à chaque nouvelle observation à partir des caractéristiques afin de détecter l'occurrence d'une faute.
- Le regroupement est une méthode de classification non-supervisée qui consiste à regrouper les observations en différentes classes homogènes. Une fois les groupes sont identifiés, nous pouvons distinguer les observations représentant les comportements normaux des observations représentant les comportements fautifs du système [Lei et al. (2008); Al-Dahidi et al. (2015)].

Le schéma de principe d'un système de diagnostic à base de données est illustré dans la figure 12.

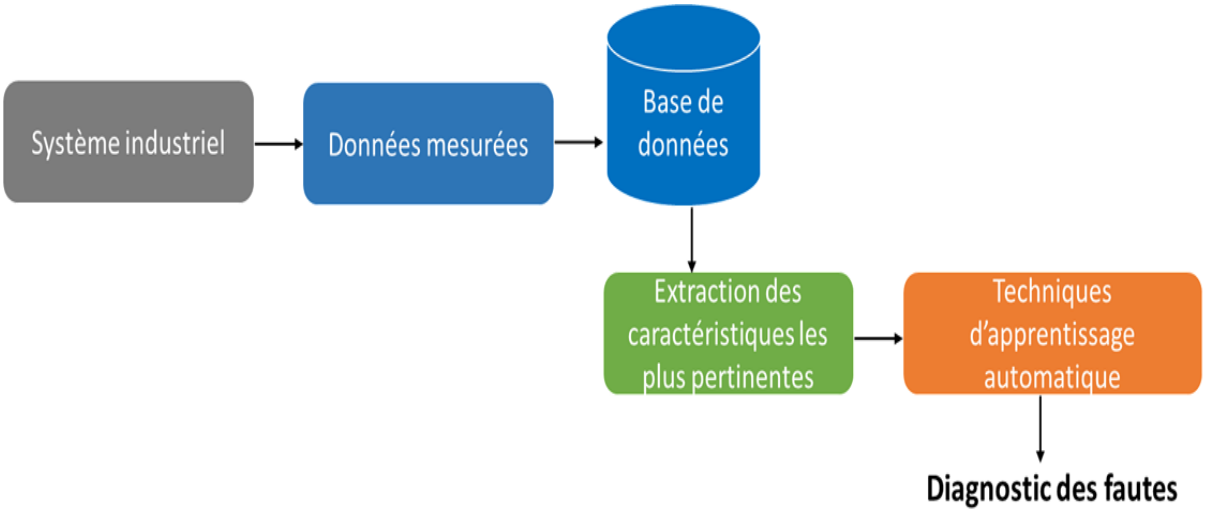


FIGURE 12 – Le principe d'un système de diagnostic à base de données.

3.4 Synthèse sur les méthodes de diagnostic

Toutes les méthodes de diagnostic mentionnées ont montré leur efficacité dans un domaine ou un autre, pour un système ou un autre. Mais lorsque nous parlons de systèmes complexes, ces méthodes rencontrent des difficultés qui les empêchent d'atteindre les objectifs de diagnostic [Philippot (2006)]. En effet, les modes de raisonnements à base de

modèles ou à base de connaissances nécessitent pour fonctionner efficacement soit un modèle analytique précis et approfondi du domaine, soit une base de connaissances qui couvre tous les comportements fautifs. Plus on a un modèle représentatif, correct et robuste ou une base de connaissance complète et cohérente, plus nous privilégions les méthodes à base de modèles ou à base de connaissances. En outre, ces méthodes ne prennent pas en compte l'apparition de nouveaux comportements fautifs dans le système. L'ensemble des fautes à diagnostiquer est fixe. L'ajout d'une nouvelle faute nécessite une mise à jour hors ligne du système de diagnostic, ce qui fait appel à un expert humain pour mettre à jour la base de connaissance ou les modèles.

En parallèle, les méthodes à base de données peuvent fonctionner avec les données disponibles sans avoir besoin de connaissances spécifiques. En effet, elles peuvent enrichir ces données en apprenant des solutions des problèmes rencontrés. Mais pour fonctionner correctement, elles nécessitent une étape de prétraitement qui doit être effectuée correctement afin d'extraire les données les plus pertinentes. Plus, les données sont pertinentes, plus les méthodes à base de données sont efficaces [Tidriri et al. (2016)].

4 Conclusion

A travers ce chapitre, nous avons présenté l'état de l'art concernant le diagnostic des SAP ayant une dynamique discrète. Nous présentons nos hypothèses de travail qui sont les suivantes :

- Les SAP à diagnostiquer sont commandés par des API ayant un fonctionnement cyclique ;
- La phase de diagnostic consiste à détecter les éventuelles fautes et à identifier leurs types et leurs emplacements.
- Les fautes à diagnostiquer sont des fautes internes, causées par la PO.

Dans un deuxième temps, nous avons pu dégager le bilan des constats suivants :

- Les approches à base de modèles sont généralement utilisées dans le cas d'une connaissance suffisante du fonctionnement du système. Elles sont capables de valider la cohérence et la complétude des fautes à diagnostiquer. Cependant, la difficulté majeure de ces approches est leurs coûts d'implémentations élevées.
- Les méthodes à base de connaissances offrent une capacité élevée en raison de la connaissance des symptômes des fautes qu'elles modélisent. Cependant, leurs difficultés majeures résident dans la formalisation des connaissances expertes et leurs mises à jour.

- Les méthodes à base de données profitent des données d'historique. Elles intègrent les méthodes d'apprentissage automatique qui sont des méthodes adaptatives qui apprennent à partir des données.

Au vu de ce bilan, nous nous intéressons dans cette thèse au diagnostic des SAP en utilisant une méthode d'apprentissage automatique. Nous présentons donc dans le chapitre suivant un état de l'art sur l'apprentissage automatique et ses utilisations pour le diagnostic des systèmes industriels.

Chapitre II

Diagnostic basé sur l'apprentissage automatique : État de l'art 2

Sommaire

Introduction

Au regard des limites que présentent les méthodes de diagnostic à base de modèles et à base de connaissances, nous nous intéressons dans le reste de ce manuscrit au diagnostic basé sur l'apprentissage automatique. Ce deuxième chapitre présente ainsi un état de l'art pour ce type de diagnostic. Il se compose de deux parties. Dans la première partie, nous commençons par définir l'apprentissage automatique, ensuite nous présentons les différents avantages de son utilisation. Puis, nous exposons les différentes méthodes ainsi les étapes de déploiement et de développement. Dans la deuxième partie de ce chapitre, nous nous intéressons au « Raisonnement à partir de cas (RàPC) » qui se présente comme une méthode d'apprentissage automatique appartenant à d'autres disciplines telles que les sciences cognitives et les systèmes à base de connaissances. Nous présentons alors son principe, les différentes étapes et les défis majeurs pour le développement d'un système de diagnostic à base de cas.

1 L'apprentissage automatique

Il existe un nombre important d'ouvrages qui s'intéressent à l'apprentissage automatique (en anglais appelé Machine Learning (ML)) [Mohri et al. (2012); Michalski et al. (2013); Bottou (2014); Shalev-Shwartz and Ben-David (2014); LeCun et al. (2015); Jordan and Mitchell (2015); Witten et al. (2016)] et ses méthodes d'apprentissage. Différentes définitions du terme apprentissage automatique y sont proposées. Pour cette raison, nous ne nous limitons pas ici à une unique définition de l'apprentissage automatique mais nous présentons celles qui nous semblent les plus importants.

D'après Arthur Samuel [Samuel (1959)] :
"L'apprentissage automatique est la discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés."

Une autre définition plus technique du terme apprentissage automatique est donnée par Tom Mitchel [Mitchell (1997)] :
"Étant donné une tache T et une mesure P, on dit qu'un programme informatique apprend à partir d'une expérience E si les résultats obtenus sur T, mesurés par P, s'améliorent avec l'expérience E."

Pour illustrer ces définitions, nous prenons à titre d'exemple, un problème académique connu qui consiste à classer une fleur (il s'agit d'une iris) selon des critères observables en trois espèces (*Iris setosa*, *Iris virginica* et *Iris versicolor*). Supposons que nous disposons initialement du jeu de données de l'iris proposé par [Fisher (1936)]. Il comprend 50

échantillons de chacune des trois classes de l'iris. Quatre caractéristiques ont été mesurées pour chaque échantillon : la longueur et la largeur des sépales et des pétales.

Maintenant la question qui se pose : Comment peut-on classer des nouvelles fleurs d'IRIS à partir de ce jeu de données ?

Pour aborder ce problème, nous pouvons utiliser l'une des deux solutions suivantes :

✓ **Solution 1 :** Elle consiste à appliquer une technique qui nécessite des compétences et des connaissances humaines. Son principe est illustré dans la figure 13 : premièrement nous étudions le jeu de données ***(a)*** ; deuxièmement, nous écrivons une liste de règles qui caractérise chacune de ces trois classes ***(b)*** ; troisièmement, nous écrivons un algorithme utilisant cet ensemble de règle afin de raisonner et déterminer la classe d'une nouvelle fleur iris ; et, quatrièmement, nous évaluons la solution proposée ***(c)***. Si elle est bonne, nous pouvons la prendre en compte pour des nouvelles prédictions ***(d)***. Sinon, lors d'une étape d'analyse d'erreur ***(e)***, nous revenons à la première étape afin d'adapter les règles au problème. Ces ajustements manuellement doivent être répéter jusqu'à trouver une réponse satisfaisante.

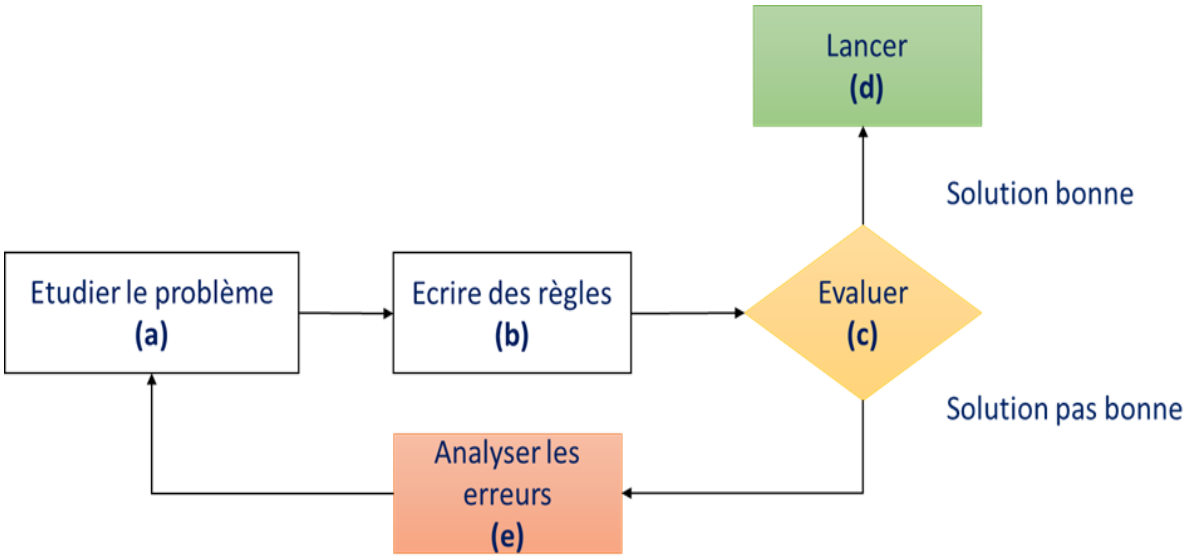


FIGURE 13 – Le principe de la 1 ère solution.

✓ **Solution 2 :** Elle consiste à utiliser une méthode d'apprentissage automatique. Son principe est illustré dans la figure 14. En premier lieu, nous étudions le problème à résoudre **(a)**. Ici, nous disposons d'un jeu de données déjà rangé et il nous faut trouver une solution qui classe les nouvelles iris à partir de celui-ci. Il s'agit donc d'un problème de classification. Pour cela, nous choisissons une méthode de classification parmi les méthodes existantes comme les K-plus proches voisins (KNN), les réseaux de neurones [Rosenblatt (1961)], les séparateurs à vaste marges (SVM) [Suykens and Vandewalle (1999)], etc. La

figure 15 montre la classification de ce jeu de données selon ces trois algorithmes. En second lieu, la méthode choisie déterminera la fonction de prédiction f à partir du jeu de données. Cette fonction permettra de prédire les types des nouvelles fleurs Iris **(b)**. En dernier lieu, nous évaluons la solution proposée **(c)** à travers une mesure de performance P pour déterminer la qualité de prédiction. Si cette performance est bonne **(d)**, nous pouvons la prendre en compte sinon une étape d'analyse d'erreur est nécessaire **(e)**.

Nous détaillons dans la section 1.4 les étapes de développement et de déploiement de ces méthodes.

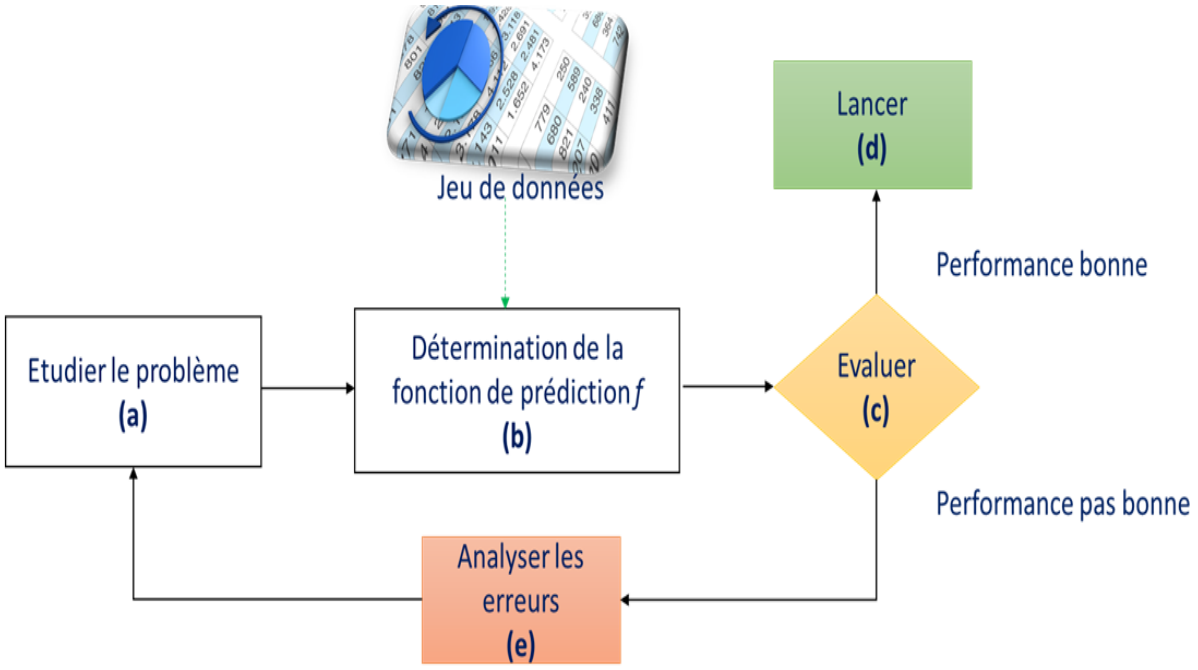


FIGURE 14 – Le principe de la 2 ème solution.

La solution 1 nécessite la présence d'un expert humain pour la description de la liste de règles et cela peut prendre beaucoup de temps et de nombreux ajustements manuels. Alors que la solution 2 n'est pas couteuse puisqu'elle ne nécessite pas la présence d'une expertise humaine, elle est rapide à mettre en œuvre et elle s'adapte automatiquement aux évolutions des données. Il suffit seulement d'entrainer une autre fois l'algorithme choisi avec le nouveau jeu de données.

Le nombre d'applications d'apprentissage automatique ne cesse de s'accroitre et touche de plus en plus de domaines critiques spécifiquement le domaine industriel. Les différentes raisons qui justifient ce succès sont présentées dans le paragraphe suivant.

1.1 Les raisons d'utilisation de l'apprentissage automatique

L'objectif de ce paragraphe est de présenter une vue d'ensemble des différentes raisons d'utilisation de l'apprentissage automatique dans les systèmes industriels. Selon les travaux réalisés au cours de ces dernières décennies, l'apprentissage automatique a été

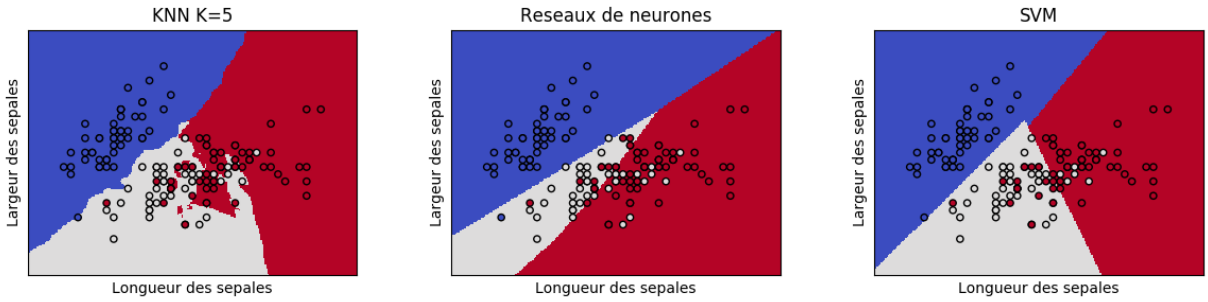


FIGURE 15 – La classification du jeu de données d'IRIS selon le KNN, les réseaux de neurones et le SVM.

utilisé avec succès pour le contrôle de la commande [Mars (2018)], la maintenance prédictive [Susto et al. (2015); Whitaker et al. (2018)], la surveillance [Shafi et al. (2018)], le diagnostic [Azadeh et al. (2013); Luo et al. (2018); Salahshoor et al. (2010); Widodo and Yang (2007)] et la reconfiguration [Shin et al. (2018)], etc.

D'après [Wuest et al. (2016)] ce succès est justifié par plusieurs raisons :

- *Gestion des grandes quantités de données*
  L'apprentissage automatique est capable de gérer de grandes quantités de données hétérogènes, complexes et provenant de différentes sources (tels que les signaux des capteurs, voix, images, données temporelles, vidéo, etc.) avec un effort raisonnable.
- *Meilleure compréhension du problème*
  L'apprentissage automatique permet de réduire la complexité des problèmes rencontrés. Elle aide les opérateurs humains de surveillance à mieux comprendre comment les problèmes sont résolus. En effet, les algorithmes peuvent être inspectés afin de découvrir ce qu'ils ont appris et comment ils ont appris. Parfois, cela révélera des corrélations et des relations entre les données permettant une meilleure compréhension du problème.
- *Apprendre de nouvelles connaissances*
  L'apprentissage automatique peut fonctionner avec les données disponibles sans avoir besoin de connaissances spécifiques et ensuite il est ensuite capable de les enrichir en apprenant à partir des solutions des problèmes rencontrés. Ces nouvelles connaissances peuvent aider les décideurs à la résolution des problème futurs [Alpaydin (2014)].
- *Adaptation automatique*
  L'apprentissage automatique peut s'adapter automatiquement à l'évolution des conditions de fonctionnement des systèmes avec un effort moyen et un coût raisonnable. En effet, les concepteurs de solutions n'ont pas besoin de prévoir et de fournir des solutions pour toutes les situations possibles.

- *De meilleurs résultats*
  L'apprentissage automatique permet de fournir des meilleurs résultats sans avoir besoin de réaliser de nombreux ajustements manuels ou décrire de longues listes de règles.

Il est évident que ces différents avantages que présente l'apprentissage automatique fait de ce domaine un axe de recherche très intéressant qui offre plusieurs méthodes que nous pouvons classer selon plusieurs critères. C'est ce que nous examinons dans le paragraphe suivant.

1.2 Les types de systèmes d'apprentissage automatique

Les systèmes d'apprentissage automatique peuvent être classés selon plusieurs critères qui peuvent être combinés tels que [Géron (2017)] :

- La présence ou l'absence de la supervision humaine (apprentissage supervisé, non supervisé, semi-supervisé ou avec renforcement) [Hastie et al. (2009); Gacquer et al. (2011)] ;
- L'apprentissage s'effectue progressivement ou non (apprentissage en ligne ou apprentissage hors ligne (groupé)) [Mohri et al. (2012)] ;
- La fonction de prédiction est-elle générale ou locale (apprentissage désireux ou apprentissage paresseux).

On distingue deux types de méthodes d'apprentissage automatique : les méthodes "eager" ou dites "désireuses" et les méthodes "lazy" ou dites "paresseuses" [Wei (2015); Wouda et al. (2016); Homayouni and Mansoori (2017); Barbaros et al. (2018); Houeland and Aamodt (2018)].

1.2.1 Les méthodes d'apprentissage de type "eager"

Les méthodes d'apprentissage "désireuses", en anglais appelées "eager" sont des méthodes qui construisent lors de la phase de l'apprentissage (appelée aussi phase d'entrainement) *une fonction de prédiction f* générale pour toutes les données disponibles. Les nouvelles observations qui se produisent après la phase d'apprentissage n'ont aucun effet sur la fonction de prédiction [Barbaros et al. (2018)]. La figure 16 décrit ce fonctionnement en trois étapes :

(a) La première étape consiste à choisir un algorithme d'apprentissage désireux parmi l'ensemble des méthodes désireuses. Dans cet exemple nous avons choisi les Réseaux de neurones.

(b) La deuxième étape consiste à l'entrainement de l'algorithme choisi à partir des données d'entrainement. Le résultat de cette étape est une fonction de prédiction globale f ;

(c) La troisième étape consiste à l'utilisation de la fonction f pour la prédiction de la valeur de la nouvelle observation x.

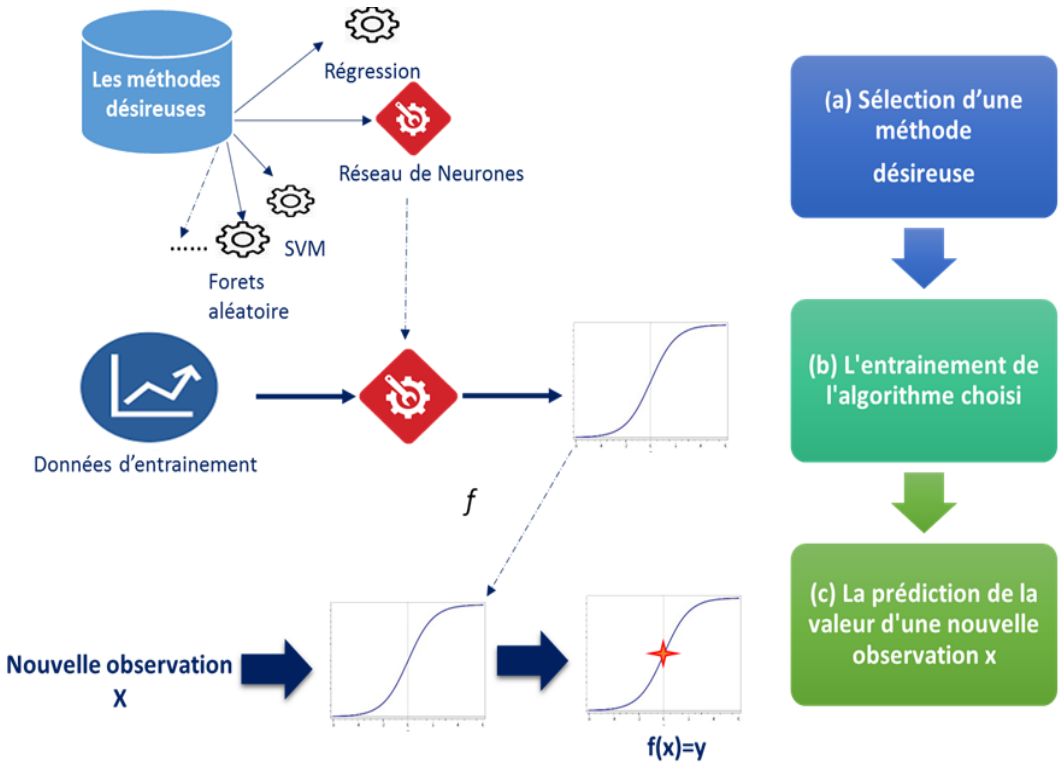


FIGURE 16 – Le fonctionnement des méthodes d'apprentissage désireuses.

Parmi les méthodes qui se basent sur ce type d'apprentissage et qui sont les plus utilisées dans l'apprentissage automatique, nous pouvons citer : les machines à vecteurs de support (voir annexe 1) , les forêts aléatoires (annexe 2), les réseaux de neurones (voir annexe 3), la régression logistique (voir annexe 4), etc.

1.2.2 Les méthodes d'apprentissage de type "lazy"

Les méthodes d'apprentissage "paresseuses", en anglais appelées "lazy" sont appelées méthodes d'apprentissage à base d'observations ou méthodes d'apprentissage à base d'instances ou méthodes d'apprentissage à base d'exemples ou méthodes d'apprentissage basées sur la mémoire [Brighton and Mellish (2002); Cazzolato et al. (2016); de Haro-García et al. (2018); Barbaros et al. (2018)]. Ces méthodes retardent le traitement de l'ensemble de données d'apprentissage jusqu'à la réception d'un nouveau problème. Cela implique qu'elles stockent dans un premier temps les données d'apprentissage en mémoire, ensuite elles cherchent un ensemble de voisins les plus proches à un nouveau problème à résoudre et à partir des prédictions de ces voisins, elles construisent la solution.

La figure 17 décrit ce fonctionnement en trois étapes :

(a) Le choix d'un algorithme d'apprentissage paresseux (dans cet exemple c'est les K-plus proches voisins (KNN)) ;

(b) La détermination de la fonction de prédiction locale f à partir des données les plus similaires de la nouvelle observation x ;

(c) L'utilisation de la fonction f pour la prédiction de la valeur de la nouvelle observation x.

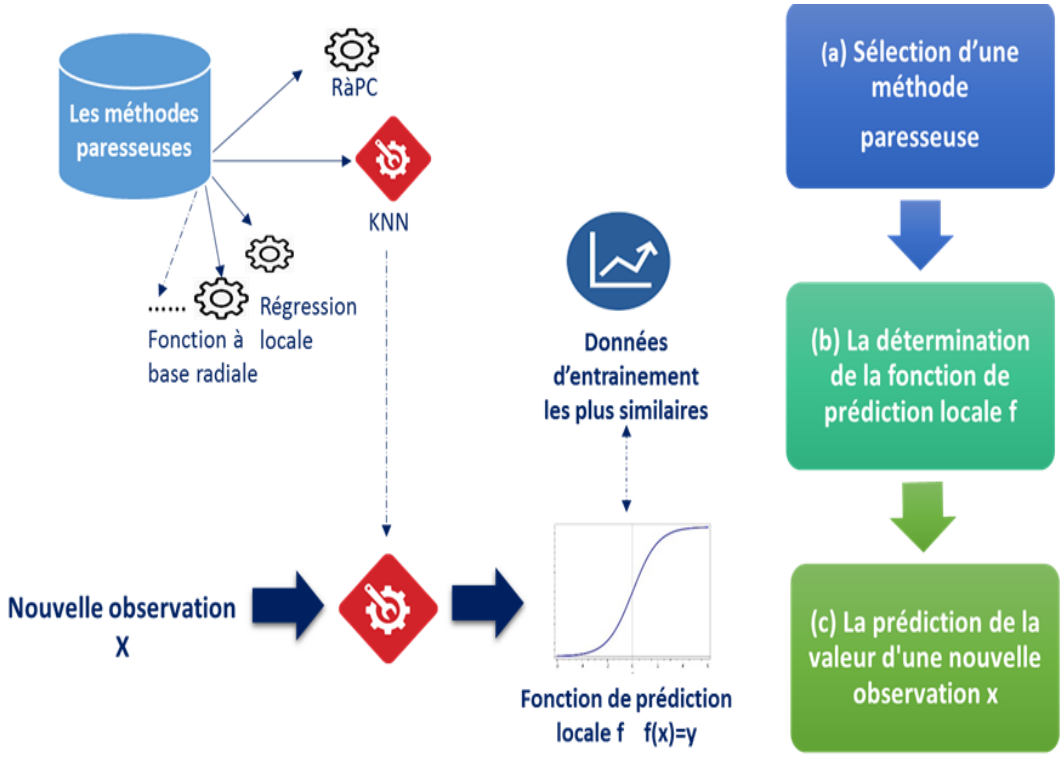


FIGURE 17 – Le fonctionnement des méthodes d'apprentissage paresseuses.

Dans les méthodes d'apprentissage "désireuses", les données d'apprentissage sont ajustées à une seule fonction de prédiction globale alors que les méthodes d'apprentissage "paresseuses" permettent d'apprendre des fonctions de prédictions locales spécifiques. En effet, pour chaque nouvelle observation, une fonction de prédiction locale est construite en se basant sur les données d'apprentissage les plus similaires de la nouvelle observation.

Parmi les méthodes qui se basent sur ce type d'apprentissage, on peut citer la méthode des k-plus proches voisins (appelée en anglais k-nearest neighbor (KNN)) qui peut être utilisée pour résoudre des problèmes de classification ou de régression. Pour fonctionner, cette méthode nécessite de posséder une base d'apprentissage, une mesure de similarité ou de distance et le nombre de voisins à prendre en compte. Pour un problème de classification, son principe est le suivant : Soit X une instance non étiquetée, la méthode cherche les k plus proches instances étiquetées de la base d'apprentissage et affecte X à la classe

qui apparait le plus souvent. Une illustration de la méthode est donnée dans la figure 18.

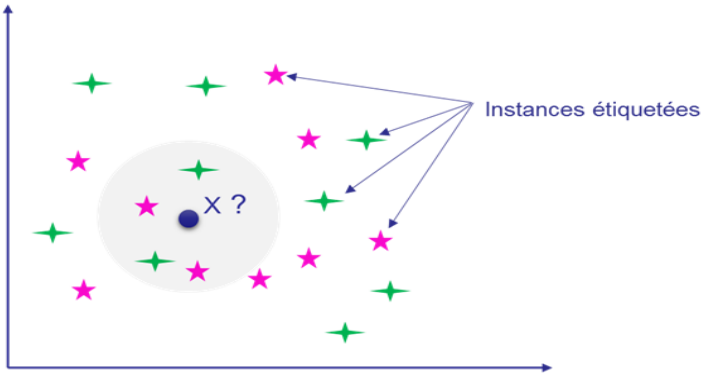


FIGURE 18 – La méthode KNN.

Une autre méthode basée sur ce type d'apprentissage est le raisonnement à partir de cas (RàPC). C'est à cette méthode que nous nous intéressons dans la section suivante. Mais avant tout, il est intéressant de présenter les étapes de développement et de déploiement des méthodes d'apprentissage automatique.

1.3 Étapes de développement et de déploiement des méthodes de l'apprentissage automatique

Le développement et le déploiement des modèles d'apprentissage automatique impliquent une série d'étapes :

a) Classer le problème : Cette étape consiste à comprendre le problème à résoudre, à déterminer les objectifs (prédiction, regroupement, etc.), à définir les critères du succès et les contraintes à respecter.

b) L'acquisition de données : Cette deuxième étape consiste à identifier et collecter les données nécessaires pour supporter le problème. Ces données peuvent provenir de plusieurs sources et peuvent être structurées (telles que les enregistrements des bases de données, les arbres, les graphes, etc) ou non structurées (telles que les images, les textes, les voix, etc).

c) Préparation des données : Dans cette étape, les données doivent être formatées selon l'algorithme d'apprentissage automatique à utiliser. Elle inclut la transformation, la normalisation, le nettoyage et la sélection des données d'apprentissage (si l'apprentissage est supervisé).

d) Entrainer l'algorithme avec des données d'apprentissage et de validation : Au cours de cette étape, les données disponibles sont divisées en trois groupes (données d'apprentissage, données de validation et données de test). L'algorithme est entrainé avec les données d'apprentissage. Ensuite, les données de validation sont utilisées pour régler les hyper paramètres [1] et les données de test sont utilisées pour les tests.

e) Tester l'algorithme sur des données de test : Cette étape consiste à vérifier les performances de l'algorithme sur les données de test. Si ces performances sont bonnes, nous pouvons passer à l'étape suivante.

f) Déploiement de l'algorithme.

2 Raisonnement à partir cas (RàPC)

Le raisonnement à partir de cas (RàPC) ou CBR "Case-Based Reasoning" [Aamodt and Plaza (1994); Watson (1999); Althoff (2001); Riesbeck and Schank (2013)] est une méthodologie de résolution de problèmes issus de l'intelligence artificielle (IA), qui appartient à plusieurs disciplines telles que les sciences cognitives, l'apprentissage automatique et les systèmes à base de connaissances [Bergmann (2002)]. Cette méthodologie est capable d'utiliser les connaissances spécifiques des expériences passées, formalisées sous forme de situations problématiques concrètes appelées des cas. Un nouveau problème est résolu en trouvant un cas passé le plus similaire et en l'utilisant pour la résolution de la nouvelle situation problématique.

Une caractéristique très importante du RàPC est son couplage avec l'apprentissage [Aha et al. (1991); Leake (1996); Diaz-Agudo and Goel (2017); Mosannenzadeh et al. (2017)]. Il permet de mettre à jour des cas, d'apprendre de nouveaux cas, de s'assurer de leur cohérence et d'apprendre des succès et des échecs. En effet, chaque nouvelle expérience est retenue à chaque fois que le problème est résolu avec succès et elle est utilisée immédiatement pour la résolution des problèmes futurs. Dans les cas d'échecs, les raisons sont identifiées et stockées afin d'éviter de commettre à nouveau les mêmes erreurs.

2.1 Résolution des problèmes à partir de cas

Le raisonnement à base de cas (RàPC) est une méthodologie de résolution de problèmes qui s'appuie sur la réutilisation par analogie des solutions des problèmes passés appelées

1. Un hyperparamètre est un paramètre dont la valeur est définie avant le début de la phase d'apprentissage. En revanche, les valeurs des autres paramètres sont apprises à partir de données.

"*cas sources*". Ces cas contiennent des connaissances qui sont utiles pour résoudre des nouveaux problèmes. Les cas sources sont stockés dans une mémoire d'expériences passées appelée "*base de cas (BC)*". Un nouveau problème présente un nouveau cas dont sa partie solution est inconnue. Ce nouveau cas peut être appelé "*cas cible*" ou "une *requête q*" qui nécessite un traitement en quatre étapes consécutives [Aamodt and Plaza (1994)], présentées dans la figure 19 :

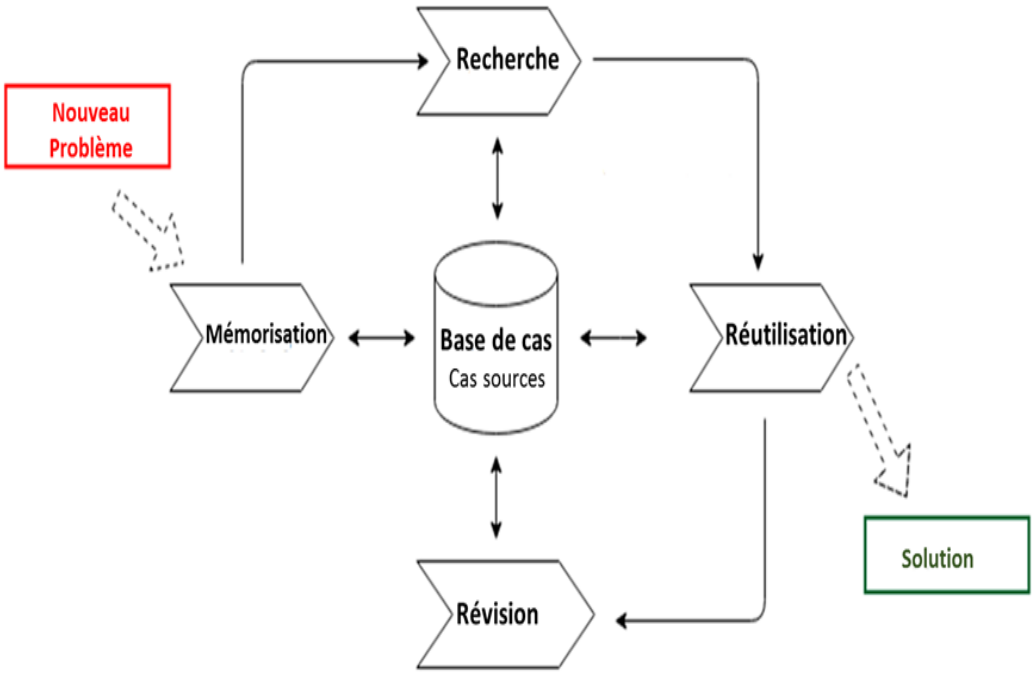


FIGURE 19 – Cycle de RàPC selon Aamodt and Plaza (1994).

2.1.1 Recherche

La première étape consiste à chercher un ou plusieurs cas jugés utiles pour soutenir la résolution du nouveau problème. Selon [Stahl (2003)], la solution idéale est de trouver un ou plusieurs cas les plus similaires et que leurs solutions soient facilement adaptables pour la résolution du nouveau problème. La recherche des cas les plus similaires se fait selon différentes approches [De Loor et al. (2011); Lejri and Tagina (2012); Yan et al. (2017)] telles que l'approche des plus proches voisins [Cover and Hart (1967)], [Liao et al. (1998)], l'approche inductive [Watson and Marir (1994)], l'approche à base de connaissances [Watson and Marir (1994)] et l'approche validée [Aamodt and Plaza (1994)].

2.1.2 Réutilisation

Après avoir sélectionné le ou les cas les plus similaire(s), l'étape de réutilisation essaie d' utiliser les informations contenues dans les solutions de ce(s) cas afin de résoudre le nouveau problème. Wilke et Bergmann [Wilke and Bergmann (1998)] ont présenté un

état de l'art de différentes méthodes d'adaptation existantes. Elles sont présentées en trois classes :

- **L'adaptation Nulle** qui signifie qu'il n'est pas nécessaire d'adapter la solution du/des cas sélectionnés ou alors que l'adaptation s'effectue manuellement par un opérateur humain. Les systèmes commercialisés de RàPC qui fonctionnent en ligne utilisent souvent ce type d'adaptation.

- **L'adaptation transformationnelle** qui consiste à modifier la solution source sélectionnée pour l'orienter afin de satisfaire le nouveau problème en utilisant des opérateurs ou des règles d'adaptation.

- **L'adaptation générative** qui permet de garder pour chaque cas passé une trace des étapes qui ont permis d'aboutir à sa solution. Pour la résolution d'un nouveau cas, une nouvelle solution est générée en appliquant les étapes du cas source le plus similaire.

2.1.3 Révision

Suite à l'étape de réutilisation, le nouveau cas doit être révisé pour s'assurer de l'exactitude de la solution suggérée. L'étape de révision dépend du domaine d'application [De Mantaras et al. (2005); Cojan and Lieber (2014)]. En effet, dans certains domaines, il est possible de tester la solution suggérée dans le monde réel. Dans ce cas, on parle d'une révision par simulation. Mais, dans certains domaines, il est impossible de tester dans le monde réel en raison des dégâts qui peuvent survenir (par exemple le diagnostic médical ou industriel). Dans ces situations, la révision doit être faite manuellement par un expert humain. Si la solution est correcte alors le système intègre le succès sinon le système ou l'expert améliore la solution en utilisant des connaissances spécifiques du domaine.

2.1.4 Mémorisation

Si le cas est résolu avec succès alors il peut être sauvegardé dans la base de cas pour la résolution des problèmes futurs. Cette étape permet de conserver une nouvelle connaissance pour enrichir la base de connaissances. Cependant, afin de contrôler cette étape plusieurs approches ont proposé une réorganisation de la base de cas lors de l'ajout d'un nouveau cas afin de ne garder que les cas les plus utiles [Stahl (2003)].

2.2 Formats de représentation de cas

Un cas est le composant élémentaire d'un système à base de cas. Dans la littérature, il n'existe pas une définition consensuelle du terme cas. En effet, sa définition est liée à

son format de représentation [Kolodner (1993); Lieber (2007)]. A partir des définitions de la littérature, nous présentons 4 types de formats de représentation de cas : structurel, textuel, conversationnel et hybride.

- **Format structurel** : les cas sont représentés selon une structure commune. Ils sont construits à partir de l'extraction des caractéristiques importantes qui modélisent le problème à résoudre. Les formalismes de représentation de connaissances peuvent être statiques ou dynamiques. Ils sont statiques s'ils ne permettent pas de prendre en compte l'aspect dynamique des éléments à modéliser dans un domaine d'application. Sinon, ils sont dynamiques. Les représentations basées sur les vecteurs de caractéristiques, les frames [Plaza (1995)] et la logique de description [Donini et al. (1996)] sont des exemples de représentation statiques. La représentation basée sur les objets [Bergmann (2002)] est un exemple de représentation dynamique.

