République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA
Faculté de Technologie
Département Génie électrique
Laboratoire ou unité de recherche de rattachement : Laboratoire de Technologie Industrielle et de l'Information*(LTII)*

THÈSE

Présentée par

KACIMI Mohand Akli

Pour l'obtention du grade de

DOCTEUR EN SCIENCES

Filière : Automatique et traitement du signal
Option : Systèmes

Thème

Contribution à la conception des systèmes flous de type Mamdani par les algorithmes d'optimisation multi-objectif

Soutenue le : **25/07/2021** Devant le Jury composé de :

Nom et Prénom	Grade		
Mr KHIREDDINE Abdelkrim	**Professeur**	Univ. de Bejaia	Président
Mr GUENOUNOU Ouahib	**Professeur**	Univ. de Bejaia	Rapporteur
Mr HAMICHE Hamid	**Professeur**	Univ. de Tizi Ouzou	Examinateur
Mr LAMAMRA Kheireddine	**Professeur**	Univ. de Oum El Bouaghi	Examinateur
Mr BENSAFIA Yassine	**MCA**	Univ. de Bouira	Examinateur
Mr ACHOUR Abdelyazid	**MCA**	Univ. de Bejaia	Examinateur

Année Universitaire : 2020-2021

Avant–Propos

Remerciements

Le travail fait durant cette thèse a était effectué au sein de l'équipe de recherche : TCA (Techniques de Commande Avancées) du laboratoire de recherche LTII (Laboratoire de Technologie Industrielle et de l'Information) de l'université A.MIRA de Bejaia. A cet effet, je tiens à exprimer ma reconnaissance et mes sincères remerciements à monsieur le Professeur : Ouahib GUENOUNOU chef de l'équipe TCA, pour son encadrement, ses encouragements incessants et sa bienveillance tout au long de la durée de cette thèse, sans oublier tous les membres de son équipe pour leurs soutiens.

Mes remerciements s'adressent également à monsieur le président des jurys le Professeur Abdelkrim KHIREDDINE, et messieurs les membres du jury : le Professeur Hamid HAMICHE, le Professeur Kheireddine LAMAMRA, le Maitre de Conférence (MCA) Yassine BENSAFIA et le Maitre de Conférence (MCA) Abdelyazid ACHOUR, d'avoir acceptés de juger le travail présenté dans cette thèse et pour leurs précieux temps consacré.

Sommaire

Chapitre II : Optimisation et algorithmes

Chapitre III : Conception mono-objectif des SIFs

Liste des figures

Liste des tableaux

Introduction générale

L'évolution de l'intelligence artificielle reçoit un accueil grandissant dans le monde de l'ingénierie, ses différentes disciplines et branches colonisent pratiquement tous les aspects de la science moderne : *Contrôle des systèmes, modélisation d'évènements, reconnaissance de formes, etc.* Bien qu'elle est une science à part entière, elle trouve son environnement très diversifié allant de l'informatique, à l'automatique, à la navigation jusqu'aux sciences les moins soupçonnées telles que la cosmologie, la pharmacologie, ou même la médecine.

L'intérêt que lui porte la communauté scientifique est d'autant plus justifié que cette dernière a pu ouvrir de nouvelles perspectives et a éveillé de nouvelles idées quant à l'exploitation du potentiel de nos machines modernes. À l'aube de son ère, le rêve été que les machines soient capables de communiquer avec l'homme. Mais aujourd'hui, on cherche plutôt à les rendre capables d'appréhender le monde comme le ferait un humain.

Les récentes années, ont été remplies de succès qui sont pratiquement au niveau des espérances tant attendus pour l'intelligence artificielle qui ne cesse jamais de nous surprendre. La première suprématie de cette intelligence est survenue en 1997, après le match d'échecs entre Garry KASPAROV et l'ordinateur d'IBM *DeepBlue* [1]. Ce succès a été suivi d'un autre encore plus surprenant, réalisé par *AlphaGo* qui a pu battre le champion du monde Lee SEDOLE en jeu de GO [2]. Ces deux exemples ne sont que quelques-uns où la puissance de l'intelligence artificielle et son potentiel se sont fait sentir, mais aujourd'hui on ne compte même plus le nombre d'exploits que réalise cette dernière.

Ces exploits et ces démonstrations de force sont le fruit d'un travail entreprit depuis les années cinquante, avec l'émergence d'une nouvelle entité appelée le neurone formel ou simplement les réseaux de neurones [3]. Ces derniers, ont permis aux spécialistes de cette science d'incorporer une nouvelle faculté à ces machines modernes. Il s'agit de la faculté d'apprentissage, qui domine récemment cette science. Par une série d'essais, les machines arrivent à achever des performances très honorables, même là-où toute autre programmation classique échouerait. Inspirer des observations faites sur le monde du vivant et s'appuyant sur les ressources immenses de ces machines modernes, les spécialistes tentent sans cesse de repousser les limites de ces dernières dans le sens à les rendre capables d'apprendre d'elles même et de s'auto-ajuster.

C'est durant cette même époque que naquit la théorie des *Ensembles flous*, plus connue par : *Logique floue*, présentée par le professeur Lotfi. A ZADEH durant les années soixante [4]. Cette théorie reprend les mêmes principes que ceux de la théorie des ensembles classiques, mais sous une appréhension purement humaine. À la place d'ensembles clairement définis, la logique floue utilise des ensembles plutôt mal définis souvent appelés : *Termes linguistiques*. Par cette théorie, le professeur a couronné l'ambition de formaliser le langage humain. Cette capacité justement à modéliser le langage humain, a rendu cette théorie très célèbre. Et elle a rapidement attiré la curiosité de nombreux chercheurs qui ont apporté des contributions significatives permettant l'exploitation de cette théorie dans les problèmes scientifiques les plus courants. *Ebrahim H. Mamdani* et *Michio Sugeno* font figures de pionniers et de principaux architectes dans l'utilisation de cette théorie. De nos jours, chacun de leurs travaux a donné naissance à un courant ayant ses propres applications et ses propres caractéristiques. EH. Mamdani, fut le premier à proposer un modèle de raisonnement flou, connu sous la désignation de « *Modèle linguistique* » [5] et il fut également le premier à l'expérimenter sur une application industrielle. M. Sugeno quant à lui, est l'auteur des modèles de raisonnement flous connus sous l'appellation de « *Modèles précis* » [6]. Cependant, l'ensemble des spécialistes actuellement s'accordent à dire que de ces deux modèles, c'est celui de Mamdani qui garantit un meilleur consensus entre la machine et l'homme.

L'application des systèmes flous de type Mamdani sur des problèmes courants de l'automatique tel que : l'asservissement et l'identification des systèmes, le diagnostic des défauts etc. ont démontré leur efficacité remarquable à assurer ces tâches et leur potentiel descriptif considérable. A l'origine, la conception de ces systèmes flous est basée essentiellement sur l'expérience d'expert, qui formule ses connaissances sous forme de

propositions simples de la forme (*Si-Alors*). Toutefois, cette approche constitue un défi de taille pour les concepteurs. D'une part, à cause du manque d'informations précises et explicites sur la tâche à accomplir et d'autre part à cause de la complexité à transcrire le raisonnement de l'expert humain sous forme de propositions simplifiées de la forme (*Si-Alors*) en plus du nombre de paramètres à ajuster simultanément pour définir complètement le système flou.

En raison des difficultés mentionnées, les chercheurs ont eu recours à la faculté d'apprentissage des machines. Ils empruntèrent alors d'autres voies de conception que l'expérience d'experts pour améliorer les performances de ces systèmes flous. De cette initiative naquirent de nombreuses hybridations du système flou avec d'autres approches. La première était celle du système flou avec les réseaux de neurones artificiels, couramment appelée: *NFS* (*Neuro Fuzzy System)* [7-8]. Cette hybridation exploite les algorithmes d'apprentissage des réseaux de neurones pour régler et concevoir les systèmes flous. Dans la seconde classe, le système flou a était hybridé avec des algorithmes d'optimisation évolutionnaires, dont l'algorithme génétique figure parmi les plus sollicités, engendrant ainsi une nouvelle branche nommée : *GFS* (*Genetic fuzzy System*) [9]. Cette hybridation est la plus abondante dans la littérature, elle a proliféré en plusieurs sous-catégories et classes suivant l'objectif et la façon d'utiliser ces algorithmes évolutionnaires.

De nos jours, la conception automatique des systèmes flous est devenue synonyme d'utilisation d'algorithmes d'optimisation. Bien que l'usage de ces algorithmes garantit souvent de bons résultats, ils restent néanmoins délicats à utilisés. C'est-à-dire : si ces algorithmes ne sont pas bien configurés, ils risquent de converger vers un état indésirable ou qui n'est pas compatible par rapport à ce que nous cherchons réellement.

Problématique

Utiliser ces algorithmes pour optimiser et/ou concevoir des systèmes flous, tout particulièrement ceux de type Mamdani, demeure un enjeu d'actualité et de taille. À nos jours, il n'existe pas de méthode efficace et toujours fiable pour aboutir systématiquement à un modèle optimal. D'une application à une autre dans un même domaine ou d'un domaine à un autre, ces méthodes peuvent plus ou moins faire émerger un modèle optimal. Par ailleurs, les récentes exigences sur les systèmes flous, telles que : *la simplicité à le comprendre, la cohérence de son*

expertise et *la minimisation de ses ressources* n'ont fait qu'augmenter d'avantage la complexité du problème.

La difficulté de satisfaire ces nouvelles exigences, réside tout d'abord dans le fait qu'elles sont mutuellement contradictoires avec la performance du système flou à assurer la tâche qui lui est assignée. Ensuite, dans la confusion qui règne sur certaines d'entre-elles, due à leurs subjectivités comme c'est le cas des deux premières exigences citées juste avant. Par conséquent, la mesure de telles exigences demeure toujours une question ouverte. Par ailleurs, l'interdépendance des paramètres définissants le système flou et leurs nombre considérables, compliquent d'avantage le problème de sa conception

Toutes ces complications nous ont conduit à penser qu'une approche de conception basée sur des algorithmes d'optimisation multi-objectif, peut être une solution adéquate pour remédier à cette question épineuse qui est : « *Comment peut-on obtenir un modèle flou optimal de type Mamdani par apprentissage automatique, tout en répondant à plusieurs exigences simultanément* ». Le potentiel de ces algorithmes à trouver les meilleurs compromis entre plusieurs objectifs contradictoires constituent l'un des points qui ont permis de développer d'avantage ce manuscrit.

Organisation du manuscrit

Pour une meilleure appréhension de ce sujet, nous avons réparti le manuscrit en quatre chapitres. Chacun développe graduellement une notion fondamentale.

Le premier chapitre est entièrement dédié à la théorie des ensembles flous et aux systèmes flous en particulier ceux de type Mamdani. Tous les fondements théoriques seront présentés en se basant sur la structure basique de ces systèmes, dite type-1 avec deux entrées et une seule sortie (*MISO*). En plus des notions fondamentales sur les systèmes flous, ce chapitre soulève également les principaux défis que requiert la conception de ces derniers.

Le second chapitre, réparti en trois sections, introduit la notion d'optimisation. La première section, présente le problème d'optimisation du point de vue mathématique ainsi que toutes les notions fondamentales relatives. La seconde, décortique d'avantage l'optimisation dite mono-objectif, en parcourant ses principaux aspects. Enfin, dans la troisième section,

l'optimisation multi-objectif sera abordée en introduisant ses principales caractéristiques et mécanismes de fonctionnement.

Le troisième chapitre expose la démarche de conception des systèmes flous par les algorithmes d'optimisations dit : mono-objectif. Dans un premier temps, ce chapitre survolera l'état de l'art de la conception dite mono-objectif en présentant les différentes variantes et les différentes démarches empruntées dans cette voie. Ensuite, la seconde partie de ce chapitre présentera en détaille notre contribution dans ce même sens.

Le quatrième chapitre présente la démarche de conception des systèmes flous par les algorithmes dit : multi-objectif. Dans ce chapitre, un résumé de l'état de l'art sur cette méthode de conception sera présenté avant d'exposer nos différents travaux et propositions dans cette classe.

Le manuscrit se terminera par quelques conclusions et perspectives.

Chapitre I

Système à inférence floue

I.1. Introduction

Les systèmes à inférence floue (*SIF*) sont souvent mentionnés dans la littérature. Ils sont très populaires chez les ingénieurs et les chercheurs qui les présentent comme étant une entité virtuelle intelligente. Une définition assez vague et ambiguë ! où chacun vend une image dépendante de son point de vue d'abord, et surtout des exigences ainsi que des besoins relatifs au problème qu'il envisage de traiter. Durant ce chapitre, nous présenterons cette entité dite : *système flou* indépendamment de ses applications en retraçant leur genèse depuis leur apparition. Au même temps, nous présenterons les notions fondamentales formant le noyau de ces systèmes, en vue de permettre une meilleure assimilation de leurs particularités et surtout de prendre conscience des questions qui entourent leur sujet.

Pour les systèmes flous, tout a commencé dès la publication de l'article du professeur Lotfi A. Zadeh intitulé : *Fuzzy Set* [4] qui signifie littéralement : *Ensemble flou.* Cet article constitue un tournant majeur dans le domaine de l'intelligence artificielle et du traitement de l'information de façon générale. Cette initiative du professeur a eu un impact si grand qu'il a pu affecter de nombreux domaines scientifiques et les tendances de recherche. Au départ, cet article a était accueilli avec beaucoup de réticence en raison de l'intitulé choisi par le professeur.

Mais, cette confusion a été vite dissipée car l'article annoncé bien une nouvelle entité mathématique ayant ses propres concepts et opérateurs.

I.2 Ensemble flou

La première fondation si existentielle des systèmes flous est la notion d'ensemble flou. Pour faire simple, un ensemble flou est un peu similaire à un ensemble classique. Sauf qu'il est mal délimité ou plutôt ses frontières ne sont pas nettement connues, mais graduellement définies. L'exemple simple pour montrer le contraste entre eux, est de considérer les deux ensembles suivants : *A* de forme circulaire de rayant *r et A'* de forme à priori circulaire dont les frontières ne sont pas clairement définies comme représentées dans la figure (I.1). Le premier est dit : *ensemble classique*, tandis que le second est dit : *ensemble flou*.

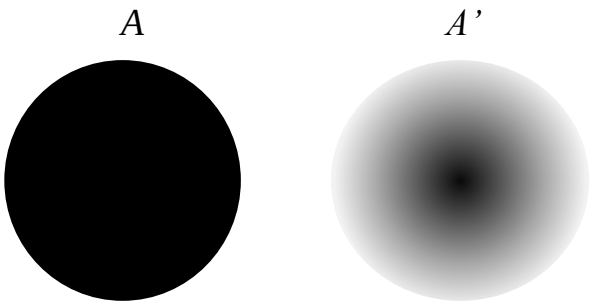


Figure I. 1: Ensemble classique et ensemble flou

Ce contraste de forme prend tout son sens dès qu'on se pose la question de l'appartenance : *Comment décrire l'appartenance d'un individu à cet ensemble flou ?.* Au moment où l'appartenance aux ensembles classiques est booléenne celle des ensembles flous est définie dans le même intervalle, mais par une équation mathématique décrivant le degré d'appartenance d'un individu à ce dernier. Cette équation mathématique est appelée : *fonction d'appartenance,* et ainsi découle la première notion des ensembles flous. La figure (I.2) présente un exemple d'ensemble flou associé à une fonction d'appartenance.

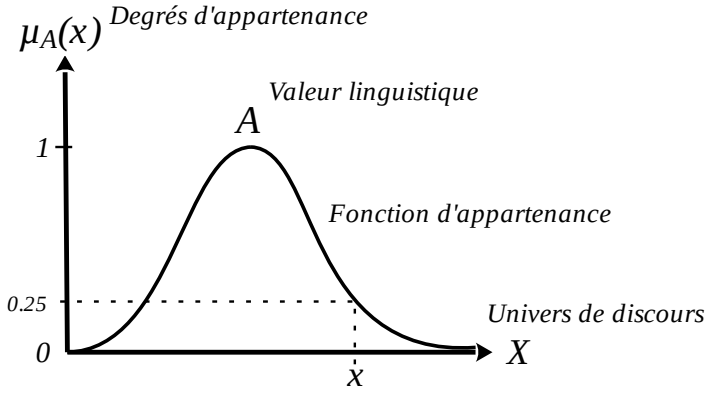


Figure I. 2 : Ensemble flou et fonction d'appartenance

Définition 1 : *Fonction d'appartenance*

*Soit **X** un ensemble d'individus **x_i**. Un ensemble flou **A** dans **X** est défini par la fonction d'appartenance **$\mu_A(x_i)$** à valeurs dans l'intervalle [0, 1] tel que :*

$$\forall\ x_i \in X\ ,\ \mu_A(x_i) \in [0,1] \tag{I.1}$$

I.2.1 Caractéristiques des ensembles flous

Un coup d'œil rapide sur la littérature des systèmes flous, permet de constater, en plus de la fonction d'appartenance, l'existence d'un ensemble de caractéristiques propres aux ensembles flous [10]. Certaines d'entre-elles semblent similaires à celles des ensembles classiques, mais en réalité la ressemblance ne dépasse pas la similitude dans la dénomination. Pour les ensembles flous, ces caractéristiques ne sont pas absolues et muettes, elles sont plutôt étroitement liées à la fonction d'appartenance définissant l'ensemble flou.

Définition 2 : *Support de l'ensemble flou*

*Le support d'un sous-ensemble flou **A** noté : **Supp(A)** est défini comme étant l'ensemble classique des éléments **x_i** pour lesquels **$\mu_A(x_i)$** est non nulle :*

$$\forall\ x_i \in X,\ Supp\ (A) = \{x_i /\ \mu_A(x_i) > 0\} \tag{I.2}$$

Définition 3 : *Noyau de l'ensemble flou*

*Le noyau d'un sous-ensemble flou **A** noté : **N(A)**, est l'ensemble classique de tous les éléments **x_i** de **X** qui appartiennent totalement à **A** :*

$$\forall\ x_i \in X\ ,\ N(A) = \{x_i / \mu_A(x_i) = 1\} \tag{I.3}$$

Définition 4 : *Hauteur de l'ensemble flou*

*La hauteur d'un sous-ensemble flou **A** notée : **h(A)**, correspond à la hauteur maximale de la fonction d'appartenance **$\mu_A(x_i)$**. Cette hauteur est généralement normalisée dans l'intervalle **[0,1]**.*

$$\forall\ x_i \in X :\ h(A) =\ Max\ (\mu_A(x_i)) \tag{I.4}$$

Définition 5 : *α-coupe de l'ensemble flou*

*Une **α-coupe** de l'ensemble flou **A**, est l'ensemble classique des **x_i** dans **X** ayant un degré d'appartenance supérieur à (**α**):*

$$\forall\ x_i \in X\ \wedge\ \forall\ \alpha \in [0,1]\ ,\ \alpha - coupe\ \ (A) = \{x_i /\ \mu_A(x_i) > \alpha\} \tag{I.5}$$

Toutes ces caractéristiques sont illustrées par la figure (I.3). Ici, la forme trapézoïdale est utilisée pour sa simplicité, mais cela reste valable pour n'importe quelle autre forme de la fonction d'appartenance.

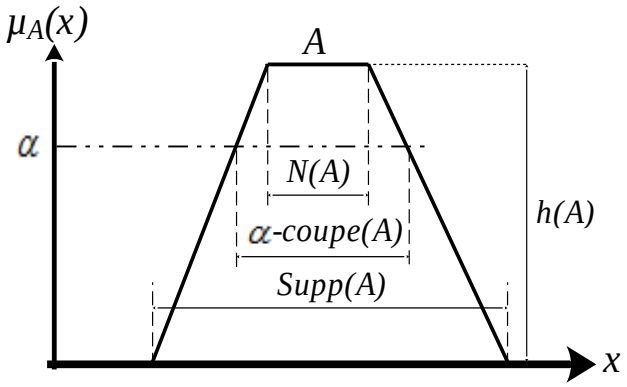


Figure I. 3 : Caractéristiques d'un sous-ensemble flou

I.3 Variable linguistique

A présent, il est très facile de décrire mathématiquement l'appartenance d'un individu à un ensemble flou. Vu qu'il s'agit d'un ensemble, ce dernier servira certainement à la même finalité que les ensembles classiques. À savoir distinguer une classe d'individus parmi tant d'autres. En effet sur ce point, l'ensemble flou ne fait pas exception. Mais il reste que sa nature fait de son utilisation un peu plus particulière.

Pour être plus claire, considérons la situation suivante : on veut décrire la vitesse d'un véhicule par l'ensemble de labels ou de termes linguistiques suivants : *Petite, Moyenne* et *Grande*. Avec des ensembles classiques, on se retrouve obligé d'abord de segmenter la plage de variation de la vitesse en plusieurs régions, puis à chaque plage on associe un adjectif unique. Bien que cette tâche soit simple à priori, elle reste néanmoins assez délicate quand il s'agit de délimiter les plages. Cette complication, dans ce cas-ci, est due à la relativité de la perception de la notion de vitesse. La vitesse vue en tant que *Grande* par un conducteur lambda est perçue en tant que *Moyenne* voir même *Petite* par un conducteur chevronné. D'où l'ambiguïté de trouver une limite nette à chaque adjectif. Avec des ensembles flous par contre, cette première phase de segmentation est beaucoup plus simple du moment qu'il n'est pas important de définir les limites avec précision. À chacun de ces adjectifs, on attribue un sous-ensemble flou décrit par une fonction d'appartenance dont le noyau recouvrira les valeurs de la vitesse communément décrites par cet adjectif. Ensuite, ces fonctions d'appartenance seront réparties sur la plage de variation de la vitesse de façon à couvrir complètement toute cette dernière, comme illustrées par la figure (I.4).

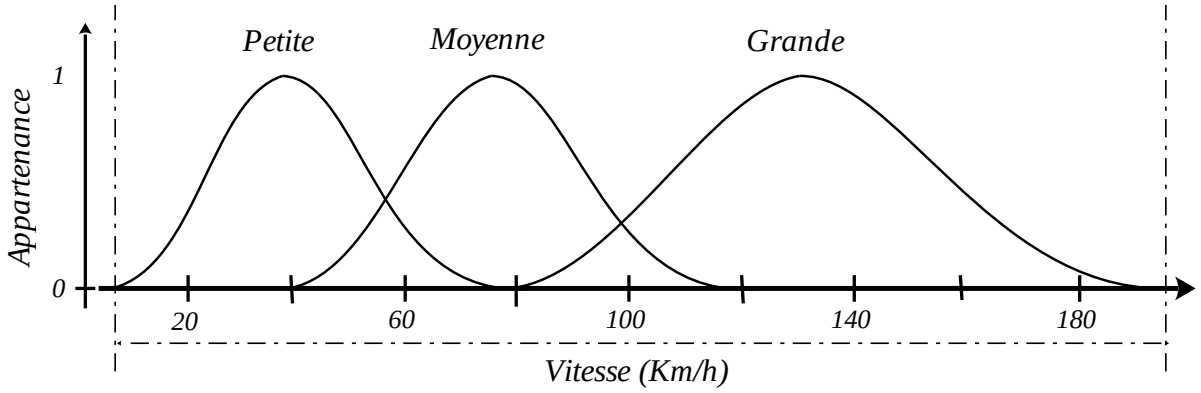


Figure I. 4 : Description de la vitesse par des ensembles flous

En plus de palier l'ambiguïté sur la délimitation des frontières pour les adjectifs, l'usage des ensembles flous offre la possibilité de modéliser la subjectivité des termes linguistiques utilisés due à la nuance de leurs sens ci-difficile à cerner par des ensembles classiques. Cette confusion de sens est exprimée par le chevauchement entre les fonctions d'appartenance, une caractéristique qui rend la description par des sous-ensembles flous beaucoup plus flexible et tolérante. Cette caractéristique est la raison majeur de la popularité de cette théorie des sous-ensembles flous et des systèmes flous en général.

I.3.1 Modificateur Linguistique

La collection des termes linguistiques, associés à leurs fonctions d'appartenance respectives, constituent ce que nous appelons un *univers de discours* ou simplement une *partition floue* notée : T_x , dans le cas de notre exemple, *x* correspond à la *vitesse*, et on écrit alors :

$$T_x = \{\mu_{Petite}(x), \mu_{Moyenne}(x), \mu_{Grande}(x)\} \quad \text{(I.6)}$$

Toute partition floue de ce genre peut être modifiée pour obtenir une seconde partition plus large et détaillée. Pour ce faire, on utilise ce que nous appelons des *modificateurs linguistiques* qui sont aussi un ensemble de termes linguistiques comme par exemple *M={Plotôt, très}* [11]. En appliquant ces modificateurs linguistiques sur la partition T_x nous obtenons une nouvelle partition notée $M(T_x)$ définie par :

$$M(T_x) = \{\mu_{TrèsPetite}(x), \mu_{PlutôtPetite}(x), \mu_{Moyenne}(x), \mu_{PlutôtGrande}(x), \mu_{TrèsGrande}(x)\} \quad \text{(I.7)}$$

I.4 Opérations floues

Dans son œuvre L. Zadah ne s'est pas contenté juste de modéliser le langage humain, mais il a également formulé mathématiquement le raisonnement vague et approché qui

caractérise la pensée purement humaine. Par une série d'opérateurs mathématiques, il a rendu possible l'expérimentation sur un calculateur classique de la pensée humaine.

Pour formuler ces opérateurs flous, L. Zadeh s'est beaucoup inspiré de la théorie des ensembles classiques. Ils partagent encore les mêmes noms que ceux de la théorie des ensembles classiques, mais pas du tout la même formulation. On retrouve par exemple l'intersection ($\cap$), l'union ($\cup$), le complément ($\bar{}$) dans la théorie floue qui sont désormais plus connus par : '*et*', '*ou*', et '*non*'. Cependant, chacun de ces opérateurs flous engendre un résultat sous forme d'ensemble flou également. Leurs définitions mathématiques reposent sur l'usage de fonctions dites t-normes ou t-conormes. Il s'agit de fonctions mathématiques ayant des propriétés auxquelles elles obéissent. Ainsi, une fonction (T) est dite t-norme, si elle peut être définie comme une opération binaire sur un intervalle unitaire $\left(T:[0,1][0,1]\rightarrow[0,1]\right)$ et qu'elle vérifie également les quatre axiomes suivants :

$$\begin{cases} T(x,1)=x \\ Si\ \left(y\le z\right)\Rightarrow T(x,y)\le T(x,z) \\ T(x,y)=T(y,x) \\ T(x,T(y,z))=T(T(y,z),x) \end{cases} \tag{I.8}$$

avec : $(x,y,z\in[0,1])$

La même définition s'applique également sur la fonction (S) dite t-conorme, sauf que pour cette dernière le premier axiome est différent. Il est défini par l'expression suivante :

$$S(x,0)=x \tag{I.9}$$

Dans la littérature, il existe des fonctions qui sont très populaires et fréquemment utilisées pour définir ces opérateurs conformément aux exigences de chaque classe de norme. Le tableau (I.1) présente quelques-unes avec leurs descriptions mathématiques. Où, A et B sont deux ensembles flous définis sur l'univers de discours T, associés respectivement aux fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$.