- **Format textuel** : Un cas peut être représenté sous forme d'un texte libre (non structuré) ou représenté sous forme d'un texte découpé en plusieurs portions étiquetées par des descripteurs (semi-structuré) [Lenz et al. (2003)].

- **Format conversationnel** : Un cas est représenté à travers trois parties (description, questions, actions). (i) Description : décrit le problème à résoudre sous forme de texte. (ii) Questions : représente une série de questions/ réponses. Chaque question a une pondération pour représenter son importance par rapport au cas. (iii) Actions : c'est une description textuelle de la solution à mettre en œuvre [Aha et al. (2001)].

- **Format hybride** : Un cas est représenté en combinant au minimum deux formats. A titre d'exemple, [Lejri and Tagina (2012)] ont proposé un système générique d'aide à la décision qui se base sur le raisonnement à base de cas pour la reconfiguration des systèmes industriels. Le format proposé est présenté dans la figure 20. Il combine entre le modèle structurel et le modèle conversationnel. Il est composé de trois parties :

  (a) *Représentation du problème* qui permet de définir l'état du système et de représenter la défaillance dont souffre le système afin de cibler l'erreur et pouvoir la traiter.

  (b) *Représentation de la solution* qui modélise les solutions utilisées pour reconfigurer le problème.

  (c) *Explications* qui présentent les règles utilisées ou le chemin suivi pour aboutir à la solution. Dans certaines situations, la résolution du problème ne peut pas être faite, alors le modèle conversationnel peut affiner la requête à travers une série de questions/réponses.

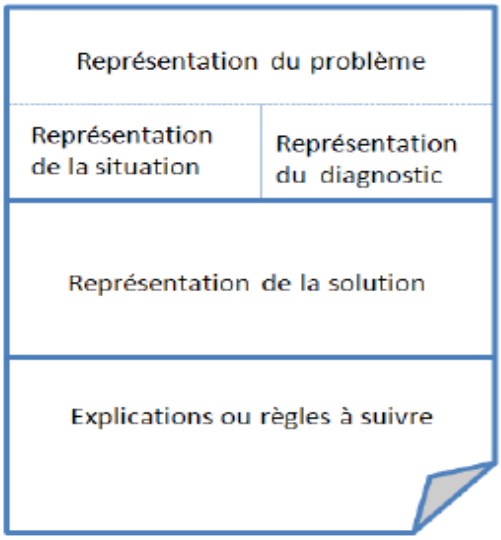


FIGURE 20 – Un exemple d'un format hybride proposé par [Lejri and Tagina (2012)].

Il est évident que le choix du format de représentation est un défi qui se pose lors de la création d'un système de RàPC. Cette étape importante peut influencer toutes les autres phases du RàPC. Afin de choisir le format le plus approprié, nous proposons dans le paragraphe suivant une série de questions avec des réponses décisives.

2.3 Les défis majeurs pour la conception d'un système de RàPC pour le diagnostic

Dans la littérature, plusieurs travaux de recherche ont utilisé le raisonnement à partir de cas pour le diagnostic des systèmes industriels. Généralement, les concepteurs des systèmes de RàPC sont face à un ensemble de problèmes qui peuvent influencer la performance de leurs systèmes. Parmi ces problèmes, on peut citer le format de représentation de cas et la mesure de similarité employée. Nous examinons dans les paragraphes suivant ces deux défis.

2.3.1 Représentation de cas

Le choix d'une représentation de cas robuste est un problème qui se pose lors la conception d'un système de RàPC. En effet, ce choix peut influencer toutes les phases du cycle de raisonnement. Du point de vue utilisateur, les formats de représentation déjà mentionnés sont équivalents mais de point du vue gestion, ils présentent un certain nombre de différences. Il est à noter qu'un bon format de représentation doit être générique indépendant des caractéristiques des systèmes à surveiller tels que son fonctionnement, ses produits, ses ressources, etc. Mais avant de choisir un format de représentation, il est nécessaire de connaitre :

i Le type de données disponibles qui peuvent provenir de diverses sources tels que les signaux des capteurs et des actionneurs, des images, des rapports écrits en texte libre, etc.

ii Le nombre de caractéristiques à extraire pour obtenir un raisonnement efficace. Etre trop spécifique rend la représentation rigide et inefficace et être trop abstrait rend la représentation inutile [Zhou et al. (2010)].

iii Le mécanisme d'extraction des caractéristiques, manuel ou automatique.

Ce défi est largement abordé dans la littérature dans plusieurs travaux qui visent à résoudre des problèmes de diagnostic [Vareilles et al. (2012)], [Behbahani et al. (2012)], [Dendani et al. (2012)], [Xiong et al. (2012)], de maintenance [Chougule et al. (2011)], [Ruiz et al. (2013)] ou de reconfiguration [Lejri and Tagina (2012)], tout en représentant la phase de diagnostic à base de RàPC. Nous présentons ci-dessous la manière dont les auteurs abordent l'étape de représentation de cas :

Dans le travail de [Ruiz et al. (2013)] et [Dendani et al. (2012)], les auteurs ont choisi une représentation basée sur les connaissances du domaine à travers la construction d'une ontologie de domaine. Un cas est décrit sous forme d'un ensemble de descripteurs où chaque descripteur est un doublet : un concept et sa valeur. Le problème de cette représentation réside dans la construction de l'ontologie de domaine qui reste une tâche difficile pour les systèmes complexes.

Les auteurs de [Behbahani et al. (2012)] ont proposé un système de RàPC pour résoudre les problèmes de la Maîtrise Statistique des Processus (MSP). Les auteurs ont proposé une représentation de cas enrichie basée sur 15 descripteurs qui présentent les caractéristiques les plus importantes du problème. Le choix des index est fait en se basant sur des livres pédagogiques de MSP spécialement les tables de matières et les mots clés des 1000 meilleurs articles de MSP. Ces index sont validés par des experts de MSP. L'inconvénient majeur de cette représentation est le choix des descripteurs qui nécessite beaucoup de temps et beaucoup d'efforts.

Les auteurs de [Vareilles et al. (2012)] ont présenté une représentation de cas qui prend en compte les différents types de connaissances : généraux et contextuels. Cependant le cas est représenté à travers 110 variables ce qui est un nombre très important. A chaque nouveau problème de maintenance, l'utilisateur doit remplir un formulaire à 110 entrées.

Dans le travail de [Chougule et al. (2011)], les auteurs ont exposé une représentation de cas en deux parties : description du problème et description de la solution. Le problème est décrit à travers un ensemble de variables tels que le ou les symptômes observés (DTC), date du DTC, numéro de véhicule, modèle du véhicule, année du modèle du véhicule, type

de moteur du véhicule, type de transmission, région de vente. La solution est décrite par une action de réparation utilisée pour résoudre le problème, présentée sous forme d'un code (LB : Labor code), une date de réparation, une description textuelle réalisée par un technicien.

Les auteurs [Xiong et al. (2012)] ont proposé une représentation de cas qui se base sur les signaux des capteurs du système à diagnostiquer. La proposition présente en premier lieu une étape de filtrage des signaux afin de purifier la lecture des capteurs et éliminer les bruits contenus dans les signaux. En second lieu, une étape d'extraction identifie les caractéristiques les plus importantes des signaux des capteurs en prenant en compte l'aspect temporel. Et en troisième lieu, les cas sont construits à partir des signaux horodatés qui sont transformés en des représentations symboliques par intervalle.

Le tableau 1 présente les problèmes à résoudre dans ces différents travaux, les conteneurs d'informations nécessaires pour construire les cas, les formats des cas adaptés et les descriptions des cas. A partir de ce tableau, nous pouvons remarquer que :

- *l'étape d'extraction des caractéristiques importantes (l'indexation)* reste dans la plus part des travaux un processus manuel réalisé par un expert humain qui doit analyser un ou plusieurs conteneurs d'informations comme les historiques des pannes, les arbres de défaillances de l'équipement ou de ses composants, les AMDEC, les signaux des capteurs, les informations disponibles dans des livres ou à partir d'une ontologie du domaine. C'est une étape qui nécessite beaucoup de temps et d'efforts.

- *la représentation de cas* la plus communément utilisée est la représentation structurée en liste de descripteurs. Un cas est généralement présenté en deux parties : une partie problème et une partie solution. Cependant, ces représentations restent statiques. En effet, elles ne s'adaptent pas au fonctionnement dynamique des systèmes industriels et ne permettent pas de manipuler des connaissances complexes. Aussi, elles pourraient générer des problèmes d'explosion combinatoire de descripteurs si le nombre de composants des systèmes à diagnostiquer est très grand.

C'est à partir de cette analyse que nous avons choisi le format de représentation des cas. Le défi suivant est la recherche des cas sources les plus similaires.

2.3.2 Mesures de similarité et de distance

La mesure de similarité permet de quantifier le degré de ressemblance entre une paire de cas. Elle joue un rôle très important dans la phase de recherche des cas les plus similaires

TABLE 1 – Résumé des représentations de cas dans certains systèmes de diagnostic utilisant le RàPC.

Références	**Problèmes à résoudre**	**Conteneurs d'informations**	**Format**	**Description du cas**
[Dendani et al. (2012)]	Diagnostic des fautes d'un système turbine à vapeur.	Ontologie du domaine	Structurel dynamique : objet	Un cas est représenté par une suite d'attributs, leurs valeurs et leurs poids.
Vareille *et al.* [Vareilles et al. (2012)]	Conception des solutions pour des problèmes industriels (maintenance, diagnostic...)..	Informations générales et contextuelles	Structurel statique	Un cas est représenté sous forme de 110 descripteurs et leurs valeurs.
Behbahani *et al.* [Behbahani et al. (2012)]	Résolution des problèmes liés à la maîtrise statistique des processus (MSP).	Informations disponibles dans des manuels de MSP (telles que les tables des matières, les mots clés, ect.).	Structurel statique	Un cas est représenté sous forme d'un vecteur de 15 index, leurs valeurs et leurs poids
Xiong *et al.* [Xiong et al. (2012)]	Diagnostic des pannes dans les SAP.	Signaux des capteurs	Structurel statique	Un cas est représenté sous forme d'un vecteur qui contient toutes les caractéristiques des signaux des capteurs et la classe de la faute.
Chougoule *et al.* [Chougule et al. (2011)]	Identification des anomalies dans les véhicules, détermination de leurs causes et actions de réparation.	Informations à partir de deux bases (BD des symptômes et BD de réclamation).	Hybride : Structurel + textuel	Un cas est un ensemble d'attributs / valeurs + des questions écrites en texte libre
Lejri *et al.* [Lejri and Tagina (2012)]	Reconfiguration d'un système industriel tout en présentant l'étape de diagnostic	Informations disponibles sur le système	Hybride : Structurel+ conversationnel	Un cas est un ensemble d'attributs, leurs valeurs et leurs poids.

[Bergmann and Gil (2014)]. C'est pour cette raison que les systèmes de RàPC sont parfois

appelés "systèmes de recherche de similarité" vue l'importance de celle-ci. Il existe deux approches principales de recherche de cas [Liao et al. (1998)] :

- Les approches appelées computationnelles ou à base de distance qui calculent la distance entre les différents cas et déterminent le cas le plus similaire en se basant sur une mesure de similarité.

- Les approches représentatives où les cas sont représentés dans des structures hiérarchiques comme les arbres ou les épisodes généralisés. Afin de rechercher le cas similaire, il suffit de parcourir ces structures.

Notons que ces deux approches peuvent être utilisées en combinaison. Notre étude présentée ci-dessous concerne seulement les approches à base de distance. En effet, les approches à base de distance sont les plus utilisées dans les applications de RàPC. Nous retrouvons dans la littérature plusieurs mesures de similarité entre cas. Dans [Liao et al. (1998)], les auteurs classent les mesures selon plusieurs critères :

i) La nature des attributs des cas : Généralement, les systèmes de RàPC sont représentés à travers des cas décrits par une liste d'attributs/valeurs. Les attributs peuvent être de types numériques ou nominaux.

ii) *La typicité des cas* : Certains cas peuvent contenir plus de connaissances que d'autres. L'importance d'un cas est décrit à travers sa typicité.

iii) *La pertinence des attributs* : Introduire les degrés de pertinence pour les attributs permet de déterminer les bons attributs dont les valeurs sont significatives par rapport à une situation donnée.

iv) *La complexité des attributs des cas* : Certains problèmes ont généralement des attributs de différents types et d'échelles de mesure. Donc, il est nécessaire de transformer ces problèmes originaux et leurs attributs en équivalent afin de faciliter le calcul de similarité entre cas et la résolution de problèmes.

v) *Les attributs avec des valeurs manquantes* : En réalité, les cas enregistrés et le cas cible peuvent contenir des valeurs manquantes ou nulles. La gestion de ces valeurs est un problème qui se pose lors du calcul de la similarité.

D'une manière générale, nous présentons ces mesures selon trois familles :

(a) **Famille 1 :** Mesures pour les cas présentés par des données quantitatives comme la distance de *Manhattan*, la distance *euclidienne* ou la distance de *Chebychev* qui sont dérivées de la distance de *Minkowski* [Bisson (2000)] en introduisant un paramètre p.

Soit X et Y deux cas avec : $X = \{x_1, x_2,..,x_k\}$, $Y = \{y_1, y_2,..,y_k\}$ et k est le nombre de descripteurs. Le tableau 2 présente les formules de calculs de ces

mesures entre X et Y en fonction du paramètre p .

Mesure	p	Formule
Distance Minkowski	p	$(\sum_{i=1}^{k} \|x_i - y_i)^p\|)^{\frac{1}{p}}$
Distance Manhattan	1	$(\sum_{i=1}^{k} \|x_i - y_i\|^1)$
Distance Euclidienne	2	$(\sum_{i=1}^{k} \|x_i - y_i\|^2)^{\frac{1}{2}}$
Distance Chebyshev	∞	$\lim_{p\to+\infty}((\sum_{i=1}^{k} \|x_i - y_i)^p\|)^{\frac{1}{p}})$

TABLE 2 – Les mesures de similarité utilisées pour les données quantitatives.

Ainsi il existe des distances pondérées qui prennent en compte les degrés de pertinence de chaque attribut comme la distance de Minkowski pondérée, la distance euclidienne pondérée, etc.

(b) **Famille 2 :** Mesures pour les cas présentés par des données qualitatives qui comportent k valeurs différentes et qui peuvent être remplacées par k attributs binaires. Les mesures à utiliser sont soit des mesures qui donnent la même importance pour tous les attributs comme la distance de *Hamming* [Hamming (1950)] ou le *Simple Matching Coefficient* (SMC) [Stahl (2003)] ; soit des mesures qui donnent une importance pour certains attributs par rapport aux autres comme la *SMC pondérée*, le *coefficient de Jaccard*, similarité *cosinus*, *coefficient de Dice* [Bisson (2000)], etc.

(c) **Famille 3 :** Mesures pour les cas présentés par des données mixtes (quantitatives et qualitatives). Il est impossible d'avoir une mesure précise. Cette dernière doit être modélisée pendant le processus d'acquisition des connaissances [Bergmann and Gil (2014)] et elle doit être simple et aussi efficace [Liao et al. (1998)].

La plus part des travaux récents comme [Dendani et al. (2012)], [Lejri and Tagina (2012)], [Chougule et al. (2011)], [Vareilles et al. (2012)], [Dendani et al. (2012)] conçoivent différentes mesures pour supporter la représentation de cas complexes. La plupart des travaux étudiés partage le principe de la similarité locale et globale présentées dans [Stahl (2003)]. En effet, la mesure de similarité locale est calculée entre les descripteurs de deux cas et la mesure de similarité globale est l'agrégation des similarités locales en prenant en compte le degré d'importance de chaque descripteur.

A titre d'exemple la mesure de similarité proposée dans le travail de [Dendani et al. (2012)] se base sur une similarité globale (équation II.1) qui mesure le degré de rapprochement entre une requête (q) et un cas (c) en se basant sur une mesure se similarité local (équation II.2) qui mesure la similarité entre deux attributs. Ces mesures se calculent comme suit :

- **Similarité globale :**

$$Sim(q,s) = \frac{\sum_{S \in CS} (Sim(q.s, c.s).\mathrm{w_s})}{|\mathrm{CS}|} \tag{II.1}$$

Avec $\mathrm{w_s}$ est le poids associé à chaque attribut s, CS est l'ensemble des attributs simples dans q et c, |CS| est son cardinal, $q.s$ représente les attributs simples de q et $c.s$ représente les attributs simples de c, Sim *(c.s,q.s)* est la similarité locale. Elle est définie comme suit :

- **Similarité locale :**

$$Sim(q.s, c.s) = \begin{cases} 0 & \text{if } \mathrm{V_{q.s}} = \mathrm{V_{c,s}} \\ 1 & \text{sinon.} \end{cases} \tag{II.2}$$

Avec $\mathrm{V_{q.s}}$ est la valeur de l'attribut s appartenant à q et $\mathrm{V_{c,s}}$ est la valeur de l'attribut s appartenant à c.

3 Conclusion

A travers ce chapitre, nous avons exposé l'état de l'art présentant une méthode d'apprentissage automatique "paresseuse" qui est le RàPC et son utilisation pour le diagnostic des systèmes industriels. Nous avons pu à travers cette étude dégager le constat suivant :

- Comme toute méthode d'apprentissage automatique, le RàPC présente beaucoup d'avantages telles que la mise en œuvre facile et simple, la capacité à gérer une grande quantité de données variées avec un effort raisonnable, la capacité à apprendre de nouvelles connaissances et à fournir des meilleurs résultats.

- Les concepteurs des solutions existantes pour le diagnostic des systèmes industriels en utilisant le RàPC sont face à deux défis qui peuvent influencer la performance de leurs systèmes. Ce sont : la représentation de cas et la recherche des cas les plus similaires.

- Les travaux présentant des solutions pour le diagnostic des systèmes automatisés

de production ayant une dynamique à évènements discrets sont peu nombreux.

- Les solutions existantes sont trop spécifiques aux systèmes industriels à diagnostiquer et ne peuvent pas, par conséquent, être généralisées.

Au vu de tous constats, nous nous proposons de nous intéresser dans la suite de cette thèse au diagnostic des systèmes automatisés de production en utilisant le RàPC. Dans le reste de ce manuscrit, nous détaillons notre approche pour résoudre cette problématique.

Chapitre III

Système d'aide au diagnostic en ligne des SAP

Sommaire

Introduction

Au regard des inconvénients que présentent les travaux de diagnostic industriel utilisant le raisonnement à partir de cas, nous proposons dans ce chapitre de définir une solution générique pour l'aide au diagnostic des Systèmes Automatisés de Production. Le chapitre se compose de deux parties. Dans la première nous définissons les systèmes d'aide à la décision et leurs différents types afin de positionner le type de systèmes qui nous intéresse. Dans la deuxième partie du chapitre, nous présentons l'architecture du système proposé et nous illustrons ses différents composants à travers l'application au système de tri de caisses.

1 Systèmes d'aide à la décision

1.1 Introduction aux SAD

Un **S**ystème d'**A**ide à la **D**écision (**SAD**) (en anglais appelé **D**ecision **S**upport **S**ystem (**DSS**)) est un système d'information interactif, flexible, adaptable et spécifiquement développé afin d'apporter l'aide et l'assistance aux décideurs dans leurs processus de décision. En effet, dans les situations complexes ou mal structurées, les SAD s'avèrent nécessaires pour fournir aux décideurs des informations pertinentes qui leur permettront de mieux comprendre ces situations. Les SAD proposent d'effectuer des choix, des tris ou des rangements pour prendre les décisions. Cette assistance par la Machine permet d'améliorer la qualité de la prise de décision et offre un équilibre entre le jugement humain et le traitement des informations par la Machine (coopération Homme-Machine) [Salem et al. (2015); Pacaux-Lemoine et al. (2016)].

Les SAD sont utilisés pour résoudre de nombreux problèmes du domaine industriel. Parmi ceux-ci, on peut citer :

- La régulation et la reconfiguration des réseaux de transports [Zidi (2007); Mejri (2012)] ;
- Le diagnostic et la maintenance des systèmes automatisés [Hedberg et al. (2018); Gao et al. (2009)] ;
- L'ordonnancement des systèmes manufacturiers [Trojanowska et al. (2017)] ;
- La maintenance des systèmes industriels [Reuss et al. (2018)].

1.2 Classification des SAD

La littérature présente plusieurs manières de classer les SAD [Stodolsky (1982); Hackathorn and Keen (1981); Holsapple and Whinston (2001); Power (2007)]. Dans ce qui suit, nous avons choisi de présenter deux classifications : celle de [Hackathorn and Keen (1981)] et celle de [Efraim et al. (2001); Power (2007)].

Un SAD peut fournir un soutien dans toutes les phases du processus de prise de décision et ce soutien peut être donné à une seule personne, un groupe de personnes ou une organisation. Selon ce critère, [Hackathorn and Keen (1981)] présentent trois catégories de SAD :

- *Les SAD individuels* qui fournissent de l'aide à une seule personne (décideur).
- *Les SAD pour groupe* qui fournissent un soutien à un groupe de personnes engagé dans des tâches distinctes mais étroitement liées.
- *Les SAD pour organisation* qui offrent un soutien à un groupe de personnes coopérant entre eux afin de réaliser des tâches ou des activités organisationnelles.

Une autre classification est donnée par [Efraim et al. (2001)] et [Power (2007)] où les SAD sont différenciés selon le type de fonctionnement (guidés par des modèles, des données, des communications, des connaissances ou le web)(voir la figure 21) :

- *Les SAD guidés par des modèles* : Ce type de SAD utilise des données et des paramètres limités afin d'aider les décideurs à analyser une situation. Il se base sur des modèles d'analyse d'hypothèses et il propose des simulations, des optimisations, des prévisions et des suggestions de décision, etc [Kadri (2014)].

- *Les SAD guidés par des données* : Ce type de SAD permet d'analyser une très grande quantité de données de différents types afin de produire des classifications, des prévisions et des associations, etc.

- *Les SAD guidés par des communications* : Ce type de SAD utilise les technologies de réseau et de communication pour faciliter la collaboration et la communication pour la prise de décision.

- *Les SAD à base de connaissances* : Ce type de SAD se base sur différents types d'expertises pour supporter la résolution des problèmes. L'expertise implique la connaissance d'un domaine bien particulier. Cette connaissance peut avoir différentes formes telles que des règles, des procédures, des recommandations, etc.

- *Les SAD guidés par le web* : Ce type de SAD est implémenté dans un environnement distribué. Il soutient le processus de prise de décision des clients de web en

leur fournissant plus d'informations avant de décider d'acheter un article ou un service.

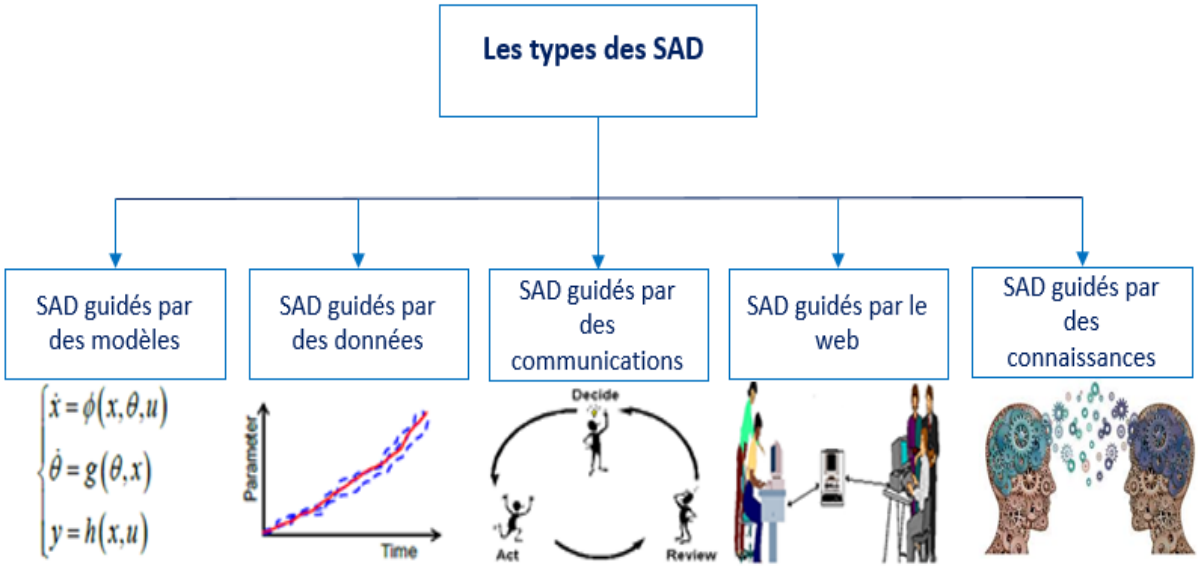


FIGURE 21 – Les types de SAD selon [Efraim et al. (2001)] et [Power (2007)].

Dans ce travail, nous nous concentrons sur les SAD à base de données étant donné que nous ne disposons que des données d'historique qui traduisent l'évolution des états de fonctionnement du système. A partir de ces données, ce type de SAD est capable de les analyser, d'extraire les caractéristiques les plus importantes et de les formaliser sous forme de cas. Pour prédire les états du fonctionnement du système à diagnostiquer, nous utilisons le RàPC.

2 Proposition d'un système d'aide au diagnostic des SAP

Jusqu'à présent la surveillance des systèmes automatisés de production (SAP) est une activité humaine exécutée par des **O**pérateurs **H**umain de **S**urveillance (**OHS**). Ces opérateurs analysent les informations acquises du système surveillé pour prendre des décisions adaptées suite à l'apparition des comportements indésirables. Les OHS sont des facteurs de fiabilité dans l'exécution des tâches puisqu'ils se basent sur leurs connaissances, leurs expériences et leur savoir-faire mais ils peuvent aussi entrainer un manque de fiabilité. En effet, gérer une grande quantité de données et agir rapidement peuvent les conduire à prendre des décisions incorrectes [Fakhfakh (2015)] qui dégradent encore plus la situation.

C'est dans cette optique, que nous proposons dans la suite de ce chapitre un **S**ystème d'**A**ide à la décision pour le **D**iagnostic des fautes dans les systèmes **A**utomatisés de

Production (**SADAP**) permettant d'apporter l'aide et l'assistance aux OHS afin d'accomplir leurs tâches. Le système proposé présente une synergie entre les SAD et le RàPC. Le raisonnement à base de cas a été utilisé pour aider à la prise de décision dans plusieurs domaines tels que la médecine [Choy et al. (2018)], [Brown et al. (2017)] et l'industrie [Abele and Weyrich (2017)], [Chen et al. (2016)], [Berman et al. (2018)]. Mais comme nous l'avons indiqué dans le chapitre précédent, il y a eu peu de travaux importants pour le diagnostic des SAP ayant une dynamique à évènements discrets. Ceci contribue à l'originalité de notre approche qui présente un système d'aide à la décision à base de cas pour le diagnostic de cette classe de systèmes.

2.1 Architecture du système proposé

Le modèle fonctionnel générique de notre SADAP comprend trois composants :

a) Une **Interface Homme-Machine (IHM)** qui se présente comme un outil de dialogue entre le SADAP et les opérateurs humains de surveillance. En effet, c'est le principal outil au travers duquel ces derniers expriment leurs choix et donnent leurs décisions finales.

b) Un **Bloc Hors Ligne (Bloc_HL)** qui permet de collecter et représenter les données les plus pertinentes afin de décrire les comportements normaux et défaillants du SAP sous la forme d'un ensemble de cas. Ce bloc s'appuie sur deux conteneurs (voir la figure 22) :

- *Un conteneur de données* qui regroupe toutes les données disponibles sur le système à diagnostiquer. Ces données vont servir à la formulation et à la construction des cas. Généralement, elles peuvent provenir de diverses sources telles que les bases de données des historiques contenant entre autres les signaux des capteurs et des actionneurs dans différents moments, les cahiers des charges des systèmes, les rapports écrits par les experts du domaine, etc.

- *Un conteneur de connaissances* qui regroupe deux bases de cas et un ensemble des mesures de similarité. Ce conteneur appartient également au bloc en ligne.
  - *Les deux bases de cas BCN et BCD* permettent de sauvegarder respectivement les deux sous-ensembles de cas (Normal et Défaillant) construites dans le Bloc_HL et mises à jour suite à chaque nouvelle observation non connue par le SADAP.
  - *L'ensemble des mesures de similarité* représente les différentes mesures de similarité permettant de quantifier la différence entre cas afin de

déterminer les cas sources les plus similaires à un nouveau cas non résolu. Le choix de la mesure se fait par l'OHS dans le Bloc_HL et elle peut être modifiée dans le bloc en ligne. En effet, dans certaines situations, l'OHS peut tester plusieurs mesures de similarité afin de déterminer la meilleure.

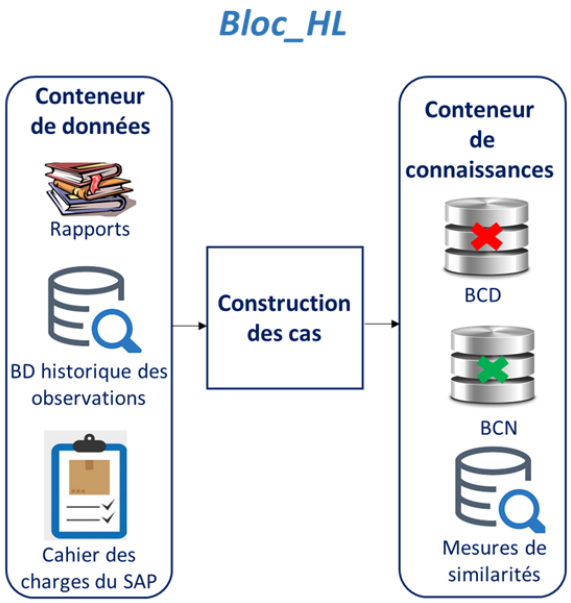


FIGURE 22 – Le Bloc_HL du SADAP.

c) Un **Bloc En Ligne (Bloc_EL)** permettant de résoudre chaque nouveau cas en déterminant sa partie solution et d'enrichir au fur et à mesure les deux bases de cas précédentes (*BCN* et *BCD*) par des nouvelles connaissances.

Ces deux blocs sont développés dans les sous-sections suivantes.

2.1.1 Le Bloc_HL

Dans ce premier bloc, nous décrivons comment mettre en forme et transformer les données existantes en connaissances lisibles et exploitables par la machine. Ces connaissances sont appelées des cas et elles sont sauvegardées dans deux bases de cas. Mais avant de décrire le fonctionnement de ce bloc, il est important de choisir le format de représentation des cas.

✠ Proposition d'un format pour la représentation des cas

La représentation des cas est la première étape de ce bloc. Comme nous avons indiqué auparavant, cette étape importante peut influencer toutes les autres phases du cycle du raisonnement à partir de cas. Dans cette partie, nous proposons un format de représentation de cas générique [Ben Rabah et al. (2017a)] capable de formaliser les différentes

données existantes dans des cas. Un cas, de manière globale, est composé de deux parties : la représentation du problème et la description de la solution. Ce format est illustré dans la figure 23.

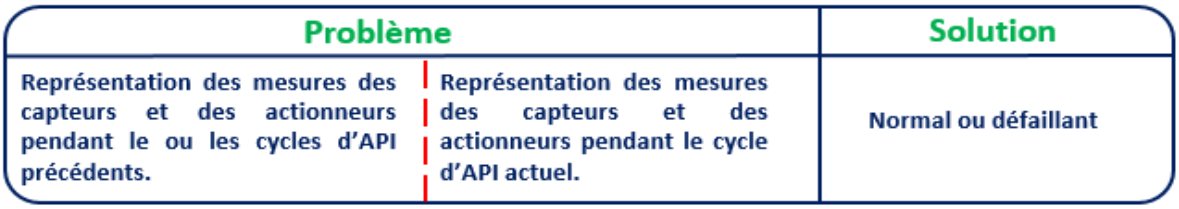


FIGURE 23 – Le format générique d'un cas.

a) Représentation du problème

Cette partie sert à représenter les caractéristiques les plus importantes du système à diagnostiquer afin de comprendre la situation à diagnostiquer. Pour ce faire, nous nous sommes basés sur les mesures des capteurs et des actionneurs obtenus pendant le cycle de l'API actuel (noté t) et le cycle ou les cycles d'API précédents ($t-1$, $t-2$, $t-3$, ..., $t-z$) avec z le nombre de cycles précédents dont les entrées/sorties évoluent (voir la figure 24). Ces mesures sont représentées par un ensemble de descripteurs où chaque descripteur est un *z+1* triplet représenté par :

$$<Co_i,\ V_t,\ V_{t-1},V_{t-2},...,V_{t-z}>$$

Co_i est l'identifiant d'un capteur ou d'un actionneur du système, V_{t-1}, V_{t-2}, V_{t-3},...,V_{t-z} sont les valeurs du composant Co_i aux cycles $t-1$, $t-2$, $t-3$,..., $t-z$ et V_t est la valeur du composant Co_i au cycle t.

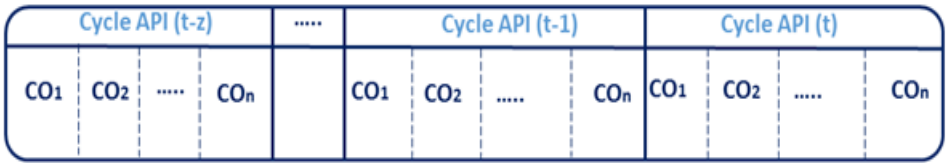


FIGURE 24 – Représentation du problème.

b) Représentation de la solution

Cette partie permet de qualifier le type de cas : normal ou défaillant. Un cas normal est un cas représentant un fonctionnement normal du système à diagnostiquer tandis qu'un cas défaillant est un cas décrivant un fonctionnement défaillant du système à diagnostiquer suite à l'occurrence d'une ou plusieurs fautes. Cette partie est composée de deux sous-parties :

- *La localisation* permettant de repérer le composant affecté.

– *L'identification* permettant d'identifier la faute détectée (son type et son identifiant).

Nous rappelons que les fautes à diagnostiquer peuvent être de quatre types : (i) blocage d'un capteur ou d'un actionneur à 0, (ii) blocage d'un capteur ou d'un actionneur à 1, (iii) passage inattendu d'un capteur ou d'un actionneur de 0 à 1 et (iv) passage inattendu d'un capteur ou d'un actionneur de 1 à 0.

✠ Construction des cas de la base d'apprentissage

Pour ce faire, un regroupement des mesures, qui sont enregistrées dans la BD historique, est nécessaire pour construire les parties problèmes des cas. Le principe de ce regroupement est détaillé à travers un exemple présenté dans la figure 25.

Soit un SAP qui possède un actionneur X et deux capteurs Y et Z. Les mesures de ces différents composants pour différents cycles d'API sont décrits dans le tableau (a) de la figure 25. Les parties problèmes sont définis à travers le regroupement d'moins deux instances consécutives et différentes. En effet, il faut prendre en compte seulement les cycles où les capteurs et les actionneurs changent de valeurs. Par exemple, le *Problème*1 est défini à travers le regroupement des valeurs de l'instance $i1$ et l'instance $i4$.

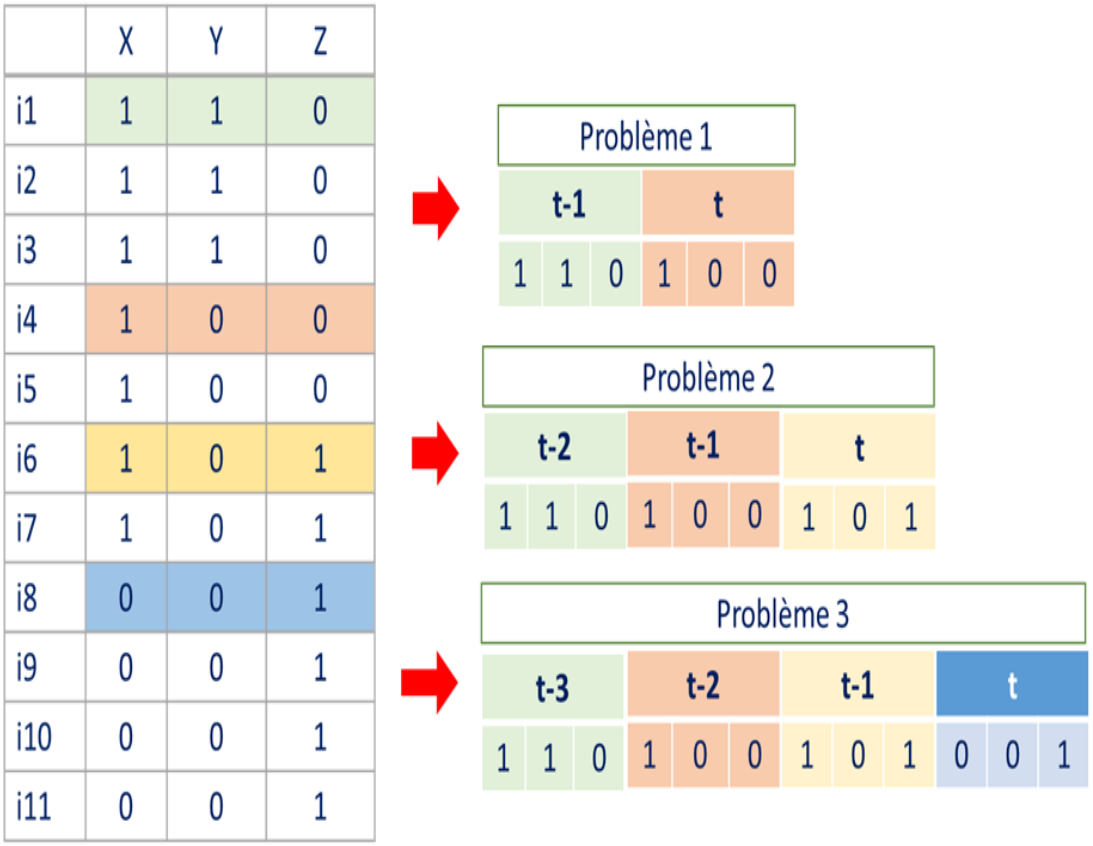


FIGURE 25 – Le principe de regroupement.

Après la formation des parties problèmes des cas, vient l'étape de labellisation des cas

où un expert humain donne son avis par rapport à chaque cas en indiquant son état du fonctionnement (c'est-à-dire normal ou défaillant) et son interprétation. Deux situations peuvent se présenter :

- Si le cas présente *un fonctionnement normal* alors l'expert définit la sous partie Etat à N et donne son interprétation pour le cas. Cette interprétation sert à expliquer la partie problème en décrivant les événements occurrents dans le système.

- Si le cas présente *un fonctionnement défaillant* alors l'expert repère le composant affecté et identifie la faute détectée.

Le sous-ensemble des cas représentant les comportements normaux du système est répertorié dans la base de cas normaux (BCN) et le sous-ensemble des cas représentant les comportements défaillants du système, est stocké dans la base de cas défaillants (BCD).

✠ Choix de la mesure de similarité

Avant d'entamer le Bloc_EL, l'OHS peut modifier la mesure de similarité choisie par défaut parmi un ensemble de mesures de similarité [Cha (2007)]. Le choix de cette mesure dépend de la représentation de cas utilisée et du type de données à comparer (nominal, ordinal, continu ou binaire, etc.). Dans nos travaux, nous exploitons des SAP avec des capteurs et des actionneurs ayant seulement deux valeurs possibles : 0 ou 1 (c'est-à-dire Tout Ou Rien). Pour cela, le système propose un ensemble de mesures de similairité et de distance pour les données binaires. Soit Cas_i et Cas_j deux cas dont leurs parties problèmes sont composés par des valeurs binaires avec n est le nombre des bits du cas Cas_i et m est le nombre des bit du cas Cas_j. A partir des variables s, q, r, p qui sont définies comme suit, ces mesures sont décrites dans le tableau 3.

- s représente le nombre d'attributs des bits dont la valeur est à 1 dans le Cas_i et dans le Cas_j ;
- q représente le nombre d'attributs des bits qui sont à 1 dans le Cas_i et 0 dans le Cas_j ;
- r représente le nombre d'attributs des bits qui valent 0 dans le Cas_i et 1 dans le cas Cas_j ;
- p représente le nombre d'attributs des bits qui sont à 0 dans le Cas_i et dans le cas Cas_j.

Nous avons choisi d'utiliser le coefficient d'appariement simple (en anglais appelé "*Simple matching coefficient (SMC)*") [Legendre and Legendre (1998)] comme une mesure par défaut (décrite dans tableau 3). Cette mesure calcule la similarité entre les attributs binaires qui sont symétriques, c'est-à-dire les attributs dont leurs valeurs ont les mêmes importances [Legendre and Legendre (1998)].

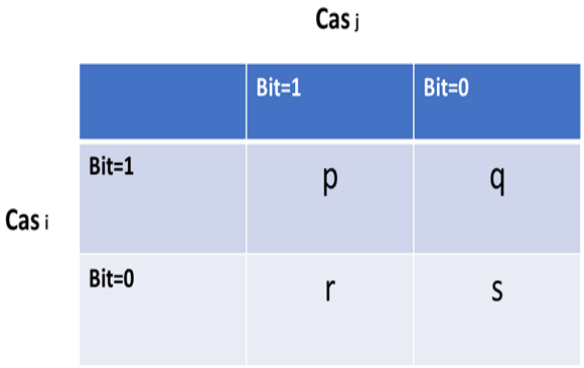


FIGURE 26 – Les valeurs de s, q, r et p pour le calcul de la similarité SMC.

TABLE 3 – Les mesures pour les données binaires.

Mesure	**Formule**
Similarité SMC	$\frac{p+s}{p+s+q+r}$
Similarité de $Jaccard$	$\frac{p}{p+q+r}$
Similarité $Dice$	$\frac{2p}{2p+q+r}$
Similarité $3W\_Jaccard$	$\frac{3p}{3p+q+r}$
Similarité $Roger\&Tanimoto$	$\frac{p+s}{p+2(q+r)+s}$
Distance de $Hamming$	$q+r$
Distance $Euclidienne$	$\sqrt{q+r}$
Distance $Vari$	$\frac{q+r}{4(p+q+r+s)}$

2.1.2 Le Bloc_EL

Dès la mise en fonctionnement du SAP, le SADAP recueille les signaux des capteurs et des actionneurs pour le cycle d'API actuel et les cycles précédents. Ensuite, il forme un nouveau cas non résolu (noté *Ncas*) et il exécute les quatre étapes suivantes (voir figure 27) afin de déterminer le type de cas :

- Recherche du ou des cas sources les plus similaires ;
- Choix de la solution la plus adéquate par l'OHS ;
- Réutilisation de la solution du cas sélectionné ;
- Sauvegarde du nouveau cas résolu dans la base de cas correspondante.

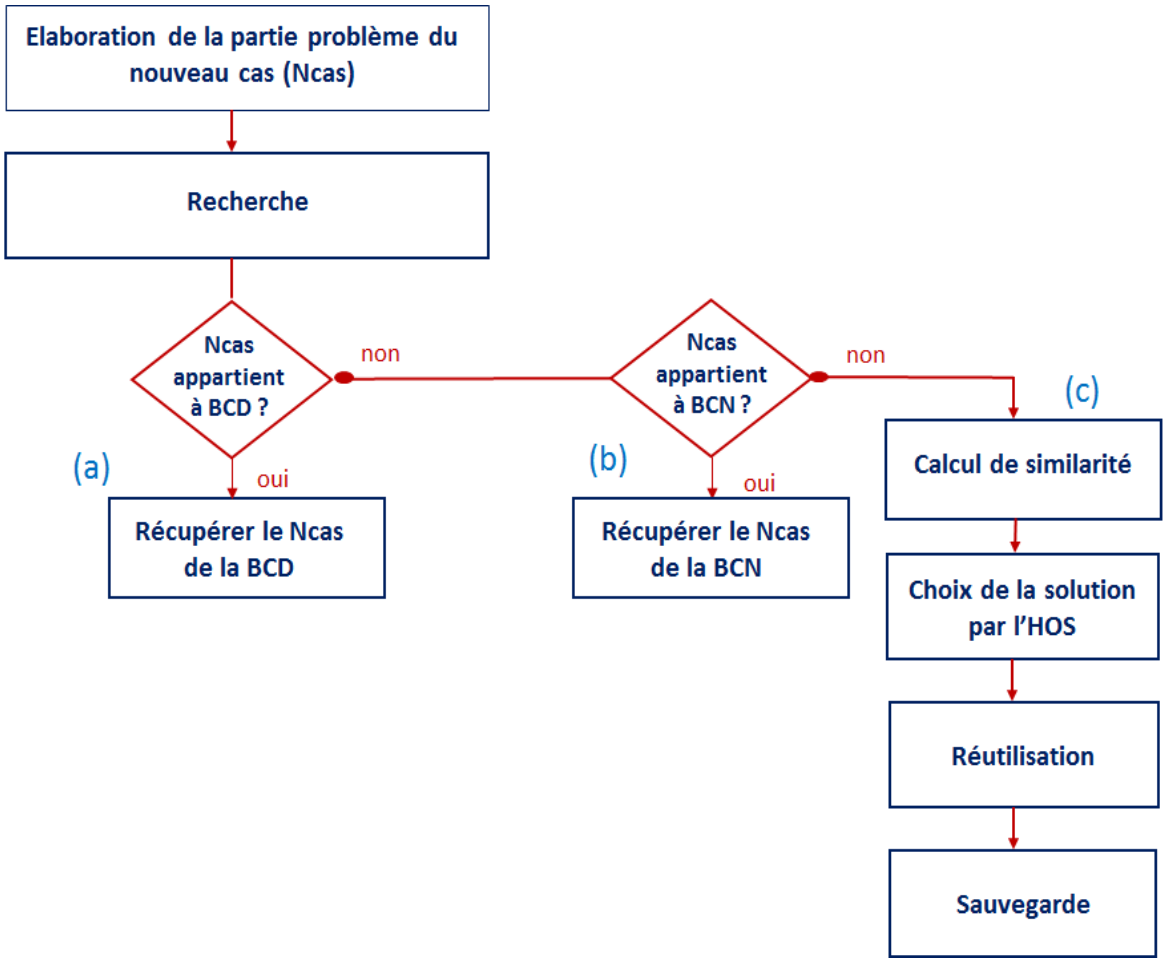


FIGURE 27 – Le Bloc_EL en ligne du SADAP.

i) Recherche

Cette première étape permet de sélectionner le cas source similaire ou les cas sources les plus similaire(s) au nouveau cas. Afin d'atteindre un tel objectif, le SADAP peut avoir trois scénarios possibles :

(a) Le *Ncas* existe dans la *BCD* alors le SADAP se retrouve dans une situation défaillante déjà connue.

(b) Le *Ncas* existe dans la *BCN* alors le SADAP se retouve dans une situation normale déja connue.

(c) Le *Ncas* n'existe pas dans les deux bases BCN et BCD alors le SADAP calcule la similarité entre ce cas et tous les cas sources des deux bases. Ensuite, il affiche, sur son interface, les résultats de calcul de similarité.

ii) Choix de la solution

Dans le troisième scénario et suite aux calculs des similarités entre le *Ncas* et les cas sources des deux bases, le SADAP demande l'avis de l'OHS afin de sélectionner le résultat qui lui semble le plus adéquat en se basant sur ses expériences, ses connaissances et son savoir-faire.

iii) Réutilisation de la solution du cas sélectionné

Dans cette étape, nous avons choisi d'utiliser une adaptation Nulle (décrite dans le chapitre précédent), ce qui veut dire qu'il n'est pas nécessaire d'adapter la solution du cas sélectionné. Le nouveau cas héritera directement la solution du cas sélectionné par l'OHS.

iv) Sauvegarde du nouveau cas résolu

Le nouveau cas résolu est sauvegardé dans l'une des deux bases pour des utilisations futures. L'étape de sauvegarde est capable d'enrichir la base de cas avec des nouvelles connaissances acquises.

2.2 Exemple illustratif

Le but de cette section est d'appliquer la méthode proposée sur un SAP afin d'illustrer notre première contribution. La phase d'expérimentation a été réalisée sous le logiciel ITS PLC (Interactive Training System for PLC) proposé par la société portugaise Real Games (www.realgames.pt). ITS PLC est un outil éducatif, de formation à l'automatisation et de validation des algorithmes de commande grâce à une expérience interactive en temps réel [Riera et al. (2010)]. Il offre des simulations 3D de parties opératives de cinq systèmes industriels (tri de caisses, système de mélange, palettiseur, robot « pick and place » et magasin automatisé). L'utilisateur de ITS PLC peut injecter des fautes dans les capteurs et les actionneurs à travers deux panneaux des pannes (voir les figure 28 et 29). Le panneau de la figure 28 permet d'injecter des fautes de blocage à 0 dans les capteurs et les actionneurs sélectionnés alors que le panneau de la figure 29 permet d'injecter des fautes de blocage à 1 dans les différents capteurs et actionneurs.

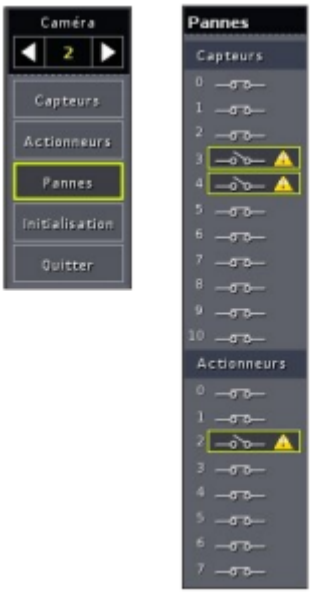


FIGURE 28 – Le panneau de création de fautes de blocage à 0.

FIGURE 29 – Le panneau de création de fautes de blocage à 1.

Dans notre étude, nous avons choisi le système de tri de caisses de l'outil ITS PLC pour appliquer notre approche. Il a été proposé dans [Saddem and Philippot (2014)] comme un système de référence. Ce système qui est présenté dans la figure 30, permet de transporter des caisses d'un convoyeur d'alimentation vers des monte-charges en les triant en fonction de leurs tailles (petites ou grandes caisses). Ce système est composé d'un convoyeur d'alimentation (A), de trois convoyeurs à bande (B, E, G), d'un plateau tournant (C), de rouleaux (D) et de deux monte-charges d'évacuation (F ou H). Le convoyeur d'alimentation (A) délivre aléatoirement des petites et des grandes caisses, chargées sur des palettes. Les palettes sont transportées au moyen d'un convoyeur à bande (B) vers un plateau tournant (C) et doivent ensuite être chargées sur les rouleaux (D). Les palettes effectuent ensuite une rotation de 90°, au moyen du plateau tournant. La taille de la caisse est détectée à l'entrée du convoyeur à bande (B). Les palettes sont envoyées sur les convoyeurs à bande (E ou G) en fonction de leur taille par la mise en rotation des rouleaux dans un sens ou dans l'autre. Enfin, les palettes sont évacuées automatiquement dès qu'elles atteignent un monte-charge (F ou H).

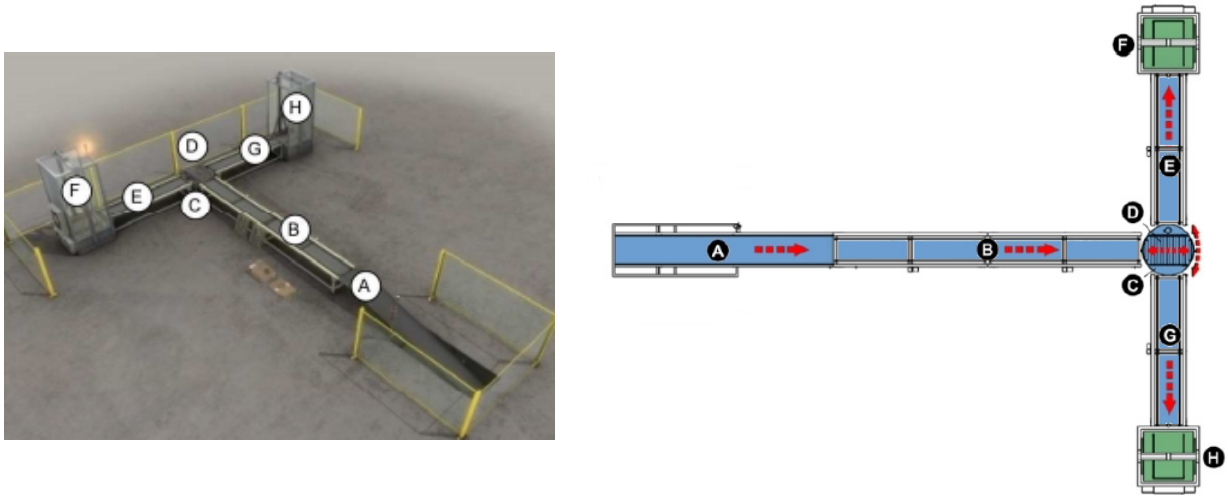


FIGURE 30 – Vue générale du système tri de caisses.

Le système dispose de 11 capteurs décrits dans le tableau 4 et de 7 actionneurs listés dans le tableau 5.

Capteur	Description
$c0$	Capteur de fin du convoyeur d'alimentation
$c1$	Capteur petite caisse
$c2$	Capteur grande caisse
$c3$	Fin du convoyeur à bande
$c4$	Plateau tournant en position chargement
$c5$	Plateau tournant en position déchargement
$c6$	Présence caisse sur le plateau tournant
$c7$	Capteur au début du convoyeur de sortie
$c8$	Capteur au début du convoyeur de sortie
$c9$	Capteur à la fin du convoyeur de sortie vers H
$c10$	Capteur à la fin du convoyeur de sortie vers F

TABLE 4 – Les capteurs du système tri de caisses issu du logiciel ITS PLC.

Actionneur	Description
$S0$	Tapis d'alimentation
$S1$	Convoyeur à bande
$S2$	Rouleaux du plateau tournant (chargement)
$S3$	Rouleaux du plateau tournant (déchargement)
$S4$	Plateau tournant
$S5$	Convoyeur de sortie vers H
$S6$	Convoyeur de sortie vers F

TABLE 5 – Les actionneurs du système tri de caisses issu du logiciel ITS PLC.

2.2.1 Base de cas normale et base de cas défaillante

L'une des étapes les plus importantes dans un système de diagnostic utilisant le RàPC est la construction de la base de cas normale et la base de cas défaillante. Dans cette partie, nous décrivons cette étape qui se base sur les mesures des capteurs et des actionneurs du système "Tri de caisses" dans différents cycles d'API. Il s'agit donc de réaliser plusieurs tests sur le système afin de récolter ces mesures pour plusieurs scénarios de fonctionnement dans une BD historique.

Plusieurs scénarios ont été produits lors de ces mesures :

- Fonctionnement normal,
- Passage inattendu de 0 à 1 du capteur c_i,
- Passage inattendu de 1 à 0 du capteur c_i,
- Capteur c_i bloqué à 1,
- Capteur c_i bloqué à 0,
- Passage inattendu de 0 à 1 de l'actionneur S_j,
- Passage inattendu de 1 à 0 de l'actionneur S_j,
- Actionneur S_j bloqué à 1,
- Actionneur S_j bloqué à 0.

avec $i = 0..10$ et $j = 0..6$.

Un regroupement de ces mesures est nécessaire pour construire les parties problèmes des cas (voir la section 2.1.1) de sorte que chaque instance représente un "cas". Ensuite, une étape de labélisation, effectuée par un expert, est essentielle afin d'étiqueter les cas normaux et les cas défaillants.

Le tableau 6 montre quelques instances de la BCN. Les instances de cette base décrivent le transport d'une petite caisse du convoyeur d'alimentation vers le convoyeur de sortie. Elles sont représentées selon 2 caractéristiques. Nous distinguons : l'"Instance" et le "Problème". Les 7 premiers bits du "Problème" représentent les mesures des 7 actionneurs (de S0 à S6) et les 11 bits suivant représentent les mesures des capteurs (de c0 à c10). Les colonnes "Etat" et "Interprétation" représentent l'étiquette établi par l'expert et son interprétation. Par exemple, l'instance 7 représente un cas normal qui décrit l'occurrence de ces événements : RS0, RS1, Rc0, Rc1, FS0, Fc0, Fc1, RS2 et Rc3. R est un front montant d'un capteur ou d'un actionneur et F est un front descendant d'un capteur ou d'un actionneur.

Le tableau 7 montre quelques instances de la BCD. Ces instances décrivent des fautes dans le système de tri de caisses. Elles sont représentées selon deux caractéristiques. Nous distinguons : l'"Instance" et le "Problème". Le "Problème" est représenté de la même manière que dans la BCN. Les colonnes "Localisation" et "Identification" représentent l'étiquette établie par l'expert en définissant le composant affecté et la faute détectée.

2.2.2 Le Bloc_EL

Dans le Bloc_EL et à chaque cycle d'API, le SADAP élabore la partie problème d'un nouveau cas non résolu *Ncas* et il commence l'étape de recherche afin d'extraire le cas

source similaire ou les cas sources les plus similaires. Le calcul de la similarité entre le nouveau cas et tous les cas sources des deux bases s'effectue suivant la formule présentée dans le tableau 3.

A titre d'exemple, le calcul de la similarité SMC entre un nouveau cas généré Ncas= "000000000001000000 100000000001000000 110000010001000000 " et l'instance 6 de la BCN (C1) , l'instance 1 de la BCD (C2) et l'instance 3 de la BCD (C3) est :

- SMC(Ncas, C1) = 0.848
- SMC(Ncas, C2) = 0.968
- SMC(Ncas, C3) = 0.88

Suite à ces calculs, l'OHS intervient pour choisir la solution la plus adéquate. Le nouveau cas héritera la solution du cas sélectionné.

2.3 Limites et critiques

Notre système proposé analyse les flux de données en ligne et rapidement par rapport à leurs fréquences d'arrivée. Il est capable de fournir l'aide et l'assistance à l'OHS dans un temps de calcul réduit. Il ne nécessite pas une connaissance complète du domaine ou des règles pour construire sa base de connaissances et il est capable d'apprendre des expériences passées. Cependant, il présente certaines limites qui sont sujet à extensions et améliorations. Nous présentons ici quelques-unes de ces limites :

- **Représentation incomplète** : Nous remarquons à travers les expérimentations effectuées sur le système "Tri de caisses" que le format de représentation de cas est incomplet. En effet, il ne permet pas de modéliser toutes les fautes existantes. Pour illustrer cette limite, regardons l'instance 8 de la BCN et l'instance 7 de la BCD présentées dans la section précédente. Nous remarquons que les deux instances sont identiques. Alors que l'instance 8 représente un comportement normal du système et l'instance 7 représente un comportement défaillant qui est le blocage de l'actionneur S4 à 0. En effet, le format de représentation de cas proposé ne permet pas de décrire cette faute. Il décrit efficacement les fonctionnements normaux du système, les fautes de type "Passage inattendu d'un capteur ou actionneur de 0 à 1" et les fautes de type "Passage inattendu d'un capteur ou actionneur de 1 à 0". Mais, il ne permet pas de couvrir toutes les fautes de type "blocage des capteurs et des actionneurs à 0 ou à 1".

- **Explosion combinatoire des descripteurs** : Le format proposé peut causer une explosion combinatoire des descripteurs si le nombre de composant du système à diagnostiquer est très grand.

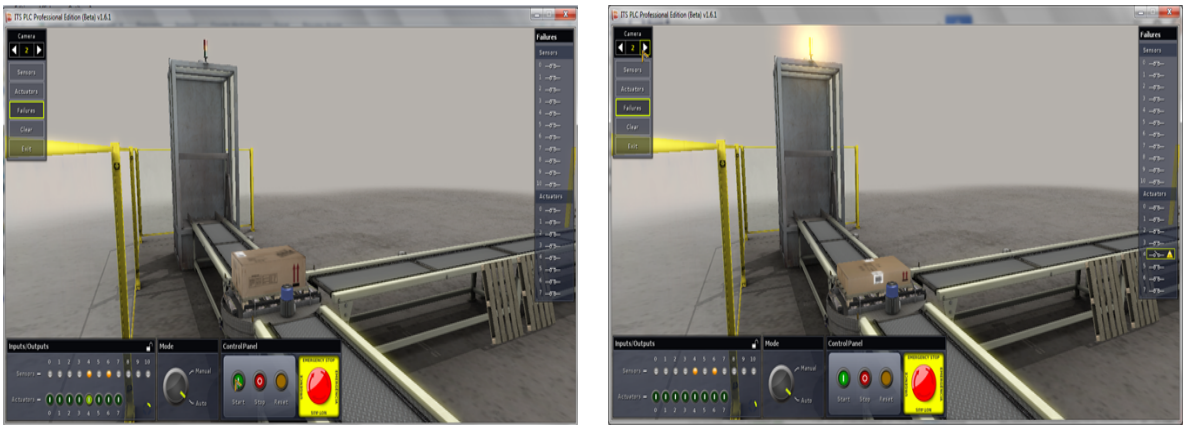


FIGURE 31 – L'instance 7 de la BCD et 8 de la BCN.

– **Lisibilité** : Le format proposé n'est pas facilement compréhensible par les utilisateurs du système.

3 Conclusion

Dans ce chapitre, nous avons proposé un système d'aide à la décision pour le diagnostic en ligne des fautes dans les SAP. Le système fournit l'aide et l'assistance aux OHS en ligne et sans avoir besoin d'une connaissance complète du domaine. Le modèle fonctionnel du système proposé comporte trois composants : les interfaces Homme-Machine, le Bloc_HL et le Bloc_EL. Le Bloc_HL permet de définir le format de représentation de cas, construire les cas de deux bases d'apprentissage et choisir la mesure de similarité. Le Bloc_EL cherche à déterminer la solution de chaque nouveau cas non résolu.

A travers ce chapitre, nous avons montré que le format de représentation de cas utilisé pour la formalisation des différentes données existantes est l'un des principaux piliers qui peut jouer sur la performance du système proposé. Un format de représentation non expressif ne permet pas de représenter tous les comportements du système à diagnostiquer et il est de plus non compréhensible et non lisible par les OHS. Ceci est dû à la non modélisation de certaines fautes (blocage des capteurs et des actionneurs à 0 ou à 1) et à la représentation binaire des signaux des capteurs et des actionneurs pour différents cycles d'API. L'une des solutions pour remédier à ce problème est de proposer un nouveau format de représentation de cas qui est assez expressif pour représenter toutes les fautes à diagnostiquer dans le SAP et il est aussi compréhensible et lisible par les OHS. Ceci sera détaillé dans le chapitre 4.

Instance	Problème	Etat	Interprétation
1	000000000001000000 100000000001000000 110000010001000000	N	$RS0, RS1, Rc0$
...			
6	000000000001000000 100000000001000000 110000010001000000 110000011001000000 010000001001000000 010000000001000000 011000000011000000	N	$RS0, RS1, Rc0, Rc1, FS0, Fc0, Fc1,$ $RS2, Rc3$
7	000000000001000000 100000000001000000 110000010001000000 110000011001000000 010000001001000000 010000000001000000 011000000011000000 011000000001000000	N	$RS0, RS1, Rc0, Rc1, FS0, Fc0, Fc1,$ $RS2, Rc3, Fc3$
8	000000000001000000 100000000001000000 110000010001000000 110000011001000000 010000001001000000 010000000001000000 011000000011000000 011000000001000000 000010000001010000	N	$RS0, RS1, Rc0, Rc1, FS0, Fc0, Fc1,$ $RS2, Rc3, Fc3, FS1, FS2, RS4, Rc6$
...			
17	010000001001000000 010000000001000000 011000000011000000 011000000001000000 000010000001010000 000010000000010000 001011000000110000 001011000000110100 001011000000100100 001011000000100000 001011000000100001 000000000000100000 000000000000000000 000000000001000000	N	$RS0, RS1, Rc0, Rc1, FS0, Fc0, Fc1,$ $RS2, Rc3, Fc3, FS1, FS2, RS4, Rc6,$ $Fc4, RS2, RS5, Rc5, Rc8, Fc6, Fc8,$ $Rc10, FS2, FS4, FS5, Fc10, Fc5, Rc4$

TABLE 6 – Base de cas normale.

Instance	Problème	Localisation	Identification
1	000000000001000000 100000000001000000 100000001101000000	Capteur $c0$	Blocage à 0 du capteur $c0$.
2	000000010001000000 100000010001000000 110000010001000000	Capteur $c0$	Blocage à 1 du capteur $c0$.
3	000000000001000000 100000000001000000 110000010001000000 110000011101000000 010000001101000000 010000000001000000 010000010001000000	Capteur $c0$	Passage inattendu de 0 à 1 du capteur c0.
4	000000000001000000 100000000001000000 110000010001000000 110000000001000000	Capteur $c0$	Passage inattendu de 1 à 0 du capteur c0.
5	000000000001000000 000000000001000000 100000000001000000 110000010001000000 110000011101000000 010000001101000000 010000000001000000 011000000011000000 011000000001000000	Capteur $c6$	Blocage à 0 du capteur $c6$.
6	000000000001000000 100000000001000000 100000000000000000	Actionneur $S4$	Blocage à 1 de l'actionneur $S4$.
7	000000000001000000 100000000001000000 110000010001000000 110000011001000000 010000001001000000 010000000001000000 011000000011000000 011000000001000000 000010000001010000	Actionneur $S4$	Blocage à 0 de l'actionneur $S4$.
8	000000000001000000 100000000001000000 100000000000000000	Actionneur $S4$	Passage inattendu de 0 à 1 de l'actionneur $S4$.
9	000000000001000000 100000000001000000 ... 000010000001010000 000010000000010000 000010000001010000	Actionneur $S4$	Passage inattendu de 1 à 0 de l'actionneur $S4$.

TABLE 7 – Base de cas défaillante.

Chapitre IV

Système de RàPC pour le diagnostic en ligne des SAP

Sommaire

Introduction

Dans le chapitre précédent, nous avons exposé une approche d'aide au diagnostic en se basant sur le RàPC. La méthode proposée s'appuie sur une phase hors ligne pour construire la base de cas initiale et sur une phase en ligne pour aider l'OHS à la prise de la décision adéquate. Nous avons conclu que le format de représentation de cas proposé et la base de cas utilisée sont deux piliers qui peuvent jouer sur la performance du système proposé. En effet, une base de cas non représentative, non compréhensible et de taille faible ne permet pas de modéliser et de reconnaitre tous les comportements défaillants du système à diagnostiquer. Aussi, les cas construits sont difficile à inspecter par les OHS.

Dans ce chapitre, nous présentons une amélioration de l'approche précédemment présentée afin de répondre à notre objectif principal qui est le diagnostic en ligne des fautes présentes dans les SAP.

Nous commençons le chapitre par décrire l'architecture du système de diagnostic proposé. Ensuite nous présentons une illustration de ses différentes étapes à travers une application pour le diagnostic du plateau tournant.

1 Architecture de l'approche proposée

La mise en œuvre de notre approche nécessite une démarche impliquant deux phases telles que la phase de construction de la base de cas initiale et la phase de raisonnement et d'apprentissage en ligne. La phase de construction de la base de cas initiale permet l'acquisition des connaissances empiriques à partir des données issues d'un SAP virtuel. Alors que la phase de raisonnement et d'apprentissage en ligne permet l'analyse en ligne des données issues d'un SAP réel afin de diagnostiquer les fautes occurrentes. Elle permet aussi la mise à jour de la base de connaissances suite l'apparition de nouveaux comportements inconnus. Notons que l'originalité de cette proposition réside dans les trois items suivants :

- Proposition d'un format de représentation de cas inspiré du formalisme **S**ignatures **T**emporelles **C**ausales (**STC**) qui s'adapte à l'aspect dynamique des systèmes à surveiller et qui est expressif, compréhensible et lisible par les OHS.
- Mise en œuvre d'une phase qui couple simulation et mise en forme de données pour la transformation des données, issues du système simulé après son émulation en mode normal et défaillant, à une base de cas.
- Présentation d'une phase de raisonnement et d'apprentissage qui permet le diagnostic en ligne du système surveillé et la mise à jour de la base de cas suite à l'apparition de nouveaux comportements inconnus.

La figure 32 illustre ces propos dont les détails sont donnés dans les sections suivantes.

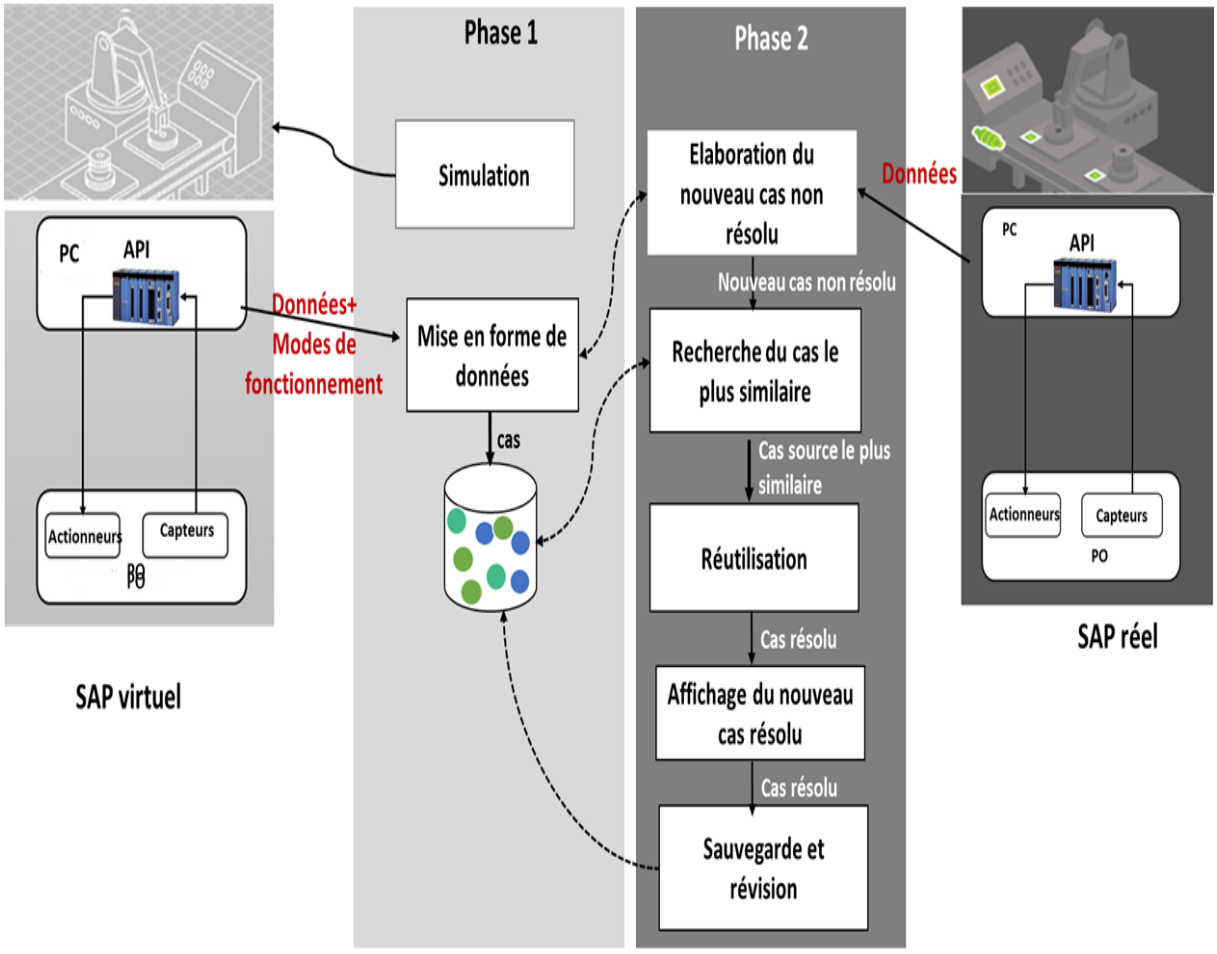


FIGURE 32 – L'architecture de l'approche proposée.

1.1 Construction de la base de cas initiale

L'objectif principal de cette phase est de construire les cas de la base de cas initiale. Afin d'atteindre cet objectif, cette phase couple un module de simulation et un module de mise en forme de données. Le module de simulation sert à décrire l'évolution du système étudié au fil du temps afin de fournir des données utiles sur son comportement dynamique dans différentes situations (y compris les situations de dysfonctionnement)[Pierreval and Ralambondrainy (1990)]. Le module de mise en forme de données permet de transformer les données fournies à des connaissances empiriques (appelées aussi des cas) qui sont exploitées par la suite par la phase de raisonnement et d'apprentissage. Ces deux modules sont détaillés dans la suite du chapitre.

1.1.1 Le module de simulation

Le module de simulation permet de fournir différentes données décrivant les comportements normaux et défaillants d'un SAP virtuel. Ces données représentent les mesures de différents capteurs et actionneurs du système virtuel. Nous avons choisi d'utiliser à ce niveau un SAP virtuel au lieu d'un SAP réel car il est possible d'y injecter différentes fautes sans avoir de dégâts matériels. Ceci veut dire que ce module n'est pas coûteux. Le principe de ce module est comme suit :

- **Étape 1** : Simuler le système virtuel en fonctionnement normal suivant plusieurs scénarios (sans injection de fautes).
- **Étape 2** : Pour chaque scénario possible, collecter les données des capteurs et des actionneurs et les mettre en forme finale (c'est-à-dire le passage des données vers les connaissances empiriques). Ce processus de mise en forme des données est détaillé dans le paragraphe suivant.
- **Étape 3** : Pour les N fautes à diagnostiquer dans le SAP, simuler le système virtuel en injectant à chaque fois une faute Fi et refaire l'étape 2.

Les scénarios de fonctionnement normal à simuler sont déduits à partir du cahier de charge du SAP alors que les scénarios de fonctionnement défaillant correspondent aux fautes que l'on souhaite diagnostiquer dans le système. Nous rappelons que nous considérons dans cette étude deux types de fautes : observables et non observables. Les fautes observables comme le passage inattendu de 0 à 1 ou de 1 à 0 d'un capteur ou d'un actionneur et les fautes non observables tels que le blocage à 1 ou à 0 d'un capteur ou d'un actionneur.

1.1.2 Le processus de mise en forme des données

Dans ce processus, les données issues du SAP virtuel sont traitées pour extraire des informations pertinentes qui sont modélisées ensuite sous forme de cas. Ces cas caractérisent et modélisent les comportements normaux et défaillants du SAP. Comme nous avons souligné auparavant, nous accordons une grande importance à l'étape de représentation de cas. De ce fait, nous proposons dans la suite un format de représentation de cas capable de caractériser et de modéliser les différents comportements des SAP et qui est compréhensible et visible par les OHS. Ce format est inspiré des Signatures Temporelles causales (STC) [Toguyeni et al. (1996); Saddem et al. (2011a, 2012); Saddem and Philippot (2014)] présentés dans le chapitre 1.

A) Pourquoi s'inspirer des STC ?

Les STC sont un formalisme de description et de reconnaissance des comportements très proche du formalisme "Chronique". Elles sont utilisées seulement pour le diagnostic

des systèmes à événements discrets alors que les chroniques sont utilisés pour la supervision et la surveillance des systèmes dynamiques [Dousson and Ghallab (1994); Micalizio et al. (2011)], la gestion des systèmes de communications mobiles [Dousson et al. (2007)], la reconnaissance des comportements dans les systèmes aérospatiale [Carle et al. (2011)] et aussi le diagnostic des fautes [Le Guillou et al. (2008); Vizcarrondo et al. (2015); Maitre et al. (2015)]. Nous rappelons qu'une STC est un sous-ensemble d'événements observables partiellement ordonnés et respectant certaines contraintes temporelles afin de caractériser un comportement défaillant. Une STC est une règle composée d'une partie condition X et d'une partie conséquence Y. La partie condition X est composée d'un ou plusieurs triplets et la partie conséquence Y décrit la faute ou les fautes résultantes suite à cette signature temporelle. Soit la STC décrite dans l'équation IV.1. Elle se compose de trois triplets.

$$(In, A, nct) * (A, B, \mathrm{ct}_1) * (B, D, ct2) => G \tag{IV.1}$$

A, B et D sont des événements ; In est un événement "*spécial*" qui est toujours occurrent, qui est introduit de manière théorique pour définir la référence temporelle des événements qui ne sont pas contraints ; $\mathrm{ct_i}$ est la i ème contrainte temporelle ; nct signifie qu'il n'y a pas de contrainte temporelle ; '*' est un opérateur de séparation entre les triplets et G est la faute qui est déduite. Le sens de l'équation (IV.1) est le suivant. Si on a l'occurrence de l'événement A, ensuite on a l' occurrence de l'événement B après A satisfaisant la contrainte $\mathrm{ct_1}$ par rapport à A et après on a l'occurrence de l'événement D après B satisfaisant la contrainte $\mathrm{ct_2}$ par rapport à B alors nous pouvons déduire la faute G.

Il est à noter que les STC sont un formalisme qui est :

(a) suffisamment expressif puisqu'il est capable de décrire formellement tous les comportements désirés du système à diagnostiquer, cela se fait grâce aux contraintes temporelles et aux opérateurs de conjonction [Saddem and Philippot (2014)].

(b) compréhensible et lisible par les opérateurs humains de supervision afin de faciliter l'inspection des règles et la découverte des corrélations et des relations entre elles.

(c) d'une grande efficacité au vu de la connaissance des symptômes des fautes à diagnostiquer.

Cependant, parmi les difficultés à surmonter pour le diagnostic à l'aide de ce formalisme ou du formalisme "chronique" se trouve l'acquisition et la mise à jour d'une base de connaissance (c'est-à-dire une base de STC ou une base de chronique) représentative et riche [Saddem and Philippot (2014); Subias et al. (2014)]. En effet, une base pauvre et non représentative peut aboutir à un mauvais système de diagnostic. Diverses techniques ont été développées durant ces dernières années pour résoudre ce problème. Elles se divisent en deux familles : à base de modèles [Guerraz and Dousson (2004); Saddem and Philippot (2014)] et à base de données [Dousson et al. (2008); Cram et al. (2012);

Subias et al. (2014)]. Les techniques à base de modèles sont des techniques de résolution de problèmes qui se basent généralement sur des modèles représentant soit le système à diagnostiquer, soit les fautes qui peuvent se présenter dans le système. Cependant, ces techniques nécessitent une connaissance suffisante du fonctionnement interne du système. Les techniques à base de données sont des techniques qui s'appuient sur les données d'historique en extrayant les caractéristiques significatives. Elles sont fondées généralement sur l'analyse des journaux d'alarmes (en anglais appelées log files of alarms) en utilisant les techniques d'exploration de données temporelles. Cependant l'inconvénient majeur de ces méthodes est qu'elles nécessitent la présence d' experts humains soit pour la qualification des chroniques [Dousson et al. (2008)], soit pour la définition des contraintes afin de guider l'algorithme de découverte des chroniques d'intérêts [Cram et al. (2012); Subias et al. (2014)].

Suite à cette étude, nous avons conclu que les STC représentent une véritable piste dont on peut s'inspirer pour représenter nos cas. Dans les paragraphes suivants nous détaillerons notre format de représentation ainsi que l'algorithme de génération automatique des cas.

B) Nouveau format de représentation

Pour pallier les limites de la représentation de cas introduite dans le chapitre 3, nous proposons d'adopter un format de représentation basé sur les objets. Ce dernier offre :

- Une facilité de maintenance et d'évolutivité puisque le format objet est facilement modifiable et il est capable de prendre en compte l'aspect évolutif des systèmes à diagnostiquer, c'est-à-dire que nous pouvons facilement modifier un des éléments d'un cas sans affecter les autres.
- Une réutilisation des éléments d'un cas à travers la relation d'héritage et le concept de polymorphisme.

Un cas est alors représenté par un objet qui possède un identifiant unique (ID) et décrit par deux attributs : *Problème* (P) et *Solution*(S). Le *Problème* P est décrit par un vecteur d'objets de type *Triplet*. Un objet *Triplet* est représenté par un "**É**vénement **R**éférent (**ER**)", un "**É**vénement **C**ontraint (**EC**)" et une "**C**ontrainte **T**emporelle de type période (**CT**)". La *Solution* S est décrite par un objet *Etat* qui détermine la nature du comportement du système à diagnostiquer. Ce comportement peut être un comportement "normal", "défaillant" ou "non identifié". L'état "non identifié" est un nouveau état qui n'est pas évoqué dans l'ancienne proposition. Il permet de garantir un bon taux de bonne détection et de diminuer le nombre de fausses alarmes. En effet, ranger un cas comme étant "non identifié" est mieux de le ranger incorrectement. Le format de représentation adopté est décrit sous la forme d'un diagramme de classe UML dans la figure 33.