Tableau I. 1 : Quelques opérateurs flous et normes

	Intersection/ 'et' T-norme	**Union /'Ou' T-conorme**	**Complément /'Non'**
Opérateur de Zadeh (Min/ Max)	$T(A,B)=\min(A,B)$	$S(A,B)=\max(A,B)$	$\mu_{\bar{A}}(x)=1-\mu_A(x)$
Produit et somme algebrique (Bandler)	$T(A,B)=(A\times B)$	$S(A,B)=A+B-A\times B$	
Produit et somme algebrique (Lukasiewicz)	$T(A,B)=\max(0,\ A+B-1)$	$T(A,B)=\min(1,A+B)$	

C'est avec tous ces ingrédients que la théorie des ensembles flous modélise le langage humain pour exprimer une idée, une expertise ou simplement une connaissance. Tous comme les mots d'une langue, les ensembles flous forment le vocabulaire de cette théorie. À la place des phrases du langage courant, cette théorie combine ensembles flous et opérateurs flous pour s'exprimer à travers de simples propositions floues. Ces dernières sont des expressions causales liant une observation ou un constat dit : *cause*, à une décision dite : *conséquence*, toutes les deux exprimées linguistiquement [12].

I.5 Proposition floue

Revenant un peu à l'exemple précédent sur la vitesse et mettons-nous dans la peau d'un moniteur de conduite. Si on veut enseigner la conduite à quelqu'un, on lui formule souvent des petites phrases simples comportant la connaissance de conduite tel que par exemple : '*Si la vitesse est petite et l'arrêt est loin, alors il faudra accélérer'*, ou bien '*Si la vitesse est grande et que l'arrêt est proche, alors il faudra ralentir'*…. Ces mêmes phrases et ces mêmes connaissances peuvent être formulées en théorie floue, résumant ainsi l'expérience du moniteur de conduite ou de l'expert sous forme de simples propositions floues (Voir figure (I.5)).

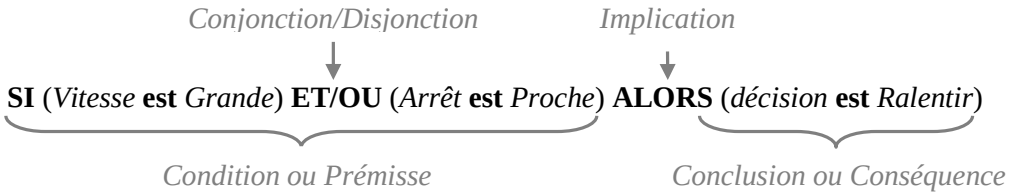


Figure I. 5 : Proposition floue ou règle floue

Du point de vu de la logique floue, chaque proposition de ce genre est un modèle de raisonnement sur lequel peut s'appuyer tout système flou pendant son fonctionnement. Surnommée aussi *règle floue*, chacune de ces dernières est d'abord formulée par l'expert exprimée dans le vocabulaire de la théorie floue. Ensuite, elle sera combinée avec d'autres pour constituer le modèle de raisonnement complet du système flou. Bien que souvent cette collection de règles soit qualifiée de l'ensemble de connaissance du système flou, elle reste obsolète pour de nombreux domaines d'ingénierie, sans d'autres mécanismes supplémentaires leur permettant d'interagir avec des entités électroniques réelles. Ces interactions se résument principalement dans la conversion de la nature de l'information du numérique au linguistique et inversement.

I.6 Système flou

À ce stade et en se basant sur ce qui a été avancé jusqu'ici, on peut définir les systèmes flous comme étant : *une entité de traitement d'information de nature symbolique ayant un support physique numérique*. Cette forme mixte les rend capables de raisonner comme un humain tout en restant une machine. Dans le paragraphe précédent, nous avons donné un aperçu de leurs constituants. Mais pour que les systèmes flous puissent fonctionner correctement, il faut impérativement définir à chaque variable d'entrée et de sortie une partition floue qui lui sera propre, en plus de l'expression mathématique des opérateurs flous. Ajouter à la collection des règles floues se forme alors le premier élément topologique des systèmes flous nommé : *la base de connaissances.* À cette base est associée deux interfaces de conversion, la première est appelée : *interface de fuzzification* qui se charge de traduire toute valeur numérique des entrées en une valeur linguistique. La seconde est appelée : *interface de défuzzification*, cette dernière se charge de convertir le résultat du calcul flou en une valeur numérique, plus facile à assimiler par la machine

Ces trois éléments sont liés à un quatrième dit : *moteur d'inférence* qui constitue le cœur du système flou. C'est lui qui réalise pratiquement l'essentiel du calcul flou en projetant les valeurs linguistiques obtenues de la fuzzification sur la base de règles afin d'induire une conséquence également linguistique.

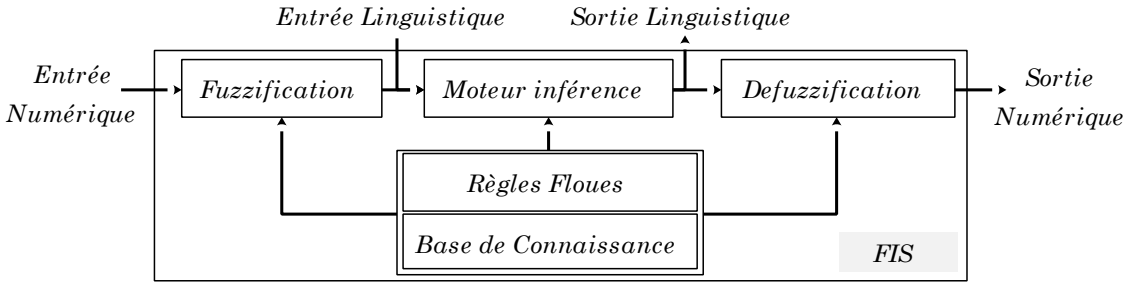


Figure I. 6 : Architecture interne d'un système flou

Cette architecture, présentée en figure (I.6), fonctionne de façon unidirectionnelle et synchronisée. L'obtention d'un résultat avec un système flou, passe immuablement par les mêmes étapes. Pour décortiquer d'avantage ce mécanisme, considérant le cas d'un système flou à deux entrées (x, y) et une seule sortie (z) respectivement associées aux partitions floues : Tx, Ty, et Tz. Chacune de ces partitions floues comporte plusieurs sous-ensembles flous tel que : $Tx = \{A_1 , A_2, \dots A_n\}$, $Ty = \{B_1 , B_2, \dots B_m\}$ et $Tz = \{C_1 , C_2, \dots C_{nm}\}$. Le modèle de raisonnement du système flou est décrit par la collection de règles floues proposant un lien de causalité entre les

valeurs linguistiques lues à partir des partitions d'entrée (*Tx, Ty*) et les décisions à prendre exprimées par la partition (*Tz*). Ces règles floues sont souvent écrites sous forme de table, qui est la forme la plus expressive. Ci-dessous un exemple de table de règles relatives à ce système étudié.

Tableau I. 2 : Table de règles floues

		Y			
	z	B_1	B_2	...	B_m
X	A_1	C_1	C_2	...	C_m
	A_2	C_{m+1}		...	C_{2m}
	⋮	⋮			⋮
	A_n	$C_{(n-1)m}$	...		C_{nm}

Cette table est lue et interprétée en parcourant les lignes et les colonnes. Associée l'une à l'autre se forme la partie prémisse de la règle et la cellule d'intersection de ces deux dernières correspond à la conséquence associée. De cette façon on peut lire facilement que : *SI (x est A_1) et (y est B_2) ALORS (z est C_2).*

I.6.1 Fuzzification

Conformément à l'architecture interne présentée dans la figure (I.6), la fuzzification est la première étape de calcul, elle convertit les valeurs numériques (x, y) des entrées en termes linguistiques précisant le degré d'appartenance de ces valeurs numériques à chacun des sous-ensembles flous présents dans les partitions des entrées.

$$\begin{cases} T_x(x) = \{\mu_{A1}(x), \mu_{A2}(x), \dots, \mu_{An}(x)\} \\ T_y(y) = \{\mu_{B1}(y), \mu_{B2}(y), \dots, \mu_{Bm}(y)\} \end{cases} \quad \text{(I.10)}$$

I.6.2 Moteur d'inférence

Les résultats de la fuzzification, autrement dit les valeurs linguistiques, sont immédiatement transmises au second étage : le moteur d'inférence. Ce dernier commence d'abord par calculer le degré de vérité de chaque règle individuellement (w_R), en considérant soit une conjonction ou une disjonction, ici représentée par l'opérateur $(\wedge)$, entre les éléments de la prémisse.

$$w_R(x, y) = \mu_{Ai}(x) \wedge \mu_{Bj}(y) \quad \text{(I.11)}$$

Une fois le degré d'activation de la règle est calculé, le moteur d'inférence projette ce dernier sur la conclusion associée à la prémisse afin de déterminer le résultat flou de la règle en

question. Cette opération liant le degré d'activation de chaque règle à l'ensemble représentant sa conclusion est dite implication floue (∘) notée comme suit :

$$\mu_R(z) = w_R(x,y) \circ \mu_{Ch}(z) \tag{I.12}$$

Le résultat de cette implication est un ensemble flou obtenu de deux façons différentes. Dans la première, la hauteur de l'ensemble flou de la conclusion est multipliée par le degré d'activation déjà calculé réduisant ainsi la hauteur de l'ensemble flou de la conclusion. Dans la seconde par contre, l'ensemble flou de la conclusion est simplement tronqué à la valeur du degré d'activation. Dans le premier cas on parle d'inférence *PRODUIT* tandis que dans le second on parle d'inférence *MINIMUM*. Une illustration graphique de ces deux méthodes est présentée dans la figure (I.7).

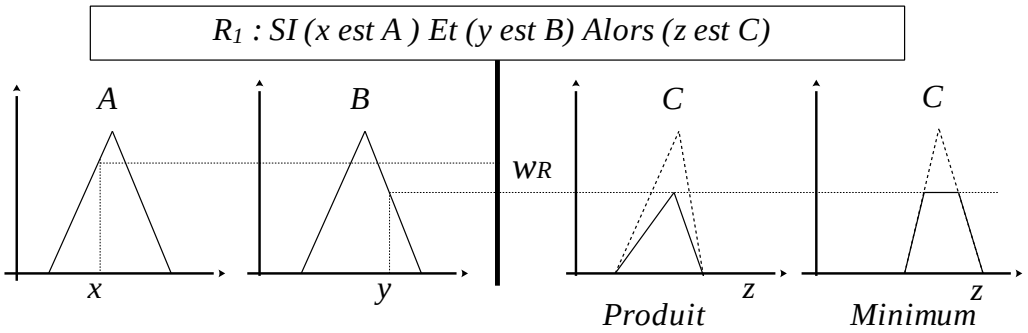


Figure I. 7 : Inférence PRODUIT et MINIMIM

I.6.3 Composition de règles floues

Le rôle du moteur d'inférence se termine dès que le résultat combiné de toutes les règles floues est obtenu. Pour ce faire, il procède simplement par agrégation des résultats élémentaires de chaque règle, ici représenté par l'opérateur (∨).

$$\mu_{aggr}(z) = \vee_{k=1}^{NR} \left(\mu_{Rk}(z)\right) \tag{I.13}$$

L'agrégation peut être vue comme une simple composition des résultats des règles floues, l'opération *SOMME* et *MAXIMUM* forment ses principales formulations. Dans la littérature la composition *SOMME* est généralement associée à l'inférence *PRODUIT*, tandis que la composition *MAXIMUM* est associée à l'inférence *MINIMUM*. Dans la figure (I.8), une composition des résultats de deux règles floues par les deux méthodes est présentée.

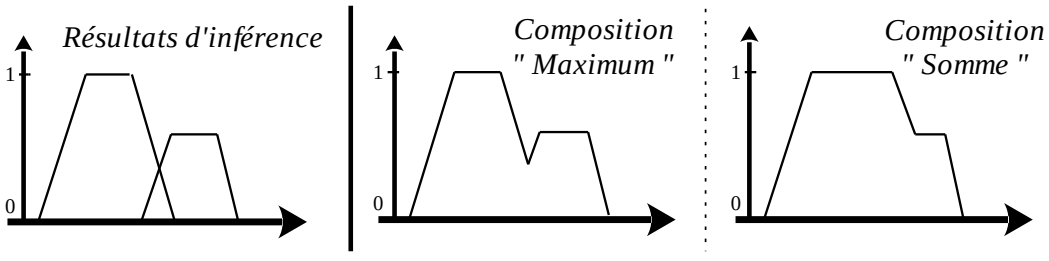


Figure I. 8 : Composition des ensembles flous

Comme nous l'avons avancé au départ, les sous-ensembles flous combinés aux opérateurs flous forment les ingrédients de base des systèmes flous. Partant de ce constat on peut à présent formuler mathématiquement le raisonnement des systèmes flous. Pour cela il suffit de retracer le chemin parcouru par les informations depuis les entrées jusqu'ici [13].

$$\mu_{aggr}(z) = \vee_{k=1}^{NR} \left(\wedge_{i=1,\, j=1}^{|Tx|\,|Ty|} \left(\mu_{A_{i,k}}(x), \mu_{B_{j,k}}(y) \right) \right) \circ \mu_{Rk}(z) \tag{I.14}$$

I.6.4 Défuzzification

La dernière phase de calcul des systèmes flous est la défuzzification. Ici, il est question de convertir le résultat flou obtenu après la composition en une valeur numérique précise. Cette tâche apparaît simple en le disant, mais elle est en réalité assez complexe : *quelle valeur numérique précise peut remplacer au mieux un ensemble flou ?*. En tentant de répondre à cette question, plusieurs propositions ont été faites, toutes visent le même objectif. Aujourd'hui, on retrouve une panoplie de techniques qui sont utilisées. Chacune d'entre elle est populaire dans un champ d'application particulier des systèmes flous. Dans le domaine du contrôle et d'identification des systèmes par exemple, c'est la méthode de centre de gravité (COG) qui est la plus utilisée [14-15].

$$z^*_{COG} = \frac{\int z.\mu_{aggr}(z)dz}{\int \mu_{aggr}(z)dz} \tag{I.15}$$

En plus de cette technique on trouve aussi la méthode du centre de surface souvent appelée : *Bisector*. Dans cette méthode, la défuzzification est assez intuitive. Elle considère l'abscisse pour laquelle la surface de l'ensemble flou résultant est partagée en deux parts égales comme étant le résultat de défuzzification. Si on considère que les limites inférieur et supérieur de l'ensemble résultant de l'agrégation sont respectivement les abscisses α et β, la valeur de défuzzification sera calculée comme suit :

$$z^*_{Bisector} = Z^* \mapsto \int_{\alpha}^{z^*} \mu_{aggr}(z)dz = \int_{z^*}^{\beta} \mu_{aggr}(z)dz \tag{I.16}$$

D'autres méthodes sont principalement basées sur la notion du maximum, comme par exemple la méthode : *FOM* (*First Of Maximum*) qui considère la valeur de défuzzification comme étant l'abscisse du premier maximum qui apparaît dans l'ensemble résultant. Ou la méthode : *LOM* (*Largest Of Maximum*) qui considère plutôt l'abscisse du dernier maximum. Comme compromis des deux, la méthode : *MOM (Mean Of Maximum)* définit la valeur de

défuzzification comme étant l'abscisse correspondant au centre du segment séparant le premier maximum du dernier [16-17].

$$z^*_{MOM} = \left(z^*_{FOM} + z^*_{LOM}\right) \Big/ 2 \qquad \text{(I.17)}$$

Pour se mettre en situation, supposant la surface en gris dans la figure (I.9) comme étant l'ensemble obtenu après composition. En appliquant les définitions déjà présentées, on peut donner une illustration des différentes méthodes de défuzzification [18].

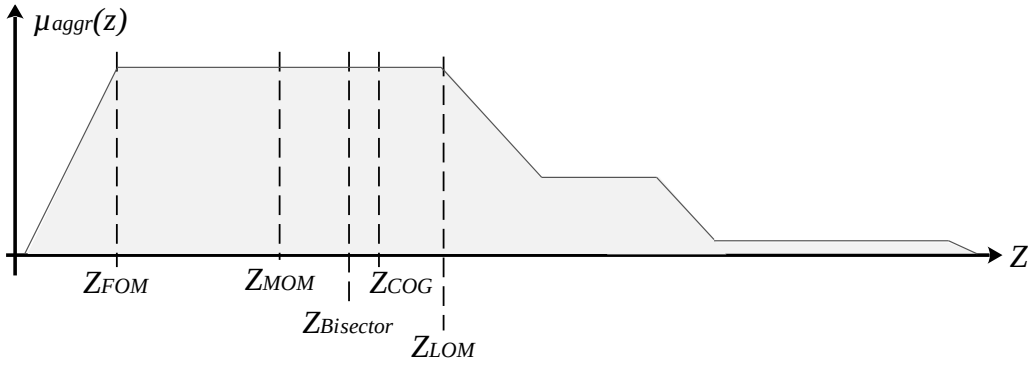


Figure I. 9 : Méthodes de défuzzification

À travers cette illustration, il est très facile de mesurer la complexité de la tâche de défuzzification et son importance également. Chacune de ces technique présente des avantages et des inconvénients, mais aucune d'entre-elles ne peut prétendre réellement se substituer à l'ensemble flou.

I.7 Genèse et taxonomie des systèmes flous

Depuis sa première mis en œuvre par E.H Mamdani [5][19], les systèmes flous ont depuis proliféré. Des entreprises et des acteurs économiques se sont rendu compte du potentiel de ces systèmes, ils les ont alors incorporés dans leurs produits pourtant destinés au grand public. Cette prolifération n'a pas été uniquement sur l'aspect applicatif des systèmes flous, mais aussi sur l'aspect théorique. D'autres chercheurs ont apporté de nouvelles visions sur le modèle de raisonnement des systèmes flous autre que celui proposé par E.H Mamdani, ces initiatives ont donné alors naissance à de nouveaux mécanismes d'inférence floue. Le modèle proposé par T. Takagi et M. Sugeno [20] est sans doute l'autre modèle flou le plus connu nommé *(TSK)*. Ce dernier forme avec le modèle de Mamdani, les deux courants des systèmes flous sur lesquels est bâtie toute la littérature.

Ces deux courants ont connu des sous-versions d'eux-mêmes. Dans son article [6], M. Sugeno a proposé une version plus simple du modèle initiale actuellement connu, par le nom :

modèle singleton. De même, le modèle linguistique a trouvé aussi une autre version de lui-même, proposé par Y. Tsukamoto [21-22]. La figure (I.10) présente une vue globale sur ces différents types du système flou, les situant les uns par rapport aux autres.

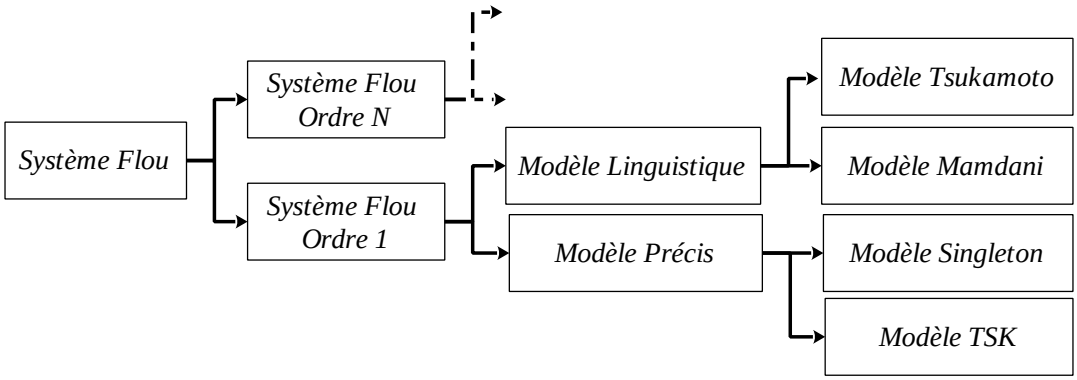


Figure I. 10 : Taxonomie des systèmes à inférence floue

La principale déférence entre les deux grandes familles des systèmes flous (*Mamdani, TSK)* réside dans la forme des conclusions des règles floues. On se retrouve avec des ensembles flous décrits par des fonctions d'appartenance dans le cas du modèle de Mamdani, et des fonctions mathématiques polynomiales dans le cas du modèle TSK.

I.7.1 Modèle flou précis

Le modèle TSK est le premier de son genre. Tentant de réduire la complexité de calcul et d'optimiser la vitesse d'exécution du système flou de Mamdani, ses auteurs ont proposé une formulation bien particulière du raisonnement flou. Au lieu de décrire les partitions floues de la sortie par des termes linguistiques, ils les décrivent par des fonctions polynomiales. Actuellement, ce modèle est considéré parmi les meilleurs approximateurs universels à côté des réseaux de neurones.

$$R_k : SI\ (x\ est\ A_i)\ ET\ (y\ est\ Bj)\ ALORS\ (z\ est\ f_k) /\quad f_k = a_{0,k} + a_{1,k}.x + a_{2,k}.y \tag{I.18}$$

Quelques années par la suite M. Sugeno a proposé une autre version plus simple et assez performante que l'originale. Dans ce modèle, les partitions floues des sorties ne sont pas décrites par des fonctions polynomiales mais plutôt par des valeurs constantes. Le remplacement des fonctions du modèle initial par des valeurs fixes, a grandement simplifié la structure du système flou et a réduit également sa complexité de calcul.

$$R_k : SI\ (x\ est\ A_i)\ ET\ (y\ est\ Bj)\ ALORS\ (z\ est\ f_k),\ avec:\ f_k = a_{0,k} \tag{I.19}$$

I.7.2 Modèle flou linguistique

Le modèle Mamdani est le plus ancien d'entre eux tous. Dans la littérature, le nom de Mamdani est régulièrement substitué par le nom linguistique. Et ceci n'est pas sans raison en effet, ce modèle de Mamdani est celui qui s'approche le plus de la pensée humaine et qui soit par conséquent le plus facilement compréhensible.

Cette caractéristique a rendu le modèle de Mamdani très sollicité dans les applications où il est question d'extraire une connaissance à partir d'une collection de données ou dans le cas où la connaissance est détenue par un expert humain. En raison de son étroit lien avec la pensée humaine, il est le plus adapté à interpréter cette expertise. Ces avantages comparés aux autres modèles flous, combinés avec ses performances d'approximations, rendent le modèle de Mamdani à ce jour au centre de nombreuses applications et de recherches notamment la nôtre. Dans la figure ci-dessous, nous avons schématisé le raisonnement flou du modèle linguistique de Mamdani en respectant le même ordonnancement décrit précédemment sur l'évolution du calcul.

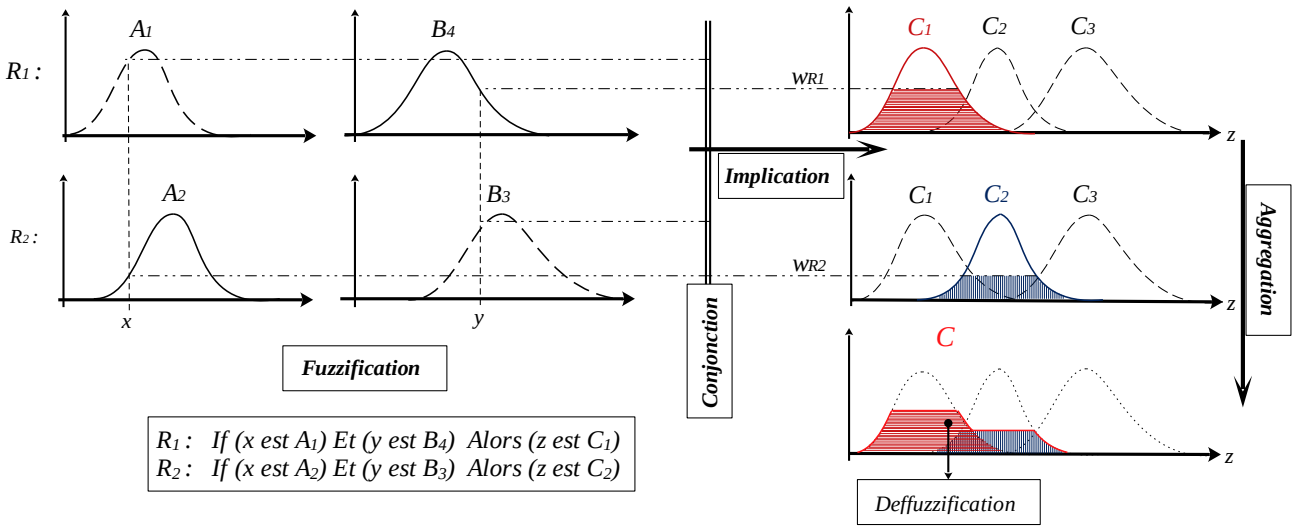


Figure I. 11 : Mécanisme d'inférence floue (Mamdani)

Le second modèle linguistique de Tsukamoto est peu connu dans la littérature. Il est très semblable à celui de Mamdani dans sa forme. En effet le modèle de Tsukamoto utilise aussi des termes linguistiques cependant, ce dernier exige une condition sur la forme des fonctions d'appartenance à utiliser dans les partitions des sorties : *elles doivent être monotones*. Dans la figure (I.12), une représentation graphique du raisonnement flou de ce modèle est donnée en contraste à celui de Mamdani.