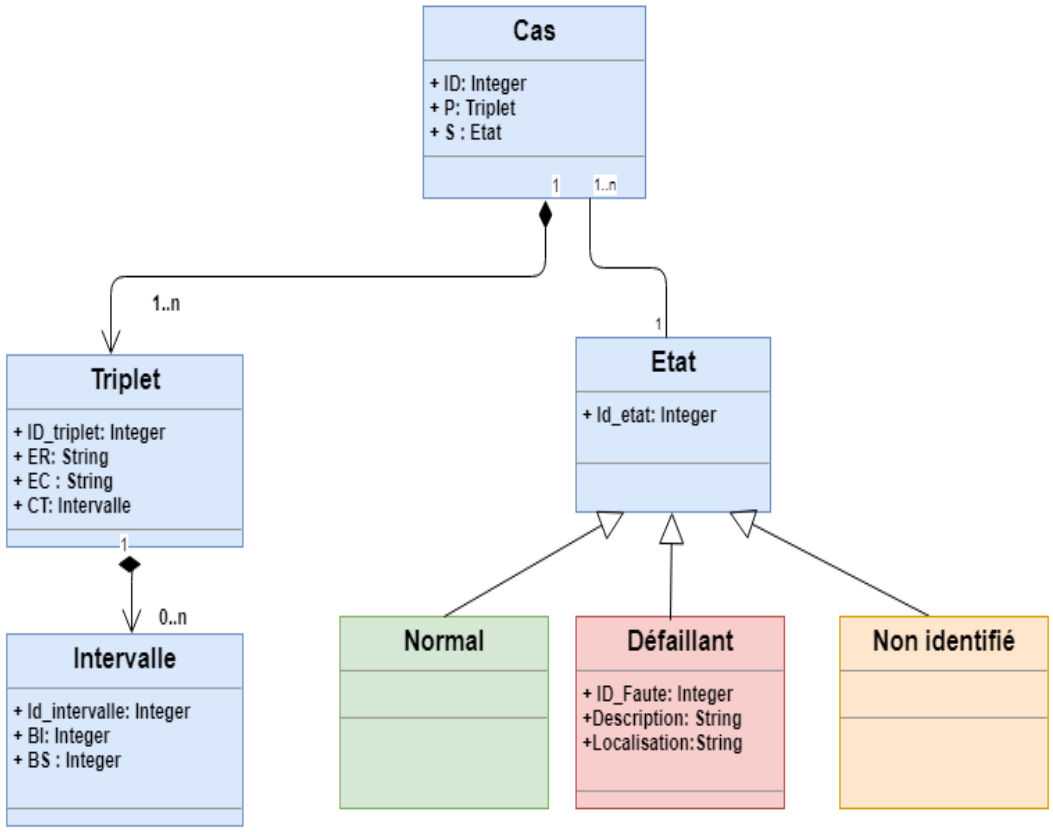


FIGURE 33 – Le format de cas proposé sous la forme d'un diagramme de classe UML.

C) Génération des parties problèmes

Selon notre représentation de cas choisie, la partie problème d'un nouveau cas est un ensemble de triplets. Chaque triplet est représenté par (ER, EC, CT). Dans cette partie, nous présentons un algorithme qui transforme les signaux des capteurs et des actionneurs calculés lors de chaque cycle d'API à un triplet et qui construit la partie problème d'un nouveau cas à partir de ce triplet et les triplets associés aux cycles d'API précédents.

Nous notons que cet algorithme est capable de décrire seulement les changements d'états des entrées/sorties qui ont une relation de causalité entre elles. Alors que les STC sont capables de décrire les changements d'états des entrées/sorties qui ont une relation de causalité entre elles et aussi les changements d'états des entrées/sorties qui ne possèdent pas nécessairement une relation de causalité entre elles.

L'algorithme proposé a comme entrées les signaux des capteurs et des actionneurs du SAP et le temps de cycle de l'API et il a comme sortie la partie problème d'un nouveau cas. Il consiste à effectuer 6 étapes qui sont illustrées à travers l'organigramme présenté dans la figure 34 et détaillées par le pseudo-code présenté dans l'annexe 1.

Les notations utilisées :

— *SignatureBinaire* : regroupe les signaux des capteurs et des actionneurs pendant un cycle API.

— *T* : cycle d'API actuel.

— *T-1* : cycle d'API précédent.

— *EC* : événement contraint.

— *ER* : événement référent.

— *CT* : contrainte temporelle.

— *BI* : borne inférieure de la CT.

— *BS* : borne supérieure de la CT.

— *TempsCyclecourant* : mesure de temps associée au cycle T.

— *TempsCycleprécédent* : mesure de temps associée au cycle T-1.

- **Étape 0** : Regrouper les signaux des capteurs et des actionneurs du SAP pendant le cycle *T* pour construire une signature binaire. Si cette signature binaire est différente de la signature binaire du cycle *T-1* alors nous passons à l'étape suivante.
- **Étape 1** : Formulation de l'événement référent (*ER*) : Si c'est le premier cycle exécuté alors l'ER est un événement "*In*" (c'est-à-dire c'est un événement non contraint) sinon l'EC du cycle *T-1* devient l'ER du cycle T.
- **Étape 2** : Formulation de l'événement contraint (*EC*) : Chaque élément de la signature binaire est transformé en un événement qui peut être soit le front montant (désigné par le préfixe "*R*") ou le front descendant (désigné par le préfixe "*F*") d'un capteur ou d'un actionneur. Cet événement est défini comme un EC.
- **Étape 3** : Formulation de la CT qui est décrite à travers deux attributs BI et BS. Si ER est égal à "*In*" alors il n'y a pas de contrainte temporelle c'est à-dire BI=0 et BS=0. Si ER est différent de "*In*" alors la CT est calculée après une estimation des erreurs aléatoires de mesure dues au matériel, aux paramètres physiques ou à l'opérateur.

  Cela conduit à exprimer premièrement le temps estimé :

$$Tempsestime = TempsCyclecourant - TempsCycleprécédent \tag{IV.2}$$

  par la suite :

$$BI = Tempsestime - \Delta T \tag{IV.3}$$

  et

$$BS = Tempsestime + \Delta T \tag{IV.4}$$

  avec ΔT représente l'incertitude sur la valeur du temps, sa valeur est >0 et elle est fixée à l'avance par un expert du domaine.

- **Étape 4** : Regroupement des 3 résultats précédents pour formuler un nouveau triplet.
- **Étape 5** : Ajout du nouveau triplet aux triplets précédents pour construire la partie problème d'un nouveau cas.

D) Étiquetage des cas

Le fonctionnement du SAP virtuel en mode normal ou en mode défaillant permet d'étiqueter automatiquement chaque cas sans avoir besoin d'un expert accompagnant le processus de formatage des données. Les opérations normales du systèmes sont représentées par un ensemble de cas étiquetés normaux tandis que les opérations défaillantes sont décrites par un ensemble de cas étiquetés défaillants. Les deux types de cas sont stockés dans la base de cas initiale.

La complexité de construction de la base de cas initiale est une complexité linéaire par rapport à la taille de la signature binaire : O (N_{cas} * Z) où N_{cas} est le nombre de cycles d'API effectués (qui est il égal aussi au nombre de cas construits) et Z est le nombre des capteurs et des actionneurs du SAP.

1.2 Phase de raisonnement et d'apprentissage en ligne

Une fois la base de cas obtenue, nous déclenchons la phase de raisonnement et d'apprentissage en ligne. Cette phase permet d'une part le diagnostic du système et d'autre part la mise à jour de la base de cas par des nouveaux comportements inconnus. Elle consiste : premièrement, à récupérer les signaux des capteurs et des actionneurs pour chaque cycle d'API ; deuxièmement, à représenter ces signaux sous la forme d'une nouvelle signature binaire (via l'algorithme décrit dans la section précédente) ; troisièmement, à transformer cette signature à un nouveau problème d'un nouveau cas appelé "*Nouveau cas non résolu*" ; quatrièmement, à rechercher le cas source similaire ou le plus similaire de ce nouveau cas ; cinquièmement, à réutiliser la solution du cas source le plus similaire et finalement à sauvegarder le nouveau cas résolu dans la base.

Dans la suite, nous allons détailler les différentes étapes : recherche, réutilisation et sauvegarde.

1.2.1 Recherche

L'étape de recherche permet d'extraire pour chaque nouveau cas, les cas similaires ou les plus similaires. Dans cette étude, cette étape est basée premièrement sur l'extraction des différents cas qui ont le même nombre de triplets que le nouveau cas et deuxièmement sur le raffinement du choix par le calcul de similarité entre le nouveau cas et le premier groupe sélectionné. Le calcul de similarité est basé sur l'utilisation d'un indice de similarité

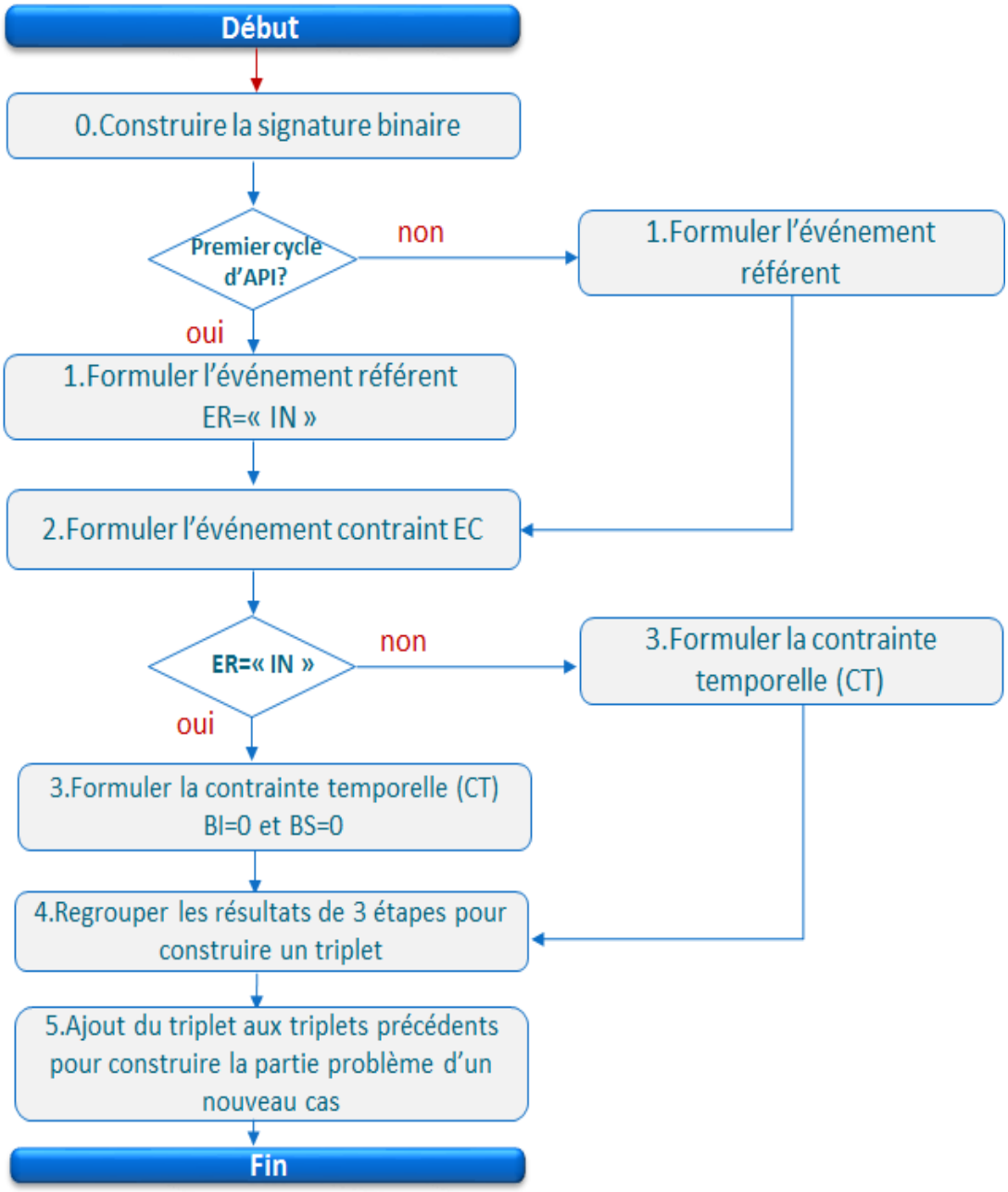


FIGURE 34 – L'organigramme de l'algorithme proposé.

qui permet de calculer le degré de ressemblance entre les cas. Elle est généralement dérivée de produits scalaires ou de mesures de dissimilarité [Bergmann and Gil (2014)].

A) Nouveau indice de dissimilarité

Dans cette partie, nous utilisons un « *calculateur de similarité* » qui est décrit dans la figure 35 afin de calculer la similarité entre deux cas. Il a comme entrée le nouveau cas non résolu et comme sortie l'indice de similarité calculé entre ce cas et chaque cas source appartenant au premier groupe sélectionné. Le calcul de l'indice de similarité entre deux

cas Cas_i et Cas_j revient à mesurer l'indice de dissimilarité noté D séparant leurs deux problèmes P_i et P_j.

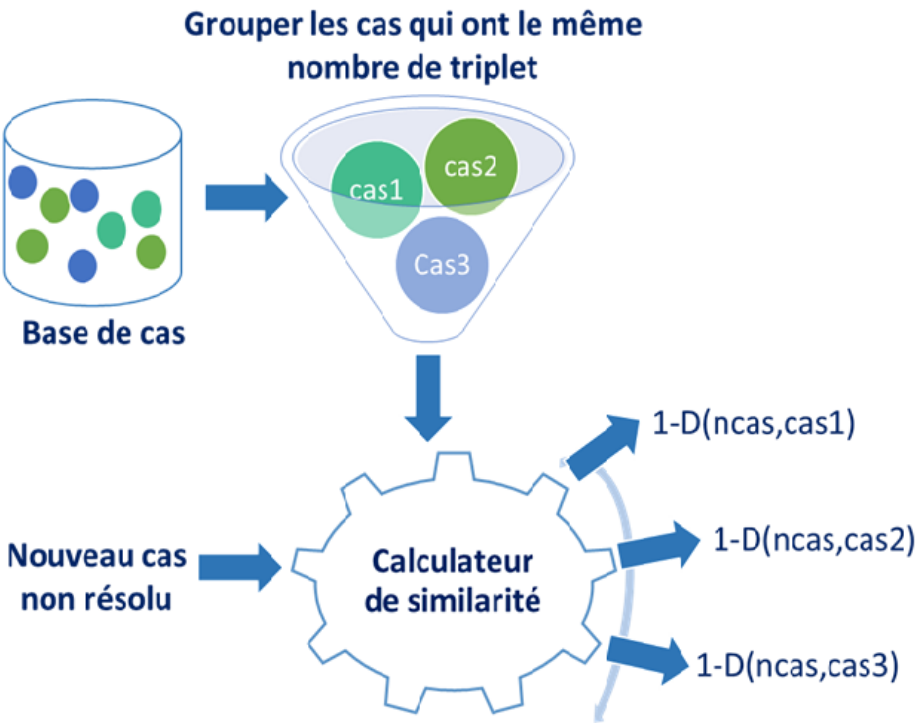


FIGURE 35 – Le calculateur de similarité.

$$S(\mathrm{Cas_i}, \mathrm{Cas_j}) = 1 - D(\mathrm{P_i}, \mathrm{P_j}) \tag{IV.5}$$

Soit D l'indice de dissimilarité défini par :

$$D : \mathrm{P_i} * \mathrm{P_j} -> [0, 1]$$

- Si $\mathrm{P_i}$ et $\mathrm{P_j}$ sont identiques et ont la même taille alors D ($\mathrm{Cas_i}$, $\mathrm{Ccas_j}$)= 0
- Si $\mathrm{P_i}$ et $\mathrm{P_j}$ n'ont pas la même taille ou aucun triplet identique alors D ($\mathrm{Cas_i}$, $\mathrm{Cas_j}$)=1

Par la suite, nous considérons les notions suivantes :

- $\mathrm{P_i}$=$\{\mathrm{Tr_{1,i}}, \mathrm{Tr_{2,i}}, \mathrm{Tr_{3,i}}, ..., \mathrm{Tr_{mt,i}}\}$ avec mt est le nombre de triplet de $\mathrm{P_i}$.
- $\mathbf{Tr_{K,i}}$=($\mathrm{ER_{K,i}}$, $\mathrm{EC_{K,i}}$, $\mathrm{CT_{K,i}}$) est le k-ème triplet de l' i-ème cas. Les événements ($\mathrm{ER_{K,i}}$ et $\mathrm{EC_{K,i}}$) sont des données qualitatives nominaux et la contrainte temporelle $\mathrm{CT_{K,i}}$ est une donnée quantitative de type intervalle.

L'indice de dissimilarité D séparant deux problèmes $\mathrm{P_i}$ et $\mathrm{P_j}$ est défini comme suit :

$$D(\mathrm{P_i}, \mathrm{P_j}) = \frac{\sum_{k=1}^{mt} DT(\mathrm{Tr_{k,i}}, \mathrm{Tr_{k,j}})}{3mt} \tag{IV.6}$$

avec mt est le nombre de triplet de deux problèmes $\mathrm{P_i}$ et $\mathrm{P_j}$, DT est l'indice de dissimilarité séparant les deux triplets $\mathrm{Tr_{K,i}}$ et $\mathrm{Tr_{K,j}}$. Il se calcule comme suit :

$$\begin{aligned} DT(\mathrm{Tr_{k,i}}, \mathrm{Tr_{k,j}}) = & DEV(\mathrm{ER_{k,i}}, \mathrm{ER_{k,j}}) \\ & + DEV(\mathrm{EC_{k,i}}, \mathrm{EC_{k,j}}) \\ & + \frac{IPOS(\mathrm{CT_{k,i}}, \mathrm{CT_{k,j}})}{\mathrm{Date_{max}}} \end{aligned} \tag{IV.7}$$

avec DEV indique l'indice de dissimilarité séparant deux événements, $IPOS$ est l'indice de positionnement entre deux contraintes temporelles et $\mathrm{Date_{max}}$ correspond à la date maximale de tous les intervalles du temps de la base de cas.

- DEV se calcule selon cette équation :

$$DEV(\mathrm{E_{k,i}}, \mathrm{E_{k,j}}) = \begin{cases} 0 & \text{Si } \mathrm{E_{k,i}} \text{ et } \mathrm{E_{k,j}} \text{ sont égaux} \\ 1 & \text{sinon.} \end{cases} \tag{IV.8}$$

- $IPOS$ se calcule comme suit :

$$IPOS([x,y],[a,b]) = \begin{cases} 0 & \text{Si } x \in [a,b] \cap [x,y] \\ MinPos([x,y],[a,b]) & \text{sinon.} \end{cases} \tag{IV.9}$$

avec :

$$MinPos([x,y],[a,b]) = \min(Pos(x,[a,b]), Pos(y,[a,b])) \tag{IV.10}$$

[a,b] est la contrainte temporelle du triplet $\mathrm{Tr_{k,i}}$, *[x,y]* est une contrainte temporelle du triplet $\mathrm{Tr_{k,j}}$ et Pos est l'indice de positionnement entre un point et un intervalle.

- Pos se calcule comme suit :

$$Pos(x,[a,b]) = \begin{cases} 0 & \text{Si } x \in [a,b] \\ Min(|x-a|, |x-b|) & \text{sinon.} \end{cases} \tag{IV.11}$$

B) Exemple récapitulatif

Supposons un nouveau problème non résolu défini par :

$$\mathrm{P_n} = (A, B, [5,10]) * (B, C, [1,5])$$

Soit quelques problèmes de la base de cas représentés dans le tableau 8 et supposons que la $\mathrm{Date_{max}} = 100$.

Le calcul manuellement de l'indice de dissimilarité séparant $\mathrm{P_n}$ et les différents instances se fait de la façon suivante :

Problèmes sources	$\mathbf{Tr_1}$			$\mathbf{Tr_2}$		
	$\mathbf{ER_1}$	$\mathbf{EC_1}$	$\mathbf{CT_1}$	$\mathbf{ER_2}$	$\mathbf{EC_2}$	$\mathbf{CT_2}$
$\mathbf{P_1}$	A	B	[5,10]	B	C	[1,5]
$\mathbf{P_2}$	A	B	[11,12]	B	C	[10,25]
$\mathbf{P_3}$	A	C	[3,4]	C	D	[5,10]
$\mathbf{P_4}$	B	D	[5,10]	D	E	[15,25]

TABLE 8 – Exemples de problèmes sources.

✓ $D(\mathrm{P_n}, \mathrm{P_1}) = (DT(\mathrm{Tr_{1,n}}, \mathrm{Tr_{1,1}}) + DT(\mathrm{Tr_{2,n}}, \mathrm{Tr_{2,1}}))/6$
$= ((DEV(A,A)+DEV(B,B)+(IPOS([5,10],[5,10])/100) +(DEV(B,B)+DEV(C,C)+$
$(IPOS([1,5],[1,5])/100))/6$
$= (0+0+(0/100)) + (0+0+(0/100))/6$
$= 0$

✓ $D(\mathrm{P_n}, \mathrm{P_2}) = (DT(\mathrm{Tr_{1,n}}, \mathrm{Tr_{1,2}}) + DT(\mathrm{Tr_{2,n}}, \mathrm{Tr_{2,2}}))/6$
$= ((DEV(A,A) + DEV(B,B) + (IPOS([5,10],[11,12])/100) + (DEV(B,B) +$
$DEV(C,C) + (IPOS([1,5],[10,25])/100))/6$
$= ((0+0+(min(Pos(5,[11,12]), Pos(10,[11,12]))/100))$
$+(0+0+(min(Pos(1,[10,25]), Pos(5,[10,25]))/100)))/6$
$= ((0+0+(min(Min(|5-11|,|5-12|), Min(|10-11|,|10-12|))/100))$
$(0+0+(min(Min(|1-10|,|1-25|), Min(|5-10|,|5-25|))/100))/6$
$= (0+0+(1/100)) + (0+0+(5/100))/6$
$= 0,01$

✓ $D(\mathrm{P_n}, \mathrm{P_3}) = (DT(\mathrm{Tr_{1,n}}, \mathrm{Tr_{1,3}}) + DT(\mathrm{Tr_{2,n}}, \mathrm{Tr_{2,3}}))/6$
$= ((DEV(A,A)+DEV(B,C)+(IPOS([5,10],[3,4])/100)+(DEV(B,C)+DEV(C,D)+$
$(IPOS([1,5],[5,10])/100))/6$
$= ((0+1+(min(Pos(5,[3,4]), Pos(10,[3,4]))/100)) + (1+1+0))/6$
$= ((0+1+(min(Min(|5-3|,|5-4|), Min(|10-3|,|10-4|))/100)) + 2)/6$
$= ((0+1+(1/100)) + 2)/6$
$= 0,501$

✓ $D(\mathrm{P_n}, \mathrm{P_4}) = (DT(\mathrm{Tr_{1,n}}, \mathrm{Tr_{1,4}}) + DT(\mathrm{Tr_{2,n}}, \mathrm{Tr_{2,4}}))/6$
$= ((DEV(A,B) + DEV(B,D) + (IPOS([5,10],[5,10])/100)$
$+(DEV(B,D) + DEV(C,E) + (IPOS([1,5],[15,25])/100))/6$
$= ((1+1+(0/100)) + (1+1+(min(Min(|1-15|,|1-25|), Min(|5-15|,|5-$
$25|))/100)))/6$
$= (2+(2+(10/100)))/6$

$= 0,683$

1.2.2 Réutilisation

L'étape de réutilisation permet de transformer la solution du cas sélectionnée à une solution adéquate au nouveau cas. Cette étape peut être effectuée manuellement par un expert ou automatiquement par copie de la solution sélectionnée ou à l'aide d'un algorithme, de formules ou de règles [Johansson and Cederfeldt (2012); Henriet et al. (2014); Reyes et al. (2015)].

Dans notre étude, nous avons utilisé une adaptation par copie où le nouveau cas héritera la solution du cas source le plus similaire. Le cas source le plus similaire doit avoir un indice de similarité maximal qui doit être supérieur à certain "*seuil de décision*". Si cet indice est inférieur ou égal à ce "*seuil de décision*" alors le cas est rangé comme un cas "*non identifié*". Le "*seuil de décision*" doit être fixé à l'avance par le concepteur du système. Nous montrons dans le chapitre suivant comment nous pouvons le choisir expérimentalement.

1.2.3 Sauvegarde et révision

Une fois la solution trouvée, elle est retenue dans la base de cas et elle est utilisée immédiatement pour la résolution de nouveaux problèmes de reconnaissance. L'ajout du nouveau cas à la base favorise un apprentissage incrémental et permet d'enrichir la base de cas par des nouveaux comportements inconnus. Afin de réduire le nombre de cas non identifié, une étape de maintenance est nécessaire afin de supprimer les cas inutiles ou redondants. Cette étape est réalisée par un expert humain qui intervient pour examiner le contenu de la base afin de garantir que les performances du système restent stables. L'intervention de l'expert se fait lorsque le nombre de cas non identifié dépasse un nombre "nid_{max}" à fixer par les utilisateurs du système.

2 Exemple illustratif

Reprenons l'exemple du système de tri de caisses présenté dans le chapitre 3. Ce système est composé de *11* capteurs et de *7* actionneurs. Chaque capteur ou actionneur peut avoir *4* types de fautes : (a) blocage du capteur ou de l'actionneur à 0, (b) blocage du capteur ou de l'actionneur à 1, (c) le passage inattendu du capteur ou de l'actionneur de 0 à 1 et (d) le passage inattendu du capteur ou de l'actionneur de 1 à 0. Bien que ce système ne soit pas constitué d'un grand nombre d'entrées/sorties (*18 E/S*), nous devons

diagnostiquer au total *72* fautes ce qui est un nombre relativement élevé. Par conséquent, afin de simplifier notre étude, nous la limitons à un seul composant du système de tri de caisses. Nous avons choisi d'étudier le « plateau tournant » puisqu'il a été utilisé comme sous-système de référence dans d'autres approches de diagnostic [Philippot et al. (2012); Saddem and Philippot (2014)]. Nous aurions pu choisir un autre composant du système, la procédure aurait été identique. A travers cette section, nous illustrons les différentes étapes de l'architecture du système proposé pour le diagnostic du plateau tournant.

2.1 Description du plateau tournant

Le plateau tournant qui est présenté dans la figure 36a est un actionneur monostable noté $S4$. Il possède deux capteurs $C4$ et $C5$. Le capteur $C4$ est le détecteur de la position de chargement et le capteur $C5$ est le détecteur de la position de déchargement. La figure 36b expose un automate avec 6 états et 10 transitions qui représente le modèle de comportement normal du plateau tournant. Le comportement normal du plateau peut être décrit à travers deux scénarios possibles :

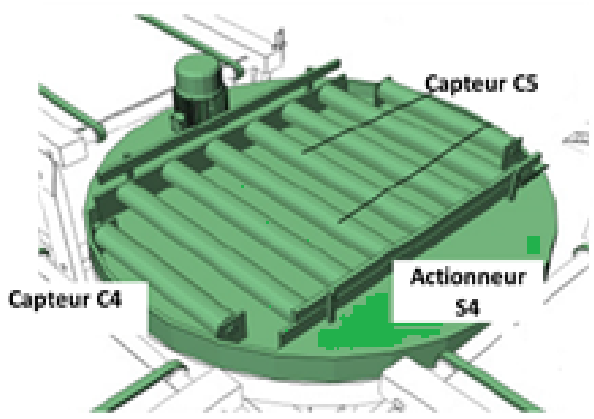
- **Scénario 1** :
  À l'état initial "0", le plateau tournant est à la position de chargement ($C4$=1). Si l'actionneur $S4$ est activé alors le plateau tournant est en mouvement et le capteur $C4$ est désactivé (passage de l'état "1" à l'état "2"). De là, si la commande est toujours active, le plateau tournant revient à la position de déchargement (transition de l'état "2" à "3"). La désactivation de l' actionneur $S4$ permet de revenir à la position initiale (l'état "4" après l'état "5" puis l'état "0").

- **Scénario 2** :
  De l'état "2", pendant le mouvement, si le contrôleur envoie la désactivation de $S4$ alors le plateau revient directement à l'état "0".

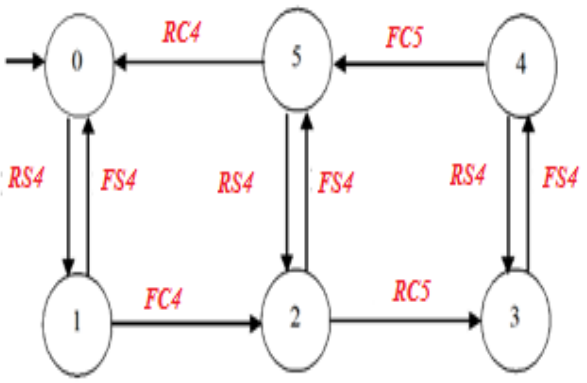
Un travail d'expertise a permis d'obtenir une liste des fautes internes qui peuvent se produire dans le plateau tournant. Cette liste est décrite dans le tableau 9 à travers deux colonnes : l'étiquette et la description de la faute.

2.2 Illustration des différentes étapes du système proposé pour le diagnostic du plateau tournant

Pour le diagnostic des fautes du plateau tournant, il est essentiel de construire la base de cas initiale. Pour se faire plusieurs cas de fonctionnement ont été produits sur le plateau tournant c'est à dire des cas de fonctionnement normal (sans injection des fautes)



(a) Le plateau tourant.



(b) Le modèle du plateau tournant : **F** : Front descendant et **R** : Front montant.

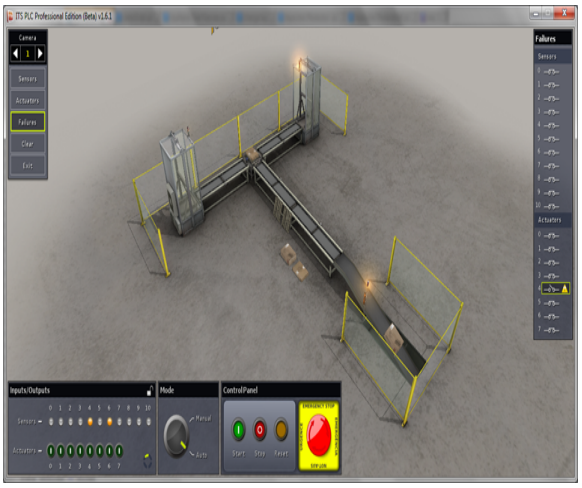
FIGURE 36 – Le plateau tournant et son modèle.

Étiquette	Description
F1	Blocage du capteur C4 à 0
F2	Blocage du capteur C4 à 1
F3	Blocage du capteur C5 à 0
F4	Blocage du capteur C5 à 1
F5	Blocage de l'actionneur S4 à 0
F6	Blocage de l'actionneur S4 à 1
F7	Passage inattendu du capteur C4 de 0 à 1
F8	Passage inattendu du capteur C4 de 1 à 0
F9	Passage inattendu du capteur C5 de 0 à 1
F10	Passage inattendu du capteur C5 de 1 à 0
F11	Passage inattendu du capteur S4 de 0 à 1
F12	Passage inattendu du capteur S4 de 1 à 0

TABLE 9 – Les fautes du plateau tournant.

et des cas de fonctionnement défaillant (avec injection des fautes). La figure 37 présente quelques exemples des fautes injectées dans le plateau tournant. La figure 37(a) met en évidence la simulation de la faute F5 qui représente le blocage de l'actionneur S4 à 0. La figure 37(b) montre le cas de blocage de l'actionneur S4 à 1. Quant à la figure 37(c), elle

met en évidence le blocage du capteur C5 à 1. Finalement un blocage du capteur C5 à 0 est illustré dans la figure 37(d). Ces quatre types de blocages entrainent une interruption de transfert des caisses.



(a) Blocage du capteur S4 à 0..

(b) Blocage de l'actionneur S4 à 1

(c) Blocage du capteur C5 à 1.

(d) Blocage du capteur C5 à 0.

FIGURE 37 – La simulation des fautes F5, F6, F3 et F4.

Chaque cas de fonctionnement qui s'est produit est représenté par un ou plusieurs cas. En effet, l'algorithme présenté dans la sous section 1.1.2 permet de transformer les signaux binaires des capteurs et des actionneurs à des nouveaux cas non résolus. L'étiquetage de ces nouveaux cas se fait selon le mode de fonctionnement du plateau tournant. Si le plateau tournant est dans l'un des scénarios normaux décrits en hauts alors le ou les nouveaux cas sont étiquetés normaux, sinon ils sont labellisés par l'étiquette de la faute injectée.

Nous supposons que notre base de cas initiale est composé de 2 cas normaux et 12 cas défaillants. Les cas défaillants sont étiquetés par les différentes fautes présentées précé-

demment. La base est présentée dans le tableau 10. Comme nous avons mentionné dans la section 1.1.2, chaque cas est caractérisé selon deux attributs. Nous distinguons : le *Problème* et la *Solution*. L'attribut *Problème* est caractérisé selon m triplets. Chaque *Triplet* est représenté par 3 attributs. Nous distinguons : l'événement référent (ER), l'événement contraint (EC) et la contrainte temporelle (CT). La contrainte temporelle est décrite par deux attributs : borne inférieur (BI) et borne supérieur (BS). L'attribut *Solution* représente l'étiquette établi automatiquement par le système proposé. Nous rappelons que si le SAP est dans un mode de fonctionnement normal alors cette étiquette est égale à "N" sinon, elle est égale à l'étiquette de la faute injectée dans le système.

L'instance 1 du tableau représente un cas de fonctionnement normal qui décrit le scénario de fonctionnement 1 (introduit ci-dessus). Le sens de l'instance est le suivant. Si nous avons : (**t1**) premièrement, l'occurrence de l'événement « front montant de l'actionneur S4 noté $RS4$ » ; (**t2** deuxièmement, l'occurrence de l'événement « front descendant du capteur C4 noté $FC4$ » entre 80 et 85 suite à l'événement $RS4$; (**t3**) troisièmement, l'occurrence de l'événement « front montant du capteur C5 noté $RC5$ » entre 2816 et 2821 après l'événement $FC4$; (**t4**) quatrièmement, l'occurrence de l'événement « front descendant de l'actionneur $S4$ noté $FS4$ » entre 6480 et 6485 après l'événement $RC5$; (**t5**) cinquièmement, l'occurrence de l'évènement « front descendant de l'actionneur $C5$ noté *FC5* » entre 80 et 85 après l'événement $FS4$ et finalement, (**t6**) l'occurrence de l'événement « front montant du capteur $C4$ noté $RC4$ » entre 2816 et 2821 après $FC5$ alors nous pouvons conclure que ce cas représente un fonctionnement normal du SAP (Solution ='N').

L'instance 3 présente un cas de fonctionnement défaillant qui décrit les symptômes de la faute $F6$. Cette faute correspond à un blocage à 1 de l'actionneur S4. Le sens de l'instance est le suivant. Si nous avons : (**t1**) l'occurrence de l'événement « front descendant du capteur $C4$ noté $FC4$ », ensuite (**t2**) l'occurrence de l'événement « front montant du capteur C5 noté $RC5$ » entre 2816 et 2821 après l'événement $FC4$ alors nous pouvons déduire la faute *F6* au sein du SAP.

Après la phase de simulation et la mise en forme des données vient la phase de raisonnement et d'apprentissage en ligne. Cette phase se base sur les connaissances empiriques stockées dans la base d'apprentissage. Le diagnostic en ligne consiste à transformer les signaux des capteurs et des actionneurs durant les différents cycles d'API à des nouveaux problèmes (des nouveaux cas non résolus). Chaque nouveau problème est résolu à travers l'étape de recherche et l'étape de réutilisation. L'étape de recherche consiste à sélectionner les différents cas de la base qui ont le même nombre de triplets que ce nouveau cas et ensuite à calculer l'indice de dissimilarité séparant ce cas des anciens cas du groupe sélectionné. Le calcul de l'indice de dissimilarité se fait suivant la formule proposée dans 1.2.1. L'étape de réutilisation permet de prédire l'étiquette de ce nouveau cas (normal,

ID	P																		S
	t1			t2			t3			t4			t5			t6			
	ER1	EVC1	CT1	ER2	EVC2	CT2	ER3	EVC3	CT3	ER4	EVC4	CT4	ER5	EVC5	CT5	ER6	EVC6	CT6	
1	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC5	[2816, 2821]	RC5	FS4	[6480, 6485]	FS4	FC5	[80, 85]	FC5	RC4	[2816, 2821]	N
2	RC4	RS4	[544, 549]	RS4	FC4	[80,85]	FC4	RC5	[2816, 2821]	RC5	FS4	[6336, 6341]	FS4	FC5	[80,85]	FC5	RC4	[2816, 2821]	N
3	IN	FC4	[0,0]	FC4	RC5	[2816, 2821]													F6
4	IN	RS4	[0,0]	RS4	FC4	[80, 85]													F3
5	IN	RS4	[0,0]	RS4	RC5	[2896, 2801]	RC5	FS4	[6864, 6869]	FS4	FC5	[80, 85]							F2
6	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC5	[2816, 2821]	RC5	FS4	[2848, 2853]							F11
7	IN	RC5	[0,0]	RC5	RS4	[22432, 22437]	RS4	FC4	[80, 85]										F4

8	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC4	[2800, 2805]								F12
9	IN	RS4	[0,0]														F5
10	IN	FC4	[0,0]														F1
11	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC5	[2736, 2741]	RC5	FS4	[6944, 6949]	FS4	RC4	[2896, 2901]		F9
12	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC5	[2816, 2821]	RC5	FS4	[6656, 6661]	FS4	FC5	[80, 85]		F8
13	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC4	[1120, 1125]	RC4	RC5	[1696, 1701]	RC5	FS4	[6880, 6885]	FS4	FC5	[80, 85]	F7
14	IN	RS4	[0,0]	RS4	FC4	[80, 85]	FC4	RC5	[2816, 2821]	RC5	FC5	[736, 741]	FC5	RS4	[6128, 6133]	FS4	RC4	[2896, 2901]	F10

TABLE 10 – Les instances de la base d'apprentissage.

défaillant ou non identifié).

Par exemple, soit un nouveau cas non résolu défini comme suit :

$$(IN, FC4, [0, 0]) * (FC4, RC5, [2900, 2905])$$

Le calcul de l'indice de dissimilarité entre ce nouveau cas et le groupe des cas possédant 2 triplets est présenté dans le tableau 11.

ID	P	S	indice de dissimilarité proposé	Similarité
3	(IN,FC4,[0,0])*(FC4,RC5,[2816,2821])	F6	0.123	0.877
4	(IN, RS4,[0, 0])*(RS4,FC4,[80,85])	F3	0.546	0.454

TABLE 11 – Calcul de l'indice de dissimilarité entre le nouveau cas non résolu et le groupe sélectionné.

Le nouveau cas héritera la solution du cas numéro 3 puisqu'il a un indice de similarité maximal qui est supérieur au *seuil de décision*. Ce nouveau cas résolu est stocké dans la base de cas et il est utilisé immédiatement pour la résolution des problèmes futurs.

3 Conclusion

Dans ce chapitre, une extension de l'approche proposée dans le chapitre 3 a été présentée. Cette extension permet premièrement de générer automatiquement un ensemble de connaissances empiriques stockées dans une base de cas. Deuxièmement, à partir de cette base et d'une phase de raisonnement et d'apprentissage en ligne, il est ainsi possible de diagnostiquer en ligne le système surveillé, d'apprendre de nouvelles connaissances et de mettre à jour la base de cas. Après avoir expliqué le principe de l'approche proposée, il convient maintenant de mesurer expérimentalement la performance de notre approche et de comparer ses résultats avec d'autres approches basées sur les modèles ou sur les techniques d'apprentissage automatique. Ceci sera détaillé dans le chapitre 5.

Chapitre V

Expérimentations et résultats

Sommaire

Introduction

Dans le chapitre précédent, nous avons présenté un système de diagnostic en ligne des fautes possibles dans les SAP. Le système s'appuie sur une méthode d'apprentissage automatique paresseuse qui est le « RàPC ». Contrairement au système présenté dans le chapitre 3, ce système prend la décision de diagnostic d'une manière autonome. Il propose un format de représentation de cas expressif et facile à inspecter par les opérateurs humains de surveillance. Pour reconnaitre le comportement d'un nouveau cas (c'est à dire une nouvelle observation), le système se base sur le comportement du cas source le plus similaire en utilisant un nouveau indice de dissimilarité. Ce nouveau indice permet de quantifier la différence entre deux cas qui sont représentés par des données catégoriques appelées aussi qualitatives et des données numériques appelées aussi quantitatives. Si l'indice de similarité séparant le nouveau cas et le cas le plus proche est supérieur à un certain "*seuil de décision*" alors le nouveau cas hérite la solution de ce cas. Sinon, il est rangé comme un cas non identifié. Nous avons ainsi illustré ces différentes étapes pour le diagnostic en ligne du plateau tournant.

Dans ce dernier chapitre, nous nous intéressons à mesurer la performance du système proposé pour le diagnostic du plateau tournant présenté dans le chapitre 4. Nous comparons les résultats de notre proposition avec certaines méthodes d'apprentissage automatique et avec une méthode de diagnostic à base de modèles.

1 Démarche expérimentale

Pour mesurer la performance du système de diagnostic proposé, nous reprenons l'exemple du plateau tournant. A partir des séries de mesures effectuées sur ce composant pour différents scénarios de fonctionnement, une base de cas contenant 348 cas a été construite. Les cas de cette base sont répartis sur les différentes classes comme suit (voir la figure 38) :

- **176** cas décrivant des fonctionnements normaux du plateau tournant. Ces cas sont étiquetés (**N**) ;
- **172** cas décrivant différentes fautes injectées sur le plateau tournant. Elles sont réparties sur les différentes classes comme suit :
  - **11** cas représentent des fautes de type "blocage du capteur C4 à 1" labellisés **F2** ;
  - **28** cas représentent des fautes de type "blocage du capteur C5 à 0" étiquetés **F3** ;

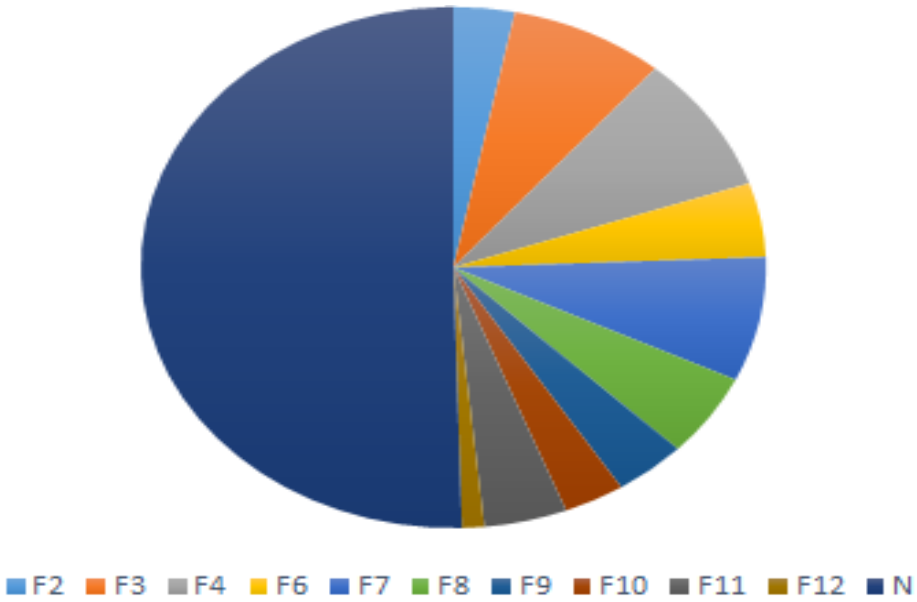


FIGURE 38 – La répartition des cas sur les différentes classes.

- **30** cas représentent des fautes de type "blocage du capteur C5 à 1" étiquetés **F4** ;
- **16** cas représentent des fautes de type "blocage de l'actionneur S4 à 1" étiquetés **F6** ;
- **26** cas représentent des fautes de type "passage inattendu du capteur C4 de 0 à 1" étiquetés **F7** ;
- **18** cas représentent des fautes de type "passage inattendu du capteur C4 de 1 à 0" étiquetés **F8** ;
- **13** cas représentant des fautes de type "passage inattendu du capteur C5 de 0 à 1" étiquetés **F9** ;
- **11** cas représentent des fautes de type "passage inattendu du capteur C5 de 1 à 0" étiquetés **F10** ;
- **15** cas représentent des fautes de type "passage inattendu du capteur S4 de 0 à 1" étiquetés **F11** ;
- **4** cas représentent des fautes de type "passage inattendu du capteur S4 de 1 à 0" étiquetés **F12**.

Les fautes de type "blocage du capteur C4 à 0" qui sont étiquetées $F1$ et de type "blocage de l'actionneur S4 à 0" qui sont étiquetées $F5$ ne sont pas représentées dans la base de cas. En effet, à chaque simulation de ces fautes, les cas construits restent les mêmes. Ils ne changent pas de valeurs. Le cas qui décrit la faute $F1$ est représenté par le triplet (IN, $FC4$, [0, 0]) et le cas qui représente la faute $F5$ est décrit par le triplet (IN,

$RS4$, [0, 0]). Pour cette raison, nous ne pouvons pas avoir plusieurs instances de ces fautes.

Afin de diagnostiquer le plateau tournant, le système de diagnostic proposé peut être considéré comme un classifieur multi-classes qui cherche à ranger correctement chaque nouveau cas dans l'une de ces classes prédéfinies. Si le nouveau cas ne correspond à aucune de ces classes alors il est rangé comme un cas "non identifié". Avant d'entamer les expérimentations et la présentation des résultats, nous décrivons dans cette section les moyens techniques et les mesures utilisées pour calculer la performance de notre système de diagnostic.

1.1 Validation croisée stratifiée

Pour évaluer un classifieur, une solution consiste à l'entrainer sur la base de données initiale (appelée aussi la base d'apprentissage) et de le tester sur la même base. L'évaluation se fait en mesurant "*l'erreur d'apprentissage*" qui calcule le nombre de fois où le classifieur a mal classé les instances de la base d'apprentissage. Mais cela nous conduit à un problème de "*surapprentissage*" appelé aussi "*sur-ajustement*" ou "*surinterprétation*" (en anglais appelée "*overfitting*" [Japkowicz and Stephen (2002); Zhang et al. (2018)]). Il se traduit par le fait que cette erreur est biaisée car le classifieur apprend par cœur cette base de données. Par contre, on ne sait pas s'il est capable d'avoir ces mêmes capacités sur des nouvelles données.

Une solution qui permet de déterminer le comportement du classifieur avec des nouvelles données consiste à partager la base de données en deux sous ensembles : le jeu d'entrainement (appelé aussi jeu d'apprentissage) et le jeu de test. Le classifieur est entrainé sur le jeu d'apprentissage et il est testé et évalué sur le jeu de test. Plusieurs portions peuvent être choisies lors de cette division mais généralement 80% des données de la base sont utilisées pour l'entrainement et 20% des données de la base sont utilisées pour le test [Géron (2017)]. L'évaluation du classifieur se fait à travers le calcul du taux d'erreur sur les nouvelles données. Ce "*taux d'erreur*" appelé aussi "*erreur de généralisation*" ou "*erreur de classification*" permet de nous informer sur le comportement du classifieur sur les instances du jeu de test. Le fait de mesurer ce taux d'erreur à de multiples reprises sur le jeu de test, le classifieur et ses hyperparamètres s'adaptent pour obtenir le meilleur modèle pour ces données. Il aura un taux d'erreur très satisfaisant. Ce qui nous mène une fois encore à un problème de "*surapprentissage*".

Pour résoudre ce problème, la base de données est partagée en trois sous ensembles : le jeu d'entrainement, le jeu de validation et le jeu de test. Le classifieur est entrainé sur le jeu d'apprentissage. L'évaluation initiale est effectuée sur le jeu de validation et lorsque

l'expérience semble réussie, l'évaluation finale peut être effectuée sur le jeu de test. Cependant, en partageant la base de données en trois ensembles, nous réduisons nécessairement le nombre d'instances pouvant être utilisées pour l'entrainement du classifieur. Dans ce cas, on peut avoir un classifieur peu performant pour le diagnostic surtout si notre base de cas initiale n'est pas de grande taille.

Une solution pour ce problème est une technique appelée "Validation Croisée" (en anglais appelée "*Cross Validation*" [Arlot et al. (2010); Pedregosa et al. (2011)]). C'est une technique qui utilise la totalité de la base de données pour l'apprentissage et la validation. Le principe de cette technique est le suivant : partager la base de données en k *parties* (en anglais appelé "*folds*") à peu prés égales. Chacune des K parties est utilisée comme un *jeu de test* et le reste des parties (*k-1*) sont utilisées pour *l'entrainement* du classifieur. A chaque "*fold*" test, l'erreur du modèle est calculée en comparant les étiquettes prédites aux étiquettes réelles des instances du "*fold*" test. L'erreur moyenne pour les k "*folds*" est calculée à partir de ces erreurs. La figure 39 montre un exemple d'une validation croisée de 4 folds. Les instances présentées en rouge sont les instances du "*fold*" test et celles qui sont en vert sont les instances pour l'entrainement du classifieur.

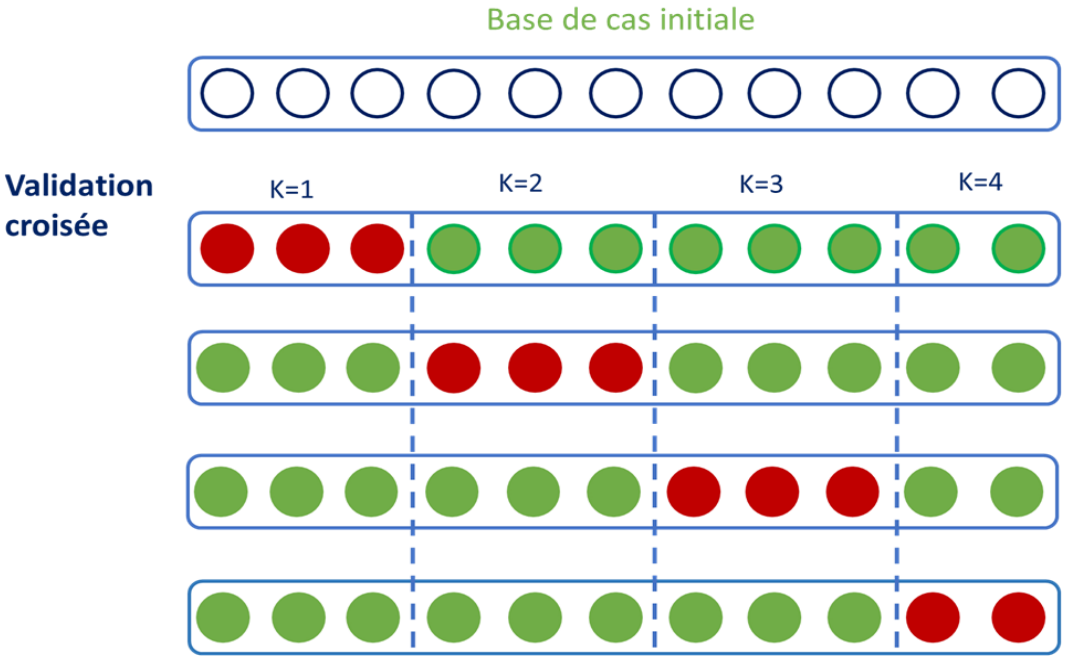


FIGURE 39 – Validaton croisée de 4 "*folds*".

La validation croisée nous permet d'utiliser l'intégralité des données pour l'entrainement du classifieur et le test du modèle. Cependant, pour un problème de classification cette technique peut nous mener à des situations de divisions où le jeu de test contient des instances étiquetées normales (N) et le jeu d'entrainement ne contient que des instances étiquetées défaillantes (D) (comme dans l'exemple de la validation croisée présentée dans

la figure 40).

La figure 40 montre un exemple d'application d'une validation croisée de 4 folds et d'une validation croisée stratifiée de 4 "*folds*". La base de cas initiale présentée se compose de 4 instances étiquetées défaillantes (D) et 7 instances labellisées normales (N). La division en 4 "*folds*" de cette base selon une validation croisée permet à titre d'exemple d'avoir un "*fold*" contenant 3 instances défaillantes alors que les autres "*folds*" contiennent des instances normales. Dans ce cas, le classificateur apprend un modèle pour classer les instances normales alors qu'il va être tester pour classer des instances défaillantes. Cela va sûrement fausser les résultats d'évaluation [Rao et al. (2008); Géron (2017)]. Pour résoudre ce problème, nous utilisons une "*validation croisée stratifiée*"[Zhang et al. (2016); Géron (2017)] qui permet de garantir que chaque "*fold*" contient les mêmes portions d'instances de chaque classe que la base de données. Elle permet de résoudre ce problème en garantissant que chaque "*fold*" contient au moins une instances défaillante (D).

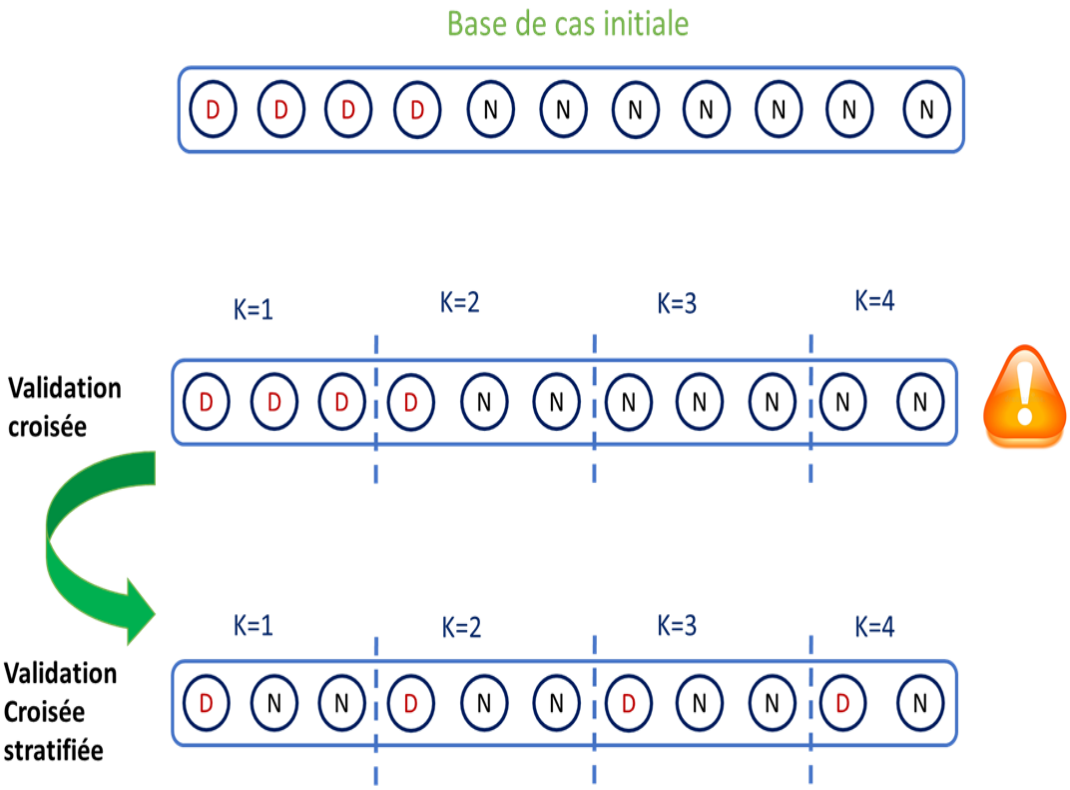


FIGURE 40 – Validaton croisée stratifiée de 4 "*folds*".

1.2 Les indicateurs de performance

Les indicateurs qui permettent de mesurer la qualité d'un classifieur peuvent être calculés à partir d'une matrice de confusion. Soit M un modèle généré par le classifieur et

t un nouveau jeu de test. La matrice de confusion MC présente dans ses lignes les valeurs *réelles* de l'attribut cible pour toutes les instances du jeu de test (i.e. l'attribut cible est la classe du cas) et dans ses colonnes les valeurs *prédites* de ce dernier. Le principe de la matrice de confusion est de compter le nombre de fois où des cas de la classe A ont été rangées dans la classe B. La matrice de confusion peut être représentée à travers quatre types de réponses :

- **V**rais **P**ositifs (**VP**) (en anglais appelés "*True Positives*") : représentent les éléments étiquetés positifs qui sont correctement prédits.
- **V**rais **N**égatifs (**VN**) (en anglais appelés "*True Negatives*") : représentent les éléments étiquetés négatifs qui sont correctement prédits.
- **F**aux **P**ositifs (**FP**) (en anglais appelés "*False Positives*") : représentent les éléments étiquetés négatifs qui sont mal prédits.
- **F**aux **N**égatifs (**FN**) (en anglais appelés "*False Negatives*") : représentent les éléments étiquetés positifs qui sont mal prédits.

Dans le cas d'un classifieur binaire c'est à dire un classifieur qui prédit seulement deux classes : la première contient des données étiquetées positifs et la deuxième contient des données étiquetées négatifs, on obtient cette matrice de confusion présentée dans la figure 41.

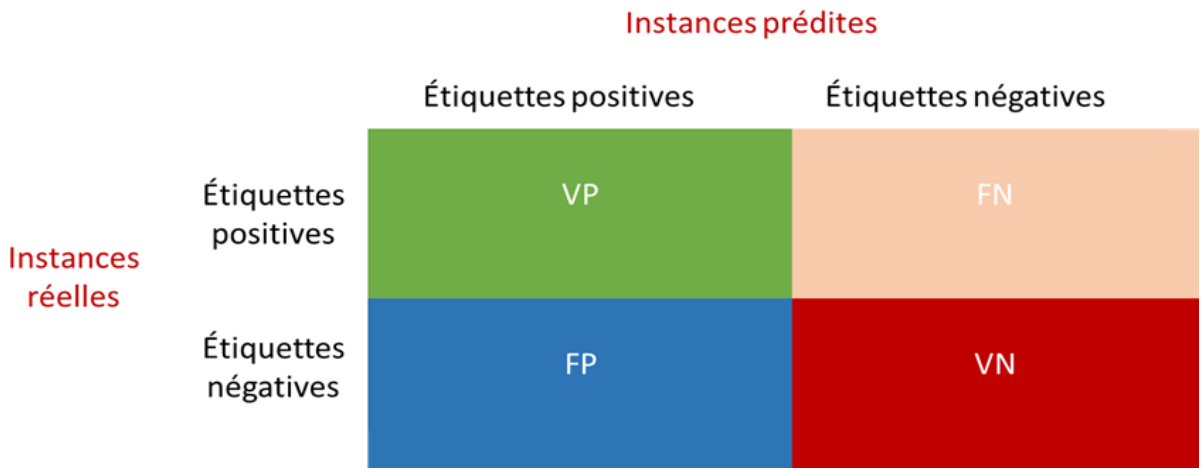


FIGURE 41 – Matrice de confusion pour un calssifieur binaire.

Un classifieur parfait aurait une matrice de confusion diagonale c'est à dire qu'il n'aurait que des VP et des VN.

Une matrice de confusion multi-classes permet d'évaluer plus que deux classes. L'évaluation de chaque classe se fait en indiquant le nombre de prédictions correctes et incorrectes. Les métriques utilisées pour une classification multi-classes sont les mêmes que

dans une classification binaire. En effet, pour chaque classe (Ci), il suffit de regrouper toutes les autres classes dans une seconde classe (Cj) et de représenter les informations VP_{Ci} , FN_{Ci}, FP_{Ci} et le VN_{Ci} pour la première classe par rapport à Cj. Pour notre système de diagnostic qui permet de distinguer 11 classes décrites précédemment, une illustration de VP_{N}, FN_{N}, FP_{N} et VN_{N} pour la classe N est présentée dans la figure 42 et pour la classe *F2* est présentée dans la figure 43.

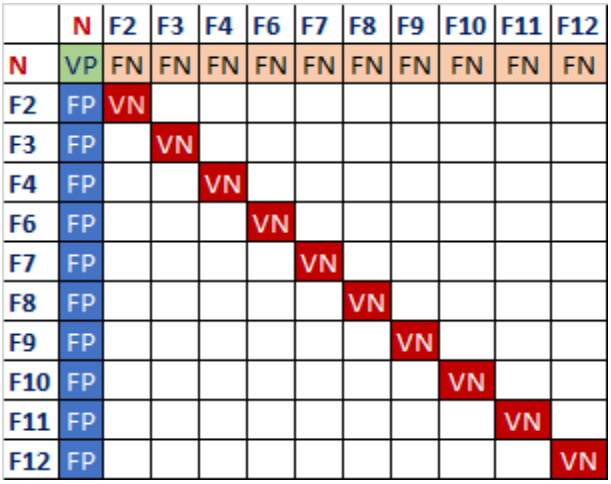


FIGURE 42 – Illustration de VP_{N}, FN_{N}, FP_{N} et VN_{N} pour la classe N.

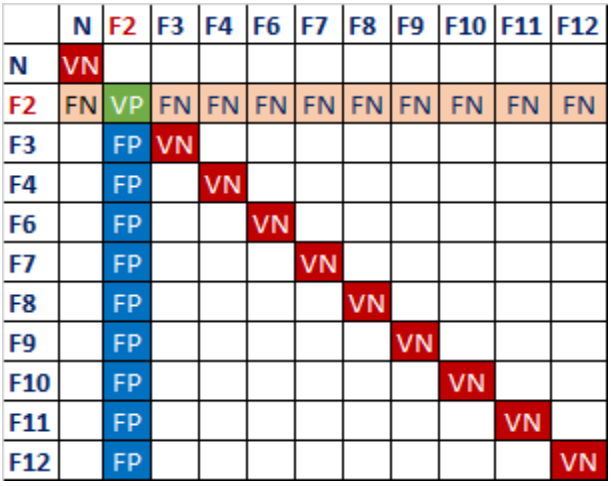


FIGURE 43 – Illustration de VP_{F2}, FN_{F2}, FP_{F2} et VN_{F2} pour la classe *F2*.

Bien que la matrice de confusion nous donne beaucoup d'informations sur la qualité du système de classification, nous pouvons ainsi calculer d'autres métriques plus concises comme la "*Précision*" (en anglais *"precision"*), le "*Rappel*" (en anglais *"recall"*), le score *F-mesure* (en anglais "*F1 score*"), le "*taux de bonne classification*" (en anglais *"average accuracy"*), etc. Dans notre étude, nous adoptons le taux de bonne classification, la précision et le rappel comme des mesures de performances. En effet, le taux de bonne classification nous informe sur le taux de bonne détection de fautes, la précision nous permet d'étudier

le taux des prédictions positives et le rappel nous indique le taux d'observations positives ayant été correctement détectées par le classifieur. Pour une classification multi-classes, ces mesures sont décrites à travers ces équations.

— *Précision*$_i$: Mesure le nombre de cas que le classificateur a correctement assigné à la classe C_i divisé par le nombre de cas attribués à cette classe.

$$\text{Précision}_\text{i} = \frac{\text{VP}_\text{i}}{\text{VP}_\text{i} + \text{FP}_\text{i}} \tag{V.1}$$

— *Rappel*$_i$ (ou Sensibilité) = Mesure la proportion de cas correctement affectés à la classe C_i divisée par le nombre de cas appartenant à cette classe.

$$\text{Rappel}_\text{i} = \frac{\text{VP}_\text{i}}{\text{VP}_\text{i} + \text{FN}_\text{i}} \tag{V.2}$$

Pour calculer la précision et le rappel moyen de toutes les classes, nous pouvons utiliser soit une *macro-moyenne* ou une *micro-moyenne*. La *macro-moyenne* consiste à calculer la précision ou le rappel pour chaque classe puis à calculer la moyenne des classes [Sokolova and Lapalme (2009)]. La *micro-moyenne* fait le calcul global de la précision ou du rappel sur l'ensemble des classes. Dans notre étude, nous avons choisi de calculer la macro-moyenne de la précision (notée $\text{Precision}_\text{M}$) et la macro-moyenne du rappel (notée Rappel_M) car pour le calcul de ces mesures finales, toutes les classes ont le même poids. Elles sont décrites respectivement à travers les équations V.3 et V.4.

$$\text{Precision}_\text{M} = \frac{\sum_{i=1}^{l} \frac{\text{VP}_\text{i}}{\text{VP}_\text{i}+\text{FP}_\text{i}}}{l} \tag{V.3}$$

$$\text{Rappel}_\text{M} = \frac{\sum_{i=1}^{l} \frac{\text{VP}_\text{i}}{\text{VP}_\text{i}+\text{FN}_\text{i}}}{l} \tag{V.4}$$

avec l est le nombre de classes.

— Average Accuracy (AC) : Mesure le taux de bonne classification. Elle est introduite dans cette équation :

$$AC = \frac{\sum_{i=1}^{l} \frac{\text{VP}_\text{i}+\text{VN}_\text{i}}{\text{VP}_\text{i}+\text{FN}_\text{i}+\text{FP}_\text{i}+\text{VN}_\text{i}}}{l} \tag{V.5}$$

— Couverture : C'est une mesure que nous introduisons afin de mesurer le taux de bonne classification en prenant en compte le nombre de cas non identifiés puisque AC calcule le taux de bonne classification en prenant en compte le nombre de cas mal classés. La couverture est définie comme suit :

$$Couverture = 1 - \frac{N_{nid}}{N} \tag{V.6}$$

avec N est le nombre de cas de la base et N_{nid} est le nombre de cas non identifiés.

2 Résultats

Afin d'estimer la fiabilité de notre système et vérifier la stabilité des résultats obtenus, nous avons utilisé une validation croisée stratifiée de 10 folds. Les tests sont effectués sur un processeur 4 cœurs avec *32* GO de RAM. La méthode proposée a été répétée **10 000** fois pour effectuer les expérimentations décrites ci-dessous. Dans cette section, nous présentons comment définir le *"seuil de décision"* et ensuite nous donnons le taux de bonne classification de notre approche ainsi que l'influence du nombre de K cas à prendre en compte pour la construction de la solution et l'indice de dissimilarité choisie sur la performance obtenue.

2.1 Choix du seuil de décision

Nous avons mentionné dans la sous section 1.2.2 du chapitre 4 que le cas source le plus similaire à un nouveau cas à résoudre doit avoir un indice de similarité maximal qui doit être supérieur à un certain seuil de décision. Si ces deux conditions sont vérifiées alors le nouveau cas hérite la solution du cas cible, sinon le nouveau cas est classé comme un cas "*non identifié*". Afin de choisir ce seuil de décision, nous avons calculé différents indicateurs dont les résultats sont décrits dans les figures 44, 45, 46 et 47 :

- La figure 44 présente le nombre de "*Faux Positifs*" (FP), le nombre de "*Faux négatifs*" (FN), le nombre de *"cas défaillants mal classés"* et le nombre de *"cas non identifiés"* pour différentes valeurs de seuil qui varient entre [0, 1]. La figure 45 est un zoom pour les valeurs de seuils qui sont comprises entre [0.750, 0.950]. Les FP représentent les fautes qui sont classées par le système en tant que comportements normaux. Les FN présentent les cas normaux qui sont classés par le système comme étant des comportements défaillants. Cette valeur représente les "*fausses alarmes*" qui déclenchent le système d'alarme sans présence de vraie faute. Les *cas défaillants mal classés* sont les cas défaillants qui sont classés dans des classes

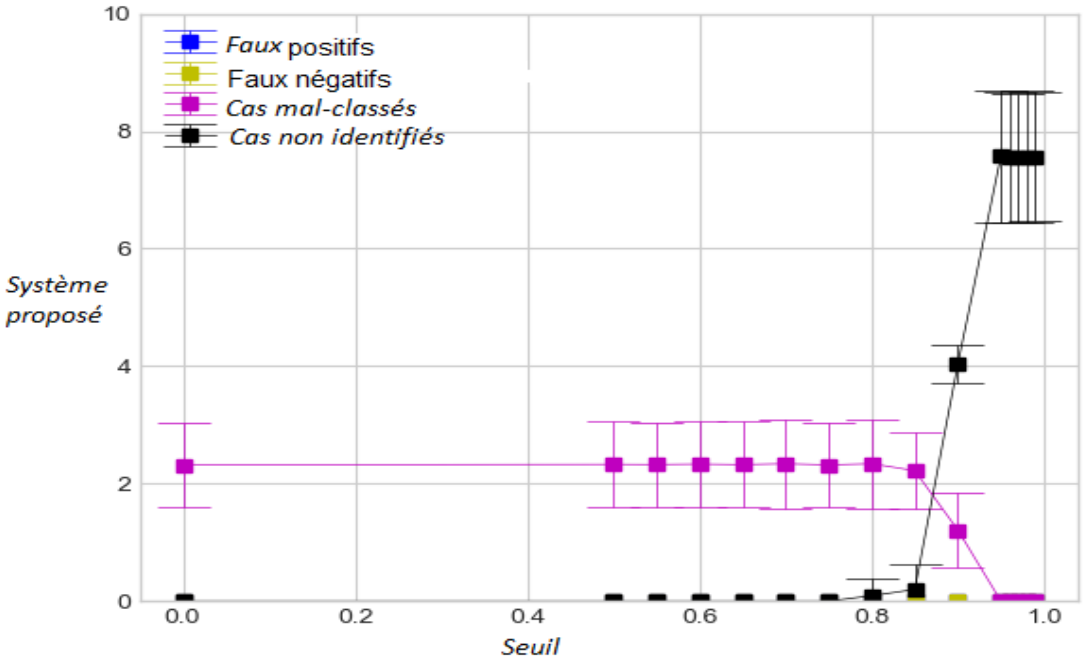


FIGURE 44 – Variations des faux positifs, des faux négatifs, de cas mal-classés et cas non identifiés en fonction des seuils de décision dans l'intervalle [0, 1].

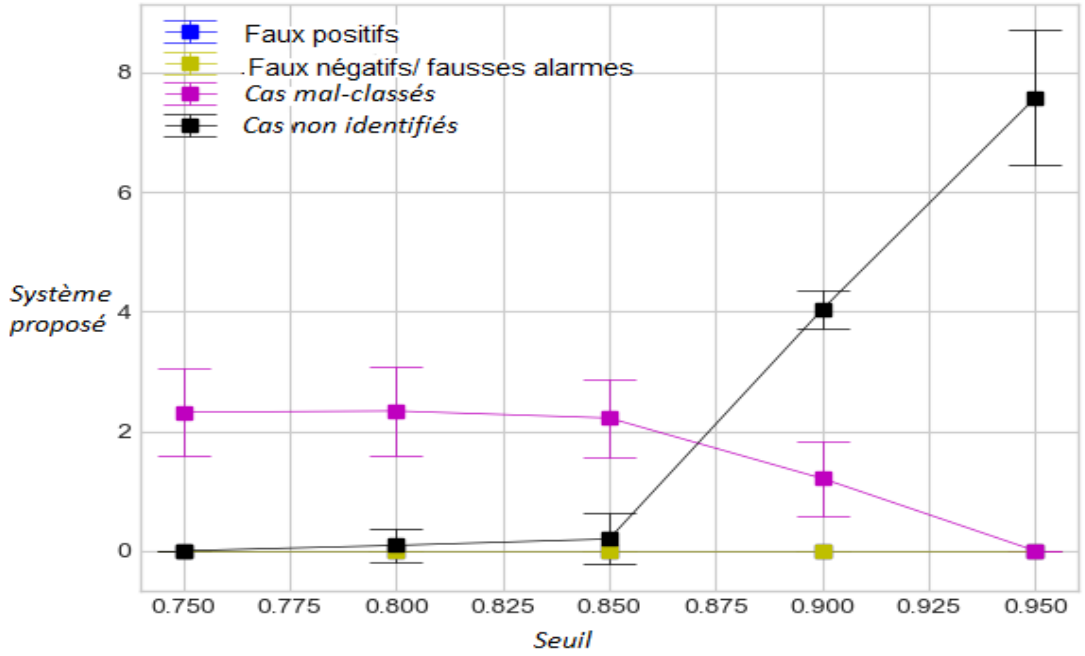


FIGURE 45 – Variations des faux positifs, des faux négatifs, de cas mal-classés et cas non identifiés en fonction des seuils de décision dans l'intervalle [0.750, 0.950].

autres que leurs classes réelles (leurs classes prédites sont différentes de leurs classes réelles).

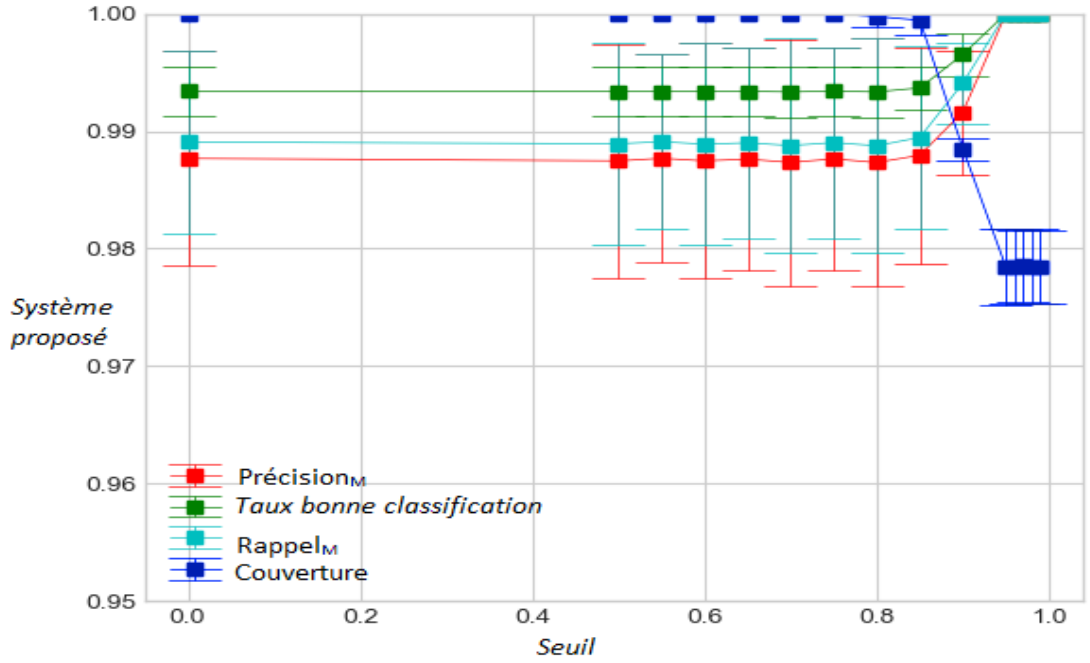


FIGURE 46 – Variations de la macro-moyenne de précision, la macro-moyenne du rappel, le taux de bonne classification et la couverture en fonction des seuils de décision dans l'intervalle [0, 1].

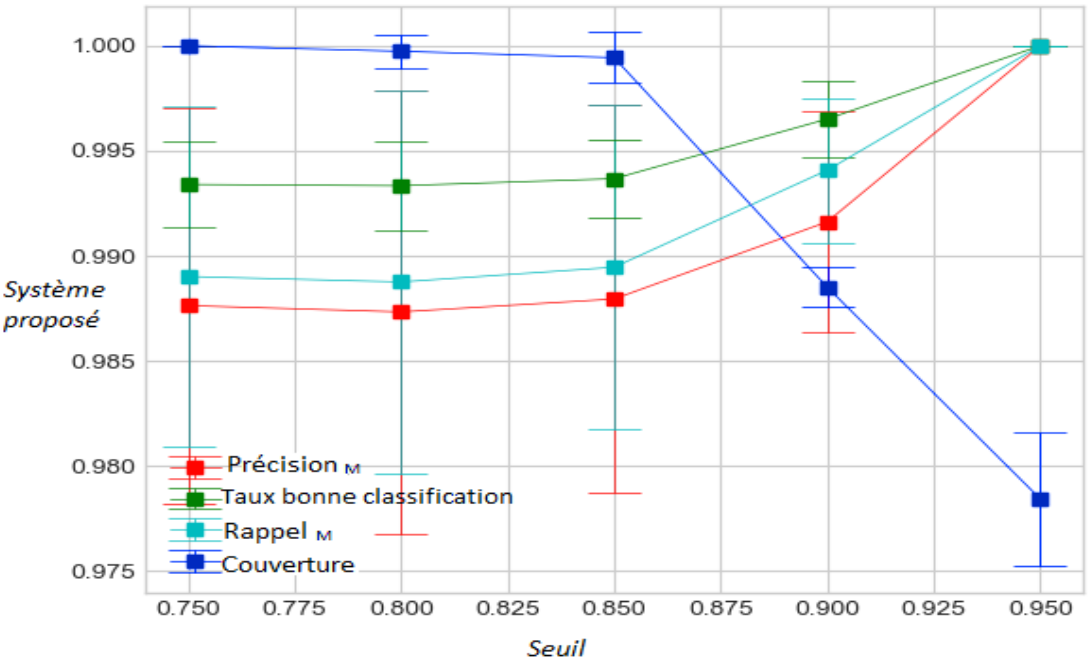


FIGURE 47 – Variations de la macro-moyenne de précision, la macro-moyenne du rappel, le taux de bonne classification et la couverture en fonction des seuils de décision dans l'intervalle [0.750, 0.950].

- La figure 46 illustre les valeurs de la macro-moyenne de précision, la "*macro-moyenne du rappel*", le "*taux de bonne classification*" et la "*couverture*" du système proposé pour différentes valeurs de seuils qui varient entre [0, 1]. La figure 47 est un zoom pour ces mesures pour les valeurs de seuils qui sont comprises entre [0.750 ; 0.950].

A partir des figures 45 et 47, nous avons choisi la valeur de seuil de décision qui est égale à "*0.85*". Cette valeur de seuil nous donne le meilleur compromis précision/rappel qui dépasse 98% et une couverture qui est égal à 99% (voir la figure 47). Ainsi à partir de cette valeur de ce seuil, le nombre de "*cas défaillants mal classés*" diminue et le nombre de "*cas non identifiés*" augmente (voir la figure 45).

La figure 48 présente la matrice de confusion multi-classes obtenue à partir d'une exécution aléatoire de notre module de raisonnement et d'apprentissage en ligne en utilisant la valeur du seuil définie précédemment. Le nombre de cas dans sa diagonale donne une image de la bonne classification : *345* cas sont correctement classés, 176 cas sont classés comme étant des cas normaux et *169* cas sont classés comme étant des cas défaillants. Aucune fausse alarme n'a été générée (voir les cases orange de la matrice) et aucune faute n' a été classée comme étant un comportement normal (voir les cases bleu de la matrice). Seulement deux instances de la faute *F8* sont mal classées et une instance de la faute *F4* n'a pas été identifiée.

	N	F2	F3	F4	F6	F7	F8	F9	F10	F11	F12
N	176	0	0	0	0	0	0	0	0	0	0
F2	0	11	0	0	0	0	0	0	0	0	0
F3	0	0	28	0	0	0	0	0	0	0	0
F4	0	0	0	29	0	0	0	0	0	0	0
F6	0	0	0	0	16	0	0	0	0	0	0
F7	0	0	0	0	0	26	0	0	0	0	0
F8	0	1	0	0	0	1	16	0	0	0	0
F9	0	0	0	0	0	0	0	13	0	0	0
F10	0	0	0	0	0	0	0	0	11	0	0
F11	0	0	0	0	0	0	0	0	0	15	0
F12	0	0	0	0	0	0	0	0	0	0	4

FIGURE 48 – Matrice de confusion multi-classes pour notre approche proposée.

2.2 Influence du nombre *K* et de l'indice de dissimilarité sur la performance du système

La méthode des K plus proches voisins (KNN) est une méthode de classification supervisée non paramétrique qui est largement utilisée dans la classification multi-classes vue sa simplicité conceptuelle, son applicabilité générale et son efficacité. Elle suppose qu'une nouvelle observation est similaire à ses K voisins. K représente le nombre d'instances (les voisins) à prendre en compte pour ranger cette nouvelle observation. Ce nombre est un hyperparamètre dont la valeur doit être fixée avant le début de l'algorithme. Il représente ainsi un des facteurs les plus importants de l'algorithme qui peut fortement influencer la qualité des prédictions.

Notre phase de raisonnement et d'apprentissage utilise le même principe que la méthode KNN. Nous pouvons donc la considérer comme une méthode KNN. La première question importante qui se pose lors l'utilisation de cette méthode est quelle valeur de K devrions-nous utiliser pour avoir un bon taux de bonne classification. Par conséquent, le but est de déterminer si l'augmentation du nombre de cas à prendre en compte pour la construction de la solution augmente ou diminue la performance de notre méthode.

La deuxième question qui se pose lors de l'utilisation d'une méthode KNN concerne le choix d'une mesure de distance ou d'une mesure de similarité ou encore d'un indice de dissimilarité ou de similarité. En effet, pour faire des prédictions avec l'approche KNN, nous devons définir ou choisir une métrique de distance ou de similarité ou un indice de dissimilarité ou de similarité. Cette métrique ou cet indice à utiliser peut aussi largement affecter les résultats de prédiction surtout que dans la littérature il y a peu de consensus sur les métriques les plus performantes et leurs domaines d'applications [Cha (2007)].