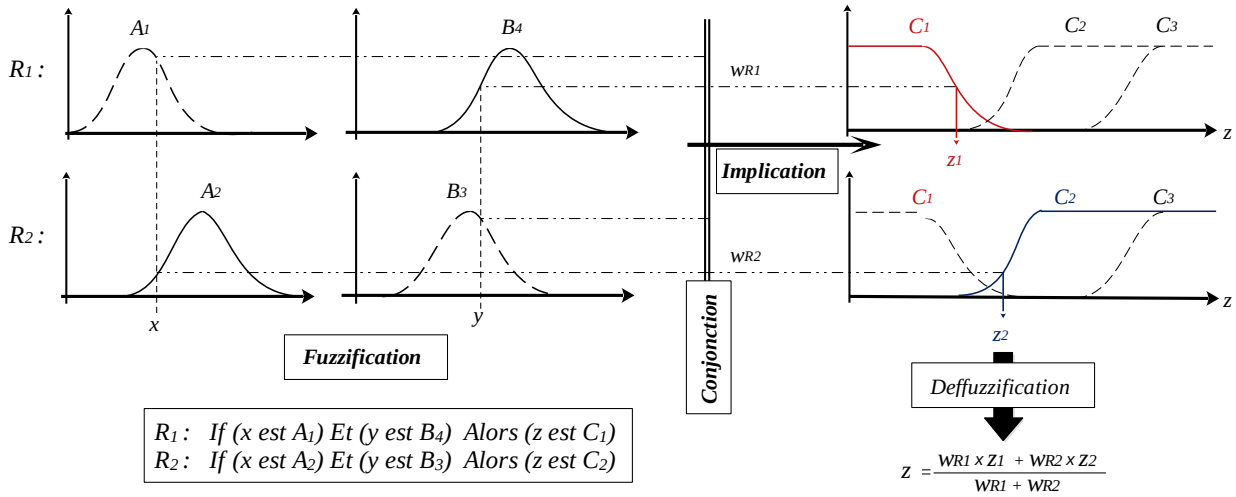


Figure I. 12 : Mécanisme d'inférence floue (Tsukamoto)

Les systèmes flous ont eu un impact considérable sur l'intelligence artificielle. Ils ont permis un traitement symbolique de l'information remplaçant la méthode traditionnelle adoptée bien avant qui est basée sur une approche purement numérique en boite noire par des réseaux de neurones. Et de ce fait, l'intelligence artificielle s'est étendue vers d'autres domaines scientifiques en s'éloignant peu à peu des applications industrielles. Cela n'a pas duré longtemps, quelques années plus tard l'intelligence artificielle a marqué un retour du numérique. Mais cette fois-ci via les réseaux de neurones et les systèmes flous simultanément. Cette démarche combinant l'avantage des deux techniques : la tolérance et la souplesse des systèmes flous et le connexionnisme des réseaux de neurones, rencontre encore une satisfaction grandissante chez les ingénieurs vu que cette démarche rallie la modélisation numérique et la représentation symbolique.

I.8 Systèmes Neuro-flous

Les systèmes Neuro-flous représentent la rencontre du traitement numérique et du traitement symbolique de l'information en intelligence artificielle. Le premier est assuré par les réseaux de neurones et le second est attelé aux systèmes flous. Cette hybridation s'articule de trois façons différentes, la première est dite : *concourante*, la seconde est dite : *coopérative*, tandis que la troisième est dite *fusionnée* ou *Fondue* [23-24].

I.8.1 Structure Neuro-flou concourante

Dans cette première forme d'hybridation, les deux entités intelligentes sont employées séparément et en cascade. Les données relevées du système réel traversent d'abord le réseau de neurones puis le système flou. Cet ordonnancement vise principalement à rendre le système

flou auto-ajustable via le réseau de neurones. Ce dernier en fonction des données mesurées du système, il réajuste les paramètres du système flou. La présente stratégie est employée principalement lorsqu'il est question d'un traitement flou de données difficiles à prévoir au préalable, ou qui changent continuellement. De ce fait, le réseau de neurones assiste le système flou continuellement et tout le temps.

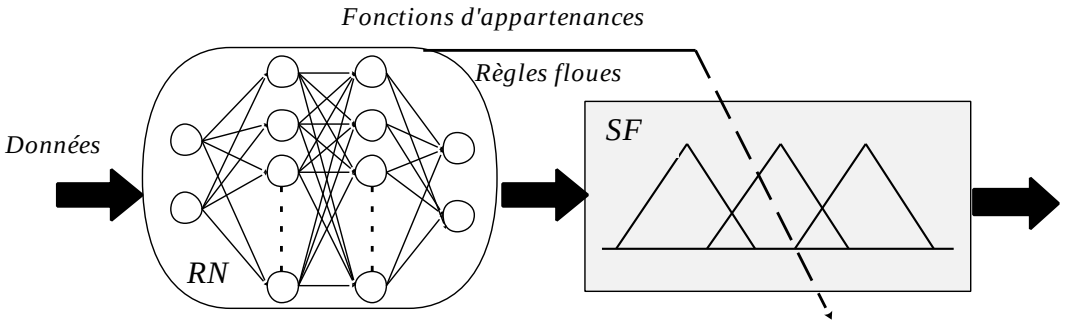


Figure I. 13 : Structure neuro-flou concourante

I.8.2 Structure Neuro-flou coopérative

Cette hybridation ressemble quelque peu à la précédente. Elle considère les deux entités intelligentes toujours séparément, mais l'emploi de ces deux dernières est différent. Ici, le réseau de neurones est utilisé pour un premier ajustement des paramètres du système flou. Une fois cette étape est terminée, le réseau de neurones passe en arrière-plan pour que l'ajustement des paramètres du système flou se poursuit par d'autres techniques tel que : les algorithmes de partitionnement par exemple.

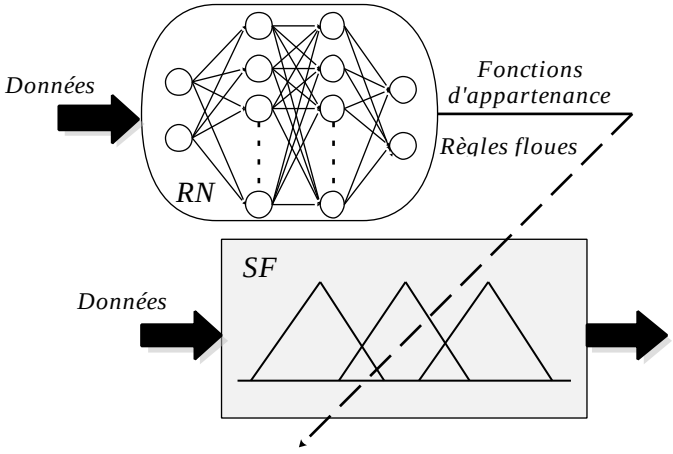


Figure I. 14 : Structure neuro-flou coopérative

I.8.3 Structure Neuro-flou fusionné

Contrairement aux deux premières stratégies, dans cette dernière les deux entités intelligentes ne sont pas considérées séparément. Le système flou est complètement fusionné avec le réseau de neurones et forment une seule et unique entité. L'intérêt de cette stratégie est

de faciliter la recherche des paramètres du système flou en exploitant les algorithmes d'apprentissage des réseaux de neurones, tout en gardant l'aspect flou du traitement des données.

À chacun des types du système flou, ou plutôt à chaque type de conclusion de la règle floue existe une forme d'hybridation qui soit la plus adéquate pour là représentée. L'architecture des hybridations ne se différencie pas uniquement selon le type de la règle qu'elles modélisent, mais aussi selon la taille du réseau qu'elle utilise. Par conséquent, on trouve que les règles relatives aux conclusions linguistiques sont implémentées souvent par des réseaux à cinq couches. Cette remarque peut être facilement constatée en analysant la structure des réseaux neuro-floues dédiés comme le : *FALCON* [25-26] ou le réseau dit : *GARIC* [27-28]. Cependant, les cinq couches, ne sont pas une règle générale, et le réseau : *NEFCON* [29-30] confirme bien ceci. Bien qu'il modélise les mêmes règles linguistiques, il n'emploie cependant que 3 couches.

Pour la seconde classe des règles floues précises, la panoplie de taille est beaucoup plus large. Les règles floues du modèle singleton de Sugeno peuvent être simplement modélisées par un réseau à seulement 3 couches. C'est le cas du modèle appelé : *NEFCLASS* [31-32], ce pendant les règles du modèle flou de type TSK nécessite jusqu'à 6 couches avec le réseau ANFIS [33-34].

D'autres structures existent dans la littérature et celles présentées juste avant ne sont que quelques-unes d'entre-elles. Dans la figure (I.15) nous présentons deux structures neuro-flou correspondant à chaque classe du modèle flou. La première est la structure ANFIS dédiée au modèle précis TSK, tandis que la seconde est la structure FALCON dédiée au modèle linguistique de Mamdani.

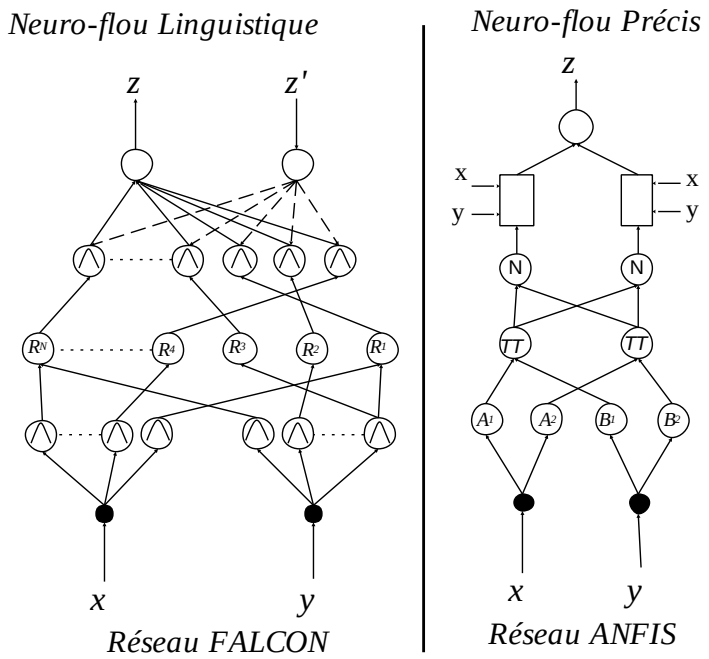


Figure I. 15 : Exemples de réseau neuro-flous fusionnés

I.9 Ingénierie des systèmes flous

À présent que les constituants des systèmes flous, leurs fonctionnements et leurs structures sont présentés, l'une des premières questions qui interpellent l'esprit est : *Comment construire un tel système ?* La réponse à cette question est assez difficile et complexe. Elle l'est effectivement car d'abord, à ce jour il n'existe pas de méthode unique et spécifique pour les concevoir. Ensuite, il est très difficile d'évaluer à priori leur qualité en raison de l'interdépendance complexe entre ses différents constituants, rendant la question de conception de ces systèmes toujours ouverte. Cependant, la démarche à suivre pour les construire demeure la partie de la question qui soit simple.

D'abord, tout commence par le choix du type du système flou à concevoir (*Linguistique* ou *précis*) une fois ce choix est fait, on procède à la définition des partitions floues des entrées/sorties. A ce stade, l'expression des conclusions des règles floues est fixée à priori par le type du modèle choisit au départ, par contre pour ce qui est des entrées il existe une multitude d'approches pour définir ces dernières.

I.9.1 Partitionnement des entrées

La définition des partitions floues des entrées est un problème largement discuté dans la littérature, ses implications et ses complications ont valu le surnom de : *la malédiction de la dimensionnalité* au problème de partitionnement de l'espace des entrées [35-36]. À première vue il s'agit d'un problème de partitionnement, mais qui conditionne fortement la complexité

du système flou et sa compréhension également. En principe, il existe trois méthodes populaires pour y procéder. Chacune d'entre elles repose sur une stratégie bien spécifique, la première est dite : *partitionnement en grille*, la seconde est dite : *partitionnement en arbre* et la troisième est connue sous le nom de : *partitionnement en secteurs.*

Pour expliquer simplement ces différentes méthodes, reprenant le système flou à deux entrées *x* et *y* étudié précédemment. Dans la première méthode, l'espace formé par ces entrées (*x*,*y*) est partagé sous forme d'une *grille*. Chaque côté de ces cellules délimite le support d'un terme linguistique dont le centre correspond au noyau de ce dernier. Dans la seconde méthode, *en arbre*, l'espace des entrées est partagé graduellement. À chaque fois une région est délimitée, ensuite l'espace restant est considéré comme un sous-espace indépendant et le partage reprend de nouveau jusqu'à ce que tout l'espace des entrées soit complètement couvert par des termes linguistiques. Contrairement à ces deux premières méthodes, l'approche de partitionnement *en secteur* partage l'espace des entrées en plusieurs secteurs qui peuvent parfois se recouvrir. Ensuite, à chaque secteur on associe un terme linguistique sur chaque entrée [37-38].

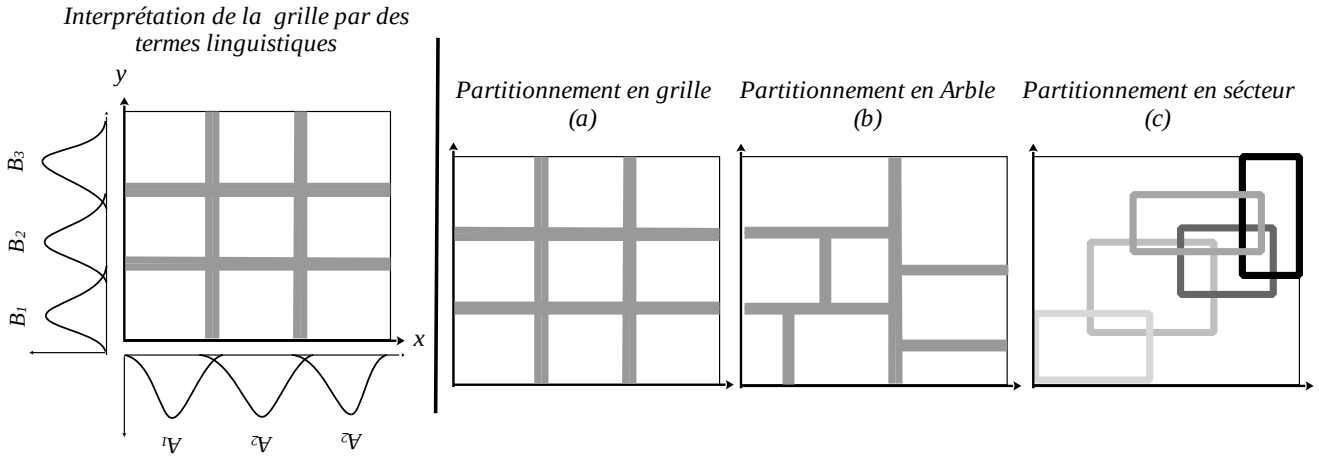


Figure I. 16 : Approches de partitionnement de l'espace des entrées

Dans la partie à droite de la figure ci-dessus, le principe de ces trois méthodes est illustré. Dans la partie à gauche, une explication graphique sur la formation des termes linguistique est donnée pour un partitionnement en grille. Ce mécanisme de formation peut être extrapolé sur les deux autres méthodes de partitionnement également. Dans la littérature, le partitionnement en arbre est peu fréquent, au moment où celui en grille est plus utilisé. Néanmoins, ces deux approches partagent le fait qu'elles engendrent souvent un grand nombre de règles floues surtout si la taille du maillage est très conséquente. Sur ce point, la méthode de partitionnement en secteur est plus intéressante. Vu qu'elle engendre moins de règles, mais au prix d'une complexité à l'emploi.

À ce problème de partitionnement s'ajoute le problème du choix de la forme des fonctions d'appartenance à utiliser pour exprimer les sous-ensembles flous dans chaque partition. Il existe dans la littérature des travaux qui ont approfondi la question en vue d'étudier l'effet que produisit la forme des fonctions d'appartenance sur la performance du système flou. Selon le type du problème traité et surtout selon la nature des données à traiter, certaines formes sont plus adéquates que d'autres. Par exemple la forme triangulaire et trapézoïdale est plus intéressante pour des systèmes flous dédiés au problème de commande. La forme gaussienne par contre, est plus intéressante dans le cas de données statistiques ou probabilistes [39-41]. Mais du point de vue conception il existe également un autre point très important, il s'agit du nombre de paramètres que requiert chaque forme pour la définir complètement. Ainsi on se retrouve avec deux paramètres pour la forme gaussienne, trois pour la forme triangulaire et quatre pour la forme trapézoïdale comme illustré dans la figure (I.17).

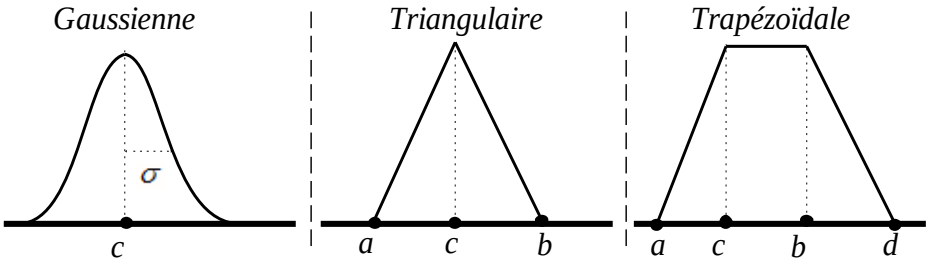


Figure I. 17 : Formes et paramètres de quelques fonctions d'appartenance

D'autre part et toujours dans le même contexte, la définition des limites de l'univers de discours peut être également un enjeu important dans certains problèmes du moment que ces limites affectent systématiquement les paramètres des fonctions d'appartenance. En raison de la difficulté de fixer à priori les limites de cet univers, il est souvent normalisé dans l'intervalle [-1, 1], puis à chaque univers de discours on associe un facteur d'échelle. De cette manière, on ajustant les facteurs d'échelles on ajuste également les limites des univers de discours.

I.9.2 Définition des opérateurs et des règles floues

La définition des règles floues est pratiquement le problème le plus connu des systèmes flous, voir même le plus traité dans la littérature. Pour faire simple, ce problème consiste à chercher les liens de causalité liant l'espace des entrées à celui des sorties. L'approche la plus ancienne pour définir ces règles floues reste fondamentalement l'expertise d'un expert humain qui connaît parfaitement le problème à traiter. Ce dernier, transcrit son expérience sous forme de propositions simples de la forme (*SI-ALORS*) qui sont ensuite adaptées pour se conformer au partitionnement déjà réalisé sur les entrées. Au même temps, l'expert définit également les

opérateurs flous à utiliser en vue d'obtenir une meilleure coopération entre les règles qu'il vient de formuler.

Cette approche est simple et ne nécessite pas beaucoup de ressources, mais elle reste néanmoins sujette à de nombreuses critiques et faiblesses. D'abord la performance du système flou obtenu est directement liée à l'expérience de l'opérateur humain et surtout à son habilité à transcrire ses connaissances sous forme de simples propositions, chose qui n'est toujours pas facile à faire. Ensuite le nombre de paramètres que l'opérateur doit définir est parfois trop grand pour que cela puisse être fait manuellement. D'autre part, cette méthode est assez simple à utiliser uniquement dans le problème de commande. Effectivement, il existe dans la littérature des tables de règles floues prédéfinies pour ce problème toutes basées sur la stratégie proposée par MacVicar-Whelan [42]. Mais même ces tables prédéfinies nécessitent parfois un ajustement qui se fait par série d'essais. Ce n'est pas le cas pour les autres classes de problème auquel le système flou peut être confronté tel que le problème : d'identification, de diagnostic, de classification... auxquels aucune table de règles n'est disponible, ce qui laisse la question sur la définition des règles floues toujours problématique.

I.10 Génération des systèmes flous

Jusqu'ici, nous avons décrit uniquement la démarche de conception des systèmes flous de façon simplifiée et intuitive en soulevant quelques complications relatives à chaque étape. Cependant, nous n'avons pas décrit les critères qui gouvernent chacune de ces étapes de conception. L'observation de ces critères permet de toujours garder en vue les buts principaux à atteindre durant la phase de conception est qui sont : la précision du modèle flou et sa facilité à être interprété. Ce dernier critère forme la particularité du système flou qui fait son exception parmi toutes les techniques de l'intelligence artificielle. Elle reflète sa capacité de traitement symbolique de l'information. Ces critères s'étendent sur tous les éléments constituant le système flou allant des partitions floues des entrées/sorties jusqu'aux règles floues en passant par la définition des opérateurs flous utilisés également.

I.10.1 Critères sur les partitions floues

Les partitions des entrées/sorties floues sont soumises principalement à l'exigence de devoir préserver l'aspect linguistique de la partition et de respecter au mieux son sens sémantique. Ce dernier critère s'exprime de différentes façons, et il existe plusieurs critères

numériques dans la littérature permettant d'estimer la qualité de la partition selon cette exigence. L'idée principale peut cependant se résumer en quelques points essentiels [43].

a. Couverture

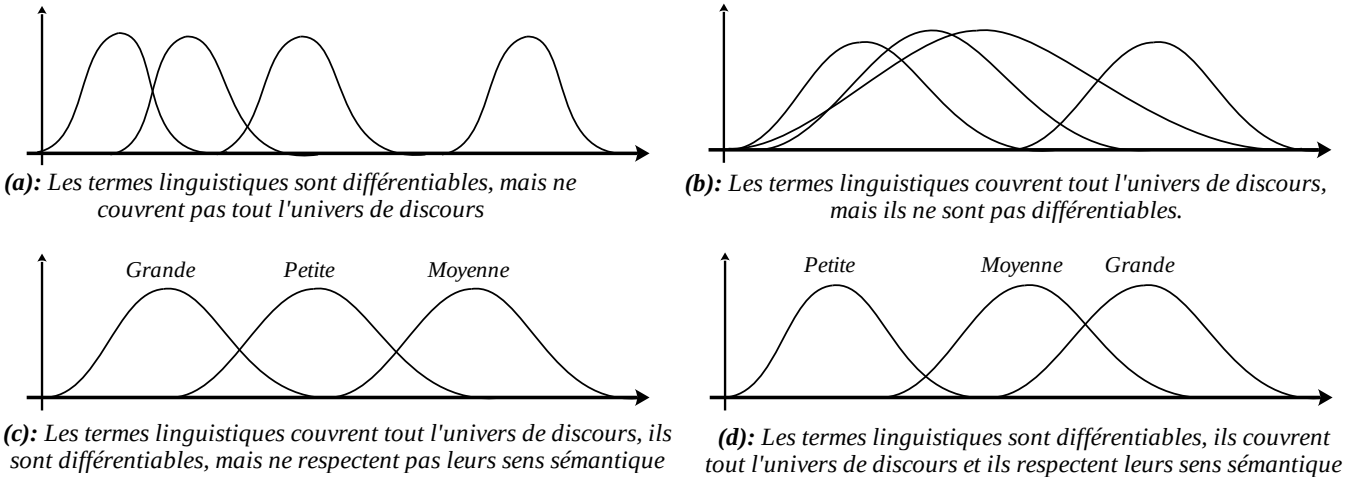
La répartition des sous-ensembles flous ou des termes linguistiques doit garantir la couverture de tout l'univers de discours, sans laisser aucune région de ce dernier non couverte par au moins un terme linguistique (voir figure (I.18-a)).

b. Distinction

Les termes linguistiques utilisés pour la définition de la partition doivent être distinguables et facilement différentiables entres eux. Il n'est pas souhaitable d'avoir des termes linguistiques qui se recouvrent mutuellement comme le cas illustré dans la figure (I.18-(b)), afin d'éviter des ambiguïtés sur la terminologie associée à la partition décrivant la variable floue.

c. Sens sémantique

La position des fonctions d'appartenance sur l'univers de discours doit correspondre et surtout refléter le sens des termes linguistiques qui leur sont associés. Par exemple dans la figure (I.18-(d)), les termes linguistiques utilisés pour décrire la valeur de la variable floue ont des fonctions d'appartenance placées de façon à respecter le sens sémantique des attributs, ce qui n'est pas le cas dans la figure (I.18-(c)).



(a): *Les termes linguistiques sont différentiables, mais ne couvrent pas tout l'univers de discours*

(b): *Les termes linguistiques couvrent tout l'univers de discours, mais ils ne sont pas différentiables.*

(c): *Les termes linguistiques couvrent tout l'univers de discours, ils sont différentiables, mais ne respectent pas leurs sens sémantique*

(d): *Les termes linguistiques sont différentiables, ils couvrent tout l'univers de discours et ils respectent leurs sens sémantique*

Figure I. 18 : Différentes configurations d'une partition floue

I.10.2 Critères sur la base des règles floues

La base de règles est également sujette à de nombreux critères qui lui sont propres, au même titre que les partitions floues. Néanmoins, on peut ramener ces différents critères à quelques-uns seulement qui sont les plus fréquemment mentionnés [44-45].

a. Continuité

Une base de règles est qualifiée de continue, si elle n'admet pas de conclusions adjacentes ayant une intersection vide. Autrement dit, les conclusions de toutes les règles floues forment une partition floue couvrant totalement l'univers de discours de la variable de sortie.

b. Complétude

Une base de règles floue est dite complète si pour toute combinaison dans l'espace des entrées, existe une valeur de sortie. Cette caractéristique garantie d'éviter la discontinuité dans la réaction du système flou. L'incomplétude de la base de règles floues apparaît plus clairement dans les problèmes de commande, sous forme de discontinuité dans la loi de commande.

c. Consistance

La consistance de la base de règles floues est la caractéristique la plus convoitée dans la littérature. Une base de règles floues est dite consistante, si elle ne contient pas de règles contradictoires. Pour faire simple, prenant le cas de deux règles floues seulement R_1 et R_2.

$$\begin{cases} R_1 : SI \ (x \ est \ A_1) \ ALORS \ (y \ est \ B_1) \\ R_2 : SI \ (x \ est \ A_2) \ ALORS \ (y \ est \ B_2) \end{cases} \tag{I.20}$$

On dit que ces deux règles sont contradictoires si elles ont les mêmes prémisses mais des conclusions différentes ou mutuellement exclusives. Ces deux conditions peuvent être exprimées analytiquement comme suit :

$$\begin{aligned} &1) \quad A_1 \cap A_2 \neq \phi \\ &2) \quad B_1 \cap B_2 = \phi \end{aligned} \tag{I.21}$$

Tous les critères présentés juste avant et bien d'autres également sont souvent regroupés sous un nom générique dans la littérature, nommé : *Intérprétabilité*. C'est sans doute l'aspect du problème de conception des systèmes flous le plus difficile à cerner et le plus difficile à formuler. À ce jour, les spécialistes dans ce domaine ne sont pas d'accord sur une définition unique d'une mesure numérique exprimant au mieux cette notion et elle reste encore une question ouverte. D'autre part, ce qui rend encore l'intérprétabilité du système flou difficile à atteindre est le fait qu'elle soit inversement proportionnelle à sa précision. Cette même précision qui est proportionnelle à sa complexité.

Ces complications relatives à la conception des systèmes flous rendent la démarche présentée précédemment dite : *manuelle*, applicable uniquement sur des structures petites du système flou, mais pas du tout adéquate face à une structure conséquente ayant plusieurs variables d'entrées-sorties et de nombreuse règles floues. À la place de cette dernière méthode, les concepteurs des systèmes flous ont tous misé sur la faculté d'apprentissage des machines

pour réaliser automatiquement cette tâche. Dans cette nouvelle voie, tous repose sur l'usage d'algorithmes dans lesquels, la conception du système flou est formulée comme un problème mathématique d'optimisation incorporant les différents critères et les différents objectifs avant d'être soumis à la machine pour le résoudre.

Bien que la conception automatique permette de soulager une grande partie du problème de conception des systèmes flous liée aux calculs répétitifs et complexes, elle reste assez délicate à utiliser. Car justement l'autre partie restante du problème de conception (*qualité du système flou*) est la plus difficile en raison de la subjectivité de ses recommandations. Ce constat à changer la position du problème et de la question posée. Maintenant, la question posée dans l'apprentissage automatique est : *Comment expliquer à la machine ce que c'est qu'un système flou idéal, qui soit précis tout en étant le plus proche possible du raisonnement humain* ?. Tous les travaux traitants le problème d'apprentissage automatique des systèmes flous tournent autour de cette question et tentent effectivement de répondre à cette dernière.