Ces deux questions peuvent être résolues expérimentalement. Pour cela nous avons réalisé une étude sur la variation des valeurs de K (entre 1 et 25) pour différentes métriques de distance et de similarité tels que la distance de Jaccard, la distance de Hamming, le coefficient d'appariement simple (Simple Matching Coefficient (SMC)) et notre indice de dissimilarité. Pour chaque variation de K, nous avons calculé le taux de bonne classification en utilisant une mesure différente. A travers cette étude, nous pouvons ainsi comparer la performance de notre indice de dissimilarité vis à vis ces métriques.

Avant de présenter les résultats de cette étude, nous rappelons les équations de différentes métriques qui sont représentées à travers les équations V.7, V.8, V.9 et V.10.

$$D_{Euclidienne}(X,Y) = \sum_{i=1}^{k} |\mathrm{x_i} - \mathrm{y_i}|^2)^{\frac{1}{2}} \tag{V.7}$$

$$D_{Hamming}(X,Y) = \sum_{i=1}^{k}[\mathrm{x_i} \neq \mathrm{y_i}] \tag{V.8}$$

$$D_{Jaccard}(X,Y) = \frac{NTF + NFT}{NTF + NFT + NTT} \tag{V.9}$$

$$D_{SMC}(X,Y) = \frac{NTF + NFT}{NTF + NFT + NFF + NTT} \tag{V.10}$$

Avec $X = \{x_1, x_2,..,x_k\}$, $Y = \{y_1, y_2,..,y_k\}$ et k est le nombre de descripteurs,

- NTT représente le nombre de variables dont la valeur est à 1 dans $\mathrm{x_i}$ et dans $\mathrm{y_j}$;
- NTF représente le nombre de variables qui sont à 1 dans $\mathrm{x_i}$ et à 0 dans le $\mathrm{y_i}$;
- NFT représente le nombre de variables qui valent 0 dans $\mathrm{x_i}$ et 1 dans $\mathrm{y_i}$;
- NFF représente le nombre de variables qui sont à 0 dans $\mathrm{x_i}$ et 0 dans $\mathrm{y_j}$.

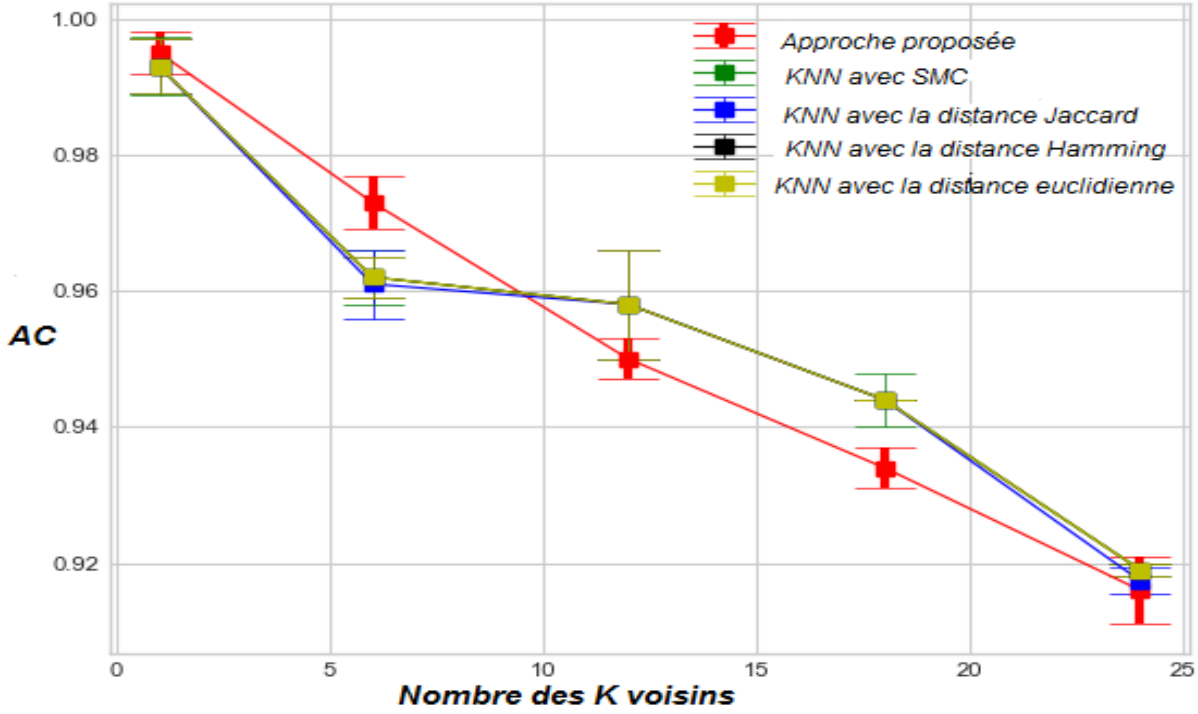


FIGURE 49 – Comparaison du taux de bonne classification des différentes méthodes KNN en fonction de K. Les barres d'erreur indiquent les performances obtenues avec la meilleure (respectivement la mauvaise) répétition de ces méthodes.

La figure 49 illustre les résultats de cette étude. Nous constatons que le taux de bonne classification (Average Accuracy) à K=*1* est aux alentours de 0.99% pour toutes les métriques ce qui conduit à une classification correcte de 345 cas parmi 348 cas. Nous

pouvons ainsi remarquer que la valeur de K augmente, le taux de bonne classification diminue avec toutes les mesures. Pour les valeurs de K dans $\in$ [1, 9] et k =*24* notre métrique donne un meilleur taux cependant pour les valeurs de K dans $\in$ [10, 23] les autres métriques fournissent des taux de bonne classification plus élevés. Par conséquent, nous pouvons déduire à partir des résultats obtenus que K=1 est la meilleure valeur de cet hyperparamètre et que pour cette valeur de K, notre métrique donne un taux de bonne classification qui est égal aux taux des autres métriques.

3 Comparaison avec les approches d'apprentissage automatique

Dans cette partie, nous nous intéressons à la comparaison de notre approche à certaines méthodes d'apprentissage automatique. Ces méthodes sont :

- "Les k plus proches voisins" connue sous le nom de *KNN* de l'anglais « *k-nearest neighbor*» ;
- "Les réseaux de neurones artificiels" (en anglais « artificial neural networks ») [Rosenblatt (1961)] ;
- "Les machines à vecteurs de support" (en anglais *Support Vector Machine* (*SVM*)) [Suykens and Vandewalle (1999)] ;
- "La régression logistique" (en anglais *Logistic regression*) [Berkson (1944)] ;
- "Les forêts aléatoires" (en anglais *Random forest*) [Breiman (2001)].

Ces méthodes sont implémentées dans une librairie libre pour l'apprentissage automatique, appelée « Scikit-learn » [Pedregosa et al. (2011)]. Elle est écrite en python. Elle implémente des algorithmes d'apprentissage supervisés pour résoudre des problèmes de classification, de regréssion et aussi des algorithmes d'apprentisage non supervisés pour résoudre des problèmes de regroupement.

Dans la suite de ce chapitre, ces méthodes vont être appelées «classifieurs» puisqu'elles sont utilisées pour résoudre un problème de classification. Le choix de ces classifieurs s'effectue de manière semi-aléatoire pour leur diversité de représentation et leur style d'apprentissage. Ils sont décrits dans l'annexe 1. Nous adoptons le taux de bonne classification (AC) comme une mesure de performance et la validation croisée stratifiée avec 10 folds comme une méthode d'évaluation. Les classifieurs utilisées sont répétés **10 000** fois pour mesurer AC et les écarts types.

Avant de présenter les résultats d'évaluation et de comparaison, il est nécessaire d'exposer premièrement comment nous avons préparé nos données pour que les classifieurs

fonctionnent correctement et deuxièmement, comment "*Scikit-learn*" réalise des classifications multi-classes en utilisant des classifieurs binaires.

3.1 Préparation des données

Avant d'entrainer un classifieur, il est important de préparer les données. En effet, certains classifieurs peuvent fonctionner directement avec des données catégoriques comme le KNN et notre approche proposée. Cependant plusieurs nécessitent que les variables d'entrées et de sorties soient numériques pour fonctionner correctement.

Les données catégoriques sont les variables qui contiennent des valeurs de type "chaine de caractères". Par exemple un attribut "Couleur" peut avoir trois couleurs "Rouge", "Vert" et "Bleu". Chaque valeur représente une catégorie différente. Certaines variables peuvent avoir une relation entre-elles. A titre d'exemple un attribut "Place" peut avoir trois valeurs "Un", "Deux", "Trois". Il y a une relation d'ordre entre les valeurs de la variable "Place" c'est pour cette raison elle est appelée "variable catégorique ordinale".

Nos cas sont formés par des données catégoriques (les événements référents, les événements contraints et la solution (S)) et aussi par des données numériques (les bornes inférieures et supérieures des contraintes temporelles). Pour cela, nous devons convertir les données catégoriques à des données numériques. Scikit-learn propose deux techniques : "label encoding" et "Onehot Encoding".

- La méthode "*Label encoding*" consiste à attribuer pour chaque catégorie une valeur entière. Par exemple pour l'attribut "couleur", elle attribue "1" pour la couleur rouge, "2" pour la couleur vert et "3" pour la couleur "bleue".

- La technique "*One hot Encoding*" est utilisée pour la représentation entière, la variable qui est codé en entier est remplacée par un ensemble de valeurs binaires.

Pour les données catégoriques qui représentent nos cas, il n'existe pas de relation d'ordre entre elles. A titre d'exemple dans la figure 50, *ER2* peut avoir quatres valeurs possibles : FC4, RS4, RC5 et FS4. Si nous appliquons la méthode "*Label encoding*", la valeur "FC4" sera remplacée par "1", "RS4" par "2", "RC5" par "3" et "RS4" par 5. Cependant les valeurs entières ont une relation d'ordre entre elles. Les classifieurs peuvent comprendre et exploiter cette relation et en conséquence, ils donnent des performances médiocres et des faux résultats. Pour palier le problème, nous avons utilisé avec cet encodage un encodage "One hot encoding" qui remplace l'attribut *ER2* par quatres variables "*Est*FC4", "*Est*RS4", "*Est*RC5", et "*Est*FS4" (voir figure 50).

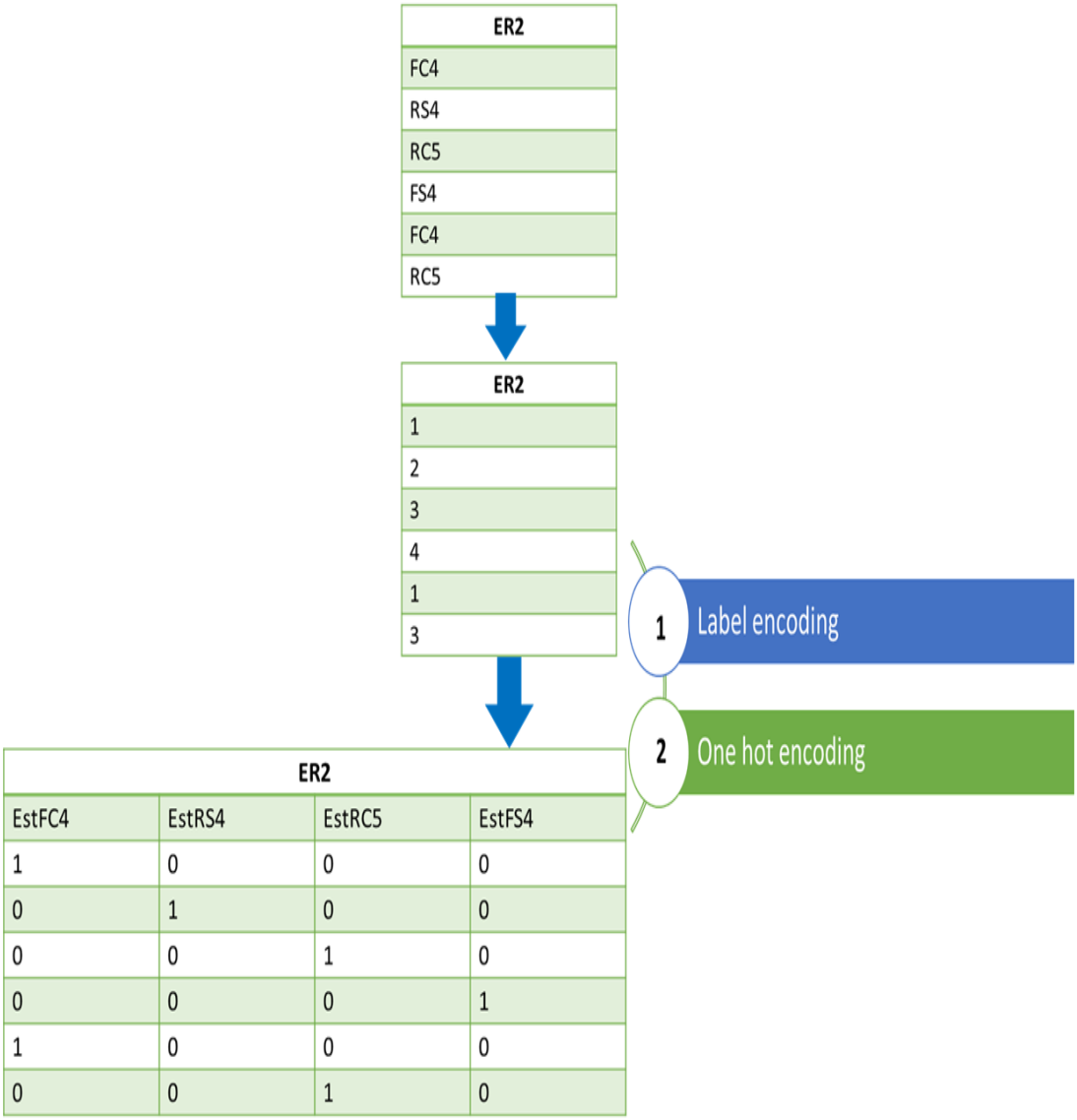


FIGURE 50 – Le codage de nos données en utilisant la méthode "Label encoding" et la méthode "One hot coding".

3.2 Classifications multi-classes en utilisant des classifieurs binaires

Certains classifieurs comme les "*forêts aléatoires*" et le "*KNN*" sont capables de gérer directement des classifications multi-classes alors que d'autres algorithmes comme les "*SVM*" et la "*régression logistique*" sont des classifieurs binaires. Pour cela, "*scikit-learn*" utilise deux méthodes permettant de réaliser des classifications multi-classes avec des classifieurs binaires. Ces méthodes sont : la méthode "*Un contre tous* " connue sous le nom de "*One vs All (OvA)*" ou "*One vs Rest (OvR)*" et la méthode "Un contre un " connue sous le nom de *"One vs One"*.

a) La méthode *OvA* consiste à entrainer un classifieur binaire pour chaque classe. A titre d'exemple, si on a trois classes classe "A", classe "B" et classe "C", pour classer une observation "x" à l'une de ces classes, la méthode entraine trois classifieurs : ((i) Un détecteur de la classe "A", (ii) un détecteur de la classe "B" et(iii) un détecteur de la classe "C"). Après il suffit d'obtenir le score de décision de toutes les classes et de ranger "x" à la classe qui a le meilleur score.

b) La méthode *OvO* permet d'entrainer un classifieur pour chaque paire de classe. Pour N classes, elle entraine "*N*(N-1)/2*" classifieurs binaires. Pour l'exemple de trois classes, la méthode *OvO* entraine trois classifieurs binaires : (i) un pour distinguer la classe "A" et la classe "B", (ii) un pour distinguer la classe "A" et la classe "C" et (iii) un pour distinguer la classe "B" et la classe "C". Après pour ranger l'observation "x" à la bonne classe, il suffit de sélectionner le classifieur qui donne le meilleur score de décision.

3.3 Résultats de comparaison

Nous présentons dans cette partie les résultats de comparaison de notre méthode avec les classifieurs présentés ci-dessus. Les implémentations de ces classifieurs sous Scikit-learn sont les suivantes :

- **MLPClassifier** [1] est un algorithme qui implémente un perceptron multicouche en utilisant pour l'entrainement la méthode de rétropropagation décrite dans l'annexe 3.2. Il supporte la classification multi-classes en utilisant la fonction "Softmax". Le nombre de couches cachées est égal à 100. La fonction d'activation utilisée est la fonction "**R**ectified **L**inear **U**nit" *ReLU* qui est une fonction linéaire qui renvoie le résultat s'il est positive sinon elle renvoie 0 (voir l'équation V.11).

$$ReLU(z) = max(0, z) \tag{V.11}$$

avec z est la somme pondérée calculée par un neurone d'une couche cachée.

Pour la fixation des poids, la méthode de gradient ordinaire calcule les gradients d'erreurs en fonction de l'ensemble du jeu d'entrainement alors la méthode de "descente de gradient stochastique" utilisée par cet algorithme, calcule les gradients d'erreurs en prenant en compte seulement une seule observation du jeu d'entrainement prise au hasard [Géron (2017)]. Cela permet de rendre l'algorithme plus rapide.

1. https ://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

- **SVC**[2](*Support Vector Classification*) est un algorithme qui implémente la méthode de Séparateur à vaste Marge (*SVM*) pour résoudre les problèmes de classification. Le principe de la classification SVM linéaire et non linéaire est présentée dans l'annexe 1. Cette implémentation de SVM utilise une fonction noyau linéaire (*Linear Kernel*) définie dans l'équation V.12 et l'hyperparamètre C qui permet de contrôler la taille de la marge qui est fixé à 1. Dans cette implémentation, la classification multi-classes est gérée selon la technique "*OvO*".

$$Noyau_{Linéaire}(x, l) = x^T l + C \tag{V.12}$$

avec x est l'observation, l est un point de repère.

- *Logistic Regression classifier*[3] (**LRC**) est l'implémentation de la méthode "*Régression logistique*" qui est détaillée dans l'annexe 4. Dans cette implémentation, la classification multi-classes est gérée selon la méthode "OvR".

- *KNeighbors Classifier*[4] **(KNC**) est l'implémentation de la méthode *KNN* qui est décrite dans la sous section 1.2.2 du chapitre 2. En se basant sur l'étude réalisée dans la sous section , nous avons fixé la valeur de K à 1 et nous avons utilisé la distance de Hamming (voir l'équation V.8).

- *Random Forest Classifier*[5] (**RFC**) est l'implémentation de la méthode de "Forêts aléatoires".

Pour analyser l'efficacité de ces méthodes, nous avons utilisé comme méthode d'évaluation une validation croisée stratifiée de 10 "*folds*". La fiabilité de ces méthodes est évaluée par le calcul du taux de bonne classification (AC) qui désigne la proportion des prédictions correctes faites par le modèle. Après avoir appliqué ces implémentations à la même base de données expérimentale, nous avons constaté que notre approche proposée effectue la meilleure prédiction avec un taux de bonne classification d'environ 99,4% suivie de KNC, SVC et RFC qui ont respectivement un AC d'environ 99,3%, 99,2% et 99,2% (voir le tableau 12). Les résultats suggèrent que ces méthodes peuvent effectuer une bonne prédiction avec le moins d'erreurs pour le diagnostic du plateau tournant. Le principal avantage de notre approche proposée est la réduction du nombre de cas mal classés en les rangeant comme des cas non identifiés. C'est pour cette raison la couverture est de l'ordre de 99.7 % (voir la figure 47).

2. https ://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
3. https ://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4. https ://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
5. https ://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

TABLE 12 – Résultats de comparaison entre l'approche proposée et les différents classifieurs.

Classifieurs	**Taux de bonne classification (AC) ± écart type**
SVC	0.992 ± 0.002
KNC	0.993 ± 0.003
LR	0.988 ± 0.002
MLPC	0.613 ± 0.008
RFC	0.992 ± 0.003
Approche proposée	0.994 ± 0.002

Cependant, du point de vue de la complexité computationnelle de l'entraînement et de la prédiction, le RFC est le plus rapide suivi par notre méthode et le KNC puis le SVC. En effet, sa complexité d'entraînement est de l'ordre de $O(M*n*m*log(n))$ et sa complexité de prédiction est de l'ordre de $(O(M*log(m)))$. Notre méthode proposée et le KNC ont une complexité d'entraînement nulle et une complexité linéaire de prédiction $(O(m*n))$. La complexité d'entraînement de SVC est de l'ordre de $O(m^2*n)$ et la complexité de la prédiction est de $O(n_{sv}*n)$ avec M est le nombre d'arbres de décision, n est le nombre d'attributs, m est la taille de l'ensemble d'apprentissage et n_{sv} est le nombre de vecteurs de support.

4 Comparaison avec une approche à base de modèle

Nous avons aussi évalué et comparé les performances de notre approche avec une méthode à base de modèles pour le diagnostic du plateau tournant. Cette méthode est l'approche diagnostiqueur proposée par [Philippot et al. (2012)] et qui est décrite dans le section 1. L'évaluation est réalisée à travers 4 indicateurs de performance : le taux de bonne détection (***TD***), la précision d'isolation (***PI***), le nombre de fausses alarmes (***FA***) et la complexité de mise en œuvre de l'approche. Le tableau 13 résume les résultats de cette comparaison.

L'indicateur *TD* indique la capacité de la méthode à détecter correctement toutes les fautes présentes dans le système. Ce critère est égal au nombre de fautes qui sont correctement détectées divisé par le nombre total de fautes. D'après les simulations, le

diagnostiqueur permet de détecter correctement toutes les fautes qui ont été introduites dans le modèle (172/172=100%) [Philippot et al. (2012)] alors que l'approche proposée ne détecte correctement que 98.2% (169/172) des fautes.

Le critère *PI* nous informe si la méthode est capable d'isoler une faute détectée avec certitude ou non. Il donne un pourcentage < 90 % s'il isole la faute avec certitude, < 10 % s'il l'isole avec deux candidats possibles et < 5 % s'il l'isole avec trois candidats ou plus. L'expérience a montré que les deux approches sont capables de localiser les fautes détectées avec certitude.

Un des objectifs d'une bonne méthode de diagnostic est de diminuer le nombre de fausses alarmes. Ces alarmes vont causer des arrêts inutiles des systèmes de production. D'après les simulations, les deux approches ne présentent aucune fausse alarme.

Pour évaluer la complexité de mise en œuvre de l'approche, nous nous basons sur deux critères : la complexité de la phase hors ligne et la complexité de la phase en ligne. La complexité de la phase hors ligne (CH) représente la complexité temporelle de construction du modèle ou de la base de cas. La complexité de la phase en ligne (CL) désigne la complexité temporelle du parcours du diagnostiqueur ou la complexité de prédiction. La complexité de construction du modèle de l'approche diagnostiqueur est égale à $O(2^{Z-1} * L + Z * L)$ avec Z est le nombre de capteurs et d'actionneurs du système et L est est le nombre des états normaux du modèle. Il s'agit alors d'une complexité exponentielle par rapport au nombre des entrées/sorties du système. La complexité du parcours du modèle est linéaire. Elle est à l'ordre de $O(L + A)$ avec A est le nombre des transitions dans le modèle. La limite majeure de cette méthode se manifeste donc dans la construction du modèle qui est une tâche très coûteuse surtout lors du diagnostic d'un système de taille plus importante. Aussi, il est à noter que ce modèle doit être correct pour ne pas avoir des fausses alarmes, robuste et complet pour garantir la détection de la totalité des fautes. Tant dis que notre approche proposée est moins coûteuse en terme de la complexité de construction de la base de cas qui est à l'ordre $O(N_{cas} * Z)$ et en terme de la complexité de prédiction qui est égale à $O(n * N_{cas})$ où N_{cas} est la taille de la base cas initiale. En plus, elle est plus facile à mettre en place puisque la construction de la base des cas initiale se fait automatiquement sans avoir besoin de la présence de l'expertise humaine.

TABLE 13 – La comparaison entre l'approche proposée et l'approche diagnostiqueur de [Philippot et al. (2012)].

Méthodes	**TD**	**PI**	**FA**	**CH**	**CL**
L'approche diagnostiqueur	100%	100%	0	$O(2^{Z-1} * L + Z * L)$	$O(L + A)$
L'approche proposée	98.2%	100%	0	$O(N_{cas} * Z)$	$O(n * N_{cas})$

5 Conclusion

Dans ce chapitre, nous avons mesuré la performance de l'approche présentée dans le chapitre 4 pour le diagnostic du plateau tournant. Ceci est fait à travers plusieurs séries de mesures effectuées sur le plateau tournant qui ont permis de construire la base de cas initiale. A partir de cette dernière, une étude de classification multi-classes a été réalisée. Elle a montré les avantages de l'approche sur le taux de bonne classification, le nombre de fausses alarmes, la précision d'isolation et la complexité de mise en œuvre de l'approche.

Nous avons aussi mené une étude comparative entre notre approche, certaines méthodes d'apprentissage automatique et une méthode de diagnostic à base de modèles. Les différentes méthodes sont testées sur le même *benchmark* qui est le plateau tournant du système de tri de caisses. Nous avons conclu que la méthode proposée possède un taux de bonne classification comparable aux meilleures méthodes d'apprentissage automatique. Cependant, elle n'a pas le meilleur taux de bonne détection par rapport à l'approche à base de modèles. En effet, cette dernière détecte toutes les fautes décrites dans le modèle. Cependant, elle est difficile à mettre en œuvre et sa complexité temporelle de construction est très coûteuse principalement lorsque le système est plus complexe. En parallèle, la méthode proposée est facile à mettre en œuvre et sa complexité temporelle de construction est moins coûteuse même avec des systèmes complexes. Ceci montre l'efficacité de l'approche proposée pour le diagnostic des fautes des SAP.

Conclusion et perspectives

La motivation de ce travail de thèse est le diagnostic des fautes dans les systèmes automatisés de production ayant une dynamique discrète. L'objectif est de proposer des solutions de diagnostic intelligentes afin de remplacer les solutions traditionnelles qui offrent de bons taux de détection de fautes mais qui sont difficiles à mettre en œuvre.

Trois familles de méthodes existent pour le diagnostic de cette catégorie de systèmes : les méthodes à base de modèles, les méthodes à base de connaissances et les méthodes à base de données. Les deux premières familles sont capables de détecter et de localiser les fautes possibles efficacement mais elles nécessitent pour fonctionner correctement soit un modèle analytique, précis et approfondi du système, soit une base de connaissance qui doit être complète et cohérente. Le modèle analytique du système à diagnostiquer est généralement difficile à établir et sa complexité temporelle de construction est très coûteuse essentiellement lorsque le système à diagnostiquer est complexe. Bien que la construction d'une base de connaissance riche et représentative peut s'effectuer automatiquement en utilisant des approches de construction à base de modèles [Guerraz and Dousson (2004); Saddem and Philippot (2014)] ou des approches de construction à base de données [Dousson et al. (2008); Cram et al. (2012); Subias et al. (2014)], la mise à jour et l'enrichissement de la base de données ne peut pas se faire automatiquement. En effet, les méthodes n'ont pas la capacité d'apprendre de nouvelles connaissances à partir des solutions des problèmes rencontrés et elles ne s'adaptent pas automatiquement aux évolutions du système à diagnostiquer. Ce sont les experts humains qui doivent intervenir pour prendre en compte les nouvelles modifications dans les spécifications du système afin de mettre à jour la base de connaissance, ces nombreux ajustements manuels nécessitent du temps.

Ces limites nous ont motivés à orienter nos travaux vers les méthodes à base de données [Venkatasubramanian et al. (2003); Moosavian et al. (2013); Dou and Zhou (2016);

Vazan et al. (2017); Han et al. (2017)] qui présentent plusieurs avantages et comblent les limites des deux premières familles. Elles gèrent de grandes quantités de données, offrent une meilleure compréhension du problème, apprennent de nouvelles connaissances à partir des solutions des problèmes rencontrés, s'adaptent automatiquement aux évolutions et donnent de bons résultats. Les méthodes à base de données se basent sur les techniques d'apprentissage automatique qui sont nombreuses. Le choix de la technique à utiliser dépend de la nature du problème à résoudre et ses objectifs (c'est-à-dire prédiction, regroupement, etc.)[Géron (2017)].

Apports de la thèse

Dans notre étude, nous avons cherché à déterminer la nature du comportement du système à diagnostiquer à partir des différents signaux binaires des capteurs et des actionneurs dans différents cycles d'API. Pour y parvenir, nous avons suggéré l'utilisation du raisonnement à partir de cas pour sa capacité de raisonnement et d'apprentissage incrémental [Aha et al. (1991); Ali et al. (2018)]. Les contributions apportées dans le cadre de ce travail de thèse se présentent sous la forme de deux propositions :

1. **La première proposition est un système d'aide à la décision pour le diagnostic en ligne des SAP en utilisant le RàPC** [Ben Rabah et al. (2017a)]. L'approche présentée propose l'aide et l'assistance aux OHS en temps réel. Elle ne requiert pas la connaissance complète sur le fonctionnement du système. Elle est capable d'apprendre à partir des problèmes rencontrés. Elle se base sur un format de représentation de cas qui représente les signaux binaires des capteurs et des actionneurs pendant différents cycles d'API. Cependant, cette représentation s'est avérée incomplète car elle ne permet pas de modéliser certaines fautes comme le blocage d'un capteur ou d'un actionneur à 0 ou à 1. De plus, si le nombre de composants du système à diagnostiquer est très important, la proposition est confronté à un problème d'explosion combinatoire des descripteurs. Aussi, cette approche n'est pas évolutive et devient inapplicable dans le contexte de l'industrie du futur où les SAP doivent être extensibles, évolutifs et reconfigurables.
2. **La seconde proposition** est une amélioration de la première proposition [Ben Rabah et al. (2017b)] et se compose de deux phases. Dans la première phase, les cas de la base de cas initiale sont construits à partir des données issues du système simulé après son émulation en mode normal et fautif. Chaque cas de la base est étiqueté (N) si le comportement du système est normal et il est étiqueté par le nom de la faute si le comportement du système est fautif [Ben Rabah et al. (2017c)]. Dans la deuxième phase, pour chaque cycle d'API, *un nouveau cas* est construit à partir des signaux des capteurs et des actionneurs. Ensuite, pour clas-

ser ce nouveau cas, la proposition agit comme un classifieur multi-classes qui cherche à ranger correctement le cas dans l'une des classes prédéfinies.

La validation de cette proposition a été réalisée pour le diagnostic de fautes sur un plateau tournant. Elle a montré ses avantages du point de vue du « taux de bonne classification », du « nombre de fausses alarmes », de la « précision d'isolation » et de la « complexité de la mise en œuvre de l'approche ». L'étude comparative menée entre l'approche proposée avec certaines techniques d'apprentissage automatique et avec une méthode de diagnostic à base de modèles a montré l'efficacité de notre approche pour le diagnostic en ligne des fautes dans les SAP.

Perspectives

Les perspectives présentées dans le cadre de cette thèse s'appuient sur les compétences complémentaires du CReSTIC et du laboratoire "**C**omputing, web services composition, **O**ptimization, **S**upervision, simulation, **M**ultiagent systems and mobility for **O**utstanding (**COSMOS**)" de l'ENSI.

Le travail développé dans la thèse semble exploitable dans le cadre de l'industrie du futur mais il comprend quelques restrictions qu'il faut lever. Pour cela, nous envisageons différents voies d'amélioration à court ou à moyen terme et nous prévoyons plusieurs perspectives d'application à plus long terme.

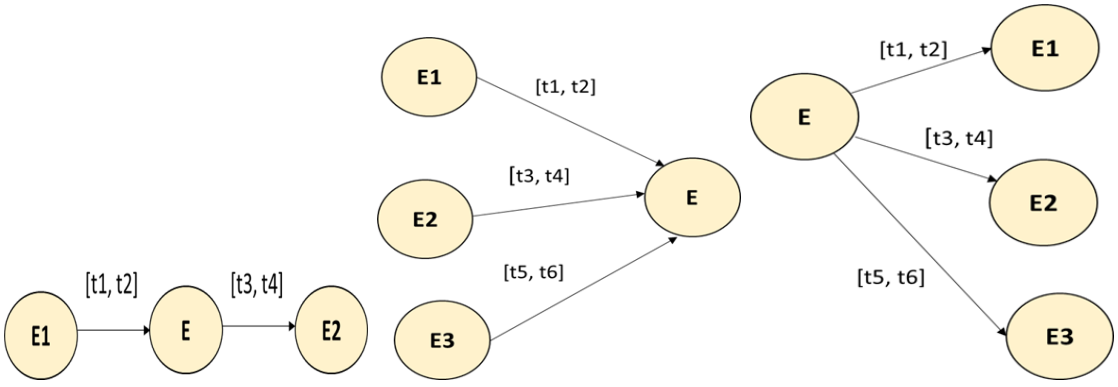
▶ Améliorations possibles

- **Test de l'approche sur un système réel ayant un jumeau numérique**

Pour valider totalement notre approche, il faut la tester sur un système réel. Grâce au projet de recherche de plateformes interconnectées « Factories of Future Champagne-Ardenne » (FFCA). financé dans le cadre du CPER 2018-2020, le CReSTIC va disposer d'un jumeau numérique de l'atelier flexible CELLFLEX [6]. Pour appliquer notre approche, nous cherchons à améliorer l'algorithme de génération de cas pour prendre en compte les changements d'états simultanés de plusieurs E/S. En effet, l'algorithme de génération automatique de cas proposé dans ce travail de thèse n'est pour le moment, pas capable de décrire les changements d'états des entrées/sorties qui ont une relation de causalité entre elles. En effet, l'hypothèse retenue prend uniquement en compte le changement d'état d'une entrée E ou d'une sortie S précédé par le changement d'état d'une seule E/S suivi par le changement d'état d'une seule E/S (voir la figure 51a). Alors que dans les

6. (http ://www.univ-reims.fr/meserp/accueil/plateau-de-formation-et-de-transfert-technologique,9485,27019.html)

systèmes plus complexes, deux autres situations peuvent se produire : la convergence et la divergence. La convergence signifie qu'il peut y avoir des changements d'états de plusieurs *E/S* au même temps suivi par un changement d'état d'une seule *E/S* (voir la figure 51*b*). La divergence caractérise le changement d'état d'une seule *E/S* suivi par les changements d'états de plusieurs *E/S* (voir la figure 51*c*). Une amélioration possible consiste alors à prendre en compte ces deux situations dans l'algorithme de génération automatique de cas.



(a) Séquentiel. (b) Convergence. (c) Divergence.

FIGURE 51 – (a) Graphes temporels des différents situations possibles.

- **Optimisation de la base de cas**

Le système de RàPC pour le diagnostic des SAP est conçu pour résoudre des problèmes tout au long du fonctionnement du système industriel. En fonctionnant sur de longues périodes, il est possible que de très nombreux cas soient générés augmentant ainsi la taille de la base de cas. La conséquence directe est la détérioration des résultats du système de diagnostic en raison d'une augmentation du temps dans l'étape de recherche des cas sources les plus similaires. Pour pallier ce problème, l'optimisation de la base de cas devient nécessaire. Elle vise à réduire la taille de la base de cas en supprimant des cas afin de diminuer le temps de recherche tout en préservant la qualité de la base de cas. La littérature présente plusieurs stratégies de suppression de cas :

- La suppression aléatoire qui consiste à éliminer aléatoirement un cas de la base dès que sa taille dépasse un certain seuil [Markovitch and Scott (1988)].
- La suppression des cas les moins utilisés qui s'effectue en calculant la fréquence d'utilisation de chaque cas. Les cas qui ont une fréquence faible sont supprimés de la base de cas [Minton (1990)].
- La suppression basée sur la redondance et l'inconsistance qui consiste à prendre chaque cas de la base pour lui faire passer dans un premier temps, un test de redondance. Si le cas n'est pas redondant, un deuxième test d'inconsistance est

réalisé. Il faut vérifier s'il y a un conflit entre la partie problème et la partie solution du cas. Si le test n'est pas vérifié alors une alerte est générée pour avertir l'utilisateur. Ce dernier peut ignorer l'alerte ou supprimer le cas [Racine and Yang (1996)].

- La suppression basée sur la distribution qui étudie l'organisation des cas dans la base de cas. Il s'agit de garder un ensemble de cas homogène car dans une base de cas non homogène, le système de RàPC résoudra les problèmes incorrectement [Smyth and McKenna (1998)].

Les deux premières méthodes sont simples à mettre en œuvre mais elles peuvent influencer la qualité de la base de cas et la performance du système de diagnostic. En effet, en supprimant des cas de la base aléatoirement, des cas de grandes importances peuvent être éliminés. En supprimant les cas les moins utilisés, des cas peuvent être retirés alors que certains cas représentent des fautes dont la fréquence d'apparition est faible. Les deux autres méthodes semblent intéressantes car elles prennent en compte soit les deux critères de redondance et d'inconsistance soit le critère d'homogénéité des cas. Regrouper ces critères dans une solution nous semble intéressant étant donné qu'ils assurent l'optimisation de la taille de la base tout en garantissant son homogénéité et sa cohérence.

▶ Perspectives d'application

- **Applicabilité de l'approche sur des systèmes embarqués**

Les systèmes embarqués intègrent du logiciel et du matériel et sont conçus pour assurer des fonctionnalités critiques. Ils se caractérisent par une interaction continue avec leur environnement, une puissance de traitement de plus en plus importante, un faible coût d'utilisation, etc. Cependant, afin d'assurer le bon fonctionnement et garantir la disponibilité de ces systèmes, il est important de concevoir un système de diagnostic fiable. Dans ce contexte, nous pouvons appliquer notre proposition pour le diagnostic des systèmes embarqués logiques. Cette classe des systèmes embarqués propose des fonctions logiques mises en œuvre à l'aide de composants numériques comme les micro-contrôleurs, FPGA, CPLD. Chaque fonction logique permet de calculer au fil du temps la valeur d'une sortie en fonction de ses entrées et de son état interne. Ses entrées/sorties sont de type Tout ou Rien (TOR). Notre approche peut être utilisée pour le diagnostic de cette classe de systèmes dans plusieurs domaines d'application comme le transport (diagnostic de la carte de commande du système de freinage d'un train [Saddem et al. (2011b)]), les télécommunications (détection d'intrusion dans des réseaux embarqués mobiles [Studnia (2015)], le médical (diagnostic des instruments de mesures), etc.

- **Orientation de l'approche vers une structure distribuée**

A long terme il est possible de proposer une architecture de diagnostic distribuée en temps réel qui repose en particulier sur notre proposition et sur les Systèmes Multi-Agents (SMA)[Ferber and Weiss (1999); Van der Hoek and Wooldridge (2008)]. En effet, les avancées scientifiques réalisées dans cette thématique de recherche montrent bien l'adaptation des SMA à la gestion efficace des systèmes complexes, dynamiques et fortement décentralisés [Ben Hmida et al. (2015)]. Ce genre de systèmes est utilisé dans une grande variété d'applications comme par exemple le contrôle du trafic aérien [Breil et al. (2017)], le diagnostic des systèmes manufacturiers [Peixoto et al. (2015); Rocha et al. (2017)], la fabrication des jeux [Ocio and Brugos (2009)], etc. Ces systèmes se caractérisent par un ensemble de propriétés qui leur permettent de résoudre les problèmes les plus importants dans les systèmes complexes [Ben Hmida (2013)]. A titre d'exemple, la décomposition en agents offre un moyen efficace pour partitionner le problème complexe en plusieurs sous problèmes plus faciles à résoudre. Dans ce cas, chaque agent remplit une partie du problème d'une manière autonome et après l'ensemble d'agents coopèrent et coordonnent leurs actions pour résoudre le problème initial. Cependant nous devons choisir une architecture multi-agents avec un nombre minimum de communication entre les agents afin de minimiser le temps de résolution du problème de diagnostic.

A Méthodes d'apprentissage automatique

Nous présentons dans cette partie le principe des méthodes d'apprentissage automatique les plus utilisées pour la résolution des problèmes de classification. Ces méthodes sont les *Machines à vecteurs de support (SVM)*, les *arbres de décision*, les *forêts aléatoires*, les *réseaux de neurones* et la *régression logistique*. Ces méthodes sont détaillées dans la suite.

1 Machines à vecteurs de support (SVM)

Les *Machines à vecteurs de support* ou *Séparateurs à Vaste Marge* (SVM)(en anglais appelés *Support Vector Machines*) sont des algorithmes d'apprentissage automatique désireuses qui sont capables de faire des classifications linéaires et non linéaires, des régressions et des détections de données aberrantes [Géron (2017)]. Dans la section suivante, nous allons présenter le principe d'une classification SVM linéaire et non linéaire.