I.11 Conclusion

Tout au long de ce chapitre, nous nous sommes intéressés aux systèmes flous d'un point de vue purement conceptuel. Les quelques notions théoriques fondamentales développées, suffisent à elles seules de se rendre à l'évidence de l'impact qu'ils ont eu sur l'évolution de l'intelligence artificielle. Ils ont fait migrer cette dernière de l'aspect numérique du traitement de l'information à l'aspect symbolique et encore, ils ont également permet un premier rapprochement concret entre la pensée humaine et celle de la machine.

Le retracement de la genèse des systèmes flous présentée dans ce chapitre, vise tout d'abord à assimiler les contrastes entre les différents modèles et structures des systèmes flous. Ensuite, à prendre conscience des questions et complications qui règnent sur leurs conceptions tout particulièrement le modèle linguistique. Les nombreux défis que présente la conception floue a mis à rude épreuve les techniques classiques pour peu de résultats, ce constat a conduit systématiquement à réfléchir sur l'apprentissage machine qui est, soit dite-en passant, la démarche adoptée de nos jours. Tel qu'il est envisagé dans la littérature, l'optimisation est le candidat favorable pour la résolution du problème de conception des systèmes flous. Un domaine ayant ces propres notions et concepts. Par conséquent, il ne nous serait pas possible d'entamer la conception automatique sans passer par l'optimisation, qui sera le sujet du prochain chapitre.

Chapitre II

Optimisation et algorithmes

II.1 Introduction

L'optimisation constitue l'un des champs scientifiques les plus populaires dans le domaine de l'ingénierie. Elle peut être vue comme la démarche de recherche de la meilleure solution offrant l'extremum d'une fonction mathématique.

Durant ce second chapitre, nous exposerons le problème d'optimisation suivant ses types et ses catégories. Dans un premier temps, nous étudierons l'optimisation dite : mono-objectif et les algorithmes dédiés à cette dernière avant d'aborder l'optimisation dite : multi-objectif et les algorithmes propres à cette classe également.

II.2 Problème d'optimisation

Un problème d'optimisation est constitué principalement de quatre composantes. L'espace de recherche du problème, les variables de décision, les contraintes à satisfaire et les fonctions objectif à minimiser ou à maximiser. Il est souvent formulé par l'équation (II.1)

$$\begin{cases} Max / Min & \left(\sum_{m=1}^{M} f_m(\vec{x}) \right) \\ \sum_{j=1}^{J} g_j(\vec{x}) & \leq & 0 \\ \sum_{k=1}^{K} h_k(\vec{x}) & = & 0 \\ \vec{x}_{Min} & \leq & \vec{x}_i & \leq & \vec{x}_{Max} \end{cases} \quad \text{(II.1)}$$

Où : f_m exprime la $m^{ième}$ fonction objectif, M représente le nombre d'objectifs à maximiser ou à minimiser. g_j et h_k : représentent respectivement les contraintes d'inégalité et d'égalité imposées aux variables. $\vec{x}$ représente le vecteur des variables de décision, et $\vec{x}_{Min}, \vec{x}_{Max}$ définissent les limites de l'espace de recherche du problème d'optimisation.

II.2.1 Classification du problème d'optimisation

Dans la littérature, on se retrouve fréquemment avec des appellations diverses du problème d'optimisation. Les différentes désignations servent principalement à définir au mieux le problème d'optimisation en raison des nombreuses configurations que ce dernier peut prendre. Ces classes sont dressées en corrélation avec les quatre composantes, du problème d'optimisation, énoncées dans l'équation (II.1). La figure (II.1) présente une vue générale sur ces classes.

Le problème peut être dit : *mono-objectif* ou *multi-objectif* suivant la valeur de *M*, ou simplement le nombre de fonctions objectif à optimiser. D'autre part, suivant la nature des variables de décisions $\vec{x}$, il peut être classé comme étant : *combinatoire* si elles consistent en une permutation sur un ensemble fini de nombre, *discret* si le vecteur $\vec{x}$ est constitué par des valeurs entières ou *continue* si ce vecteur est composé par des valeurs réelles.

La présence des deux équations (g_j et h_k) ou d'au moins l'une d'entre elles, et aussi un critère de classification. Ainsi, il peut être dit : *problème avec contraintes* quand elles sont présentes et *problème sans contraintes* dans le cas contraire. Un autre critère de classification existe aussi, il s'agit du type des fonctions objectif. Dans le cas où toutes ces fonctions sont linéaires, le problème est dit : *linéaire*. Dans le cas contraire il est dit : *non linéaire*. La taille du vecteur $\vec{x}$ constitue également un critère de différentiation. Le problème sera nommé mono-variable si le vecteur est défini uniquement par une seule valeur ou multi-variable dans le cas contraire. [46]

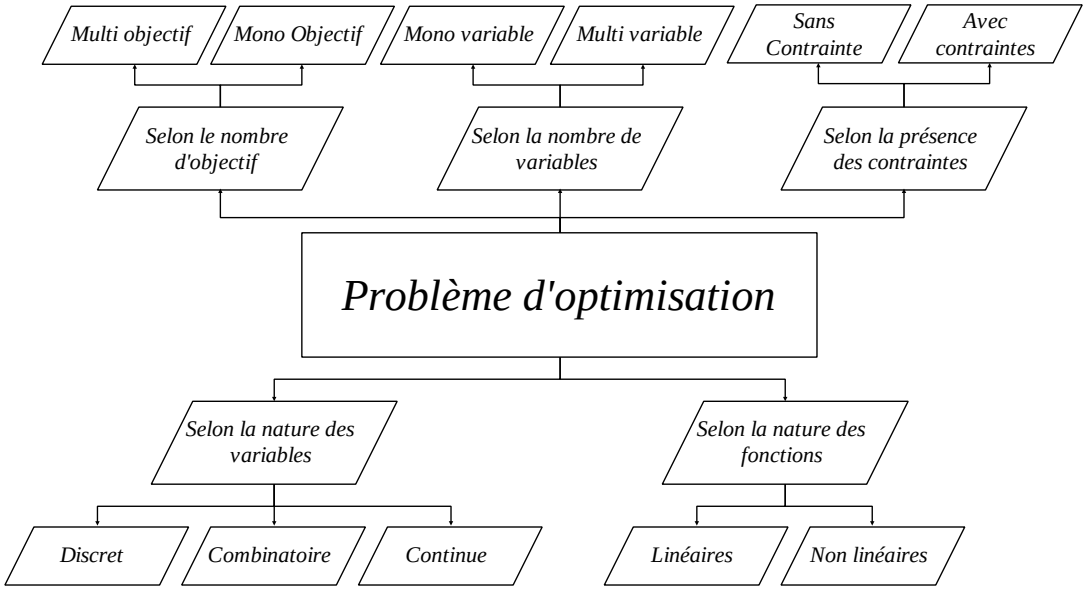


Figure II. 1 : Différente classes du problème d'optimisation

II.2.2 Espace de recherche

Tout problème d'optimisation a son propre espace de recherche. Il s'agit en principe de l'ensemble de solutions candidates du problème**,** défini par l'ensemble des contraintes que comporte le problème et bien sûr les limites de ses variables de décision. La Figure (II.2) présente une interprétation graphique d'un espace de recherche. L'ensemble des valeurs du vecteur $\overrightarrow{x_i}$ comprises entre les limites inférieurs (x_i^{Min}) et supérieurs (x_i^{Max}) forment l'ensemble de solutions potentielles du problème d'optimisation, ici noté: $\mathbb{C}$. Chacune de ces valeurs est vérifiée par rapport aux différentes contraintes (g_j, h_k), celles d'entre elles qui ne vérifient pas ces contraintes seront systématiquement écartées tandis que les autres seront retenues. Ce sont uniquement ces dernières qui seront considérées pour la résolution du problème d'optimisation, elles déterminent l'espace de : *solutions faisables* noté $\mathbb{C}^*$.

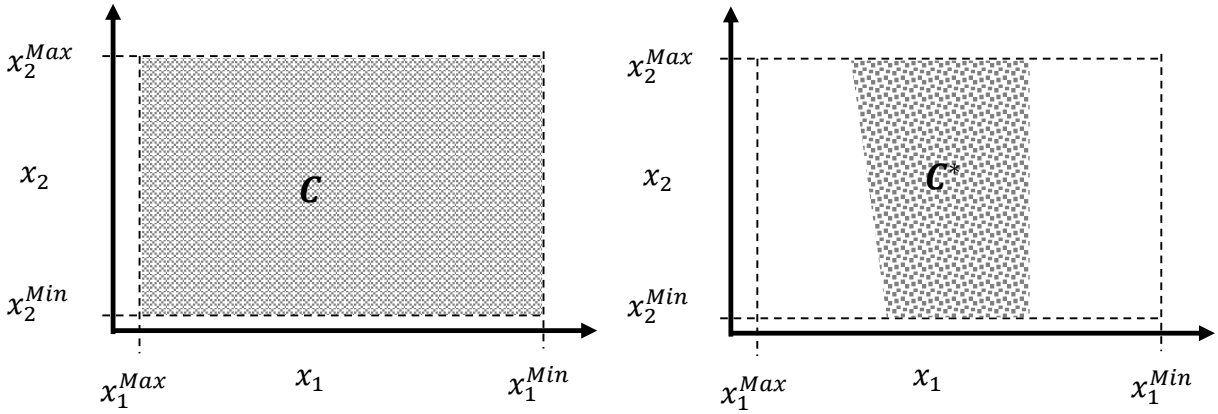


Figure II. 2 : Espace de recherche

Pour la suite de ce chapitre, toutes les notions qui seront présentées et discutées partent de la considération d'un scénario de minimisation. D'un autre côté, nous considérons uniquement la distinction selon le nombre d'objectif que comporte le problème d'optimisation, vu que ce critère est le seul considéré durant la conception des systèmes flous. Ainsi, soit le problème est mono-objectif ou multi-objectif. Chacune de ces deux dernières classes a ses propres notions et concepts toutefois, ceux de l'optimisation mono-objectif sembles beaucoup plus facile à assimiler que ceux de l'optimisation multi-objectif. Pour cette raison, nous poursuivrons d'abord avec l'optimisation mono-objectif avant d'aborder l'optimisation multi-objectif.

II.3 Optimisation mono-objectif

L'optimisation mono-objectif est un concept purement théorique, ou du moins très loin des cas réels. Elle consiste à chercher le minimum d'une seule et unique fonction mathématique exprimant le problème d'optimisation donné par l'équation (II.1) avec M =1.

L'ensemble des solutions faisables définit précédemment, est soumis à la fonction objectif, produisant ainsi une image de l'espace de recherche. Cette image correspond à ce qui est couramment appelé : *espace objectif*. C'est dans cet espace que les solutions sont réellement évaluées en vue de sélectionner celle correspondante à la meilleure d'entre-elles.

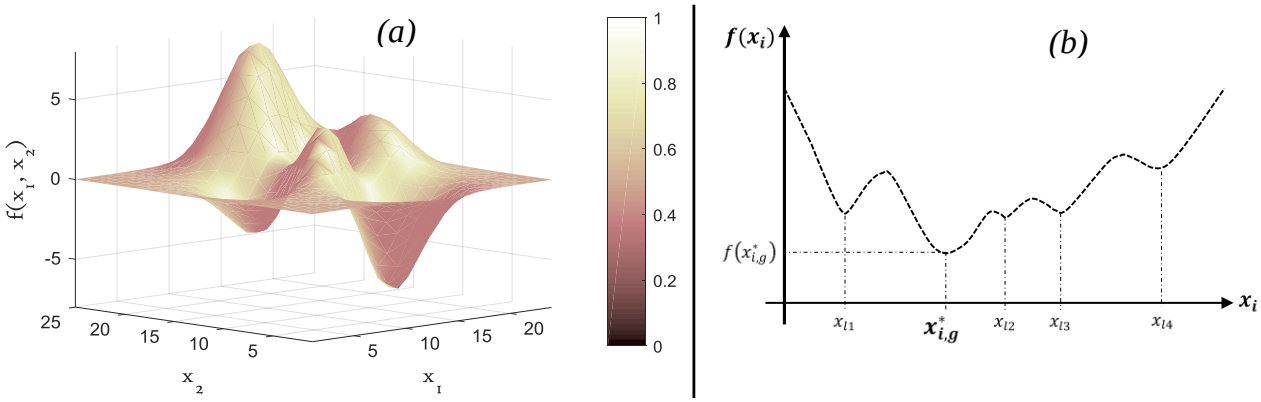


Figure II. 3 : Espace objectif du problème mono-objectif

La figure (II.3) illustre quelques exemples de l'espace objectif, dans la partie (a), le problème comporte deux variables de décisions induisant un espace de recherche bidirectionnel ou sous forme de surface, tandis que dans la partie (b) le problème est mono-variable ce qui engendre un espace de recherche unidirectionnel.

Du point de vue d'une méthode de résolution ou d'un algorithme d'optimisation, il est question de fouiller dans l'espace de recherche faisable du problème pour trouver la solution

pour laquelle la fonction objectif est la plus petite (ou la plus grande dans le cas d'une maximisation). Cette valeur minimale ($x_{i,g}^{*}$ dans la figure II.3-b) est nommée : *le minimum global* et elle est mathématiquement définie comme suit :

$$\forall x_i \in C^* : f\left(x_{i,g}^{*}\right) \leq f\left(x_i\right) \tag{II.2}$$

Cette valeur minimale est plus au moins difficile à localiser en présence de plusieurs vallées plus au moins basses (x_l^*) dites : *minimums locaux.* Selon qu'un algorithme d'optimisation est capable de trouver le minimum global ou seulement les minimums locaux, il sera qualifié soit de *méthode globale* ou de *méthode locale*.

II.4 Algorithmes mono-objectif

La résolution de ce genre de problèmes d'optimisation n'a pas été toujours facile. Entre les problèmes dits combinatoire et continue, les anciennes méthodes et algorithmes d'optimisation se perdaient facilement. Il été très difficile de réconcilier les deux natures, même si une méthode est efficace pour une classe de problème elle reste non opérationnelle pour une autre. Ce dilemme à demeurer longtemps jusqu'à l'émergence d'une nouvelle classe d'algorithmes qui a considérablement simplifié la résolution du problème d'optimisation, il s'agit des *Méta-heuristiques*. En effet, cette nouvelle classe d'algorithmes a réussi à réconcilier les deux natures du problème. Elle s'applique bien pour des problèmes combinatoires et s'adapte aussi bien aux problèmes continus.

II.4.1 Méta-heuristique

Le nom Méta-heuristique est une composition de deux mots « *méta* » et « *heuristique* ». Le premier mot signifie littéralement : *au-delà,* et dans ce contexte : *haut niveau* tandis que le second mot désigne en grec : *trouver*. Alors souvent dans la littérature, la méta-heuritique est définit comme étant : *une méthode générique et stochastique de recherche de solution* [47]. Dans certaines terminologies, le mot : *méta* est interprété comme la capacité d'incorporer d'autres heuristiques locales.

Les méta-heuristiques regroupent plusieurs algorithmes d'optimisation, chacun inspiré d'un domaine scientifique particulier : la théorie d'évolution pour les algorithmes génétiques (AGs) [48], la physique pour l'algorithme du recuit simulé (SA) [49] et l'algorithme de recherche gravitationnelle (GSA) [50], l'éthologie pour l'algorithme des colonies de fourmis (ACO) [51], algorithme de luciole (FFA) [52], algorithme de libellule (DA) [53] et l'essaim de particules (PSO) [54] …. Et il existe encore une large panoplie d'autres algorithmes aussi

différents les uns des autres, qui nous sera très difficile de récapituler en quelques lignes. Néanmoins, dans ce qui suit, nous détaillerons certains de ces algorithmes qui sont les plus représentatifs de l'état de l'art touchant à la conception des systèmes flous.

II.4.2 Algorithmes génétiques (AGs)

Les algorithmes génétiques sont le produit de la théorie d'évolution énoncée par Charles Darwin et formulé par J. HOLLAND en 1975 [48]. Ils reproduisent pratiquement le même mécanisme d'évolution des espèces dans leurs environnements au niveau génétique. À travers une population de solutions potentielles, initialisées généralement de façon aléatoire, l'algorithme génétique utilise différent opérateurs génétiques exprimés sous forme d'équation ou simplement d'algorithmes pour faire évoluer cette population vers la solution optimale. La particularité de cet algorithme est qu'il n'utilise pas les variables elles-mêmes, mais une représentation de ces dernières. Les variables sont représentées par des codes structurés nommés *chromosomes*, imitant ainsi la notion de génotype et phénotype en biologie. Chaque chromosome est d'abord décodé, ensuite soumis à une fonction objectif pour évaluer la performance de l'individu représenté. Cette fonction est souvent formulée comme suit :

$$Fitness = f \circ g \tag{II.3}$$

La fonction *f* exprime directement la performance visée et la fonction *g :* représente l'adaptation de la fonction *f soit* à une minimisation ou à une maximisation. Chaque individu, à travers sa performance ou *fitness*, participe dans la compétition à la survie. Cette compétition est l'opérateur dit : *sélection*. Différentes méthodes peuvent être employées pour cet opérateur, les plus connues restent la sélection par tournoi et la roulette biaisée de Goldberg [55]. Les individus sélectionnés joueront le rôle de *parents* géniteurs, par croisement de leur codes génétiques (*chromosomes*) ils donnent naissance à une nouvelle population appelée *Enfant.* Tout comme dans la nature, la reproduction est parfois soumise à des perturbations dans la transcription des codes génétiques. Cette perturbation est interprétée par l'opérateur de mutation.

Ces deux derniers opérateurs (croisement et mutation) dépendent fortement du codage utilisé pour la représentation des individus. Par exemple, pour un codage en binaire ou en entier, le croisement en un point ou multi-points [56] est souvent employé. Par contre, dans le cas du codage réel, il existe une multitude d'opérateurs de croisement comme, le croisement barycentre [57], le croisement Laplacien [58] etc. De même pour la mutation, elle consiste simplement à complémenter à 1 chaque bit du chromosome, si le code est binaire ou un tirage

au sort dans la plage qui définit les entiers contenus dans le chromosome si le code est entier. Dans le cas du codage réel, la procédure de mutation est beaucoup plus complexe et on peut citer entre autres la mutation non-uniforme [59] et la mutation de puissance [60]. Ces deux opérateurs sont sujet à des probabilités de réalisation notées respectivement par *: Pc* et *Pm.*

Enfin, les deux populations (enfants et parents) sont combinées pour former la nouvelle génération. Ainsi, les individus les moins performants se voient disparaître au fur et à mesure que les générations se succèdent. Dans le but d'améliorer d'avantage la robustesse de cet algorithme, une procédure d'élitisme et aussi incorporée. Elle consiste en principe à sauvegarder un individu ou plusieurs individus de la génération précédente pour les insérer dans la nouvelle génération, si aucun des nouveaux individus n'est plus performant qu'eux. Dans l'algorithme-1, un pseudo-code de l'algorithme génétique est présenté résumant l'essentiel des étapes à suivre pour son implémentation.

Algorithme-1 : Pseudo-code d'un Algorithme génétique

```
01 : Définition de la fonction objectif : F(x),
02 : Définition des opérateurs génétiques et leurs probabilités respectives (Ps, Pc, Pm)
03 : Initialisation les chromosomes (hi), mettre le compteur de génération à 1
04 : Faire :
05 : Décodage des chromosomes
06 : Evaluation de la Fitness des individus représentés
07 : Sélection des parents
08 : Croisement des parents
09 : Mutation des enfants
10 : Jusqu'à ce qu'un critère d'arrêt soit atteint
```

L'algorithme génétique est sans doute la méta-heuristique la plus populaire dans la littérature et la plus sollicitée pour la résolution des différents problèmes d'optimisation. Sa popularité est justifiée, vue que ce dernier est pratiquement adaptable à toute forme et toute nature des problèmes d'optimisation. Une analyse un peu plus profonde du fonctionnement de cet algorithme fait émerger deux sous-mécanismes qui font sa puissance à savoir : le mécanisme d'exploration et celui d'exploitation. Le premier permet à l'algorithme de découvrir de nouvelles régions dans l'espace de recherche et par la même occasion de s'extraire des optimums locaux, tandis que le second lui procure la possibilité d'investiguer en profondeur la nouvelle région découverte.

II.4.3 Algorithme d'essaim de particules (PSO)

L'algorithme d'essaim de particules est aussi une heuristique utilisant une population de solutions potentielles, proposé par J. Kennedy et RC. Eberhart en 1995 [54]. Chaque solution est appelée : *particule* et l'ensemble est appelé : *Essaim,* d'où son nom. Contrairement aux AGs,

le PSO n'utilise pas d'opérateurs complexes et n'a pas besoin de beaucoup de paramètres à initialisés, ce qui constitue une différence significative par rapport aux AGs. En plus de cette simplicité de mise en œuvre, il est également très intuitif ce qui justifie son attraction immense dans la littérature.

Dans cet algorithme, chaque particule P_i est caractérisée par deux attributs : la vitesse V_i et la position P_i. D'autre part, l'essaim est structuré suivant une topologie bien particulière qui régit les différentes interactions entre les différentes particules. Dans la figure (II.4), quelques topologies les plus populaires dans la littérature sont illustrées. Cependant, des études analysant l'influence de la forme de la topologie sur la performance de l'algorithme ont montré que la structure en anneau est beaucoup plus intéressante que les autres [61].

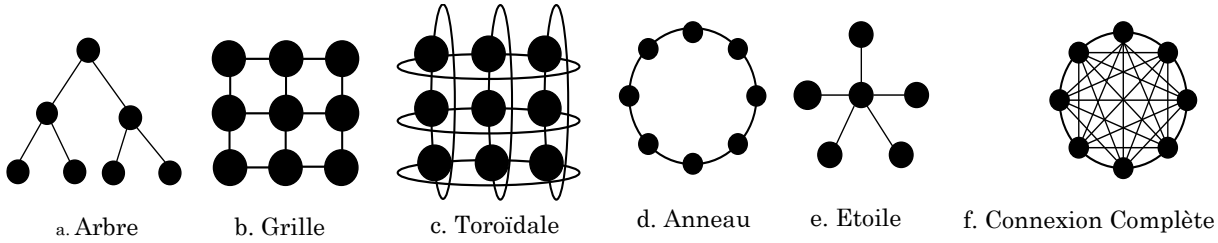


Figure II. 4 : Topologies d'un essaim de Particule

Par une simple formule mathématique, l'algorithme PSO déplace ses particules à travers tout l'espace de recherche en ajustant successivement leurs vitesses puis leurs positions. Cette équation est référée comme l'équation de mise à jour :

$$\begin{cases} V_i^{k+1} = V_i^k + c_1 r_1 (P_i^{k,best} - P_i^k) + c_2 r_2 (P_k^{Global} - P_i^k) \\ P_i^{k+1} = P_i^k + V_i^{k+1} \\ i = 1..N \quad k = 1..T_{\max} \end{cases} \tag{II.4}$$

Avec V_i^k, P_i^k représentent respectivement la vitesse et la position de la particule (i) à la $k^{ième}$ itération et T_{max} le nombre maximum d'itérations. Le terme $P_i^{k,Best}$ représente la meilleure position passée de la particule (i), tandis que $P_k^{Globale}$ représente la meilleure position trouvée par son voisinage qui dépend de la topologie régissant l'essaim. c_1, c_2 constituent les paramètres intrinsèques du PSO appelés coefficients d'accélération ou d'apprentissage et r_1, r_2 sont des variables aléatoires définies entre [0, 1] et N est la taille de l'essaim. Le fonctionnement de l'algorithme PSO standard est illustré dans le pseudo-code donné dans l'algorithme-2 ci-après.

Algorithme-2: Pseudo-code de l'algorithme PSO standard	
01:	*Définition de la fonction objectif : F(x), x=(x1, x2, x3,..., xD)*
02:	*Initialiser les positions et les vitesses pour chaque particule et mettre itération =1*
03:	*Initialiser les meilleures positions personnelles des particules à leurs positions initiales*
04:	***Faire***
05:	*Evaluer toutes les positions des particules*
06:	*Mettre à jour les vitesses de toutes les particules*
07:	*Mettre à jour les positions de toutes les particules*
08:	*Mettre à jour les meilleures positions personnelles de toutes les particules ainsi que la meilleure position de tout l'essaim.*
09:	***Jusqu'à ce qu'un critère d'arrêt soit atteint***

La figure ci-dessous présente une interprétation graphique de l'équation de la vitesse (II.4). En traits discontinus sont représentées les distances séparant la particule (*i*) de sa meilleur position dans le passé ($P_i^{k,best}$) et de la meilleure position trouvée par son voisinage (P_k^{Global}). Le premier vecteur (V_i^k) représente la vitesse initiale de la particule lui permettant de ne pas changer brusquement de direction, préservant ainsi la stabilité du processus de recherche. Le second vecteur $c_1 r_1 \left(P_i^{k,Best} - P_i^k\right)$ exprime l'aspect cognitif de la particule. Il modélise en principe la tendance de chaque particule dans l'essaim à revenir à sa position précédente la plus satisfaisante. Le troisième et dernier vecteur $c_2 r_2 \left(P_k^{Globale} - P_i^k\right)$ exprime l'aspect social de l'algorithme PSO et modélise le partage d'information entre les particules.

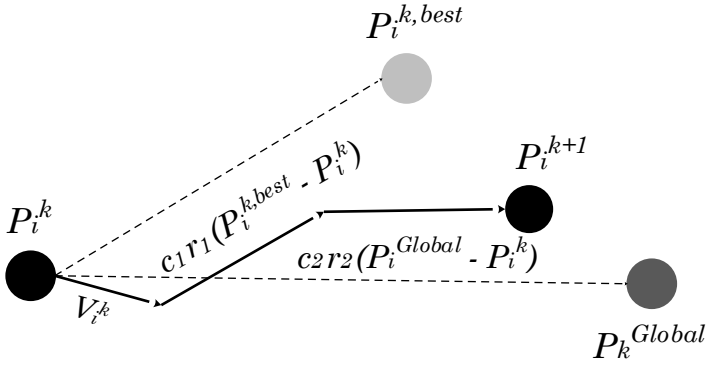


Figure II. 5 : Mécanisme de l'algorithme PSO standard

Dans cette formulation standard de l'algorithme PSO, il est très difficile de déceler les deux mécanismes vus précédemment dans l'algorithme génétique (*Exploration et exploitation*). La nuance floue entre ces deux mécanismes constitue l'une des raisons qui affaiblit l'algorithme dans sa version standard [62]. Actuellement, l'amélioration de ces deux mécanismes dans l'algorithme PSO est devenue le principal objectif de plusieurs travaux de recherche. On trouve par conséquent une large gamme de versions de l'algorithme PSO dans la littérature mais toutes visent à améliorer ces deux mécanismes.

Dans certains travaux, les auteurs ont proposé un ajustement dynamique du jeu de paramètres de l'algorithme [63-66]. D'autres par contre ont proposé d'ajouter de nouveaux

paramètres à l'algorithme [67-68]. Contrairement aux cas précédents, d'autres auteurs ont adressé le problème d'un autre point de vu, ils ont proposé d'améliorer l'aspect social et/ou cognitif de l'algorithme [69-75]. Certains ont même introduit des hybridations de l'algorithme PSO avec d'autres heuristiques afin d'y remédier [76-83]. D'autres travaux ont proposé des approches reposantes sur l'usage de plusieurs essaims distincts simultanément.

II.4.3.1 Algorithme à essaim multiple

L'usage de plusieurs essaims dans l'algorithme PSO est inspiré principalement des symbioses observées dans le monde du vivant comme le : *mutualisme*, *commensalisme,* et le *parasitisme* [84]. Tous ces phénomènes peuvent être implémentés dans l'algorithme PSO, mais le commensalisme reste le plus adapté à cet algorithme [85]. Ce choix est justifié car l'idée même de cette symbiose est d'encourager la coopération entre les individus sans pour autant nuire à aucun d'entre eux. D'autre part, elle est beaucoup plus compatible avec le mécanisme intrinsèque du PSO renforçant ainsi les mécanismes déjà existant dans l'algorithme.

La stratégie d'essaim multiple emploie plusieurs essaims séparés qui coopèrent et communiquent ensemble pour aboutir à la solution optimale ou du moins l'approximer au mieux. Un rapide regard sur l'état de l'art de l'algorithme PSO, permet de se rendre en compte que cette stratégie est considérée comme une solution potentielle pour améliorer la capacité de recherche de l'algorithme PSO. Cependant, elle reste une stratégie assez complexe à employer ou à améliorer également. Car elle nécessite à la fois l'élaboration d'un mécanisme d'interaction sociale entre les particules d'un même essaim et entre les essaims eux-mêmes, en plus de confectionner un plan pour le partage d'information au sein de l'ordre social établi.

L'implémentation de cette stratégie à essaim multiple dans l'algorithme PSO se fait de différentes façons, mais elles peuvent être regroupées en deux grandes familles. La première considère que toutes les particules des différents essaims sont de la même espèce ayant les mêmes habilités et les mêmes caractéristiques, tandis que pour la seconde famille les particules sont issues d'espèce différentes ayant des capacités et des caractéristiques différentes. L'algorithme proposé par X. Zhao [86] constitue un exemple de la première approche et l'algorithme proposé par J. Jie [87] représente la seconde approche. Dans ce dernier, deux espèces nommées (*S1*) et (*S2*) collaborent ensemble pour achever le processus d'optimisation. La première espèce (*S1*) est principalement dédiée à l'exploration de l'espace de recherche visant à découvrir de nouvelles régions, tandis que la seconde (*S2*) est consacrée à l'exploitation des régions trouvées par la première espèce.

Durant cette thèse, nous avons proposé une version dans cette deuxième catégorie rivalisant même avec les plus récents algorithmes à essaim multiple. Notre proposition [88] s'inspire de l'organisation sociale adoptée par les sociétés du vivant. L'observation de ces sociétés permet de déceler un ordre bien déterminé entre les différents individus d'un même groupe et même entre les différents groupes constituant la société. Cet ordre social conditionne les interactions entre les individus du même groupe et les interactions entre les différents groupes également par des règles basées sur la performance de chacun. L'algorithme proposé utilise un environnement multi-rangs pour les individus au sein de leur groupe respectif et multi-niveaux pour leurs groupes également. Ces niveaux et rangs attribués ne sont pas immuables, ils changent continuellement en fonction de la performance des groupes et des individus à chaque itération.

L'algorithme commence par partager l'espace de recherche en (*Ns*) régions, chacune recevra un essaim uniformément réparti sur elle. Et à chaque itération, tous ces essaims participent à une compétition visant à designer le groupe *leader* qui va piloter le processus de recherche. Une fois toutes les particules sont soumises à la fonction objectif, la moyenne de la performance (*équation II.5*) des particules constituant chaque essaim forme le score avec lequel participe l'essaim dans la compétition.

$$Q(Sj) = \frac{1}{|Sj|}\sum_{h=1}^{|Sj|} F\left(P_{Sj,h}^{k}\right) \tag{II.5}$$

$Q(S_j), F\left(P_{Sj,h}^{k}\right), |S_j|$ correspondent respectivement à la performance moyenne de l'essaim (*Sj*), à la valeur de la fonction objectif de la particule (*h*) à l'itération (*k*) et au cardinal de l'essaim (*Sj*). Dans la figure (II.6) une représentation de la structure sociale adoptée en plusieurs niveaux pour les essaims et en plusieurs rangs pour les particules est présentée.

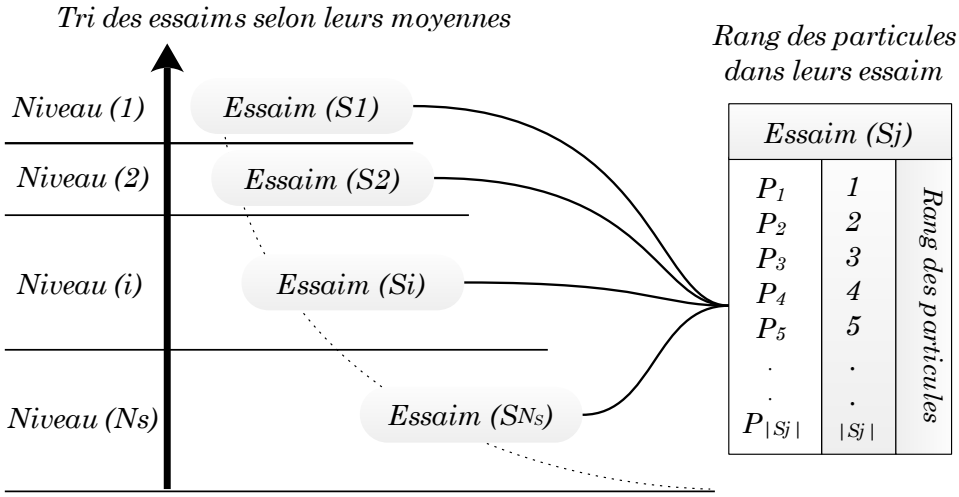


Figure II. 6 : Organisation social de l'algorithme proposé

L'usage de la moyenne des performances des particules pour déterminer l'essaim leader permet de garantir que l'essaim choisi pour piloter le processus de recherche dispose de toutes les meilleures sources d'informations disponibles dans la population. Cet essaim leader est dédié à l'exploration et la découverte de nouvelles zones dans l'espace de recherche. Pour augmenter cette capacité, la stratégie de perturbation est utilisée sur la meilleure position trouvée par cet essaim. Par conséquent la mise à jour des particules constituant l'essaim leader se fait de la manière suivante :

$$\begin{cases} V_{SL,i}^{k+1} = w_k V_{SL,i}^{k} + c_1 r_1 (P_{SL,i}^{k,best} - P_{SL,i}^{k}) + c_2 r_2 (P_{k,A}^{Global} - P_{SL,i}^{k}) & if\ (\alpha \geq 0.5) \\ V_{SL,i}^{k+1} = w_k V_{SL,i}^{k} + c_1 r_1 (P_{SL,i}^{k,best} - P_{SL,i}^{k}) + c_2 r_2 (P_{k,P}^{Global} - P_{SL,i}^{k}) & if\ (\alpha < 0.5) \\ \alpha \in [0,1] \end{cases} \quad \text{(II.6)}$$

Avec: $P_{k,A}^{Globale}, P_{k,p}^{Global}$ représentent respectivement la position exacte et la position perturbée avec une incertitude $\sigma \in [1e^{-3}, 0.8]$ calculée comme suit :

$$\begin{cases} P_P^{Global} = (-1)^i \sqrt{-2\sigma^2 \ln(\lambda)} + P_A^{Global} \\ \lambda \in [0,1] \end{cases} \quad \text{(II.7)}$$

La perturbation de la meilleure position de l'essaim leader, permet d'améliorer le mécanisme d'exploration sans pour autant nuire au processus d'optimisation globale de l'algorithme. Les nouvelles positions des particules dans cet essaim leader sont déterminées de la même façon qu'un PSO standard :

$$P_{SL,i}^{k+1} = P_{SL,i}^{k} + V_{SL,i}^{k+1} \quad \text{(II.8)}$$

Une fois l'essaim leader est mis à jour, sont rayon d'activité sera utilisé pour confiner les essaims subordonnés en leurs imposant ce rayon comme une limite à leurs vitesses. Quant

à leurs positions, elles sont mises à jour différemment. Pour chaque particule (*h*) dans les essaims subordonnés (*Sj*), son expérience personnelle est combinée avec une position virtuelle représentant la tendance de l'essaim leader du point de vue de la particule considérée. La formule suivante montre en détail le calcul relatif à la mise à jour de ces essaims :

$$\begin{cases} V_{Sj,h}^{k+1} = w_k V_{Sj,h}^{k} + c_1 r_1 (P_{Sj}^{k,best} - P_{Sj,h}^{k}) + r_2 \phi_{Sj} (\gamma_h - P_{Sj,h}^{k}) \\ P_{Sj,h}^{k+1} = P_{Sj,h}^{k} + V_{Sj,h}^{k+1} \\ j = 1,\ldots, Ns \\ r_1, r_2 \in [0,1] \end{cases} \quad \text{(II.9)}$$

Deux paramètres auto-adaptatifs sont ajoutés à l'équation de mise à jour des essaims subordonnés, permettant ainsi de calibrer le lien qu'auront ces particules avec celles de l'essaim leader. Ces deux paramètres sont calculés de la façon suivante :

$$\begin{cases} \phi_{Sj} = 1 - \left(\dfrac{F_{Sj}^{Best}}{Max[F^{Best}(S_1,\ldots,Sj,\ldots,S_{Ns})]} \right) \\ \gamma_h = \dfrac{1}{|SL|} \displaystyle\sum_{i=1}^{|SL|} \dfrac{F\left(P_{Sj,h}^{k}\right)}{Max\left(F_{Sj}\right)} \left(P_{SL,i}^{k} - P_{Sj,h}^{k}\right) \end{cases} \quad \text{(II.10)}$$

Avec : $F_{(.)}^{Best}, F(.), |.|$ sont respectivement le meilleur score de l'essaim(.), la valeur de la fonction objectif et le cardinal de l'essaim. ϕ_{Sj} et γ_h sont les deux paramètres propres à cet algorithme, le premier est un coefficient de pondération qui est utilisé comme une accélération pour apprendre du groupe leader, tandis que le second est une position virtuelle représentant la tendance globale de l'essaim leader. Un pseudo-code récapitulant l'approche proposée est donné dans l'algorithme-3.

Algorithme-3: Pseudo-code l'approche proposée

Définition de la fonction objectif : F(x), x = (x1, x2, x3,...xD) ^{T}
Diviser l'espace de rechercher en plusieurs sous-espace.
Initialiser chaque sous-essaim dans une seule région
Faire :
Evaluer toutes les positions des particules,
Calculer le score moyen de tous les essaims
Trier le score des essaims et trouver le leader d'entre eux
Essaim Leader :
    - Mettre à jour les vitesses des particules dans l'essaim Leader
    - Mettre à jour les positions des particules dans l'essaim Leader
Essaim Subordonné :
    - Calculer les paramètres (ϕ, γ)
    - Mettre à jour les vitesses des particules
    - Appliquer le confinement sur la vitesse des particules
    - Mettre à jour la position des particules
Mettre à jour les bests locaux des particules
Jusqu'à ce qu'un critère d'arrêt soit atteint

II.4.3.2 Etude expérimentale

Pour situer notre proposition parmi celles qui la précèdent, nous avons réalisé une étude comparative avec quelques algorithmes les plus représentatifs de l'état de l'art du PSO comme : GPSO [89], LPSO [90], FIPSO [91-92], CLPSO [93], DMSPSO [94], SLPSO [95] et MCPSO [63]. Ces algorithmes en plus du nôtre, ont été testés sur un ensemble de 10 fonctions de test conçues pour le CEC'2008 [96]. Cet ensemble contient des fonctions dites : uni-modales et d'autres dites multimodales. Le test est réalisé comme suit : chaque algorithme est exécuté 30 fois sur une dimension de 30^D avec un critère d'arrêt fixé à 200.000 évaluations de la fonction objectif.

Les résultats de simulation sont donnés dans le tableau (II.1). Chaque ligne de ce tableau correspond à une fonction spécifique et chaque colonne correspond à chaque algorithme. Les meilleurs résultats obtenus pour chaque fonction sont indiqués en gras et la dernière colonne du tableau illustre le classement de notre algorithme parmi tous les autres algorithmes utilisés durant cette étude comparative.

À partir du tableau (II.1) on peut facilement constater le potentiel de l'approche proposée et sa capacité à améliorer la performance de l'algorithme PSO. En effet, sur les 10 fonctions test, notre algorithme a réussi à obtenir pratiquement le minimum global sur 8 fonctions, ce qui est une performance considérable.

Tableau II. 1: Performance de notre algorithme en comparaison avec les 7 autres versions du PSO

		GPSO	LPSO	FIPSO	CLSPO	DMSPSO	SLPSO	MCPSO	MCSPSO	Rank
F_1	**Mean**	1.25E-61	8.48E-35	6.20E-70	4.76E-19	3.30E-14	4.24-E-90	0.00E+00	**0.00E+00**	**1/8**
	(Std)	2.82E-61	2.85E-34	1.44E-69	1.92E-19	1.27E-13	5.26E-90	0.00E+00	**0.00E+00**	
F_2	**Mean**	8.49E-07	4.42E-05	2.37E+00	4.31E+00	1.90E+00	1.17E-24	4.29E-89	**0.00E+00**	**1/8**
	(Std)	1.01E-06	2.32E-05	1.17E+00	6.84E-01	7.85E-01	8.37E-25	2.10E-88	**0.00E+00**	
F_3	**Mean**	7.33E+00	6.67E-01	1.13E-38	7.54E-12	8.48E-11	1.50E-46	0.00E+00	**0.00E+00**	**1/8**
	(Std)	1.39E+01	2.58E-00	5.7E-39	2.50E-12	8.37E-25	5.34E-47	0.00E+00	**0.00E+00**	
F_4	**Mean**	4.22E+03	3.65E-01	1.21E+00	1.13E+03	9.79E+01	4.66E-07	3.35E-60	**0.00E+00**	**1/8**
	(Std)	5.08E+04	3.83E-01	6.59E-01	2.89E+02	7.31E+01	2.48E-07	5.72E-60	**0.00E+00**	
F_5	**Mean**	6.05E+03	5.18E+01	3.53E+01	9.28E+00	5.60E+01	2.15E+01	**6.12E+00**	2.72E+01	4/8
	(Std)	2.32E+04	3.68E+01	2.71E+01	1.03E+01	3.28E+01	3.41E+00	**1.09E+01**	1.10E-01	
F_6	**Mean**	4.65E+01	5.02E+01	3.86E+01	5.83E-09	2.70E-13	1.55E+01	7.08E-06	**0.00E+00**	**1/8**
	(Std)	2.55E+01	2.25E+01	1.04E+01	5.02E-09	8.41E-13	3.19E+00	3.46E-05	**0.00E+00**	
F_7	**Mean**	1.21E-02	2.46E-03	2.07E-13	8.40E-12	1.76E-02	0.00E+00	1.91E-14	**0.00E+00**	**1/8**
	(Std)	1.58E-02	6.64E-03	5.03E-13	1.45E-11	2.56E-02	0.00E+00	3.26E-14	**0.00E+00**	
F_8	**Mean**	1.36E-14	7.67E+00	6.69E-15	2.99E-10	6.11E-09	5.51E-15	6.38E-12	**8.88E–16**	**1/8**
	(Std)	4.34E-15	9.97E+00	1.83E-15	9.47E-11	1.89-08	1.59E-15	5.09E-11	**0.00E+00**	
F_9	**Mean**	5.62E+03	3.07E+03	2.98E+03	**6.06E-13**	5.74E-08	1.50E+03	1.32E-03	4.39E+00	4/8
	(Std)	2.91E+03	7.80E+02	7.78E+01	**8.88E-13**	6.02E-10	9.10E+01	2.16E-03	1.50E+03	
F_{10}	**Mean**	7.32E-04	7.32E-04	1.35E-32	3.31E-19	1.47E-03	1.35E-32	2.54E-08	**0.00E+00**	**1/8**
	(Std)	2.84E-03	2.84E-03	2.83E-48	8.67E-20	3.87E-03	0.00E-00	2.61E-21	**0.00E+00**	

II.5 Optimisation multi-objectif

L'optimisation multi-objectif constitue le problème le plus étudié dans le monde de l'ingénierie durant ces dernières années. De par sa capacité à considérer plusieurs objectifs simultanément, elle formalise parfaitement les problèmes issus du monde réel. Mathématiquement, ce problème est exprimé par la même équation (II.1) vue précédemment, avec $M \geq 2$. La première différence avec l'optimisation mono-objectif est que dans l'optimisation multi-objectif, il n'y a pas qu'une solution optimale unique mais plutôt un ensemble de solutions formant les meilleurs compromis entre les différents objectifs considérés. Par conséquent la notion d'optimalité n'a plus de raison d'être du moins dans sa forme classique, on parle plutôt de dominance et d'optimalité au sens de Pareto. Il existe une littérature immensément grande à ce sujet. Néanmoins, nous présenterons dans la suite de ce chapitre les quelques notions les plus fondamentales permettant de cerner le problème d'optimisation multi-objectif ainsi que les algorithmes dédiés.

Définition 1 : ***Dominance :*** *On dit que le vecteur $\vec{x}_1$ domine le vecteur $\vec{x}_2$ noté $(\vec{x}_1 \prec \vec{x}_2)$ ssi :*

$$\begin{cases} \forall\ m \in \{1, M\} & f_m(\vec{x}_1) \leq f_m(\vec{x}_2) \\ \exists\ m \in \{1, M\} & f_m(\vec{x}_1) < f_m(\vec{x}_2) \\ \vec{x}_1, \vec{x}_2 \in C* & \end{cases} \qquad \text{(II.11)}$$

La dominance est la notion fondamentale de l'optimisation multi-objectif. Elle exprime un ordre de préférence entre plusieurs solutions vis-à-vis de plusieurs fonctions objectif simultanément. L'usage de cette notion conduit systématiquement à un ensemble de solutions qui forme les meilleurs compromis entre tous les objectifs considérés, cet ensemble-là est connu sous le nom de : *front de Pareto* que nous verrons en détail un peu plus loin dans les paragraphes suivants.

La figure (II.7) ci-dessous, présente une interprétation graphique de cette notion pour un problème d'optimisation à deux objectifs. Ici cinq solutions (x_1, x_2, x_3, x_4, x_5) sont considérées, elles sont représentées sur la figure par leurs images respectives (*a, b, c, d, e*) dans l'espace objectif. Selon la définition-1, les images *a* et *e* correspondent aux solutions non-dominées de ce problème tandis que : *d, b* et *c* correspondent aux solutions dominées [97].

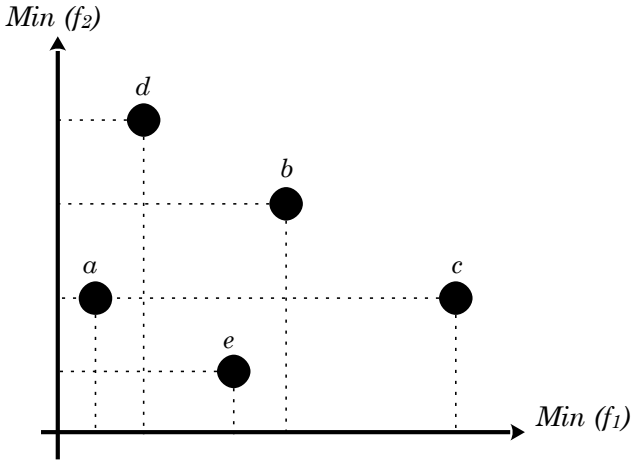


Figure II. 7 : Notion de dominance

II.5.1 Front de Pareto

Le front de Pareto et/ou surface de compromis, peut être défini comme étant l'ensemble des images des solutions non-dominées du problème multi-objectif. Dans la figure (II.7), les solutions x_1 et x_5 ayant les images (*a* et *e)* sont l'ensemble de solutions non-dominée. Par conséquent, elles forment la surface de compromis (front de Pareto) du problème illustré.

La figure II.8-(a) donne une représentation graphique du front de Pareto pour un problème d'optimisation bi-objectif, où *F* représente l'espace objectif des solutions faisables. Cette figure illustre également deux vecteurs très importants qui existent pour tout problème multi-objectif, il s'agit du vecteur *Idéal* et du vecteur *Nadir*. Le premier correspond aux coordonnées du point minimum du problème d'optimisation, obtenu par minimisation des objectifs individuellement et le second représente sont opposé. La figure II.8-(b) illustre les quatre formes généralisées du front de Pareto dans le cas de problème bi-objectif, selon que l'on minimise ou maximise les deux objectifs f_1 et f_2.

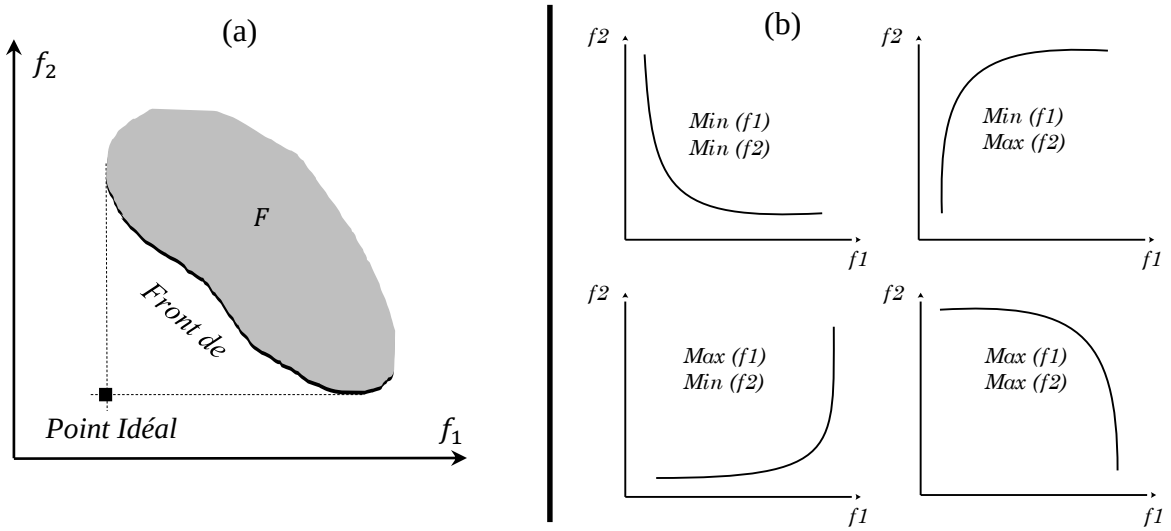


Figure II. 8 : Caractéristiques du problème d'optimisation multi-objectif

II.6 Algorithmes multi-objectif

Il existe actuellement une large panoplie d'algorithmes dédiés à la résolution des problèmes d'optimisation multi-objectif et de nouvelles versions sont encore publiées de nos jours. Néanmoins, ces algorithmes sont répertoriés en plusieurs classes suivant qu'on se place du point de vue utilisateur ou concepteur.

Selon le premier point de vue (utilisateur), on retrouve trois classes des méthodes de résolutions multi-objectif dites : *à priori, à posteriori,* ou *itérative*. Selon le second point de vue (concepteur) on distingue trois classes également : *les méthodes agrégatives, les méthodes Pareto,* et *les méthodes non-agrégatives et non-Pareto*. La principale différence entre ces trois dernières classes réside dans la façon avec laquelle l'ensemble des solutions non dominées est obtenu.

II.6.1 Algorithmes scalaires (agrégatives)

C'est l'approche la plus simple à mettre en œuvre. Elle consiste à transformer le problème multi-objectif en un problème mono-objectif en combinant les composantes f_m. L'agrégation additive ou la somme pondérée est la méthode de transformation couramment utilisée, elle s'exprime comme suit :

$$f_{obj} = \sum_{m=1}^{M} \lambda_m . f_m \qquad \text{(II.12)}$$

Cette approche a le mérite d'être simple et facilement utilisable avec des algorithmes mono-objectif. Néanmoins, elle comporte d'énormes difficultés. Il est très difficile de fixer à priori les valeurs de λ_m, elle implique également la normalisation des objectifs qui ne sont pas toujours commensurables. D'autre part, elles nécessitent plusieurs exécutions pour trouver tout le front de Pareto. À chaque exécution, pour chaque choix des paramètres de pondération (λ_m) correspond une seule solution sur le front de Pareto (voir Figure II.9). Par conséquent, cette méthode se trouve très démunie face à des espaces non-convexes rendant ainsi cette approche de loin un bon candidat à la résolution des différents problèmes multi-objectif [98].

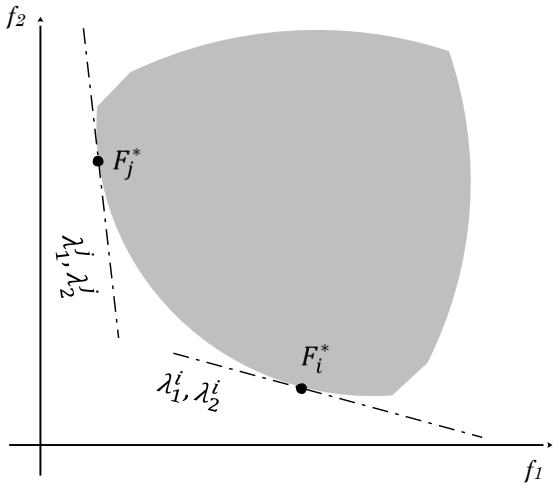


Figure II. 9 : Principe des méthodes scalaires

II.6.2 Algorithmes non-agrégatives et non-Pareto

Comme leur nom le sous-entend, ces algorithmes ne transforment pas le problème multi-objectif par agrégation et ne le traitent pas également en considérant toutes les fonctions objectif simultanément à leurs états bruts. Dans la littérature, il existe deux techniques représentatives de cette classe. La première traite les fonctions objectif séparément et les optimise individuellement comme l'algorithme *VEGA* (*Vector Evaluated Genetic Algorithm),* tandis que la seconde approche est dite *lexicographique.*

a. L'algorithme Génétique (VEGA)

Proposée par Schaffer en 1985 [99], cette méthode est basée sur l'usage d'algorithme génétique simple. Dans cette technique la sélection est différente de celle employée dans les algorithmes génétiques classiques. Sur une population à *n* individus et pour *k* fonctions objectif, à chaque fois on sélectionne (*n/k*) individus selon la fonction objectif (*k*). Les *k* sous-populations obtenues son ensuite utilisées pour former à nouveau des populations de *n* individus enfants.

b. Méthode lexicographique

Cette méthode est proposée par Fourman en 1985 [100]. Dans cette approche, les fonctions objectif sont préalablement rangées selon un ordre d'importance. Une fois le tri est fait par le décideur, l'optimisation commence en se focalisant d'abord sur la fonction la plus importante puis la seconde plus importante est ainsi de suite.