1.1 Classification SVM linéaire

Pour expliquer le principe d'une classification SVM linéaire, supposons que nous disposons de deux ensembles de formes géométriques différentes : un ensemble qui regroupe les triangles et un ensemble qui rassemble les cercles. Les deux ensembles sont représentés dans la figure 52. Pour ranger une nouvelle forme géométrique, par exemple l'étoile verte de la figure 52, le classifieur SVM linéaire dessine une ligne droite qui sépare les formes en deux classes (c'est à dire une classe pour les cercles et une pour les triangles). Après il peut repérer si cette nouvelle forme est rangée comme étant un triangle ou un cercle. Dans cet exemple, la nouvelle observation est classée comme étant un cercle.

L'emplacement de la ligne par rapport aux formes nous informe sur la qualité du modèle. En effet, si la ligne droite est très proche des formes de deux classes alors il s'agit d'un mauvais modèle. Il ne donnera pas probablement des bons résultats pour des nouvelles observations. Si la ligne droite est distante par rapport aux formes les plus proches comme dans le cas de la figure 52 alors le modèle est bon. Cette classification est appelée une "*classification à large marge*".

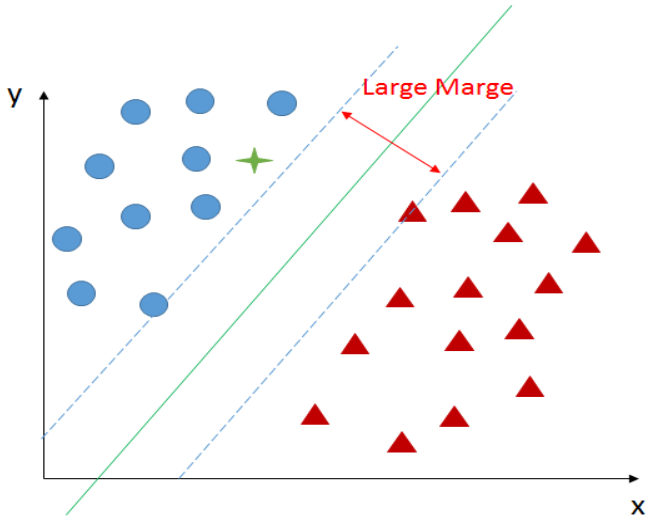


FIGURE 52 – Classification SVM linéaire à large marge.

Cependant en ajoutant d'autres données, nous pouvons les avoir à l'intérieur du chemin (représenté par les deux lignes pointillées). Pour cette raison, il est mieux d'utiliser un modèle qui fait un compromis entre une large marge et limite le nombre de données à l'intérieur du chemin. L'hyperparamètre "C" permet de contrôler cette proportion. Si la valeur de C est grande alors la marge est petite comme dans l'exemple présenté dans la figure 53. A l'inverse avec une petite valeur de C, la marge devient large et dans ce cas nous aurions plus de données à l'intérieur du chemin (voir l'exemple de la figure 52).

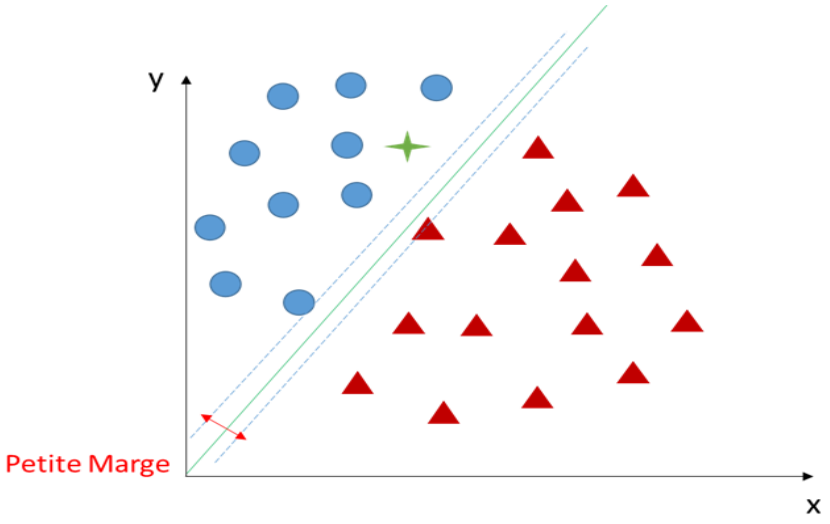


FIGURE 53 – Classification SVM linéaire à petite marge.

Les classifieurs SVM linéaires sont efficaces cependant ils sont valables seulement pour les jeux de données linéairement séparables. Dans la section suivante, nous présentons comment nous pouvons classer les jeux de données qui ne sont pas linéairement séparables.

1.2 Classification SVM non linéaire

Pour gérer les jeux de données qui ne sont pas linéairement séparables avec un classifieur SVM linéaire, une solution consiste à ajouter des nouvelles données à l'ancien jeu de données. Le nouveau jeu de données est généralement linéairement séparable. L'ajout des nouvelles données se fait en ajoutons pour chaque donnée une nouvelle donnée calculée en utilisant par exemple une fonction polynomiale du second degré. Cette technique est efficace avec un faible degré polynomiale pour les données simples mais elle n'est pas efficace pour les données complexes avec un faible degré polynomial. En effet, pour qu'on puisse traiter les données complexes nous avons besoin d'avoir un degré polynomial élevé [Géron (2017)]. Mais cette solution ralentit le traitement. Pour résoudre ce problème, une solution consiste à utiliser la technique de "*noyau*" qui permet d'avoir plus de données comme la technique précédente mais sans ajouter les données réellement. Cela permet de diminuer le temps de traitement.

Une autre solution pour la classification des jeux de données non linéaire consite à ajouter des données calculées à travers une fonction de similarité qui calcule la similarité entre chaque donnée du jeu de données et un *"point de repère"* . Un exemple de fonction de similarité est "la fonction de base radiale gaussienne" (en anglais **R**adial **B**asic **F**unction (**RBF**)) définie à travers l'équation 13.

$$\phi(x, l) = exp(-\gamma \|x - l\|^2) \tag{13}$$

avec x est l'observation, l est un point de repère et γ est un hyperparamètre. L'emplacement de chaque donnée du jeu de données est définie comme un point de repère.

2 Forêts aléatoires

En apprentissage automatique, il existe des méthodes d'apprentissage appelées "*ensemblistes*". Elles combinent les prédictions de plusieurs modèles qui sont construites avec un algorithme d'apprentissage donné. Il existe deux types des méthodes ensemblistes :

- Les méthodes ensemblistes hétérogènes combinent un ensemble de modèles produit par des algorithmes différents sur un même jeu de données.
- Les méthodes ensemblistes homogènes combinent un ensemble de modèles produits par un même algorithme d'apprentissage sur un même jeu de données.

Ainsi nous pouvons distinguer une autre classification pour ces méthodes :

- Les méthodes ensemblistes de calcul de la moyenne qui consistent à construire plusieurs modèles indépendamment puis de faire la moyenne de leurs prédictions.

En général le modèle combiné est meilleur que les autres modèles. L'algorithme des forêts aléatoires est un algorithme ensembliste qui s'appuie sur ce principe.

- Les méthodes ensemblistes de boosting [Freund et al. (1996); Schapire (2003)] qui consistent à construire plusieurs modèles séquentiellement. Le principe de ces méthodes consiste : (*i*) à construire un premier modèle et l'évaluer ; (*ii*) à augmenter les poids des observations qui induisent en erreur afin de leur donner plus d'importance ; (*iii*) à construire un deuxième modèle avec le nouveau jeu de données pondéré ; (*iv*) à construire plusieurs modèles qui s'appuient chacun sur les erreurs effectuées par le modèle précédent et à pondérer à chaque fois le nouveau jeu de données et finalement à combiner ces modèles afin d'obtenir un modèle plus performant. AdaBoost [Freund et al. (1996); Freund and Schapire (1997)] est un algorithme de boosting qui s'appuie sur ce principe.

Dans cette section, nous allons détailler le principe des forêts aléatoires qui est une méthode d'apprentissage ensembliste homogène. Elle se présente comme l'une des méthodes les plus puissantes [Breiman (2001)]. Elle permet de produire un ensemble de modèles en se basant sur un même algorithme qui est les "*arbres de décision*". En effet, cet ensemble d'arbres de décision est entrainé sur différents sous-ensembles du jeu d'entrainement. Pour déterminer la classe d'une nouvelle observation x, il suffit d'obtenir les prédictions de chacune des arbres puis de choisir la classe qui a le plus grand nombre de votes.

Puisque les arbres de décision sont les composants fondamentaux de cette méthode, nous décrivons dans la suite leurs principes.

2.1 Arbres de décision

Ce sont des algorithmes d'apprentissage automatique permettant de résoudre des problèmes de classification et de régression. Ils sont très puissants et ils s'adaptent facilement à des jeux de données complexes [Géron (2017)]. Aussi, ils sont simples, faciles à comprendre et à interpréter par les opérateurs humains puisqu'ils sont capables de retracer les questions à poser pour effectuer leurs choix.

La figure 54 montre un exemple d'un arbre de décision pour la classification des fleurs IRIS en utilisant le jeu de données d'IRIS [Fisher (1936)] introduit dans le chapitre 2. Le sens de cet arbre est le suivant : si nous voulons ranger une nouvelle fleur d'IRIS, nous commençons par vérifier la largeur du pétale du noeud racine de l'arbre.

- Si la largeur du pétale de cette fleur est inférieure ou égale à *0.8 cm* alors nous descendons vers le nœud coloré en orangé qui représente un noeud terminal. Il est appelé "*feuille*" puisqu'il n'a pas de nœuds fils. Afin de déterminer la classe prédite, il suffit de voir l'attribut *"class"*. Dans ce cas la nouvelle fleur est un "IRIS setosa".

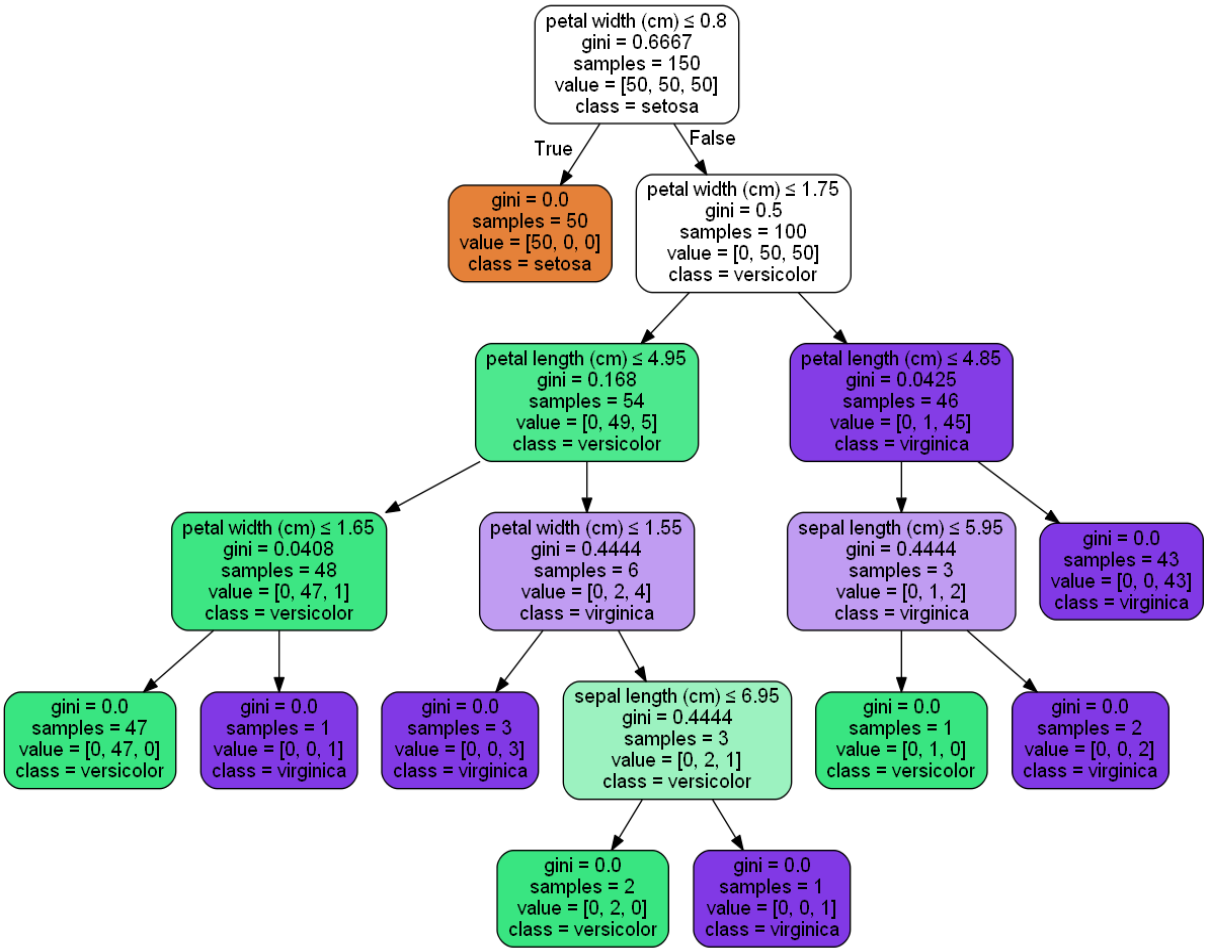


FIGURE 54 – Arbre de décision IRIS.

– Si la largeur du pétale de cette fleur est supérieure à *0.8 cm* alors nous descendons vers le nœud inférieur à droite coloré en blanc. Ce nœud n'est pas un nœud terminal. Pour cette raison nous vérifions la condition "la largeur de pétale est elle inférieure ou égale à *1.75 cm* ?". Si la condition est vrai alors nous descendons vers le nœud inférieur à gauche (coloré en vert). Dans ce cas la fleur est un "*IRIS versicolor*". Sinon, nous descendons vers le nœud inférieur à droite qui n'est pas un nœud terminal. Dans ce cas la fleur est un "*IRIS verginica*". Nous continuons ce raisonnement jusqu'à arriver aux feuilles qui sont responsables de la prise de décision.

L'attribut "*samples*" représente le nombre d'observations du jeu d'entrainement qui ont vérifié (ou non) la condition du noeud du niveau précédent. A titre d'exemple, dans le nœud présenté en orangé, il y 50 observations qui vérifient la condition "largeur du pétale est inférieur ou égale à *0.8* cm". L'attribut "*value*" indique la répartition de ces observations dans les différentes classes *[x, y, z]* avec x est le nombre d'observations de la classe "*setosa*", y est le nombre d'observations de la classe "*versicolor*" et z est le nombre d'observations de la classe "*virginica*". Dans l'exemple du noeud présenté en orangé, toutes

les observations retenues appartiennent à la classe "*setosa*".

L'attribut "*gini* de chaque nœud mesure son impureté. Cette notion a été introduite par Breiman en 1984 [Breiman (1984)]. Elle caractérise le degré de mélange des classes dans un même nœud. En effet, le fonctionnement d'un arbre de décision se base sur l'ordre des attributs selon leur emplacement dans chaque nœud. Le choix des attributs se fait par rapport à la capacité de séparer les différentes observations en classes homogènes. Cette capacité est calculée à travers une mesure d'impureté [Breiman (1996b)] qui peut être la mesure « *gini* » (présenté à travers l'équation 14) ou la mesure d' «*entropie*»(décrite dans l'équation 15).

Mesure gini :

$$\mathrm{G_i} = \sum_{K=1}^{n} (p_{i,k})^2 \tag{14}$$

Mesure d'entropie :

$$\mathrm{Entropie_i} = \sum_{K=1}^{n} p_{i,k} log(p_{i,k}) \tag{15}$$

avec n est le nombre de classes et $p_{i,k}$ est le pourcentage d'observations de la classe K parmi toutes les observations d'entrainement dans le $i^{ème}$ nœud. Une fois cette mesure est calculée pour les différents attributs, nous choisissons l'attribut qui a la mesure la plus élevée.

Plusieurs algorithmes exploitent les arbres de décision, parmi ces algorithmes nous citons : ID3 (Iteractive Dichotomizer 3) qui est développé en 1986 par « *Ross Quinlan* » [Quinlan (1986)], *C4.5* qui est extension de *ID3* [Quinlan (1993)], *C5.0* est le successeur de *C4.5* et *CART* (Classification and Regression Tree (CART)). Scikit-learn utilise une version optimisé de CART : *Decision Tree Classifier* pour les classifications multi-classes et *Decision Tree Regressor* pour les problèmes de régression [Pedregosa et al. (2011)].

Comme nous venons d'indiquer au début de cette section, une forêt aléatoire est un ensemble d'arbres de décision qui utilise le même algorithme d'entrainement sur des sous-ensembles différents du jeu d'entrainement. Le fait d'utiliser le même classifieur (arbre de décision) sur différents sous-ensembles du jeu d'entrainement, sélectionnés aléatoirement, permet d'avoir des résultats diversifiés. Il suffit d'obtenir les prédictions de chacun des arbres puis de choisir la classe ayant le plus grand nombre de votes.

Il existe deux méthodes de sélection aléatoire : le « bagging »[Breiman (1996a)] et le « pasting » [Breiman (1999)]. Le principe de ces deux méthodes est détaillé dans la sous

section suivante.

2.2 Bagging et Pasting

Lorsque la sélection des sous-ensembles du jeu d'entrainement s'effectue **avec remise**, la méthode est appelée « bagging » (l'abréviation d'agrégation bootstrap)[Breiman (1996a); Quinlan et al. (1996)] et lorsque la sélection se fait **sans remise**, la méthode est appelée « pasting »[Breiman (1999)]. La figure 55 présente un exemple d'un jeu d'entrainement initial qui comporte 4 balles rouges, 3 vertes et 2 bleues. Après un tirage aléatoire avec remise, le sac 1 contient 2 balles bleues et 4 rouges et le sac 2 contient deux balles bleues et 2 vertes. Puisque le tirage est avec remise, les deux balles bleues sont tirées dans le sac 1 et le sac 2.

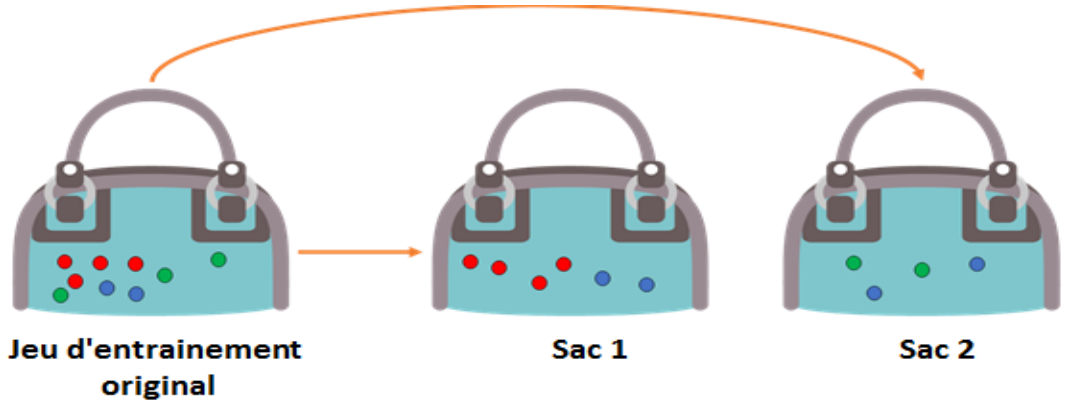


FIGURE 55 – La méthode bagging.

Suite à un tirage sans remise (figure 56), le sac 1 contient 4 balles rouges et 2 bleues alors que le sac 2 contient 2 balles vertes. En effet, le sac 2 ne peut pas contenir ni des balles rouges ni des balles bleues car elles sont utilisées dans le sac 1.

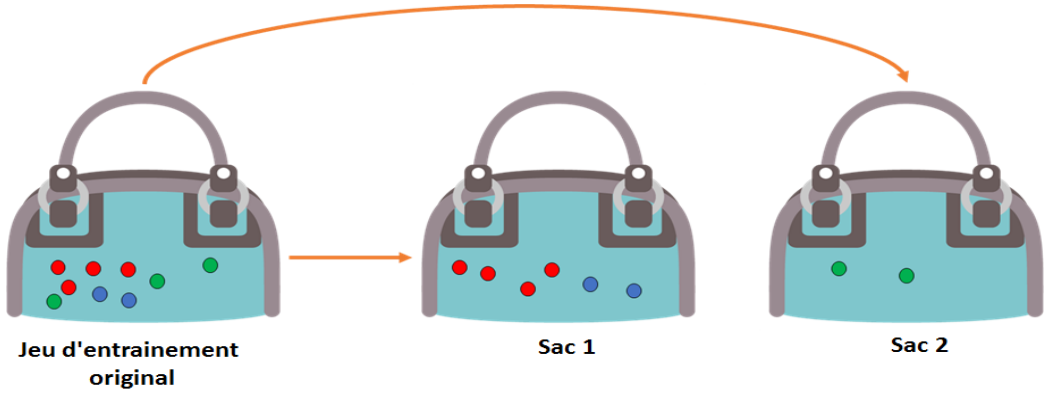


FIGURE 56 – La méthode pasting.

L'objectif principal de ces deux méthodes est de prélever plusieurs fois des observations du jeu d'entrainement et de les utiliser pour l'entrainement de plusieurs prédicteurs.

Les arbres de décision représentant une forêt aléatoire sont généralement entrainés selon la méthode *bagging* mais aussi ils peuvent être entrainés selon la méthode *pasting*.

Pour entrainer une forêt aléatoire avec scikit-learn, il existe deux solutions : (*a*) utiliser la classe *Bagging Classifier* pour l'échantillonnage et ensuite la classe *Decision Tree Classifier* pour l'entrainement. (b) utiliser directement la classe *Random Forest Classifier* qui est plus optimisée [Géron (2017)].

3 Réseaux de neurones

Les réseaux de neurones sont un ensemble de neurones interconnectés. Ces interconnexions peuvent avoir plusieurs architectures. Dans la suite nous présentons le perceptron [Rosenblatt (1957)] qui est l'une des premières architectures les plus simples. Nous présentons ensuite les perceptrons multicouches [Rosenblatt (1961)].

3.1 Perceptron simple

Le perceptron est l'une des architectures des réseaux de neurones les plus simples qui est inventée par Rosenblatt en 1957 [Rosenblatt (1957)]. Cette architecture est présentée dans la figure 57.

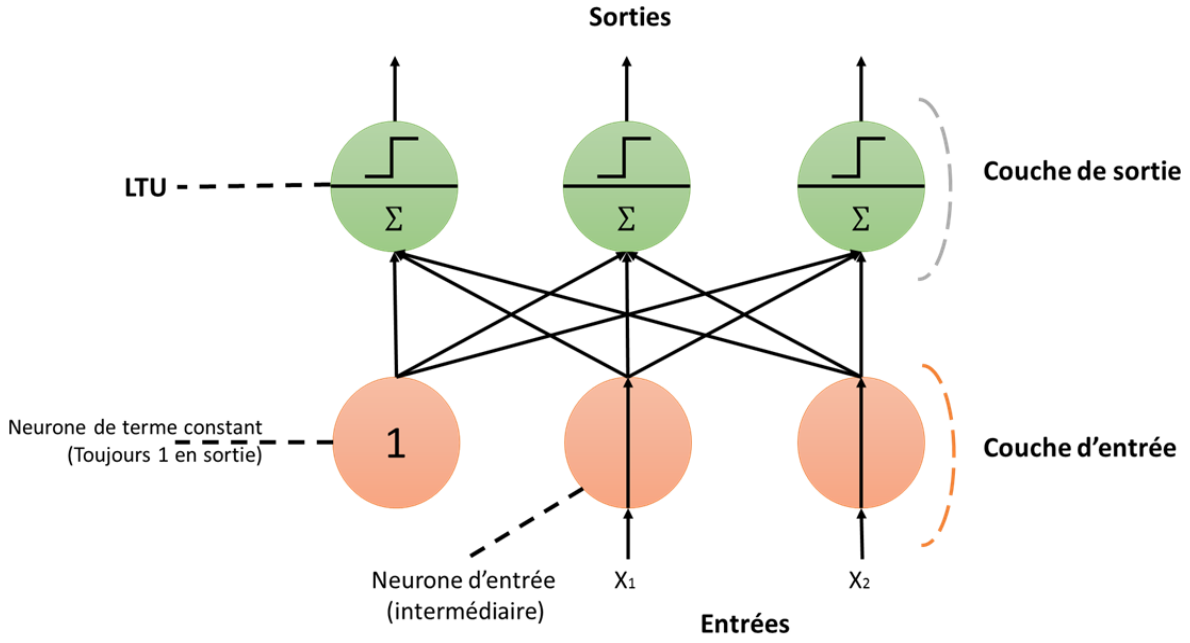


FIGURE 57 – Un perceptron.

Elle est formée de deux couches : une couche d'entrée et une couche de sortie. La couche

d'entrée représente les neurones d'entrée qui permettent de sortir les mêmes entrées fournies. La couche de sortie représente un type de neurone artificiel différent des autres, appelé «**U**nité **L**inéaire à **S**euil (**ULS**)» en anglais appelé "*Linear Threshold Unit(LTU)*"(voir la figure 58). Chaque neurone ULS est connecté à toutes les neurones d'entrées et aussi à un neurone particulier appelé "*neurone de biais*" dont la sortie est toujours égale à 1 (neurone de terme constant). Les neurones de la couche d'entrée sont connectés à toutes les neurones ULS à travers des poids W_{nm} entre les n neurones d'entrée les m neurones de sortie.

L'ULS sert à calculer une somme pondérée de toutes les entrées puis il applique une "fonction d'activation" (appelée aussi une "fonction de transfert") pour cette somme. La fonction *Heaviside* présentée à travers l'équation 16 et la fonction *sign* décrite à travers l'équation 17 sont deux exemples des fonctions de transfert.

$$Heaviside(z) = \begin{cases} 0 & \text{Si } z < 0 \\ 1 & \text{sinon } z >= 1 \end{cases} \tag{16}$$

$$Sign(z) = \begin{cases} -1 & \text{Si } z < 0 \\ 0 & \text{Si } z = 0 \\ 1 & \text{Si } z > 0 \end{cases} \tag{17}$$

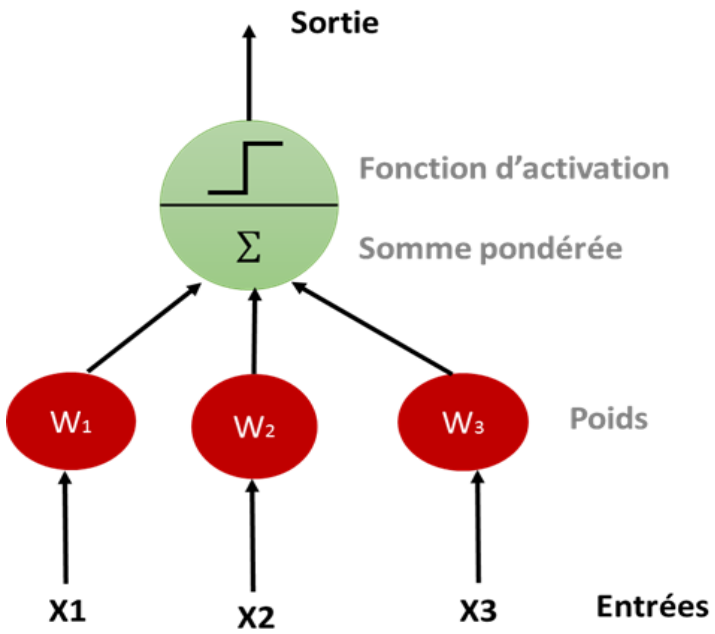


FIGURE 58 – Unité linéaire à seuil.

Pour entrainer un perceptron c'est à dire déterminer les poids de connexion, le principe se base sur la règle de Hebb proposée par "*Donald Hebb*" [Hebb et al. (1949)] qui suggère que "si un neurone biologique déclenche un autre alors la connexion entre ces

deux neurones se renforce". Cela veut dire que chaque neurone de sortie qui produit une prédiction juste, ses connexions liées aux neurones d'entrée sont renforcées. Donc pour l'entrainement du perceptron en utilisant un jeu d'entrainement, l'algorithme est itératif car à chaque instance d'entrainement, il effectue des prédictions et pour chaque neurone de sortie qui produit une prédiction juste il renforce ses connexions suivant la règle décrite dans l'équation 18. Cette règle prend en compte l'erreur commise pendant l'apprentissage.

$$W_{ij} = W_{ij} + \eta(y_i - Y_i)x_i \tag{18}$$

— W_{ij} représente le poids de la connexion entre le $i^{ème}$ neurone de la couche entrée et le $j^{ème}$ neurone de la couche de sortie ;

— x_i correspond à la $i^{ème}$ valeur d'entrée de l'instance d'entrainement courante ;

— y_j correspond à la sortie prédite du $j^{ème}$ neurone de sortie pour l'instance d'entrainement courante ;

— Y_j correspond à la sortie réelle du $j^{ème}$ neurone de sortie pour l'instance d'entrainement courante ;

— η est le taux d'apprentissage dans le réseau.

Le perceptron simple permet de résoudre uniquement les problèmes qui sont linéairement séparables. Pour cette raison, il est incapable d'apprendre des motifs complexes. En 1969, Minsky ont dévoilé plusieurs faiblesses des perceptrons simples qui sont incapables de résoudre des problèmes simples comme le problème de classification de "OU exclusif". Pour résoudre ce problème, des chercheurs ont proposé une autre architecture des résaux de neurones appelée "les perceptrons multicouches PMC"(en anglais Multilayer Perceptron).

3.2 Perceptron multicouche (PMC)

L'architecture PMC [Rosenblatt (1961)] est constituée d'une couche d'entrée, d'une ou plusieurs couches ULS qui sont appelées des couches cachées et d'une dernière couche ULS qui présente la couche de sortie. L'architecture est présentée dans la figure 59. La couche d'entrée et la ou les couches cachées comprennent un "*neurone de biais*" dont la sortie est toujours est égale à 1.

L'entrainement du PMC consiste à donner au réseau les instances d'entrainements et ensuite de lui demander de modifier ses poids afin de retrouver les sorties réelles de ces instances de traitement. L'algorithme d'entrainement de PMC est appelé "Rétropropaga-

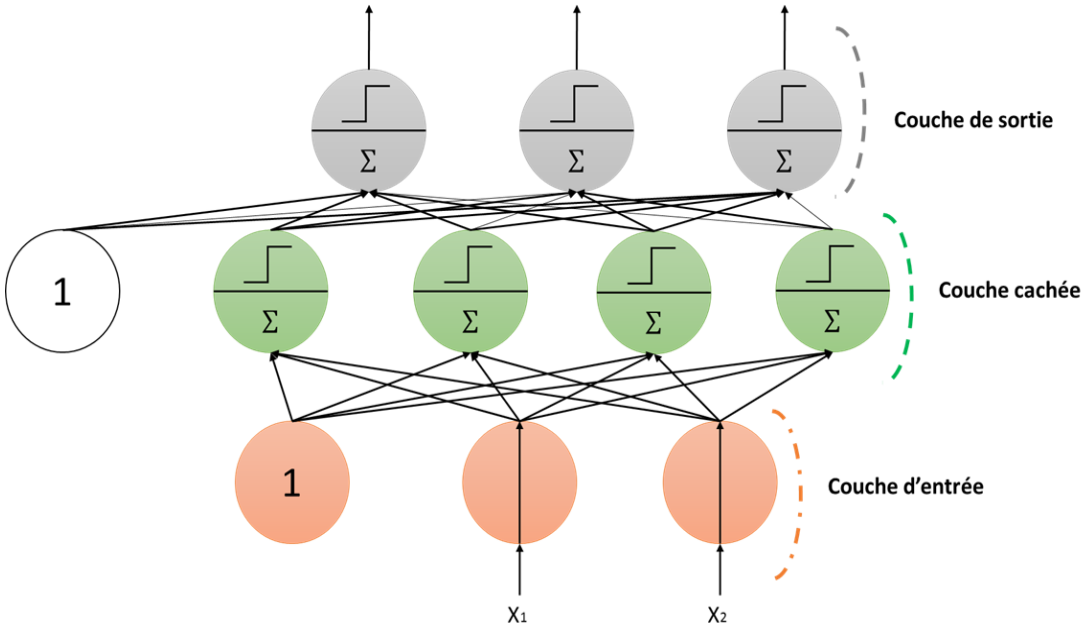


FIGURE 59 – L'architecture du PMC.

tion" [Rumelhart et al. (1985)]. Pour chaque instance d'entrainement et après avoir affecté des poids aléatoires dans le résau, il consiste à :

a) Calculer la sortie de chaque neurone dans chaque couche consécutive (de la couche d'entrée vers la couche de sortie) ;

b) Calculer l'erreur du réseau en comparant la sortie calculée (c'est-à-dire la sortie prédite) avec la sortie réelle ;

c) Pour chaque neurone de sortie, calculer la contribution à l'erreur des neurones des couches cachées qui précédent le neurone de sortie (il s'agit d'un passage en arrière : de la couche de sortie vers la couche d'entrée). Cette étape mesure le "gradient d'erreur" pour tous les connexions du réseau.

d) Ajuster les poids de connexions entre les couches en utilisant la méthode de "descende de gradient" et les gradients d'erreurs calculés pour chaque connexion dans l'étape précédente. La descente de gradient est un algortithme d'optimisation qui consiste à corriger petit à petit les poids de connexions afin de minimiser l'erreur dans le réseau. La modification des poids se fait dans le sens inverse du gradient d'erreur.

Les PMC peuvent être utilisés pour la classification binaire et aussi pour la classification multi-classes. Dans le cas d'une classification multi-classes les fonctions d'activations de la couche de sortie sont remplacées par une fonction partagé "Softmax". Cette fonction calcule pour chaque observation x et pour les k classes les probabilités P_k d'appartenance

de x au différents classe k (voir l'équation 19). La sortie de chaque neurone de la couche de sortie est la probabilité estimée pour la classe qui la représente. Pour l'observation x le perceptron multicouche prédit alors la classe ayant la plus grande probabilité.

$$\mathrm{P_k} = \frac{exp(\mathrm{z_k})}{\sum_{i=1}^{k} exp(\mathrm{z_i})} \tag{19}$$

avec K est le nombre de classes et $\mathrm{z_k}$ est la somme pondérée calculée par le neurone de sortie de la classe k.

4 La régression logistique

L'algorithme de *régression logistique* [Berkson (1944, 1951)] appelé aussi *régression logit* peut être utilisé pour la régression (estimer la valeur numérique de la variable cible) et aussi pour la classification binaire. Il permet d'estimer la probabilité d'appartenance d'une observation x à une classe particulière. Si la probabilité calculée est supérieure ou égale à 50% alors le modèle range l'observation x à cette classe (classe positive), sinon il range l'observation à l'autre classe (classe négative).

Pour l'estimation des probabilités, le modèle de régression logistique calcule premièrement une somme pondérée pour toutes les variables d'entrées (notée s) et deuxièmement il calcule une fonction logistique de cette somme pondérée (équation 21). Cette fonction logistique (noté σ) est une fonction "sigmoïde" qui est définie dans l'équation 20 et représentée dans la figure 60.

$$\sigma(\mathrm{s}) = \frac{1}{1 + exp(-s)} \tag{20}$$

$$p = \sigma(w^T * x) \tag{21}$$

Avec w est le vecteur de pondérations des différentes variables de l'observation x, w^T est le transposé du vecteur w.

Après le calcul de la probabilité qu'une observation x appartienne à la classe positive, le modèle peut facilement ranger x soit à la classe positive (si p >=0.5) soit à la classe négative (si p <0.5).

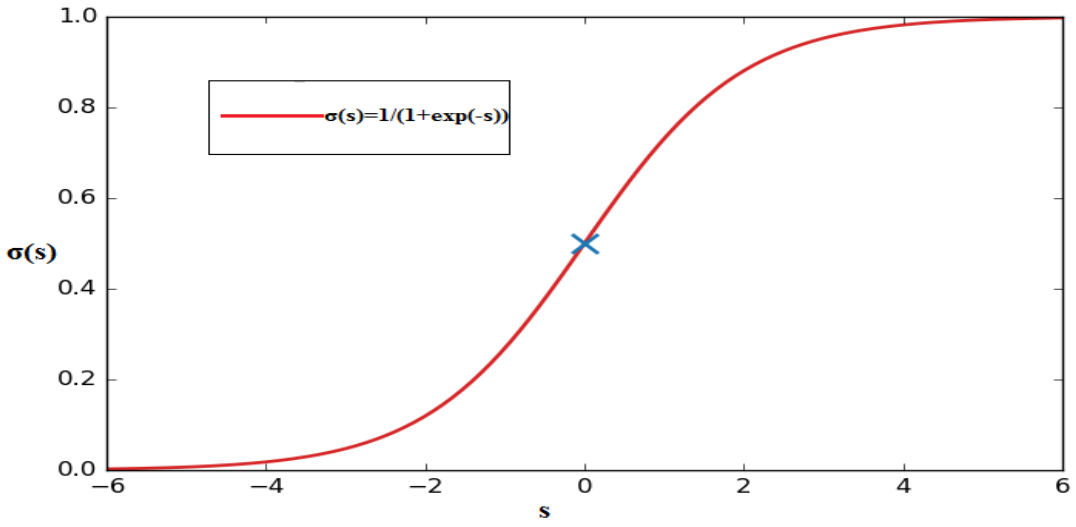


FIGURE 60 – Fonction logistique.

Annexe B

1 Algorithme de génération automatique des parties problèmes

Algorithme 1 Algorithme de génération automatique des parties problèmes

Entrées: $O, I, TempsCycleAPI$ {O est un tableau d'objets contenant les signaux de n actionneurs, I est un tableau d'objets contenant les signaux de m capteurs et *TempsCycleAPI* est le temps du cycle de l'API}

Sorties: *PartieProbleme*

1: $EVT \leftarrow$""
2: $tampon \leftarrow$""
3: $PartieProbleme \leftarrow$ [] {*La partie problème*}
4: $TempsCyclecourant \leftarrow 0$
5: $TempsCycleprécédent \leftarrow 0$
6: **pour** chaque cycle d'API **faire**
7: $SignatureBinaire \leftarrow$""
8: ***Etape 0 : Formulation de la SignatureBinaire***
9: **pour** $i = 1$, i++, $i <= n$ **faire**
10: $SignatureBinaire \leftarrow SignatureBinaire + str(O[i])$ {*Convertir les n éléments de O et les assigner à la Signature Binaire*}
11: **fin pour**
12: **pour** $i = 1$, i++, $i <= m$ **faire**
13: $SignatureBinaire \leftarrow SignatureBinaire + str(I[i])$ {*Convertir les m éléments de I et les assigner à la Signature Binaire*}
14: **fin pour**
15: $SignatureBinaire \leftarrow SignatureBinaire + Remplacer("True", 1)$
16: $SignatureBinaire \leftarrow SignatureBinaire + Remplacer("False", 0)$
17: $TempsCyclecourant \leftarrow TempsCyclecourant + TempsCycleAPI$
18: **si** tampon != SignatureBinaire **alors**
19: ***Etape 1 : Formulation de l'événement référent (ER)***
20: **si** tampon == "" **alors**
21: $tampon \leftarrow SignatureBinaire$
22: $TempsCycleprécédent \leftarrow TempsCyclecourant$
23: $EVT \leftarrow$""

24: **sinon**
25: ***Etape 2 : Formulation de l'événement contraint (EC)***
26: **pour** i de 1 à longueur de la SignatureBinaire **faire**
27: **si** (tampon[i] !=SignatureBinaire[i]) **alors**
28: **si** (tampon[i]=="0" et SignatureBinaire[i]=="1") **alors**
29: $EVT \leftarrow EVT + "R"$ {R représente un front montant}
30: $EVT \leftarrow EVT + Appoint(i)$ {Appoint(i) permet d'associer pour chaque élément i le nom du capteur ou de l'actionneur qui lui correspond}
31: $EC \leftarrow EVT$
32: **sinon**
33: $EVT \leftarrow EVT + "F"$ {F représente un front descendant}
34: $EVT \leftarrow EVT + Appoint(i)$ {Appoint(i) permet d'associer pour chaque élément i le nom du capteur ou de l'actionneur qui lui correspond}
35: $EC \leftarrow EVT$
36: **fin si**
37: **fin si**
38: **fin pour**
39: $tampon \leftarrow SignatureBinaire$
40: ***Etape 3 : Formulation de la CT***
41: $BI \leftarrow (TempsCyclecourant - TempsCycleprécédent) - \Delta T$
42: $BS \leftarrow (TempsCyclecourant - TempsCycleprécédent) + \Delta T$
43: **si** ER="" **alors**
44: $ER \leftarrow "IN"$
45: $BI \leftarrow 0$
46: $BS \leftarrow 0$
47: **fin si**
48: ***Etape 4 : Formulation d'un nouveau triplet***
49: T=new Triplet(ER, EC, BI, BS)
50: ***Etape 5 : Formulation d'une nouvelle partie problème***
51: $PartieProbleme.ajouter(T)$
52: $TempsCycleprécédent \leftarrow TempsCyclecourant$
53: $ER \leftarrow EC$
54: **fin si**
55: **fin si**
56: **fin pour**

Bibliographie

Aamodt, A. and Plaza, E. (1994). Case-based reasoning : Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1) :39–59.