II.6.3 Algorithmes Pareto

En opposition aux deux classes précédentes, les approches Pareto peuvent trouver le front de Pareto en une seule exécution et en évitant toute transformation du problème d'optimisation. Cette classe comporte elle aussi deux types distingues d'algorithmes. Le premier type est dit : *Algorithmes non-élitistes* qui sont relativement anciens. Tandis que le second type est dit : *Algorithmes élitistes.*

Les algorithmes non-élitistes recherchent continuellement le front de Pareto sans sauvegarder les bons individus déjà retrouvés durant les itérations précédentes. Bien que ces algorithmes présentent des performances acceptables, ils restent moins efficaces durant la recherche vue qu'ils perdent souvent les bonnes sources d'information pour piloter la recherche à l'itération suivante. Ces algorithmes sont pratiquement les premières tentatives de recherche du front de Pareto. L'absence d'élitisme n'est pas l'unique faiblesse de ces algorithmes, leurs mécanismes de gestion de la recherche sont aussi remis en cause. Toutes ces raisons ont fait que ces algorithmes ont été reformulés avec des améliorations rendant leurs performances plus intéressantes.

L'algorithme NSGA est le parfait exemple à citer dans ce contexte. Après sa proposition, son auteur à remédier non seulement à l'absence de l'élitisme mais aussi au mécanisme de gestion de la population durant le processus de recherche introduisant ainsi l'un des algorithmes les plus puissants et les plus connus dans la littérature de nos jours, le NSGA-II [101]. D'autres algorithmes ont connus pratiquement un sort similaire à celui du NSGA, comme le SPEA-II [102].

L'observation de l'évolution de ces deux algorithmes est la meilleure façon d'assimiler les principes régissant les algorithmes multi-objectif dits Pareto. Ils incorporent souvent deux mécanismes distingues de sélection des meilleurs sources d'informations disponibles dans la population, *la sélection environnementale* et *la sélection à la reproduction*. La première adresse la question : *Comment guider la recherche vers le front de Pareto ?*, tandis que la seconde s'adresse plutôt à la question : *Quels individus doit-ont retenir durant le processus de recherche ?* En plus de ces deux sélections qui permettent de maximiser le nombre de solutions trouvées et de minimiser leurs distances au front de Pareto réel du problème étudié, ces algorithmes ont perfectionné la mesure de la dispersion des solutions sur le front de Pareto. Chose qui a significativement amélioré la qualité du front de Pareto trouver par ces algorithmes.

a. **Algorithme NGSA II** (*Non-dominated Sorted Genetic Algorithm-II*)

À chaque génération *t* de l'algorithme, une population parent P_t de taille *N* et une population enfants Q_t de même taille sont assemblées pour former une population R_t de taille *2N*, cet assemblage permet d'assurer l'élitisme. La population R_t est ensuite répartie en plusieurs fronts $(F_1, F_2, \dots F_n)$ par une procédure de tri basée sur la dominance. Une nouvelle population parent (P_{t+1}) est alors formée en ajoutant les fronts un par un et ce, jusqu'à atteindre une taille totale de *N* individus. Si le nombre d'individus présents dans (P_{t+1}) est inférieur à *N*, une procédure de *crowding* est appliquée sur le premier front suivant F_i non inclus dans (P_{t+1}). Le but de cet opérateur est d'insérer les $(N - |P_{t+1}|)$ meilleurs individus de F_i qui manquent dans la population (P_{t+1}), c'est la sélection environnementale. Une fois que les individus de la population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1}) est créée par sélection, croisement et mutation, c'est la sélection à la reproduction. La sélection se fait par tournoi, mais le critère de sélection est maintenant basé sur l'opérateur $(\prec n)$ que nous allons aborder juste après. Le processus se répète d'une génération à l'autre jusqu'à satisfaction d'un critère d'arrêt. La figure (II.10) résume le principe de cet algorithme.

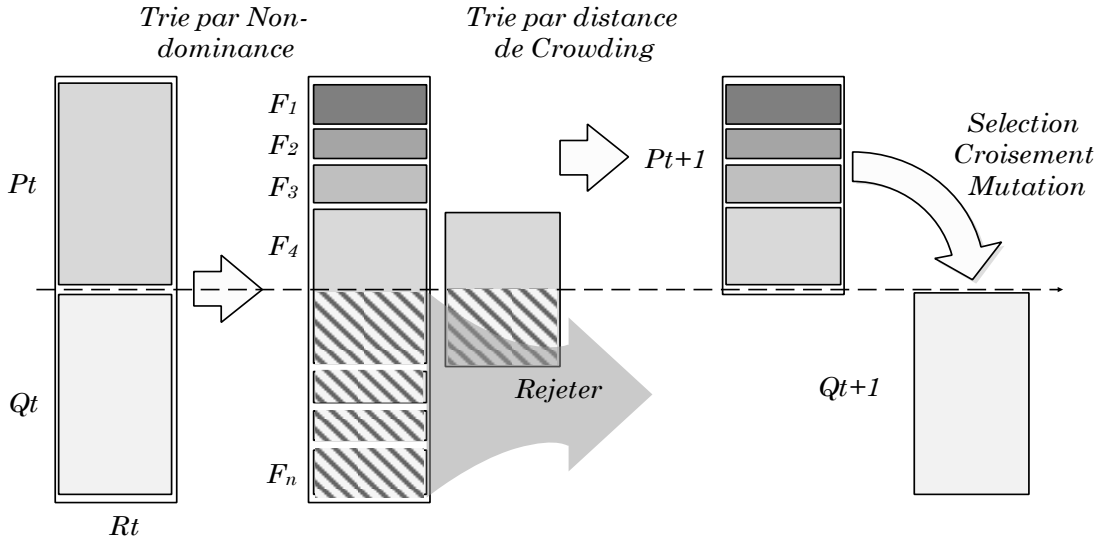


Figure II. 10 : Principe de l'algorithme NSGA II **[101]**

a. 1. Opérateur de crowding de comparaison ($\prec_n$)

Cet opérateur est utilisé pour guider le processus de sélection comme suit : chaque solution *i* de la population est identifiée par son rang i_{rank} et sa distance de crowding i_{distance}. Si les deux solutions se trouvent dans deux fronts différents alors l'opérateur sélectionne celle au front le plus petit, si par contre les solutions se trouvent dans un même front alors l'opérateur choisit celle caractérisée par une grande distance.

$$(x \prec_n y) = \begin{cases} x_{rank} < y_{rank} \\ (x_{rank} = y_{rank}) \wedge (x_{distance} > y_{distance}) \end{cases} \quad \text{(II.13)}$$

b. **Algorithme SPEA II** *(Strengh Pareto Evolutionnary Algorithm II)*

Initialement une population P_t de taille *N* est créée ainsi qu'une archive Q_t de taille N_a, combinés à chaque génération pour former l'ensemble des solutions faisables du problème posé. Chaque individu se voit assigner une fitness qui exprime son taux d'adaptation. Les individus non dominés sont sélectionnés pour former la prochaine archive Q_{t+1}. Une sélection par tournoi est organisée au sein de la population d'individus non sélectionnés pour l'archive et les gagnants de ce tournoi constitueront une population temporaire qui se reproduira via les deux opérateurs de croisement et de mutation. Les individus enfants résultants de ce processus donneront la nouvelle population P_{t+1}. Cette séquence d'événement ce poursuivras tant que la condition d'arrêt n'est pas satisfaite.

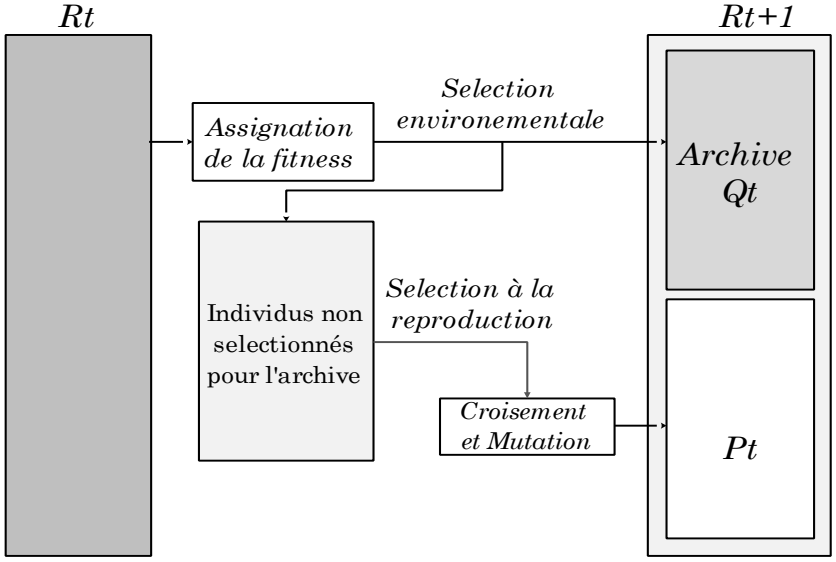


Figure II. 11 : Principe de l'algorithme SPEA II **[102]**

b. 1. Procédure d'assignement de la fitness

La première étape de l'assignation de la valeur d'adaptation s'effectue sur l'ensemble $\{P_t \; U \; Q_t\}$. Cette valeur est déterminée en calculant pour chaque individu $x_i \in \{P_t \cup Q_t\}$ la valeur $S(i)$ qui représente le nombre de solutions que x_i domine :

$$S(i) = \left|\{j \,/\, j \in (Pt + Qt) \wedge i \prec j\}\right| \quad \text{(II.14)}$$

À partir de cette valeur $S(i)$ (*strength value*), on définit une autre valeur $R(i)$ appelée valeur d'adaptation brute (*Raw Fitness*) qui est égale à la somme des forces $S(j)$, où *j* désigne

les individus qui domine *i*. Par conséquent, si un individu n'est pas dominé par aucun autre, il aura une valeur *R* nulle.

$$R(i) = \sum_{j \in Pt+Qt,\, j \prec i} S(j) \tag{II.15}$$

À un stade avancé dans le processus d'optimisation, cette technique peut échouer lorsqu'aucun individu ne domine l'autre, une information de densité est alors nécessaire pour distinguer entre les individus ayant des valeurs d'adaptation brutes *R* identiques. On définit alors une fonction de densité donnée par :

$$D_k(i) = \frac{1}{\sigma_i^k + 2} \tag{II.16}$$

Où σ_i^k représente la distance entre x_i et son $K^{ième}$ plus proche voisin. Le chiffre 2 au dénominateur sert à obtenir $D_k(i) < 1$ avec $k = \sqrt{\|Pt\| + \|Qt\|}$. Finalement la valeur de la fitness (adaptation) est obtenue à partir de $R(i)$ et de $D_k(i)$ comme suit :

$$F(i) = R(i) + D_k(i) \tag{II.17}$$

b. 2. Sélection environnementale

Pour mettre au point ce mécanisme vu précédemment, on commence par sélectionner tous les individus non dominés, c'est-à-dire, ceux dont la valeur d'adaptation F_i est inférieure à 1 et ainsi mettre à jour l'archive de la génération suivante.

$$Q_{t+1} = \{i / i \in Pt + Qt \wedge F(i) < 1\} \tag{II.18}$$

Arriver à ce stade trois cas peuvent se présenter :

1. $\|Q_{t+1}\| = Na$: Fin de la procédure de sélection environnementale.
2. $\|Q_{t+1}\| < Na$: Sélectionner les meilleurs individus de l'ensemble $Na - \|Q_{t+1}\|$.
3. $\|Q_{t+1}\| > Na$: Une procédure de troncature itérative supprimera les individus dominés jusqu'à ce que $\|Q_{t+1}\| = Na$.

La troncature se fait suivant un critère de dominance employant à la base la mesure de la distance σ_{ij}^k, ce critère est défini par l'équation (II-19). Cette définition de la dominance permet d'obtenir des individus éloignés, de façon à toujours maintenir la diversité dans le processus d'optimisation.

$$i \leq_d j = \begin{cases} \forall\ 0 < k < |Q_{t+1}| : \sigma_i^k = \sigma_j^k \\ \exists\ 0 < k < |Q_{t+1}| : (\forall\, 0 < l < k : \sigma_i^l = \sigma_j^l) \wedge (\sigma_i^k < \sigma_j^k) \end{cases} \tag{II.19}$$

b. 3. Sélection à la reproduction (matting selection)

La sélection à la reproduction opère sur les individus les moins performants qui ne sont pas sélectionnés pour former l'archive. Elle est basée sur un tournoi binaire dont l'objectif est de construire une population temporaire P_t(de taille *N*). Le critère de comparaison employé cette fois-ci est la valeur d'adaptation *Fi*, l'individu au *Fi* le plus faible sera sélectionné, ensuite cet ensemble résultant P_t sera soumis aux opérateurs de croisement et de mutation qui constituent la dernière étape avant l'obtention de la nouvelle population P_{t+1}.

II.7 Evaluation de performance en multi-objectif

Contrairement au problème mono-objectif, l'évaluation des résultats obtenus par algorithmes multi-objectif est beaucoup plus complexe du moment qu'ils fournissent un ensemble de solutions au lieu d'une seule. Dans les problèmes bien connus (*fonctions test*), où le front de pareto optimale est connu, la qualité du résultat de l'agorithme est évaluée par deux métriques principales. La première mesure le degré de proximité des solutions trouvées par rapport à celles optimales, tandis que la seconde mesure la dispersion des solutions sur le front trouvé. Pour la première mesure, plus elle est petite plus le résultat est meilleur, tandis que pour la seconde mesure, plus la dispersion est grande plus le résultat est meilleur [101]. La première mesure exprimant la convergence de l'algorithme est donnée par l'équation (IV.20) et celle évaluant la dispersion des solutions est exprimée dans l'équation (II.21). Ces deux mesures sont interprétées graphiquement dans la figure (II.12).

$$\zeta(t) = \frac{1}{N}\sum_{i=1}^{N} m_i \tag{II.20}$$

$$\Delta(t) = \frac{d_f + d_l + \sum_{i=1}^{N-1}\left|d_i - d_{moy}\right|}{d_f + d_l + (N-1)d_{moy}} \tag{II.21}$$

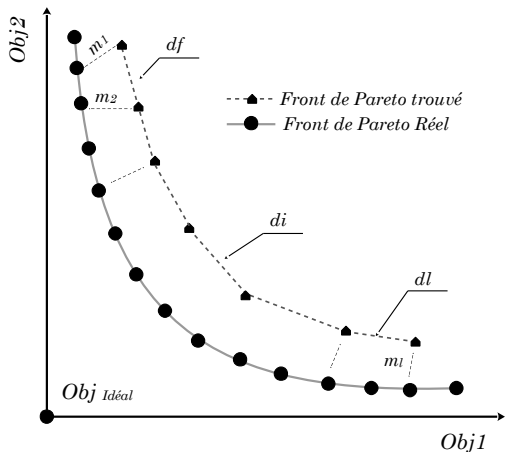


Figure II. 12 : Evaluation du front de Pareto

II.8 Optimisation multi-objectif et décision multi-critères

Dans les sections précédentes nous avons présenté le problème d'optimisation et les principaux outils de traitement. Et si dans le cas de l'optimisation mono-objectif, le problème est directement résolu par l'algorithme lui-même, ce n'est pas entièrement le cas pour l'optimisation multi-objectif. En effet, du moment qu'on ne va pas implémenter toutes les solutions trouvées (Front de Pareto), il demeure toujours une question: *Quelle solution adoptée pour le problème, si toutes celles trouvées sont dites incomparables?*. Car effectivement, aucune de ces solutions n'est meilleure sur tous les objectifs simultanément, ce qui rend le choix d'une seule d'entre-elles loin d'être une mince affaire.

Dans la littérature, cette question est traitée par différentes méthodes qui s'apparentent toutes au domaine de « *Décision multicritère* ». Ces techniques reposent dans leur ensemble sur la définition d'une matrice dite : *Matrice de décision* . Où, le décideur attribut un ordre de préférence au différents critères de décision. Ces critères peuvent êtres les fonctions objectif elles-mêmes. Autrement dit, ces techniques interprètent en réalité un choix préalablement pensé mais pas explicitement exprimé par le décideur.

Alors, une fois l'algorithme multi-objectif trouve toutes les solutions alternatives du problème, la sélection de la solution optimale à implémenter sera une question de peférence qui requière à la fois une parfaite connaissance du problème étudié en plus d'une très bonne capacité d'estimation de la qualité des solutions une fois appliquées au problème. Ainsi, la démarche complète de résolution du problème multi-objectif se ramène à l'usage d'un algorithme multi-objectif et d'une méthode de décision multi-critères comme illustrée dans la figure (II.13).

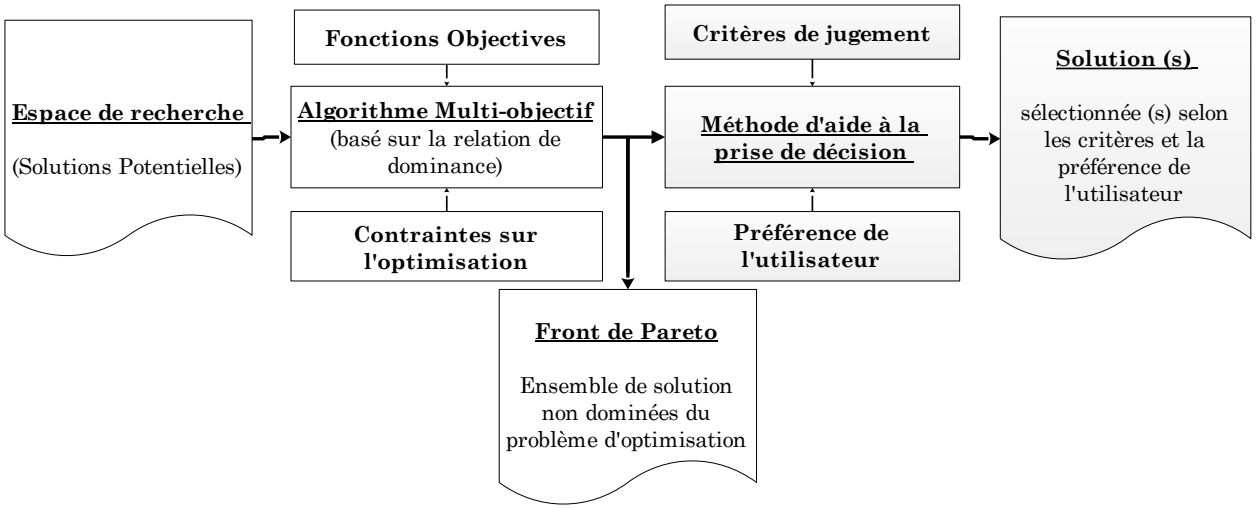


Figure II. 13 : Utilisation des algorithmes multi-objectif et les méthodes de décision multi-critères

La synthèse et la discription de toutes les techniques d'aide à la prise de décision multi-critères est très difficile à faire dans une seule section. Par ailleurs, l'étude de cette classe de technique n'est pas l'objectif de cette thèse. Alors dans ce qui suit nous allons présenter une vue d'ensemble sur ces techniques et sur comment le problème d'aide à la prise de décision multi-critère est formulé dans la littérature.

Le problème de la décision multi-critère peut se présenter sous différentes formes qui peuvent être regroupées en trois classes. Le tableau (II.2) présente ces trois grandes classes selon la formulation de la problématique, la procédure adoptée et le résultat obtenu à la fin de l'exécution [46].

Tableau II. 2 : Classification des problèmes d'aide à la décision multi-critères.

Problèmatique	Résultat	Procedure
Choix d'un sous-ensemble des actions, les « meilleures » ou a défault, les plus « satisfaisantes »	Choix	Sélection
Tri par affectation des actions à des catégories prédéfinies	Tri	Affectation
Rangement par classes d'équivalances composées d'actions, ces classes étant ordonnées de façon complete ou partielle.	Rangement	Classement

Ces techniques sont connues dans la littérature sous des noms de familles regroupant une large variété d'approches ayant pratiquement le même principe. On trouve par exemple la famille de méthode ELECTRE [103], la famille PROMETHEE [104], les méthodes d'analyse hiérarchisée (AHP) [105-106]. Ces trois familles diffèrent entre-elles par leurs formulations et leurs objectifs, par concéquent à chaque type de la problématique de décision multi-critère, une famille d'entre elle est plus appropriée.

II.9 Conclusion

Comme nous l'avons mentionné tout au début de ce chapitre, ce dernier vise à souligner les aspects essentiels de ce domaine d'optimisation et son potentiel à résoudre le problème de conception des systèmes flous présenté dans le premier chapitre.

Tout au long de ce chapitre, nous nous sommes intéressés à la classe d'algorithmes dite méta-heuristique. Cet intérêt est dû au fait que d'abord ces algorithmes constituent un tournant majeur dans le domaine de l'optimisation et ensuite ils constituent l'outil principal souvent utilisé dans le domaine d'apprentissage automatique des systèmes flous. Dans ce chapitre nous

avons également souligné l'intérêt de combiner l'optimisation multi-objectif à la décision multicritère en vue de sélectionner une solution à adopter pour le problème.

Maintenant que les outils de conception sont présentés, une seule question se pose alors : *comment l'optimisation et ses algorithmes peuvent contribuer à la conception des systèmes flous ?* La réponse justement à cette question sera le sujet des prochains chapitres.

Chapitre III

Conception mono-objectif des SIFs

III.1 Introduction

Comme nous l'avons déjà vu dans le premier chapitre, la conception des systèmes flous constitue un défi assez complexe qui est régit par de nombreuses contraintes. Cela même a conduit la communauté spécialisée à se tourner vers la conception automatique ou précisément à l'optimisation. Dans le second chapitre, nous avons montré qu'il existe plusieurs classifications possibles du problème d'optimisation. Cependant, du point de vue conception des systèmes flous, seul le nombre de fonction objectif compte. Ainsi, soit la conception est mono-objectif soit elle est multi-objectif.

Durant ce présent chapitre, seule la conception dite : mono-objectif sera traitée. Nous allons d'abord présenter un bref résumé de l'état de l'art sur ce sujet avant de détailler notre contribution.

III.2 Conception automatique des systèmes flous

Dans la littérature, il existe une large gamme de méthodes et d'approches destinées à la conception automatique du système flou. Toutes ces approches peuvent être synthétisées par un simple diagramme comme celui de la figure (III- 1). Elles sont en principe le fruit de la

rencontre ou de l'hybridation du système flou avec *les réseaux de neurones,* et *les algorithmes d'apprentissage automatique*. Ces trois outils forment les ingrédients de base de l'apprentissage automatique en ingénierie.

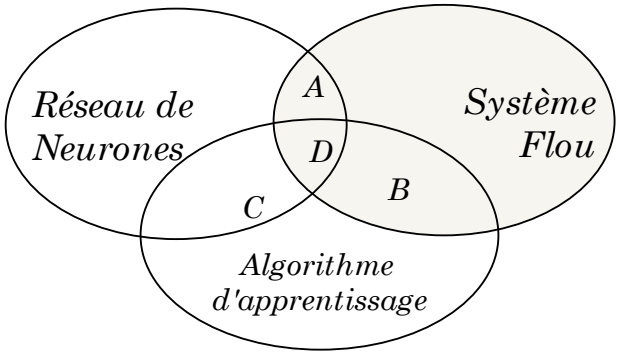


Figure III. 1 : Position de la conception des systèmes flous

Ces hybridations donnent lieu à trois grandes familles d'approches de conception des systèmes flous (figure (III.1)). La première classe (*A*) est l'hybridation du système flou avec les réseaux de neurones dite : *neuro-flou*, cette classe est déjà présentée dans le chapitre I (*Séction I.8, pp. 20*). La seconde classe (*B*) est une combinaison du système flou avec les différents algorithmes d'apprentissage. C'est cette classe d'approche qui nous intéresse le plus dans ce travail, en particulier l'utilisation des algorithmes de classe « *méta-heuristiques* ». Comme nous l'avant avancé juste avant et selon le nombre d'objectifs considérés par ces algorithmes, cette classe se scinde également en deux catégories. La première catégorie est basée sur les algorithmes mono-objectif et sera détaillée dans ce chapitre, la seconde est basée sur les algorithmes multi-objectif fera l'objet du dernier chapitre de la thèse.

La troisième classe (D) est l'hybridation du système flou avec les réseaux de neurones et les algorithmes apprentissage. Dans ce cas l'hybridation neuro-flou n'est pas forcement identique à celle présentée auparavant. Certains auteurs préfèrent désigner cette classe par l'appellation *DFS* pour (*Deep Fuzzy learning*) qui signifie littéralement : *apprentissage flou profond* [107]. Cette dernière classe adopte une structure spécifique du système flou et du réseau de neurones.

III.3 Conception des SIFs par algorithme mono-objectif

La conception automatique des systèmes flous par ces algorithmes d'optimisation désigne deux manières de procéder difficilement distinguables. La première consiste à affiner les valeurs des paramètres du système flou initialement définis par le concepteur, tandis que la

seconde vise à définir ces paramètres sans aucune indication du concepteur. On parle alors de : *réglage automatique* dans le premier cas et d'*Apprentissage automatique* dans le second.

L'usage des algorithmes mono-objectif pour la conception des systèmes flous est l'approche la plus ancienne, et la plus populaire dans la littérature de conception des systèmes flous. Cette popularité est due, en grande partie, à la simplicité d'implémentation de ces algorithmes et à leurs efficacités. La figure (III.2) offre une vue globale de l'état de l'art sur la méthode. Cet état de l'art est formulé principalement pour l'usage des algorithmes génétiques, mais cela reste valable également pour les autres méta-heuristiques [108-109].

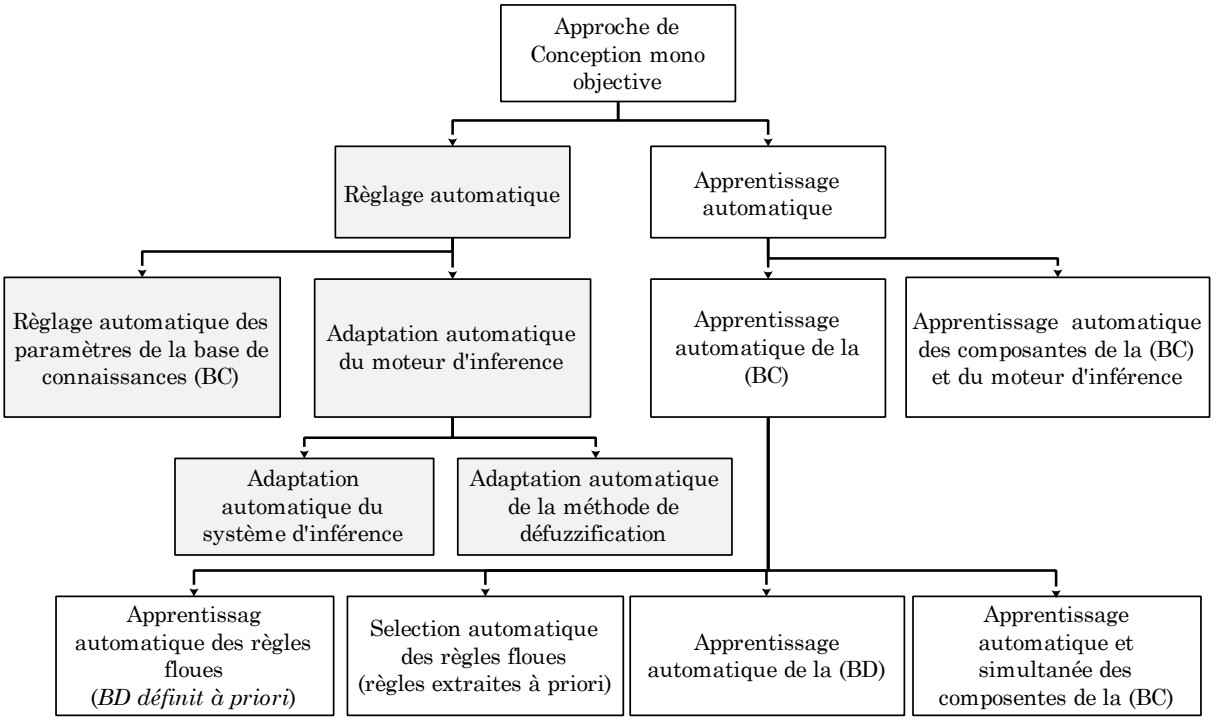


Figure III. 2 : Etat de l'art de la conception mono-objectif des systèmes flous

III.3.1 Réglage automatique

Le réglage automatique est utilisé pour améliorer l'expertise de l'expert lors de la conception du système flou. Une fois la base de règles floues (BR) est définie, l'algorithme d'optimisation se voit attribuer la fonction d'affiner le réglage soit de la base données (BD) ou du moteur d'inférence (MI).

a. Réglage de la base de données

Une fois les règles floues sont définies, l'algorithme d'optimisation est appliqué sur les fonctions d'appartenance en vue d'améliorer la performance du système flou. Il existe plusieurs façons de réaliser cette tâche, soit l'algorithme ajuste uniquement la forme des fonctions d'appartenance sans changer leur nombre [110]. Soit en utilisant des modificateurs

linguistiques donnant ainsi plus de flexibilité au réglage [111]. Dans d'autres travaux, l'algorithme combine un réglage latéral avec un ajustement de l'amplitude des fonctions d'appartenance pour faire face aux espaces de recherche très large [112].

b. Réglage du Moteur d'inférence

Le réglage du moteur d'inférence comporte deux classes, la première est dite : *Adaptation du système d'inférence* [113-115] tandis que la seconde est dite : *Adaptation de la méthode de deffuzification* [116]. Dans la première classe, l'algorithme d'optimisation tente de chercher une configuration ou une définition optimale des opérateurs flous. Le but étant d'augmenter la coopération entre toutes les règles floues tout en préservant le sens sémantique de ces dernières. Par contre dans la seconde classe, l'algorithme d'optimisation est dédié au réglage de la méthode de deffuzification afin d'améliorer la précision du système flou.

III.3.2 Apprentissage automatique

Contrairement aux cas précédents où il été juste question de raffiner le choix fait par le constructeur, ici il est question de trouver les éléments de la base de connaissance tout en les raffinant. Tout le travail de conception dans ce cas est entièrement fait par l'algorithme(s). Autrement dit, l'algorithme(s) doit trouver la base de règles floues en plus de la base de données soit simultanément ou séparément.

a. Apprentissage simultané de la BD et de la BR

Lorsque l'apprentissage de la base de connaissance (base de règles et base de données) est réalisé (Figure III.3-(A)), l'algorithme d'optimisation est employé pour chercher au même temps les paramètres des fonctions d'appartenance ainsi que les règles floues. Bien que cette méthode aboutis souvent à de très bons résultats, elle reste cependant assez difficile et gourmande en raison de la dimension de l'espace de recherche engendré [117-118].

b. Apprentissage séparé de la BD et de la BR

D'autre part lorsque l'apprentissage est fait séparément, soit l'algorithme est orienté à l'extraction des règles floues (Figure III.3-(B)), soit il est orienté à la définition de la base de données (Figure III.3-(C)).

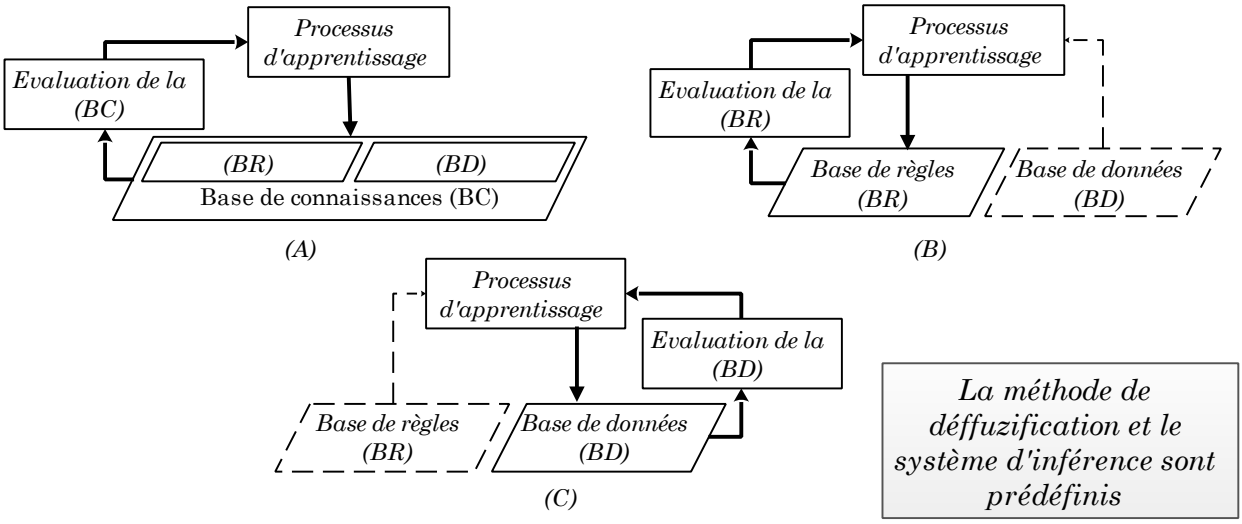


Figure III. 3 : Apprentissage automatique de la base connaissances

b.1 Apprentissage des règles floues

Dans ce cas, les fonctions d'appartenance sont définies au préalable, souvent en nombres impairs par partition [3 5 7 …]. En se basant sur ces fonctions d'appartenance, l'algorithme forme les règles floues. Cette approche est fréquemment employée lorsqu'il est question de traiter des collections de données numériques. Cependant, ce mécanisme d'extraction des règles floues peut générer des règles redondantes, erronées ou même des règles conflictuelles qui perturbent le comportement global du système flou. Pour remédier à cette situation, un processus de sélection automatique est souvent prévu pour obtenir un sous-ensemble optimisé de règles à partir de la première collection.

b.2 Apprentissage de la base de données

L'apprentissage de la base de données peut être réalisé de deux façons différentes. Dans la première approche, l'apprentissage des fonctions d'appartenance est combiné avec l'apprentissage des règles floues. À chaque fois que les partitions floues sont formulées, elles sont utilisées pour induire une collection de règles floues (Figure III.4-(1)). Dans la seconde approche, l'apprentissage des fonctions d'appartenance vise la définition de la forme des fonctions d'appartenance et leurs paramètres. Ce n'est qu'une fois que les partitions floues sont complètement définies que les règles floues sont induites (figure III.4-(2)).

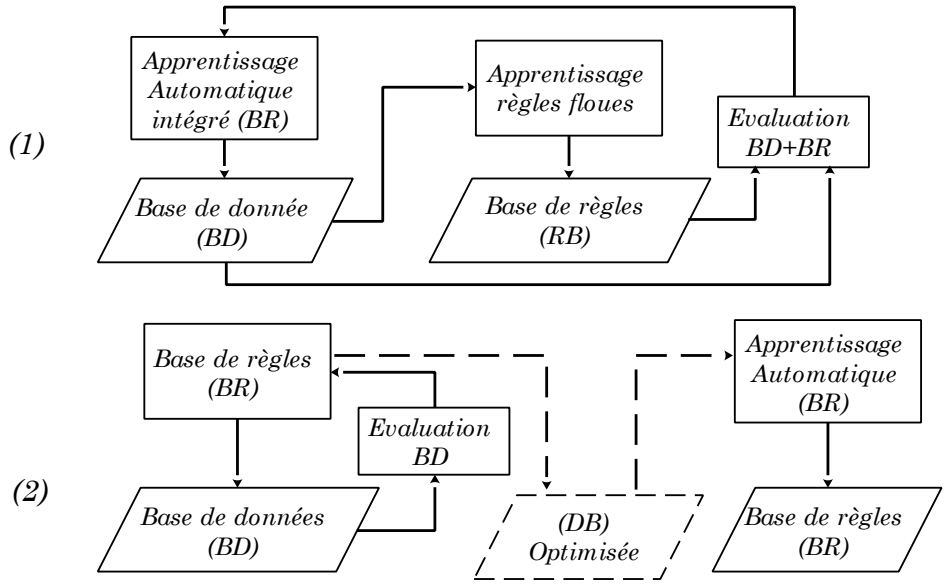


Figure III. 4 : Apprentissage automatique de la base de données

Il est important de souligner que le réglage automatique et l'apprentissage automatique ainsi que leurs différentes variantes ne sont pas mutuellement exclusive, ils peuvent être combinés lors de la même phase de conception. La combinaison des deux techniques est soit incorporée dans un même algorithme, soit réalisée par plusieurs algorithmes où chacun a une de ces vocations. Par exemple pour les problèmes d'identification, le concepteur peut définir une base de données à priori et en utilisant un algorithme d'apprentissage, il extrait la base règles floues. Ensuite, il affine le choix fait sur la base de données par un algorithme de réglage.

III.4 Codage et représentation du système flou

Le codage et la représentation des paramètres du système flou est la première étape importante du processus de conception. La méthode adoptée conditionne fortement le mécanisme d'évolution de l'heuristique d'optimisation. Dans ce contexte, il existe en principe deux grandes méthodes de codage des paramètres du système flou. La figure (III.5) illustre ces différentes méthodes.

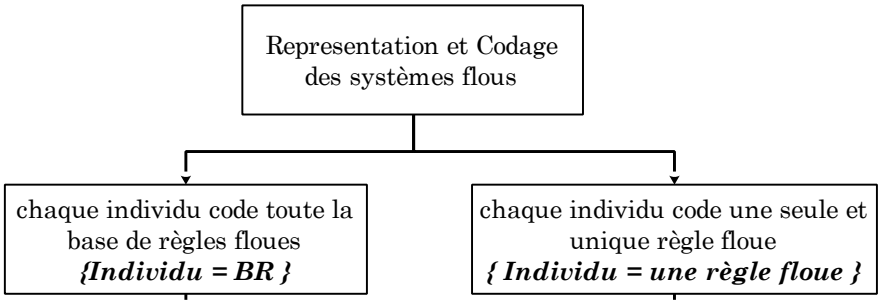


Figure III. 5 : Codage et représentation de la base des règles floues

La première méthode est la plus répandue dans la littérature. Nommée « *méthode Pittsburg* » [119], elle consiste à coder tous les paramètres du système flou par un seul individu de sorte à ce que la population d'individus correspondra à plusieurs systèmes flous distincts. Par contre dans la seconde méthode nommée « *méthode Michigan* » [120], chaque règle floue est codée par un seul individu. Par conséquent, la base de règles du système flou est codée par toute la population d'individus.

Pour clarifier le principe de ces deux méthodes de codage, considérons le cas où il est question d'un système flou ayant (*M*) règles floues. La forme de la population et des individus sera différente selon qu'on utilise la première ou la seconde méthode comme indiquée dans la figure (III.6)

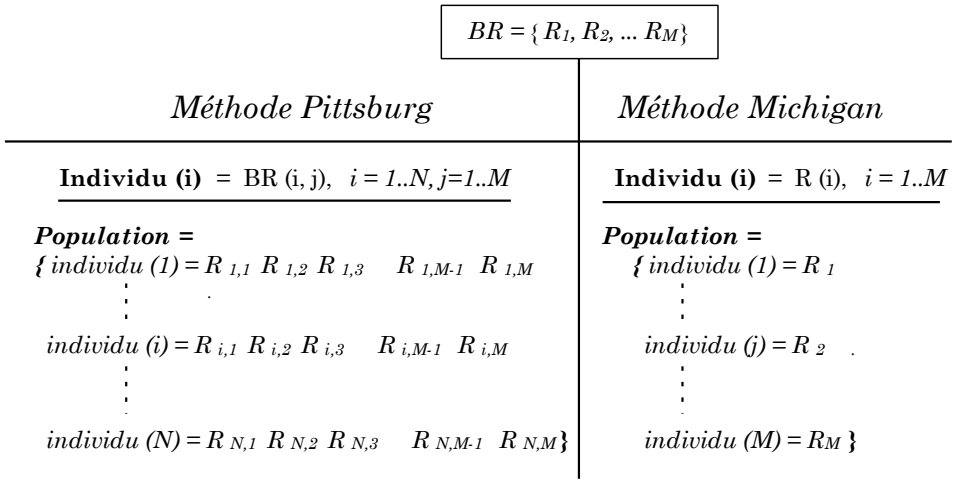


Figure III. 6 : Codage du système flou avec les deux méthodes : Pittsburg et Michigan

III.5 Optimisation des partitions floues (MFs)

L'optimisation des fonctions d'appartenance comporte elle-même deux sous-problèmes distincts non-exclusif, mais plutôt complémentaires. Le premier est la recherche de la forme de la fonction d'appartenance à utiliser, tandis que le second est la recherche du jeu de paramètres associés à ces fonctions.

Comment nous l'avant déjà avancé dans le premier chapitre (§1 *pp.*25), la première question peut être résolue à priori. Pour la seconde question, par contre, il n'existe pas de méthodes permettant de fixer a priori les paramètres, d'où l'intérêt d'utiliser un algorithme d'optimisation. Dans ce cas, tous les paramètres définissant les fonctions d'appartenance sont regroupés dans un seul vecteur représentant le codage de la partition floue associée. Dans la figure (III-7) une illustration est donnée pour deux formes de fonction (Gaussienne et triangulaire).

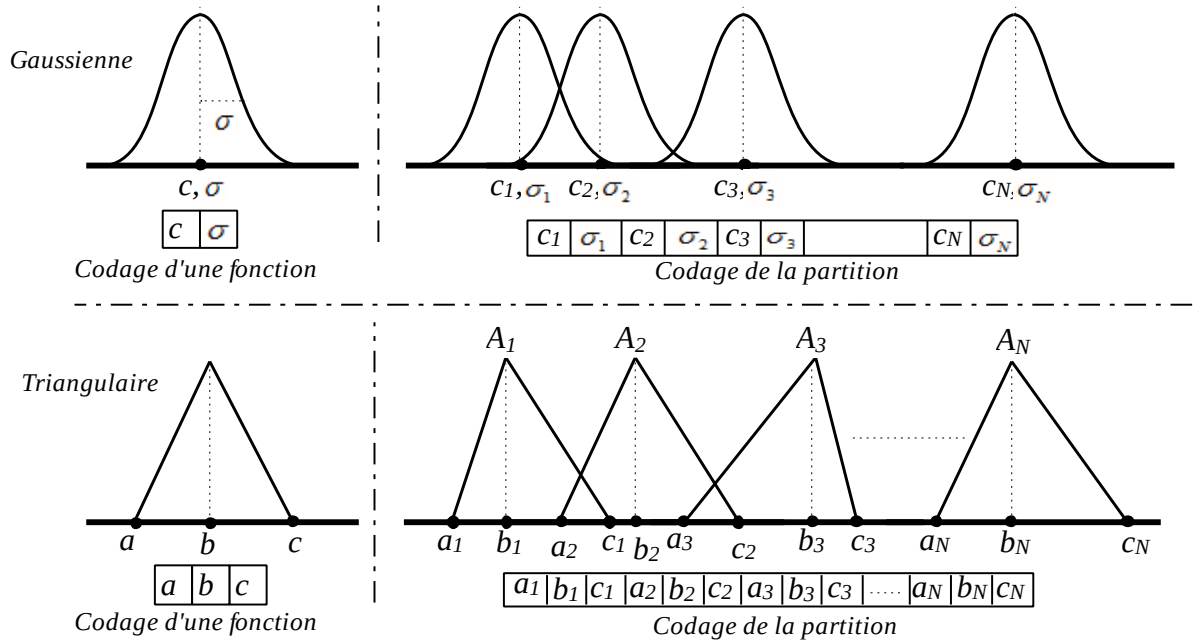


Figure III. 7 : Codage des partitions floues

Ce codage intuitif est vraisemblablement très simple. Cependant, pour que la partition floue puisse toujours garder son sens sémantique, il faudra ajouter aussi des contraintes sur les paramètres qui dépendent essentiellement de la forme des fonctions choisies. Ce qui rend le problème d'optimisation plus difficile à résoudre. Par exemple, dans le cas des fonctions triangulaires, les paramètres de chaque fonction d'appartenance doivent obéir à la condition (III.1).

$$a_i < b_i < c_i \ , \ i = 1...N \tag{III.1}$$

Cette condition est combinée avec d'autres pour garantir une partition floue sémantiquement acceptable. Ces autres conditions peuvent être énoncées comme suite :

$$\begin{aligned} b_{i-1} \le a_i \le c_{i-1} \ , \ i = 2...N \\ a_{i+1} \le c_i \le b_{i+1} \ , \ i = 1...N-1 \end{aligned} \tag{III.2}$$

L'inconvénient majeur de cette représentation est qu'elle engendre un nombre important de contraintes et de paramètres qui peuvent limiter l'efficacité de la procédure de recherche. Pour contourner cette difficulté et simplifier le codage, Michael. A.L [121] et N. Yubaziki [122] ont proposé une autre méthode pour laquelle le nombre de paramètres requis est réduit de manière significative. L'idée est de faire partager le même paramètre par plusieurs fonctions d'appartenance. La figure (III.8) reprend la même partition précédente avec des formes triangulaires, mais cette fois avec un partage de paramètres.

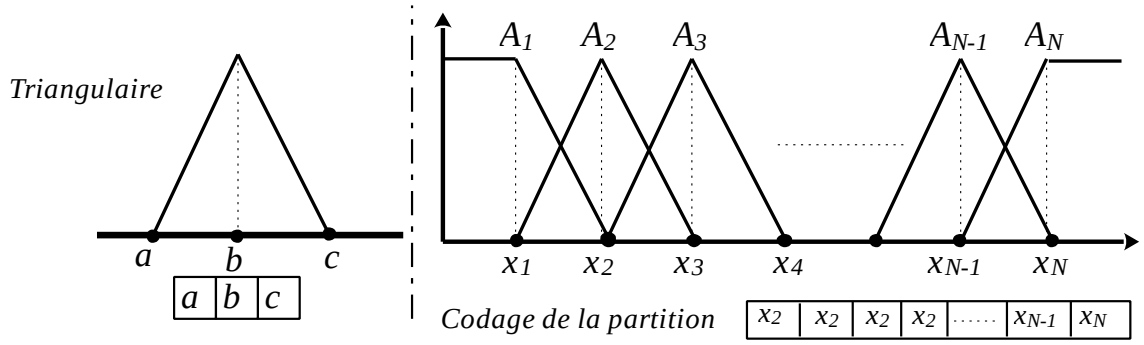


Figure III. 8 : Codage des partitions floues avec partage de paramètres

Contrairement au premier codage, le partage de paramètres entre les fonctions d'appartenance à significativement réduit la taille du code. Il est passé de (*3xN)* paramètres pour *(N)* fonctions d'appartenance à seulement *(N)* paramètres. Bien que cette idée à l'avantage de réduire le codage des partitions floues, elle reste toutefois trop rigide du moment que le degré de chevauchement entre les fonctions d'appartenance adjacentes n'est jamais accessible et demeure toujours égal à (1/2).

III.6 Optimisation des règles floues

L'optimisation des partitions floues, présentée juste avant, est aussi en effet une optimisation de la base des règles. En déplaçant les fonctions d'appartenance dans leur univers de discours respectives, la structure sémantique des règles change également. De nombreux travaux dans la littérature reposent sur cette vision [123-124]. Les auteurs définissent d'abord la base de règles floues puis, ils procèdent à l'ajustement des paramètres des fonctions d'appartenance.

Cependant, cette approche comporte plusieurs faiblesses. Si l'optimisation des fonctions d'appartenance n'est pas soumise à des contraintes comme celles présentées dans les équations (III.1 - III.2), elle risque d'aboutir à des partitions non-conformes (*voir section I.10.1, pp28*). D'autre part, si l'optimisation est fondée sur l'idée de M.A. Lee et N. Yubazaki, le sens sémantique des partitions sera préservé mais limitera également l'exploration de nouvelles possibilités de configuration des fonctions d'appartenance.

Pour remédier à toutes ces complications et permettre une meilleure exploration et recherche des règles floues, des auteurs ont proposé d'agir sur les conclusions de ces dernières. En intervertissant les conclusions des règles floues, on préserve la structure sémantique des règles tout en réduisant le temps de recherche. Cette méthode consiste à coder toutes les conclusions des règles floues dans un seul vecteur. Vu que ces dernières sont de nature

linguistique elles sont souvent codées par des nombres entiers. La figure (III-9) illustre le principe de construction de ce vecteur. Dans certains travaux, les conclusions des règles floues sont combinées avec les partitions floues dans le même vecteur.

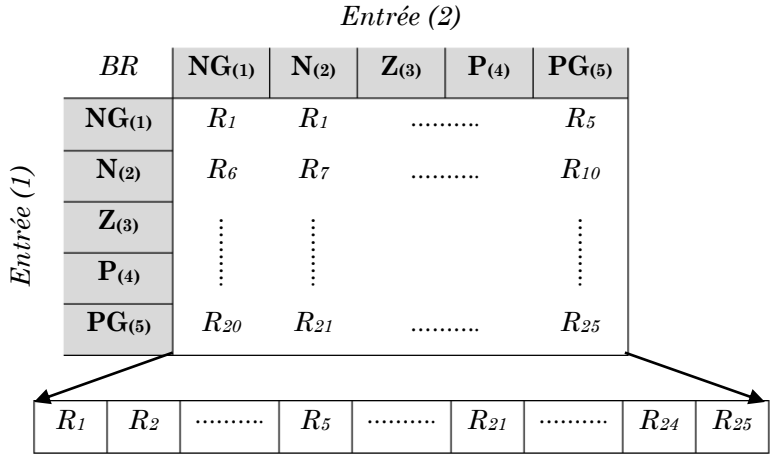


Figure III. 9 : Codage des conclusions des règles floues

III.7 Critères de performances

Il existe une multitude d'objectifs que la conception automatique des systèmes flous vise à atteindre simultanément. Toutefois, ces objectifs sont souvent contradictoires et ne peuvent être facilement réconciliés dans une seule fonction mathématique. Par conséquent, lorsque l'heuristique utilisée pour l'optimisation est de nature mono-objectif, c'est la précision du système flou qui est souvent considérée. On peut trouver dans la littérature une large panoplie de critères numériques permettant l'évaluation de la précision du système flou. Ils sont en grande majorité basés sur l'estimation de l'erreur ou simplement l'écart entre la réponse du système flou et la consigne désirée. MSE (*erreur quadratique moyenne*), RMSE (*racine de l'erreur quadratique moyenne*), IAE (*Intégrale de l'erreur absolue*), ISE (*Intégrale du carré de l'erreur*), ITAE (*Intégrale temporel de l'erreur absolue*) sont quelques-uns des plus connus.

$$MSE = \frac{1}{N}\sum_{k=1}^{N}\left(e(k)\right)^2 \qquad \text{(III.3)}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(e(k)\right)^2} \qquad \text{(III.4)}$$

$$IAE = \sum_{k=1}^{N}\left|e(k)\right| \qquad \text{(III.5)}$$

$$ISE = \sum_{k=1}^{N}\left(e(k)\right)^2 \qquad \text{(III.6)}$$

$$ITAE = \sum_{k=1}^{N}\left|e(k)\right| \times kT_e \qquad \text{(III.7)}$$

Le terme *k* correspond au numéro de l'échantillon mesuré, *N* : le nombre total d'échantillons collectés et *Te* : la période d'échantillonnage. Tous ces critères reflètent la

précision du système flou à suivre une consigne désirée, ils peuvent être utilisés individuellement ou combinés par agrégation.

III.8 Approche de conception Proposée

La conception automatique des systèmes flous a était longtemps gouvernée par les algorithmes génétiques. Le choix de cette méta-heuristique est en grande partie justifié par l'absence d'autres algorithmes rivaux en plus de sa robustesse et surtout sa capacité à surmonter la présence des optimaux locaux. Cependant, cet algorithme est très gourmand en termes de ressource en raison de sa complexité algorithmique. Cet inconvénient fut la raison principale qui a incité les chercheurs à introduire de nouvelles heuristiques tels que : *FFA* (*firefly algorithm*), *GSA* (*gravitational search algorithm*), *PSO* (*particle swarm optimization algorithm*) … qui sont beaucoup plus intéressants que l'algorithme génétique du point de vue complexité [125]. Actuellement, l'algorithme PSO gagne de plus en plus d'intérêt par rapport aux autres heuristiques pour l'optimisation des systèmes flous.

Bien qu'il réussit facilement à optimiser les systèmes flous précis de Sugeno [126-129], son usage pour le type linguistique de Mamdani reste encore très modeste en raison d'incompatibilité entre la nature du PSO et le jeu de paramètres du système flou de type Mamdani. En effet, l'algorithme PSO est intrinsèquement dédié aux espaces de recherche continus formés par des variables de nature réelle. Cependant, le système flou de type Mamdani comporte en plus des paramètres réels (*MFs*) d'autres paramètres codés en entier (*Conclusions des règles, paramètres du moteur d'inférence etc.*), ce qui rend son espace de recherche beaucoup plus difficile à parcourir par l'algorithme PSO.

Notre proposition [130] consiste principalement à résoudre cette incompatibilité en rendant l'algorithme PSO capable de manipuler d'autres types de variables que le réel et d'adapter cet algorithme pour la conception du système flou de type Mamdani. Contrairement aux méthodes standards, notre proposition permet un ajustement simultané de tous les paramètres du système flou linguistique, et ce sans aucune connaissance à priori.