Abele, S. and Weyrich, M. (2017). Decision support for joint test and diagnosis of production systems based on a concept of shared knowledge. *IFAC-PapersOnLine*, 50(1) :15227–15232.

Aha, D. W., Breslow, L. A., and Muñoz-Avila, H. (2001). Conversational case-based reasoning. *Applied Intelligence*, 14(1) :9–32.

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1) :37–66.

Al-Dahidi, S., Di Maio, F., Baraldi, P., and Zio, E. (2015). Ensemble clustering for fault diagnosis in industrial plants. *Chemical Engineering Transactions*, 43 :1225–1230.

Aldrich, C. and Auret, L. (2013). *Unsupervised process monitoring and fault diagnosis with machine learning methods*. Springer.

Ali, R., Khatak, A. M., Chow, F., and Lee, S. (2018). A case-based meta-learning and reasoning framework for classifiers selection. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, page 31. ACM.

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Althoff, K.-D. (2001). Case-based reasoning. In *Handbook of Software Engineering and Knowledge Engineering : Volume I : Fundamentals*, pages 549–587. World Scientific.

Antsaklis, P. J. and Koutsoukos, X. D. (2003). Hybrid systems : Review and recent progress. *Software-Enabled Control : Information Technology for Dynamical Systems*, 273 :298.

Araújo, A. F., Varela, M. L., Gomes, M. S., Barreto, R. C., and Trojanowska, J. (2018). Development of an intelligent and automated system for lean industrial production, adding maximum productivity and efficiency in the production process. In *Advances in Manufacturing*, pages 131–140. Springer.

Arlot, S., Celisse, A., et al. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, 4 :40–79.

Azadeh, A., Saberi, M., Kazem, A., Ebrahimipour, V., Nourmohammadzadeh, A., and Saberi, Z. (2013). A flexible algorithm for fault diagnosis in a centrifugal pump with corrupted data and noise based on ann and support vector machine with hyper-parameters optimization. *Applied Soft Computing*, 13(3) :1478–1485.

Bakkari, M., Rachidi, A., and Khatory, A. (2015). Evolution of automated production systems in smes : what are the consequences for the employees ? In *Xème Conférence Internationale : Conception et Production Intégrées*.

Barbaros, V., van Hoof, H., Abdolmaleki, A., and Megerl, D. (2018). Eager and memory-based non-parametric stochastic search methods for learning control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE.

Behbahani, M., Saghaee, A., and Noorossana, R. (2012). A case-based reasoning system development for statistical process control : Case representation and retrieval. *Computers & Industrial Engineering*, 63(4) :1107–1117.

Ben Hmida, F. B. (2013). *Evaluation des performances des systemes multi-agents*. PhD thesis, UNIVERSITÉ BORDEAUX I ; UNIVERSITÉ DE LA MANOUBA, TUNISIE.

Ben Hmida, F. B., Chaari, W. L., and Tagina, M. (2015). Evaluation of organization in multiagent systems for fault detection and isolation. In *Agent and Multi-Agent Systems : Technologies and Applications*, pages 69–79. Springer.

Ben Rabah, N., Saddem, R., Hmida, F. B., Carre-Menetrier, V., and Tagina, M. (2017a). Intelligent case based decision support system for online diagnosis of automated production system. In *Journal of Physics : Conference Series*, volume 783, page 012009. IOP Publishing.

Ben Rabah, N. B., Saddem, R., Hmida, F. B., Carre-Menetrier, V., and Tagina, M. (2017b). Approche originale utilisant le raisonnement à partir de cas pour le diagnostic en ligne des systèmes automatisés de production.

Ben Rabah, N. B., Saddem, R., Hmida, F. B., Carré-Ménétrier, V., and Tagina, M. (2017c). Automatic acquisition and update of a causal temporal signatures base-for faults diagnosis in automated production systems. In *ICINCO (1)*, pages 262–269.

Bergmann, R. (2002). *Experience management : foundations, development methodology, and internet-based applications*. Springer-Verlag.

Bergmann, R. and Gil, Y. (2014). Similarity assessment and efficient retrieval of semantic workflows. *Information Systems*, 40 :115–127.

Berkson, J. (1944). Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227) :357–365.

Berkson, J. (1951). Why i prefer logits to probits. *Biometrics*, 7(4) :327–339.

Berman, A. F., Maltugueva, G. S., and Yurin, A. Y. (2018). Application of case-based reasoning and multi-criteria decision-making methods for material selection in petrochemistry. *Proceedings of the Institution of Mechanical Engineers, Part L : Journal of Materials : Design and Applications*, 232(3) :204–212.

Bertrand, O. (2009). *Détection d'activités par un système de reconnaissance de chroniques et application au cas des simulations distribuées HLA*. PhD thesis, Paris 13.

Bertrand, O., Carle, P., and Choppy, C. (2007). Chronicle modelling using automata and colored petri nets. In *The 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 229–234.

Bisson, G. (2000). La similarité : une notion symbolique/numérique. *Apprentissage symbolique-numérique*, 2 :169–201.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine learning*, 94(2) :133–149.

Boufaied, A. (2003). *Contribution à la surveillance distribuée des systèmes à événements discrets complexes*. PhD thesis, Université Paul Sabatier-Toulouse III.

Boulanger, J.-L. (2013). *Safety of Computer Architectures*. John Wiley & Sons.

Breil, R., Delahaye, D., Lapasset, L., and Féron, E. (2017). Multi-agent systems to help managing air traffic structure. (Preprint) :1–30.

Breiman, L. (1984). Classification and regression trees.

Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2) :123–140.

Breiman, L. (1996b). Some properties of splitting criteria. *Machine Learning*, 24(1) :41–47.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine learning*, 36(1-2) :85–103.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1) :5–32.

Brighton, H. and Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2) :153–172.

Brown, D., Aldea, A., Harrison, R., Martin, C., and Bayley, I. (2017). Temporal case-based reasoning for type 1 diabetes mellitus bolus insulin decision support. *Artificial intelligence in medicine*.

Carle, P., Choppy, C., and Kervarc, R. (2011). Behaviour recognition using chronicles. In *2011 Fifth IEEE International Conference on Theoretical Aspects of Software Engineering*, pages 100–107. IEEE.

Cassandras, C. G. and Lafortune, S. (2009). *Introduction to discrete event systems*. Springer Science & Business Media.

Cazzolato, M. T., Costa, A. F., Blanco, G., Rodrigues Jr, J. F., Traina, A. J., and Traina Jr, C. (2016). Fire detection from social media images by means of instance-based learning. In *Enterprise Information Systems : 17th International Conference, ICEIS 2015, Barcelona, Spain, April 27-30, 2015, Revised Selected Papers*, volume 241, page 23. Springer.

Cellier, F. E. and Kofman, E. (2006). *Continuous system simulation*. Springer Science & Business Media.

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2) :1.

Chen, S., Yi, J., Jiang, H., and Zhu, X. (2016). Ontology and cbr based automated decision-making method for the disassembly of mechanical products. *Advanced Engineering Informatics*, 30(3) :564–584.

Chougule, R., Rajpathak, D., and Bandyopadhyay, P. (2011). An integrated framework for effective service and repair in the automotive domain : An application of association mining andcase-based-reasoning. *Computers in Industry*, 62(7) :742–754.

Choy, K. L. T., Siu, K. Y. P., Ho, T. S. G., Wu, C., Lam, H. Y., Tang, V., and Tsang, Y. P. (2018). An intelligent case-based knowledge management system for quality improvement in nursing homes. *VINE Journal of Information and Knowledge Management Systems*, 48(1) :103–121.

Chu, M., Liu, X., Gong, R., and Liu, L. (2018). Multi-class classification method using twin support vector machines with multi-information for steel surface defects. *Chemometrics and Intelligent Laboratory Systems*, 176 :108–118.

Cojan, J. and Lieber, J. (2014). Applying belief revision to case-based reasoning. In *Computational Approaches to Analogical Reasoning : Current Trends*, pages 133–161. Springer.

Combacau, M., Berruet, P., Zamai, E., Charbonnaud, P., and Khatab, A. (2000). Supervision and monitoring of production systems. *IFAC Proceedings Volumes*, 33(17) :849–854.

Commission, I. E. et al. (1993). Plcs–part 3 : programming languages.

Cordier, M.-O. and Dousson, C. (2000). Alarm driven monitoring based on chronicles. *IFAC Proceedings Volumes*, 33(11) :291–296.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1) :21–27.

Cram, D., Mathern, B., and Mille, A. (2012). A complete chronicle discovery approach : application to activity analysis. *Expert Systems*, 29(4) :321–346.

Danancher, M., Roth, M., Lesage, J.-J., and Litz, L. (2011). Diagnostic des sed basé sur un modele : trois approches évaluées sur une même étude de cas. In *4èmes Journées Doctorales/Journées Nationales MACS (JD-JN-MACS'11)*, pages 165–170.

Darkhovski, B. and Staroswiecki, M. (2003). A game-theoretic approach to decision in fdi. *IEEE Transactions on Automatic Control*, 48(5) :853–858.

de Haro-García, A., Pérez-Rodríguez, J., and García-Pedrajas, N. (2018). Combining three strategies for evolutionary instance selection for instance-based learning. *Swarm and Evolutionary Computation*.

De Kleer, J. and Williams, B. C. (1987). Diagnosing multiple faults. *Artificial intelligence*, 32(1) :97–130.

De Loor, P., Bénard, R., and Chevaillier, P. (2011). Real-time retrieval for case-based reasoning in interactive multiagent-based simulations. *Expert Systems with Applications*, 38(5) :5145–5153.

De Mantaras, R. L., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., T COX, M., Forbus, K., et al. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3) :215–240.

Debouk, R., Lafortune, S., and Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems*, 10(1-2) :33–86.

Dendani, N., Khadir, M., and Guessoum, S. (2012). Hybrid approach for fault diagnosis based on cbr and ontology : using jcolibri framework. In *Complex Systems (ICCS), 2012 International Conference on*, pages 1–8. IEEE.

Derbel, H. (2009). *Diagnostic à base de modèles des systèmes temporisés et d'une sous-classe de systèmes dynamiques hybrides*. PhD thesis, Université Joseph-Fourier-Grenoble I.

Diaz-Agudo, B. and Goel, A. K. (2017). Report on the 24th international conference on case-based reasoning research and development (iccbr-2016). *AI Magazine*, 38(4) :89–90.

Donini, F. M., Lenzerini, M., Nardi, D., and Schaerf, A. (1996). Reasoning in description logics. *Principles of knowledge representation*, 1 :191–236.

Dou, D. and Zhou, S. (2016). Comparison of four direct classification methods for intelligent fault diagnosis of rotating machinery. *Applied Soft Computing*, 46 :459–468.

Dousson, C., Clerot, F., and Fessant, F. (2008). Method for the machine learning of frequent chronicles in an alarm log for the monitoring of dynamic systems. US Patent 7,388,482.

Dousson, C., Gaborit, P., and Ghallab, M. (1993). Situation recognition : representation and algorithms. In *IJCAI*, volume 93, pages 166–172.

Dousson, C. and Ghallab, M. (1994). Suivi et reconnaissance de chroniques. *Revue d'intelligence artificielle*, 8(1) :29–61.

Dousson, C., Pentikousis, K., Sutinen, T., and Makela, J. (2007). Chronicle recognition for mobility management triggers. In *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pages 305–310. IEEE.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Efraim, T., Jay, E. A., Liang, T.-P., and McCarthy, R. (2001). Decision support systems and intelligent systems. *Upper Saddle River, NK : Prentice Hall*.

Erard, P.-J. and Déguénon, P. (1996). *Simulation par événements discrets*. PPUR presses polytechniques.

Fakhfakh, O. (2015). *Flow monitoring and diagnosis of cyclic production workshops*. Theses, Ecole Centrale Lille.

Fanti, M. P. and Seatzu, C. (2008). Fault diagnosis and identification of discrete event systems using petri nets. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, pages 432–435. IEEE.

Ferber, J. and Weiss, G. (1999). *Multi-agent systems : an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2) :179–188.

Fournier, O. (2002). *Conception de la commande d'un système automatisé de production : Apport des graphes et de l'ordonnancement cyclique*. PhD thesis, Université de la Réunion.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1) :119–139.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer.

Gacquer, D., Delcroix, V., Delmotte, F., and Piechowiak, S. (2011). Comparative study of supervised classification algorithms for the detection of atmospheric pollution. *Engineering Applications of Artificial Intelligence*, 24(6) :1070–1083.

Gao, Y., Shang, Z., and Kokossis, A. (2009). Agent-based intelligent system development for decision support in chemical process industry. *Expert Systems with Applications*, 36(8) :11099–11107.

Gao, Z., Cecati, C., and Ding, S. X. (2015). A survey of fault diagnosis and fault-tolerant techniques—part i : Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6) :3757–3767.

Géron, A. (2017). *Machine Learning avec Scikit-Learn : Mise en oeuvre et cas concrets*. Hors collection. Dunod.

Groover, M. P. (2007). Automation, production systems, and computer-integrated manufacturing.

Guerraz, B. and Dousson, C. (2004). Chronicles construction starting from the fault model of the system to diagnose. In *International Workshop on Principles of Diagnosis (DX04)*, pages 51–56. Citeseer.

Hackathorn, R. D. and Keen, P. G. (1981). Organizational strategies for personal computing in decision support systems. *MIS quarterly*, pages 21–27.

Hamdi, F. (2010). *Contribution à la synthèse d'observateurs pour les systèmes hybrides*. PhD thesis, Université de Batna 2.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell Labs Technical Journal*, 29(2) :147–160.

Han, T., Jiang, D., Zhao, Q., Wang, L., and Yin, K. (2017). Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control*, page 0142331217708242.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer.

He, K., Jia, M., and Liu, C. (2018a). A review of optimal sensor deployment to diagnose manufacturing systems. *IEEE Access*.

He, Y., Guo, J., and Zheng, X. (2018b). From surveillance to digital twin : Challenges and recent advances of signal processing for industrial internet of things. *IEEE Signal Processing Magazine*, 35(5) :120–129.

Hebb, D. O. et al. (1949). The organization of behavior : A neuropsychological theory.

Hedberg, T., Feeney, A. B., and Camelio, J. (2018). Toward a diagnostic and prognostic method for knowledge-driven decision-making in smart manufacturing technologies. In *Disciplinary Convergence in Systems Engineering Research*, pages 859–873. Springer.

Henriet, J., Leni, P.-E., Laurent, R., and Salomon, M. (2014). Case-based reasoning adaptation of numerical representations of human organs by interpolation. *Expert Systems with Applications*, 41(2) :260–266.

Holsapple, C. W. and Whinston, A. B. (2001). Decision support systems : a knowledge-based approach. *Studies in Informatics and Control*, 10(1) :73–76.

Homayouni, H. and Mansoori, E. G. (2017). A novel density-based ensemble learning algorithm with application to protein structural classification. *Intelligent Data Analysis*, 21(1) :167–179.

Houeland, T. G. and Aamodt, A. (2018). A learning system based on lazy metareasoning. *Progress in Artificial Intelligence*, 7(2) :129–146.

Isermann, R. (2017). Supervision, fault-detection and fault-diagnosis methods–a short introduction. In *Combustion Engine Diagnosis*, pages 25–47. Springer.

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem : A systematic study. *Intelligent data analysis*, 6(5) :429–449.

Jia, F., Lei, Y., Guo, L., Lin, J., and Xing, S. (2018). A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines. *Neurocomputing*, 272 :619–628.

Johansson, J. and Cederfeldt, M. (2012). Interactive case based reasoning through visual representation-supporting the reuse of components in variant-rich products. In *DS 70 : Proceedings of DESIGN 2012, the 12th International Design Conference, Dubrovnik, Croatia*.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning : Trends, perspectives, and prospects. *Science*, 349(6245) :255–260.

Kadri, F. (2014). *Contribution à la conception d'un système d'aide à la décision pour la gestion de situations de tension au sein des systèmes hospitaliers. Application à un service d'urgence.* PhD thesis, Université de Valenciennes et du Hainaut-Cambrésis.

Kim, N. and Wysk, R. A. (2012). Consideration of human operators in designing manufacturing systems. In *Manufacturing System*. InTech.

Kiss, I., Genge, B., Haller, P., and Sebestyén, G. (2014). Data clustering-based anomaly detection in industrial control systems. In *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*, pages 275–281. IEEE.

Kolodner, J. (1993). Morgan kaufmann publishers inc. *San Mateo, California*.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial intelligence review*, 6(1) :3–34.

Le Guillou, X., Cordier, M.-O., Robin, S., Rozé, L., et al. (2008). Chronicles for on-line diagnosis of distributed systems. In *ECAI*, volume 8, pages 194–198.

Leake, D. B. (1996). *Case-Based Reasoning : Experiences, lessons and future directions*. MIT press.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553) :436.

Legendre, P. and Legendre, L. (1998). Numerical ecology : second english edition. *Developments in environmental modelling*, 20.

Lei, Y., He, Z., Zi, Y., and Chen, X. (2008). New clustering algorithm-based fault diagnosis using compensation distance evaluation technique. *Mechanical Systems and Signal Processing*, 22(2) :419–435.

Lejri, O. and Tagina, M. (2012). Representation in case-based reasoning applied to control reconfiguration. In *ICDM*, pages 113–120. Springer.

Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., and Wess, S. (2003). *Case-based reasoning technology : from foundations to applications*, volume 1400. Springer.

Liao, T. W., Zhang, Z., and Mount, C. R. (1998). Similarity measures for retrieval in case-based reasoning systems. *Applied Artificial Intelligence*, 12(4) :267–288.

Lieber, J. (2007). Application of the revision theory to adaptation in case-based reasoning : The conservative adaptation. *Case-Based Reasoning Research and Development*, pages 239–253.

Liu, R., Yang, B., Zio, E., and Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery : A review. *Mechanical Systems and Signal Processing*, 108 :33–47.

Longo, F., Nicoletti, L., and Padovano, A. (2017). Smart operators in industry 4.0 : A human-centered approach to enhance operators' capabilities and competencies within the new smart factory context. *Computers & industrial engineering*, 113 :144–159.

Luo, B., Wang, H., Liu, H., Li, B., and Peng, F. (2018). Early fault detection of machine tools based on deep learning and dynamic identification. *IEEE Transactions on Industrial Electronics*.

Maitre, G., Pencolé, Y., Subias, A., and Gougam, H. E. (2015). Modélisation et analyse de chroniques pour le diagnostic. In *Modélisation des Systèmes Réactifs (MSR 2015)*.

Markovitch, S. and Scott, P. D. (1988). The role of forgetting in learning. In *Machine Learning Proceedings 1988*, pages 459–465. Elsevier.

Mars, P. (2018). *Learning algorithms : theory and applications in signal processing, control and communications*. CRC press.

Mejri, H. (2012). *Un système d'aide à la régulation d'un réseau de transport multimodal perturbé : réponse au problème de congestion*. PhD thesis, Ecole Centrale de Lille.

Micalizio, R., Scala, E., and Torasso, P. (2011). Intelligent supervision for robust plan execution. In *Congress of the Italian Association for Artificial Intelligence*, pages 151–163. Springer.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning : An artificial intelligence approach*. Springer Science & Business Media.

Miljković, D. (2011). Fault detection methods : A literature survey. In *MIPRO, 2011 proceedings of the 34th international convention*, pages 750–755. IEEE.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42(2-3) :363–391.

Mitchell, T. M. (1997). Does machine learning really work ? *AI magazine*, 18(3) :11.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

Moosavian, A., Ahmadi, H., Tabatabaeefar, A., and Khazaee, M. (2013). Comparison of two classifiers ; k-nearest neighbor and artificial neural network, for fault diagnosis on a main engine journal-bearing. *Shock and Vibration*, 20(2) :263–272.

Moreno, A. P., Santiago, O. L., de Lazaro, J. M. B., and Moreno, E. G. (2013). Comparative evaluation of classification methods used in fault diagnosis of industrial processes. *IEEE Latin America Transactions*, 11(2) :682–689.

Mosannenzadeh, F., Bisello, A., Diamantini, C., Stellin, G., and Vettorato, D. (2017). A case-based learning methodology to predict barriers to implementation of smart and sustainable urban energy projects. *Cities*, 60 :28–36.

Mosier, K. L. and Skitka, L. J. (2018). 10 human decision makers and automated decision aids : Made for each other ? *Automation and human performance : Theory and applications*, page 120.

Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4) :049901.

Niguez, J., Amari, S., and Faure, J.-M. (2015). Fault-tolerant control of discrete event systems : Comparison of two approaches on the same case study. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pages 1–4. IEEE.

Niguez, J., Amari, S., and Faure, J.-M. (2017). Active fault-tolerant control of timed automata with guards. *IFAC-PapersOnLine*, 50(1) :13648–13653.

Ocio, S. and Brugos, J. A. L. (2009). Multi-agent systems and sandbox games. *AISB 2009*.

Pacaux-Lemoine, M.-P., Trentesaux, D., and Rey, G. Z. (2016). Human-machine cooperation to design intelligent manufacturing systems. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 5904–5909. IEEE.

Pacaux-Lemoine, M.-P., Trentesaux, D., Rey, G. Z., and Millot, P. (2017). Designing intelligent manufacturing systems through human-machine cooperation principles : A human-centered approach. *Computers & Industrial Engineering*, 111 :581–595.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn : Machine learning in python. *Journal of machine learning research*, 12(Oct) :2825–2830.

Peixoto, J. A., Oliveira, J. A. B., Rocha, A. D., and Pereira, C. E. (2015). The migration from conventional manufacturing systems for multi-agent paradigm : The first step. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pages 111–118. Springer.

Perrin, J., Binet, F., Dumery, J.-J., Merlaud, C., Trichard, J.-P., and Perrin, J. (2004). *Automatique et informatique industrielle : Bases théoriques, méthodologiques et techniques*. Nathan Technique.

Philippot, A. (2006). *Contribution au diagnostic décentralisé des systèmes à événements discrets : Application aux systèmes manufacturiers*. PhD thesis, Université de Reims-Champagne Ardenne.

Philippot, A., Marangé, P., Carré-Ménétrier, V., and Riera, B. (2012). Implementation of diagnosis approach for discrete event systems. In *International Symposium on Security and Safety of Complex Systems, 2SCS'12*, page CDROM.

Pichard, R., Rabah, N. B., Carre-Menetrier, V., and Riera, B. (2016). Csp solver for safe plc controller : Application to manufacturing systems. *IFAC-PapersOnLine*, 49(12) :402–407.

Pierreval, H. and Ralambondrainy, H. (1990). A simulation and learning technique for generating knowledge about manufacturing systems behavior. *Journal of the Operational Research Society*, 41(6) :461–474.

Plaza, E. (1995). Cases as terms : A feature term approach to the structured representation of cases. In *International Conference on Case-Based Reasoning*, pages 265–276. Springer.

Power, D. J. (2007). A brief history of decision support systems. *DSSResources. COM, World Wide Web, http ://DSSResources. COM/history/dsshistory. html, version*, 4.

Proctor, R. W. and Van Zandt, T. (2018). *Human factors in simple and complex systems*. CRC press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1) :81–106.

Quinlan, J. R. et al. (1996). Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730.

Quinlan, R. J. (1993). C4. 5 : Programs for machine learning.

Racine, K. and Yang, Q. (1996). On the consistency management of large case bases : the case for validation. In *To appear in AAAI Technical Report-Verification and Validation Workshop*, page 1. Citeseer.

Rajaoarisoa, L. and Sayed-Mouchaweh, M. (2017). Adaptive online fault diagnosis of manufacturing systems based on devs formalism. *IFAC-PapersOnLine*, 50(1) :6825–6830.

Rao, R. B., Fung, G., and Rosales, R. (2008). On the dangers of cross-validation. an experimental evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 588–596. SIAM.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial intelligence*, 32(1) :57–95.

Reuss, P., Stram, R., Althoff, K.-D., Henkel, W., and Henning, F. (2018). Knowledge engineering for decision support on diagnosis and maintenance in the aircraft domain. In *Synergies Between Knowledge Engineering and Software Engineering*, pages 173–196. Springer.

Reyes, E. R., Negny, S., Robles, G. C., and Le Lann, J.-M. (2015). Improvement of online adaptation knowledge acquisition and reuse in case-based reasoning : Application to process engineering design. *Engineering Applications of Artificial Intelligence*, 41 :1–16.

Riera, B., Benlorhfar, R., Annebicque, D., Gellot, F., and Vigario, B. (2011). Robust control filter for manufacturing systems : application to plc training. In *18th World Congress of the International Federation of Automatic Control*.

Riera, B., Marangé, P., Gellot, F., Nocent, O., Magalhaes, A., and Vigario, B. (2010). Complementary usage of real and virtual manufacturing systems for safe plc training. *IFAC Proceedings Volumes*, 42(24) :89–94.

Riesbeck, C. K. and Schank, R. C. (2013). *Inside case-based reasoning*. Psychology Press.

Rocha, A. D., Lima-Monteiro, P., Parreira-Rocha, M., and Barata, J. (2017). Artificial immune systems based multi-agent architecture to perform distributed diagnosis. *Journal of Intelligent Manufacturing*, pages 1–13.

Rohee, B. (2005). Répartition dynamique d'activité sur un automate programmable industriel à moniteur non préhemptif. *Mémoire de DEA de Production Automatisé, LURPA, encadré par De Smet O*, 6.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Ruiz, P. A. P., Kamsu-Foguem, B., and Noyes, D. (2013). Knowledge reuse integrating the collaboration from experts in industrial maintenance management. *Knowledge-Based Systems*, 50 :171–186.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Saddem, R. (2012). *Diagnosticabilité modulaire appliquée au Diagnostic en ligne des Systèmes Embarqués Logiques*. PhD thesis, Ecole Centrale de Lille.

Saddem, R., Armand, T., and Moncef, T. (2012). Algorithme d'interprétation d'une base de signatures temporelles causales pour le diagnostic en ligne des systèmes à événements discrets. In *9th International Conference on Modeling, Optimization & SIMulation*.

Saddem, R. and Philippot, A. (2014). Causal temporal signature from diagnoser model for online diagnosis of discrete event systems. In *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on*, pages 551–556. IEEE.

Saddem, R., Toguyeni, A., and Tagina, M. (2011a). A model-checking approach for checking the Consistency of a set of Causal Temporal Signatures. In *9th European Workshop on Advanced Control and Diagnosis*, pages 1–6, Budapest, Hungary.

Saddem, R., Toguyeni, A., and Tagina, M. (2011b). Diagnostic des systèmes embarqués critiques : Application à la carte de commande du système de freinage d'un train. *Journal Européen des Systèmes Automatisés (JESA)*, 45(1-3) :205–220.

Saddem, R., Toguyeni, A., and Tagina, M. (2011c). A model-checking approach for checking the consistency of a set of causal temporal signatures. In *9th European Workshop on Advanced Control and Diagnosis*, pages 1–6.

Salahshoor, K., Kordestani, M., and Khoshro, M. S. (2010). Fault detection and diagnosis of an industrial steam turbine using fusion of svm (support vector machine) and anfis (adaptive neuro-fuzzy inference system) classifiers. *Energy*, 35(12) :5472–5482.

Salem, M., Lakatos, G., Amirabdollahian, F., and Dautenhahn, K. (2015). Would you trust a (faulty) robot ? : Effects of error, task type and personality on human-robot cooperation and trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 141–148. ACM.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, 40(9) :1555–1575.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. C. (1996). Failure diagnosis using discrete-event models. *IEEE transactions on control systems technology*, 4(2) :105–124.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3) :210–229.

Schapire, R. E. (2003). The boosting approach to machine learning : An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer.

Shafi, U., Safi, A., Shahid, A. R., Ziauddin, S., and Saleem, M. Q. (2018). Vehicle remote health monitoring and prognostic maintenance system. *Journal of Advanced Transportation*, 2018.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning : From theory to algorithms*. Cambridge university press.

Shin, H.-J., Cho, K.-W., and Oh, C.-H. (2018). Svm-based dynamic reconfiguration cps for manufacturing system in industry 4.0. *Wireless Communications and Mobile Computing*, 2018.

Smyth, B. and McKenna, E. (1998). Modelling the competence of case-bases. In *European Workshop on Advances in Case-Based Reasoning*, pages 208–220. Springer.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4) :427–437.

Stahl, A. (2003). *Learning of knowledge-intensive similarity measures in case-based reasoning*. dissertation. de.

Stodolsky, D. (1982). Steven l. alter : Decision support systems : Current practice and continuing challenges. reading, massachusetts : Addison-wesley publishing co., 1980, 316 pp. *Behavioral Science*, 27(1) :91–92.

Studnia, I. (2015). *Détection d'intrusion pour des réseaux embarqués automobiles : une approche orientée langage*. PhD thesis, Institut national des sciences appliquées de Toulouse.

Su, R. (2004). *Distributed diagnosis for discrete-event systems*. University of Toronto.

Subias, A., Travé-Massuyès, L., and Le Corronc, E. (2014). Learning chronicles signing multiple scenario instances. *IFAC Proceedings Volumes*, 47(3) :10397–10402.

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. (2015). Machine learning for predictive maintenance : A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3) :812–820.

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3) :293–300.

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., and Sui, F. (2018a). Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9-12) :3563–3576.

Tao, F., Qi, Q., Liu, A., and Kusiak, A. (2018b). Data-driven smart manufacturing. *Journal of Manufacturing Systems*.

Tidriri, K., Chatti, N., Verron, S., and Tiplica, T. (2016). Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring : A review of researches and future challenges. *Annual Reviews in Control*, 42 :63–81.

Toguyeni, A., Craye, E., and Gentina, J. (1990). A method of temporal analysis to perform online diagnosis in the context of flexible manufacturing system. In *Industrial Electronics Society, 1990. IECON'90., 16th Annual Conference of IEEE*, pages 445–450. IEEE.

Toguyeni, A., Craye, E., and Gentina, J. (1996). A framework to design a distributed diagnosis in fms. In *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, volume 4, pages 2774–2779. IEEE.

Toguyeni, A., Craye, E., and Gentina, J. (1997). Time and reasoning for on-line diagnosis of failures in flexible manufacturing systems. In *Proceedings of the 15th IMACS world congress on scientific computation, modeling, and applied mathematics*, volume 6, pages 709–714.

Toguyeni, A. K. A. (1992). *Surveillance et diagnostic en ligne dans les ateliers flexibles de l'industrie manufacturière*. PhD thesis, Lille 1.

Trentesaux, D. (1996). *Conception d'un système de pilotage distribué, supervisé et multicritère pour les systèmes automatisés de production*. PhD thesis, Institut National Polytechnique de Grenoble-INPG.

Trentesaux, D. and Millot, P. (2016). A human-centred design to break the myth of the "magic human" in intelligent manufacturing systems. In *Service orientation in holonic and multi-agent manufacturing*, pages 103–113. Springer.

Trojanowska, J., Varela, M. L. R., and Machado, J. (2017). The tool supporting decision making process in area of job-shop scheduling. In *World Conference on Information Systems and Technologies*, pages 490–498. Springer.

Van der Hoek, W. and Wooldridge, M. (2008). Multi-agent systems. *Foundations of Artificial Intelligence*, 3 :887–928.

Vareilles, E., Aldanondo, M., De Boisse, A. C., Coudert, T., Gaborit, P., and Geneste, L. (2012). How to take into account general and contextual knowledge for interactive aiding

design : Towards the coupling of csp and cbr approaches. *Engineering Applications of Artificial Intelligence*, 25(1) :31–47.

Vazan, P., Janikova, D., Tanuska, P., Kebisek, M., and Cervenanska, Z. (2017). Using data mining methods for manufacturing process control. *IFAC-PapersOnLine*, 50(1) :6178–6183.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., and Yin, K. (2003). A review of process fault detection and diagnosis : Part iii : Process history based methods. *Computers & chemical engineering*, 27(3) :327–346.

Vizcarrondo, J., Aguilar, J., Exposito, E., and Audine, S. (2015). Distributed chronicles to faults recognition. *Ciencia e Ingeniería*, 36(2) :73–83.

Vogel-Heuser, B., Fay, A., Schaefer, I., and Tichy, M. (2015). Evolution of software in automated production systems : Challenges and research directions. *Journal of Systems and Software*, 110 :54–84.

Vogl, G. W., Weiss, B. A., and Helu, M. (2016). A review of diagnostic and prognostic capabilities and best practices for manufacturing. *Journal of Intelligent Manufacturing*, pages 1–17.

Watson, I. (1999). Case-based reasoning is a methodology not a technology. In *Research and Development in Expert Systems XV*, pages 213–223. Springer.

Watson, I. and Marir, F. (1994). Case-based reasoning : A review. *The knowledge engineering review*, 9(4) :327–354.

Wei, C.-C. (2015). Comparing lazy and eager learning models for water level forecasting in river-reservoir basins of inundation regions. *Environmental Modelling & Software*, 63 :137–155.

Whitaker, D. A., Egan, D., OBrien, E., and Kinnear, D. (2018). Application of multivariate data analysis to machine power measurements as a means of tool life predictive maintenance for reducing product waste. *arXiv preprint arXiv :1802.08338*.

Widodo, A. and Yang, B.-S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical systems and signal processing*, 21(6) :2560–2574.

Wilke, W. and Bergmann, R. (1998). Techniques and knowledge used for adaptation during case-based problem solving. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 497–506. Springer.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann.

Wouda, F. J., Giuberti, M., Bellusci, G., and Veltink, P. H. (2016). Estimation of full-body poses using only five inertial sensors : an eager or lazy learning approach ? *Sensors*, 16(12) :2138.

Wu, C., Liu, F., and Zhu, B. (2015). Control chart pattern recognition using an integrated model based on binary-tree support vector machine. *International Journal of Production Research*, 53(7) :2026–2040.

Wuest, T., Weimer, D., Irgens, C., and Thoben, K.-D. (2016). Machine learning in manufacturing : advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1) :23–45.

Xiong, N., Olsson, T., and Funk, P. (2012). Case-based reasoning supports fault diagnosis using sensor information. *eMaintenance*, page 63.

Yan, A., Yu, H., and Wang, D. (2017). Case-based reasoning classifier based on learning pseudo metric retrieval. *Expert Systems with Applications*, 89 :91–98.

Zambal, S., Eitzinger, C., Clarke, M., Klintworth, J., and Mechin, P.-Y. (2018). A digital twin for composite parts manufacturing : Effects of defects analysis based on manufacturing data. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 803–808. IEEE.

Zaytoon, J. et al. (2001). Systèmes dynamiques hybrides. *Hermes*.

Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2) :308–320.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. (2018). A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv :1804.06893*.

Zhang, Y.-D., Yang, Z.-J., Lu, H.-M., Zhou, X.-X., Phillips, P., Liu, Q.-M., and Wang, S.-H. (2016). Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine, and stratified cross validation. *IEEE Access*, 4 :8375–8385.

Zhou, M., Chen, Z., He, W., and Chen, X. (2010). Representing and matching simulation cases : A case-based reasoning approach. *Computers & Industrial Engineering*, 59(1) :115–125.

Zidi, S. (2007). *SARR : Système d'aide a la régulation et la reconfiguration des réseaux de transport multimodal*. PhD thesis, Lille 1.

APPROCHE INTELLIGENTE À BASE DE RAISONNEMENT À PARTIR DE CAS POUR LE DIAGNOSTIC EN LIGNE DES SYSTÈMES AUTOMATISÉS DE PRODUCTION

Les systèmes automatisés de production (SAP) représentent une classe importante de systèmes industriels qui sont devenus complexes et sensibles aux dysfonctionnements avec des conséquences importantes sur la productivité, la qualité de production et la sécurité des biens et des personnes. Un défi majeur est de mettre en œuvre des approches systématiques d'aide ou de diagnostic afin de garantir la sûreté de fonctionnement. Cette thèse s'intéresse donc au diagnostic en ligne des comportements indésirables des SAP ayant des capteurs et des actionneurs délivrant des signaux binaires et dont la dynamique correspond à celle des systèmes à évènements discrets (SED). L'efficacité d'une approche de diagnostic au service de ce type de systèmes se mesure à travers le taux de bonne détection de fautes, la précision d'isolation, le nombre de fausses alarmes et la complexité de mise en œuvre de l'approche. L'objectif de ce travail est donc de concevoir des solutions de diagnostic intelligentes qui satisfont les critères mentionnés sans requérir la connaissance complète du fonctionnement interne du système, capables de se mettre à jour en temps réel pour améliorer les performances et ayant un faible coût de mise en œuvre. L'approche proposée dans cette thèse est basée sur le Raisonnement à Partir de Cas (RàPC) qui est à la fois une méthodologie de raisonnement par analogie et une méthodologie d'apprentissage issue du domaine de l'intelligence artificielle. L'originalité des travaux réside en finalité dans les quatre items suivants : (1) l'approche proposée utilise le RàPC pour le diagnostic des SAP ayant une dynamique de type SED, (2) elle propose un format de représentation de cas inspiré du formalisme Signatures Temporelles Causales qui s'adapte à l'aspect dynamique des systèmes à surveiller, (3) elle introduit une phase qui couplent simulation et mise en forme de données pour la transformation des données issues du système simulé après son émulation en mode normal et défaillant, et (4) elle présente une phase de raisonnement et d'apprentissage qui permet non seulement le diagnostic en ligne du système surveillé mais aussi la mise à jour de la base de cas suite à l'apparition de nouveaux comportements inconnus.

MOTS CLES : Diagnostic, apprentissage automatique, raisonnement à partir de cas, système d'aide à la décision, systèmes à évènements discrets, systèmes automatisés de production, métriques de distance, systèmes à base de données.

INTELLIGENT APPROACH USING CASE-BASED REASONING FOR ONLINE DIAGNOSIS OF AUTOMATED PRODUCTION SYSTEMS

Automated Production Systems (APS) represent an important class of industrial systems. They have become complex and susceptible to malfunctions, with significant negative consequences on productivity, production quality and security of property and people. The main challenge when using such systems is to implement systemic assistance or diagnosis approaches in order to ensure their operating safety. Within this framework, we focus, in this thesis, on the online diagnosis of the APS equipped with sensors and actuators emitting binary signals. These systems can be considered as 'Discrete Event Systems' (DES). The effectiveness of an approach for diagnosing such systems is measured through good detection rate, isolation accuracy, false alarms number and the method implementation complexity. The objective of this work is, thus, to propose intelligent diagnosis solutions satisfying the mentioned criteria without complete knowledge about the system's internal functioning. The solutions, whose implementation is not expensive, are able to update themselves in real time in order to improve their performance. The introduced approach is based on a reasoning and learning methodology derived from 'Artificial Intelligence' (AI) which is the 'Case Based Reasoning' (CBR). The originality of our research work is observed in the following 4 aspects : (1) the developed approach uses the CBR for the diagnosis of the APS with DES dynamics, (2) it suggests a case representation format inspired by the 'Causal Temporal Signatures' (CTS) which is able to adapt to the dynamic aspect of the systems to be monitored, (3) it exploits data acquired from a digital twin after its emulation in both normal and faulty modes to build an empirical knowledge called 'cases' and finally (4) it presents a reasoning and learning phase that allows not only the online diagnosis of the monitored system, but also the updating of the case base following the appearance of the new unknown behaviors.

KEYWORDS: Diagnosis, machine learning, case-based reasoning, decision support system, discrete event systems, automated production systems, distance metrics, data driven systems.

Discipline : Automatique, Génie Informatique, Traitement du Signal et Image

Établissement –Université de Reims Champagne Ardenne, École doctorale Sciences Du numérique et de l'ingénieur

N° et nom de laboratoire. CReSTIC, EA 3804

UFR Sciences Exactes et Naturelles- Moulin de la Housse-BP 1039- 51687 Reims CEDEX 2

Université de Reims Champagne-Ardenne – École doctorale Sciences Du numérique et de l'ingénieur

CReSTIC, EA 3804

UFR Sciences Exactes et Naturelles- Moulin de la Housse-BP 1039- 51687 Reims CEDEX 2