III.8.1 Codage des paramètres

L'approche proposée repose sur la stratégie dite *(Pittsburg)* pour le codage des paramètres. Ainsi chaque particule code un système flou entier et l'essaim code une collection

de systèmes flous distincts. La figure (III.10) illustre la structure d'une seule particule de l'essaim utilisé.

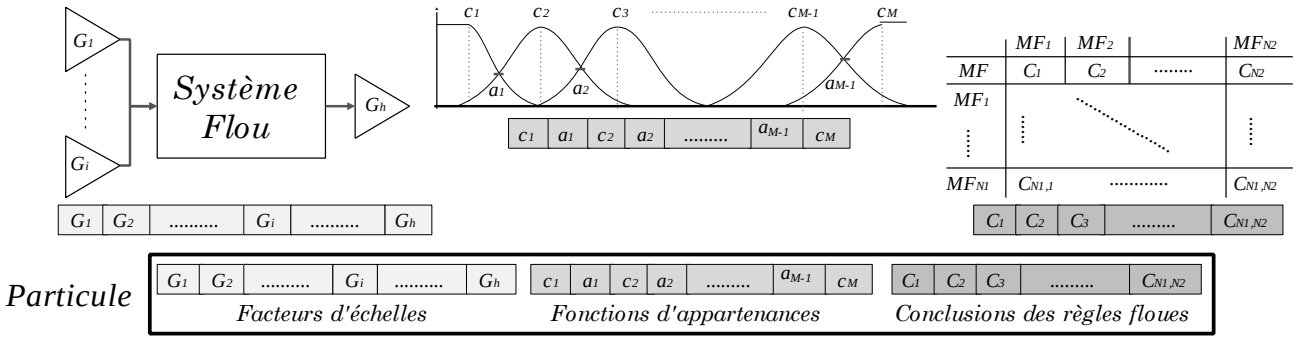


Figure III. 10 : Codage des paramètres et structure de la particule

Les facteurs d'échelle ainsi que les paramètres des fonctions d'appartenance sont codés par des nombres réels normalisés dans l'intervalle [0, 1]. Chacun de ces paramètres réels est associé à un intervalle bien spécifique pour le décoder. Les conclusions des règles floues par contre, sont codées par des nombres entiers dans l'intervalle [1, *Nr*]. Où *Nr* correspond au nombre total de termes linguistiques utilisés pour définir la partition de la variable de sortie. Les termes linguistiques des entrées/sorties sont définis par des fonctions gaussiennes asymétriques.

III.8.2 Pilotage des conclusions floues

L'usage d'un codage mixte tel que celui proposé dans cette approche constitue une complication pour l'algorithme PSO. Pour résoudre cette difficulté, une fonction de contrôle ∅ est introduite [130] pour être un intermédiaire entre les vitesses des conclusions qui sont de type réel et leurs positions qui sont de type entier. Cette fonction joue le rôle de processus de décision qui détermine dans quelle direction la conclusion aller se déplacer. Selon sa vitesse ($V_{i,j}^{iter+1}$), la future position de la conclusion est soit *incrémentée,* soit *décrémentée,* ou soit *inchangée*. Ainsi, la conclusion se déplace toujours d'un seul pas pour éviter tout changement brusque pouvant conduire à une exploration chaotique.

Pour prendre cette décision, la fonction utilise un seuillage dynamique et auto-adaptatif (τ_j^{iter}) associé à chaque conclusion $(C_{i,j})$. Ce seuil représente la vitesse de confiance requise pour que chaque conclusion passe de sa position actuelle à la suivante. La formulation mathématique de la fonction de contrôle (∅) introduite est donnée par les équations (III.8) et (III.9). La première est une forme analytique constituée de deux fonctions sigmoïdes tandis que la seconde est une forme algorithmique plus simple à mettre en œuvre. Ces formulations sont

élaborées en respectant les exigences expliquées précédemment, elles font que les conclusions changent en douceur avec un seul pas, puisqu'elles ne peuvent générer que les nombres entiers suivants : {-1, 0, ou 1}. Ces trois nombres correspondent respectivement aux trois décisions de {*Décrémenter*, *inchanger* et *incrémenter*}.

$$\phi\left(V_{i,j}^{iter+1}, \tau_j^{iter}\right) = \frac{1}{2}\left(\frac{V_{i,j}^{iter+1} - \tau_j^{iter}}{\left|V_{i,j}^{iter+1} - \tau_j^{iter}\right|} + \frac{V_{i,j}^{iter+1} + \tau_j^{iter}}{\left|V_{i,j}^{iter+1} + \tau_j^{iter}\right|}\right) \quad \text{(III.8)}$$

$$\phi\left(V_{i,j}^{iter+1}, \tau_j^{iter}\right) = \begin{cases} -1 & if \quad V_{i,j}^{iter+1} < -\tau_j^{iter} \\ 0 & if \quad -\tau_j^{iter} \le V_{i,j}^{iter+1} \le \tau_j^{iter} \\ 1 & if \quad V_{i,j}^{iter+1} > \tau_j^{iter} \end{cases} \quad \text{(III.9)}$$

Une interprétation graphique du mécanisme de la fonction de contrôle (∅) est donnée par la Figure (III.11). L'axe X correspond à la vitesse de la conclusion et l'axe Y à la décision à prendre. La Figure (III.11-(a)) interprète l'équation (III.8) et la Figure (III.11-(b)) interprète l'équation (III.9).

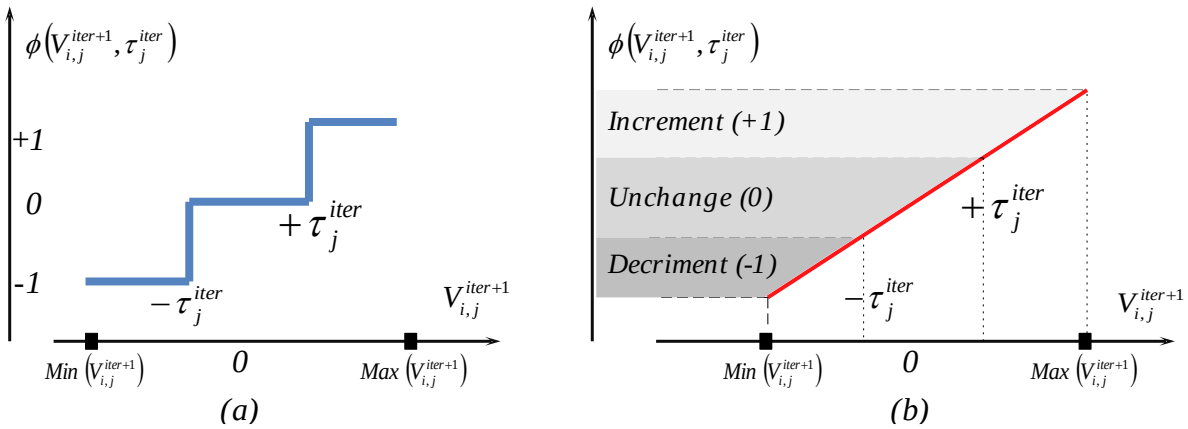


Figure III. 11 : Interprétation graphique de la fonction de contrôle

Comme le montrent ces deux figures, la décision à prendre dépend de la vitesse de la conclusion et du seuil associé. Par exemple, considérons les trois cas suivants pour une même valeur du seuil ($\tau_j^{iter} = 0.3$) : dans le premier cas ($V_{i,j}^{iter+1} = 0.7$), ici la valeur de la vitesse est supérieure au seuil, donc la décision à prendre est "d'*incrémenter*" la conclusion, soit : $C_{i,j}^{iter+1} = C_{i,j}^{iter} + 1$; Dans le second cas, la valeur de la vitesse est ($V_{i,j}^{iter+1} = -0.33$), ici la conclusion associée sera décrémentée : $C_{i,j}^{iter+1} = C_{i,j}^{iter} - 1$; Enfin, dans le troisième cas ($V_{i,j}^{iter+1} = 0.3$), la conclusion associée restera inchangée : $C_{i,j}^{iter+1} = C_{i,j}^{iter} + 0$.

III.8.3 Seuillage auto-adaptatif

Comme mentionné précédemment, le seuil τ_j^{iter} change dynamiquement pendant le processus d'optimisation. Au départ, il est fixé à 0 pour toutes les conclusions. À chaque fois

que l'algorithme trouve une nouvelle meilleure position (P_{Global}^{iter}), il met à jour un masque *(M)* qui nous permet de déterminer la nouvelle valeur de τ_j^{iter} pour chaque conclusion $C_{i,j}$ comme exprimé dans le pseudo-code de l'algorithme-1. L'équation de mise à jour des conclusions des règles floues proposée est donnée par l'équation (III.10). Le nouvel algorithme PSO modifié dédié à l'optimisation du système flou linguistique de type Mamdani selon l'approche proposée est illustré dans la figure (III-12).

Algorithme-1 : Mise à jour du seuil

Si (nouveau $(\boldsymbol{P_{Global}^{iter}})$*);*
  Pour j = 1:D, avec : D=cardinal $\left(\boldsymbol{P_{Global}^{iter}}\right)$
    Si ($\boldsymbol{P_{Global,j}^{iter} \neq P_{Global,j}^{iter-1}}$ *);*
      $\boldsymbol{M_j^{iter} = M_j^{iter-1} + 1}$*;*
    Fin Si
    $\boldsymbol{\tau_j^{iter} = M_j^{iter}/iter}$*;*
  Fin Pour
Fin Si

$$\begin{cases} V_{i,j}^{iter+1} = w^{iter} V_{i,j}^{iter} + c_1 r_1 (C_{i,j,Best}^{iter} - C_{i,j}^{iter}) + c_2 r_2 (C_{j,Global}^{iter} - C_{i,j}^{iter}) \\ C_{i,j}^{iter+1} = C_{i,j}^{iter} + \phi\left(V_{i,j}^{iter+1}, \tau_j^{iter}\right) \\ j = 1...D,\ i = 1...N \end{cases} \quad \text{(III.10)}$$

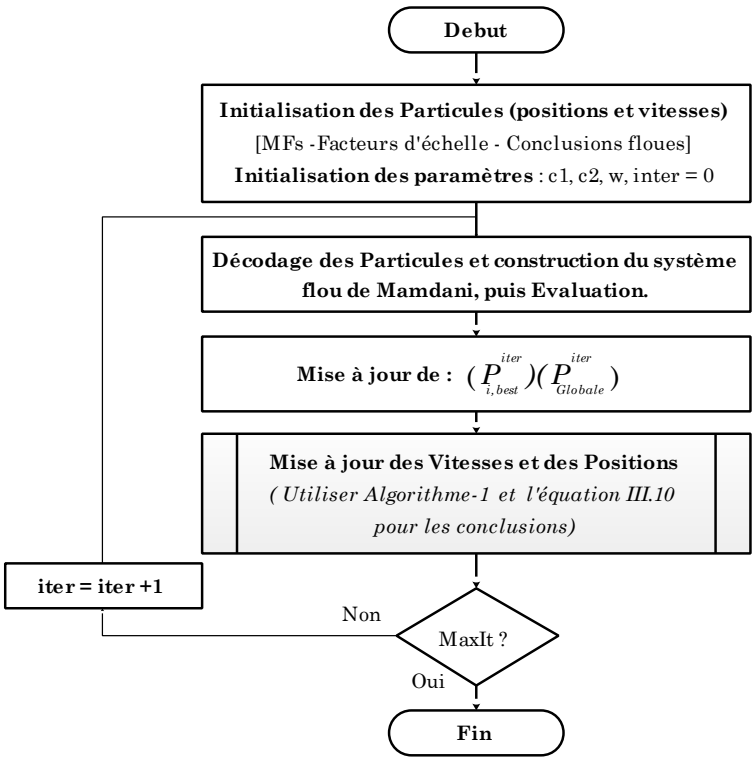


Figure III. 12 : Algorithme PSO proposé pour l'optimisation des conclusions linguistiques

III.9 Etude Expérimentale

L'étude expérimentale adoptée pour évaluer la performance de notre nouvelle approche comporte deux tests. Le premier est une étude comparative visant à situer notre proposition par rapport à d'autres dans la littérature. Le second test, quant à lui, permet de vérifier la validité et l'utilité de notre approche vis-à-vis des applications d'ingénierie à temps réel.

III.9.1 Etude comparative

Durant cette étude, on se propose d'identifier l'un des systèmes les plus connus et les plus largement étudiés dans la littérature sur la conception floue. Il s'agit d'un four à gaz à une seule entrée *u(k)* (*débit de gaz*) et une seule sortie *y(k)* (*concentration de* CO_2), décrit par une collection de données chronologiques constituée de 269 paires de données (entrée-sortie) recueillies avec un temps d'échantillonnage de 9 secondes [131].

Pour identifier ce système, l'approche proposée est utilisée avec une version standard de l'algorithme PSO, employant 25 particules dans une topologie en anneau. Les particules sont soumises à une inertie (w) décroissante et des coefficients d'accélération (c_1, c_2) égaux respectivement à 1.5 et 2.0. Le critère d'arrêt considéré est exprimé par le nombre maximal d'itérations fixé à 2000. La fonction objectif permettant l'évaluation de la qualité du système flou est l'erreur quadratique moyenne (MSE) donnée par l'équation (III.3).

Chaque particule utilise un codage mixte pour coder simultanément les paramètres des fonctions d'appartenance, les conclusions des règles floues et les facteurs d'échelle comme illustré à la Figure (III-10). La structure choisie pour le système flou est constituée de deux entrées : *y(k-1)* et *u(k-4)* et une seule sortie *y(k)*. Dans la littérature, certains auteurs affirment que cette structure est la meilleure pour identifier ce système [132]. Les deux entrées sont partitionnées en (n_1=n_2=3) fonctions nommées : {*N, Z, P*}, ce qui induit ($n_1.n_2$ = 9) règles floues. Cependant, la sortie est partitionnée en (n_3=7) fonctions nommées : {*NL, NM, NS, ZR, PS, PM, PL*}. Chaque centre de ces fonctions est décodé dans un intervalle prédéfini. Entre-temps, tous les degrés chevauchements entre les fonctions d'appartenance adjacentes sont décodés dans le même intervalle comme indiqué dans le tableau (III-1). Les entrées et la sortie floue sont décrites par un univers de discours normalisé associé à des facteurs d'échelle nommés : $G_{y(k-1)}$, $G_{u(k-4)}$, $G_{y(k)}$. Ces derniers sont décodés respectivement dans les intervalles suivants [0.1 0.5], [0.1 0.5], [7.5 10] et qui ont été identifiés par analyse des données du four

à gaz. Les conclusions des règles floues sont codées par 9 nombres entiers appartenant à l'intervalle [1, 7]. Ce codage est illustré par la figure (III.13)

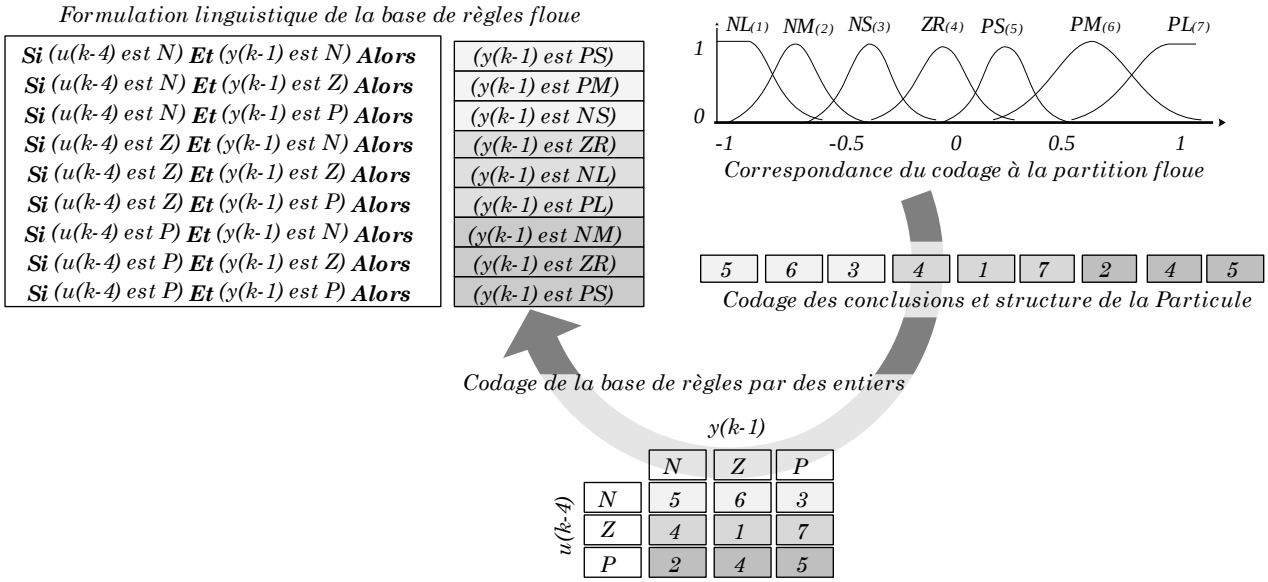


Figure III. 13 : Interprétation du codage entier des conclusions des règles floues

Tableau III. 1 : Intervalles de décodage des centres des fonctions et des degrés de chevauchement

	Fonctions d'entrées			Fonctions de Sortie							
	N	**Z**	**P**	**NL**	**NM**	**NS**	**ZR**	**PS**	**PM**	**PL**	**Chevauchement α_i**
Min	-1	-0.5	0.55	-1	-.68	-.38	-.1	.12	.4	.7	0.05
Max	-0.55	0.5	1	-.7	-.4	-.12	.1	.38	.68	1	0.7

Le tableau (III.2) énumère un large éventail de travaux portant sur la conception du système flou pour identifier le four à gaz de Box-Jenkins avec le même ensemble de données. Ces travaux utilisent différents types du système flou et différentes stratégies d'apprentissage avec différents algorithmes d'optimisation. Dans ce tableau, les approches sont listées suivant les variables utilisées en entrées, le nombre de règles floues utilisées et leur valeur MSE respective.

Les travaux mentionnés couvrent les principaux algorithmes d'optimisation habituellement utilisés dans la littérature tels que l'algorithme de recherche Tabou, l'algorithme génétique (GA), l'algorithme différentiel évolution (DE), l'algorithme des abeilles artificielles (ABC), l'algorithme de l'essaim de particules (PSO), en plus d'autres algorithmes classiques. Certains d'entre eux utilisent plus d'un algorithme pour concevoir le système flou, comme dans [141] où les auteurs ont utilisé trois algorithmes indépendants pour concevoir le système flou.

Ces travaux couvrent également les deux types de système flou : le type Sugeno (Takagi-Sugeno-Kang) et le type Mamdani. Même pour ceux qui utilisent le type Mamdani, certains auteurs ont incorporé des modifications sur la structure mise en œuvre. Pour plus de conformité, dans le tableau

(III.2), les systèmes flous de Mamdani ayant des conclusions singleton [148], [150-151] sont considérés comme un type Sugeno, en raison de la similarité entre les conclusions des deux types.

Tableau III. 2 : Modèles flous pour l'identification du système Box-Jenkins

Auteurs et années	Type du modèle	Entrées	N^{bre} de règles	MSE
Box and Jenkins, 1976 [131]	**Linéaire**	*y(k-1),y(k-2),u(k-3), u(k-4), u(k-5)*	---	0.202
Tong, 1980. [133]	**Mamdani**	*y(k-1), u(k-4)*	19	0.469
Pedrycz, 1984. [134]	**Mamdani**	*y(k-1), u(k-4)*	81	0.320
Takagi and Sugeno, 1985. [135]	**Sugeno**	*y(k-1), u(k-3), u(k-4)*	6	0.190
Xu and Lu, 1987. [136]	**Mamdani**	*y(k-1), u(k-4)*	25	0.328
Sugeno and Tanaka, 1991. [137]	**Sugeno**	*y(k-1), y(k-2), y(k-3),u(k-3), u(k-4), u(t-5)*	2	0.068
	Sugeno	*y(k-1), u(k-4)*	2	0.359
Sugeno and Yasukawa, 1993. [138]	**Sugeno**	*y(k-1), u(k-3), u(k-4)*	6	0.190
Wang and Langari, 1996. [139]	**Sugeno**	*y(k-1), u(k-4)*	5	0.158
Kim and al, 1997. [140]	**Sugeno**	*y(k-1), y(k-2), y(k-3), u(t), u(k-1), u(k-2)*	2	0.055
Farag and al, 1998. [141]	**Mamdani**	*y(k-1), u(k-4)*	37	0.111
Kang and al, 2000. [142]	**Sugeno**	*y(k-1), u(k-4)*	5	0.161
Evsukoff and al, 2002. [143]	**Mamdani**	*y(k-1), u(k-3)*	36	0.153
	Mamdani	*y(k-1), u(k-4)*	90	0.090
Khosla and al, 2005. [144]	**Sugeno**	*y(k-1), u(k-4), u(k-3)*	6	0.190
Bagis, 2008. [145]	**Sugeno**	*y(k-1), u(k-4)*	4	0.148
Zhao and al, 2010. [146]	**Sugeno**	*y(k-1), u(k-4)*	3	0.127
Su and al, 2012. [147]	**Sugeno**	*y(k-1), u(k-4)*	4	0.125
	Sugeno	*y(k-1), y(k-2), u(k-3),u(k-4)*	3	0.058
	Sugeno	*y(k-1), y(k-2), y(k-3), u(k-1), u(k-2), u(k-3)*	3	0.054
Wadhawan and al, 2013. [148]	**Linear**	*y(k-1), y(k-2), u(k-3), u(k-4), u(k-6)*	---	0.193
	Mamdani	*y(k-1), u(k-4)*	8	0.007
	Mamdani	*y(k-1), u(k-4), u(k-3)*	16	0.154
Habbi and al, 2015. [149]	**Sugeno**	*y(k-1), u(k-4)*	5	0.078
Bagis and Konar, 2016. [150]	**Sugeno**	*y(k-1), u(k-4)*	5	0.132
			10	0.116
	Mamdani	*y(k-1), u(k-4)*	10	0.120
Konar and Bagis, 2016. [151]	**Sugeno**	*y(k-1), u(k-4)*	10	0.138
			10	0.127
			10	0.154
Turki, and Sakly, 2017. [152]	**Sugeno**	*y(k-1), u(k-4)*	3	0.139

L'étude comparative s'appuie d'une part sur la précision obtenue et exprimée par le critère *(MSE)* et d'autre part sur la complexité de la structure utilisée. Habituellement, cette dernière est liée au nombre de règles floues incorporées dans la structure et au nombre de paramètres à optimiser. Afin de simplifier cette étude comparative, nous avons résumé les performances obtenues par chaque type de modèle (Mamdani, Sugeno et linéaire) dans le tableau (III.3). Ce dernier illustre la performance statistique de chacun des trois modèles indépendamment de la stratégie utilisée pour les concevoir.

Tableau III. 3 : Comparaison de notre modèle et ceux issus de la littérature

	Nombre d'entrée	Nombre de règles	MSE
modèle linéaire	5.0000 ∓ 0.0000	-----	0.1975 ∓ 0.0064
modèle Sugeno	2.8500 ∓ 1.4609	5.2000 ∓ 2.7834	0.1384 ∓ 0.0681
modèle Mamdani	2.1111 ∓ 0.3333	35.7778 ∓ 30.0241	0.1947 ∓ 0.1462
Notre modèle	2.0000 ∓ 0.0000	9.0000 ∓ 0.0000	0.1237 ∓ 0.0000

Par rapport à ces modèles statistiques, nous pouvons clairement voir la supériorité de notre modèle qui ne comporte que 9 règles avec deux entrées. La structure utilisée a réalisé une performance très honorable avec un MSE de 0,12373. Cette valeur correspond à 63,54 % du MSE moyen de toutes les structures Mamdani considérées, à 89,40 % du MSE moyen de toutes les structures de type Sugeno et à 62,64 % du MSE moyen des modèles linéaires. En d'autres termes, notre modèle offre un gain de 36,45 % en termes de précision par rapport à la moyenne statistique de Mamdani, 10,59 % par rapport à celle de Sugeno et 37,35 % par rapport à celle des modèles linéaires.

En termes de complexité, notre modèle semble également supérieur. En effet, il présente 4 fois moins de règles que la structure moyenne de type Mamdani correspondant à une réduction de 74,83 % pour un nombre équivalent d'entrées ($2 \cong 2.111$). Par rapport au type Sugeno, ce dernier présente moins de règles, mais le nombre de règles ne donne pas nécessairement une indication fiable du nombre de paramètres inconnus de ce type de modèle à ajuster. Par exemple, dans [147] qui correspond à la structure la plus précise, seules 3 règles sont utilisées, mais d'un autre côté il utilise 6 entrées et plus de 39 paramètres. Chose qui rend cette structure très complexe.

Maintenant, si nous comparons notre modèle à chaque structure de type Mamdani présentée dans le tableau (III.2). Nous pouvons remarquer que notre structure surpasse la majorité des structures proposées en termes de précision, à l'exception de celles présentées dans : [143], [141] et [148]. Pour la dernière structure, les auteurs ont utilisé seulement 10% de l'ensemble de données pour entraîner le système flou et 20% de l'ensemble de données pour le tester. En d'autres termes, le MSE relatif de 0,007 avec 8 règles et de 0,154 avec 16 règles ne représente pas la valeur réelle de ce critère pour tout l'ensemble des données.

Toutefois, par rapport aux deux premières structures floues, notre modèle reste proche de celles-ci en termes de précision, mais meilleur en termes de complexité. Il est quatre fois moins complexe que la structure proposée par Farag et al, [141] et dix fois moins que la structure proposée par Evsukoff et al [143]. Par rapport à cette dernière structure, notre modèle est plus simple et plus intuitif à régler. En effet, notre structure ne nécessite pas de contraintes supplémentaires pour régler les fonctions d'appartenance, et plus important encore, elle a moins de paramètres à ajuster. En comparaison avec les modèle de type Sugeno, notre modèle surpasse la majorité des modèles présentée en termes de précision (MSE) à l'exception des modèles présentées par : [137], [140], [147], et [149] qui semble être la plus intéressant. Les trois premières structures utilisent plus de 4 entrées induisant un grand nombre de paramètres inconnus à ajuster et rendant la structure très difficile à mettre en œuvre.

Compte tenu de la stratégie et des algorithmes utilisés pour obtenir la structure optimale, notre approche semble très compétitive. Au lieu d'utiliser beaucoup d'algorithmes indépendants ou d'algorithmes complexes très gourmands en terme de ressource tel que : les algorithmes génétiques,

l'algorithme de l'abeille artificielle.... ou une structure floue modifiée comme la structure neuro-floue, elle utilise un seul algorithme qui est la version classique de l'algorithme PSO et la structure classique du système flou de Mamdani sans aucune hybridation ou modification. Cette caractéristique rend notre approche plus adaptée aux applications d'ingénierie, puisqu'elle ne nécessite que l'algorithme PSO, peu de paramètres à configurer et peu de ressources sans aucun autre algorithme ou hybridation. Ce qui rend le processus de conception plus simple, intuitif et accessible à tous.

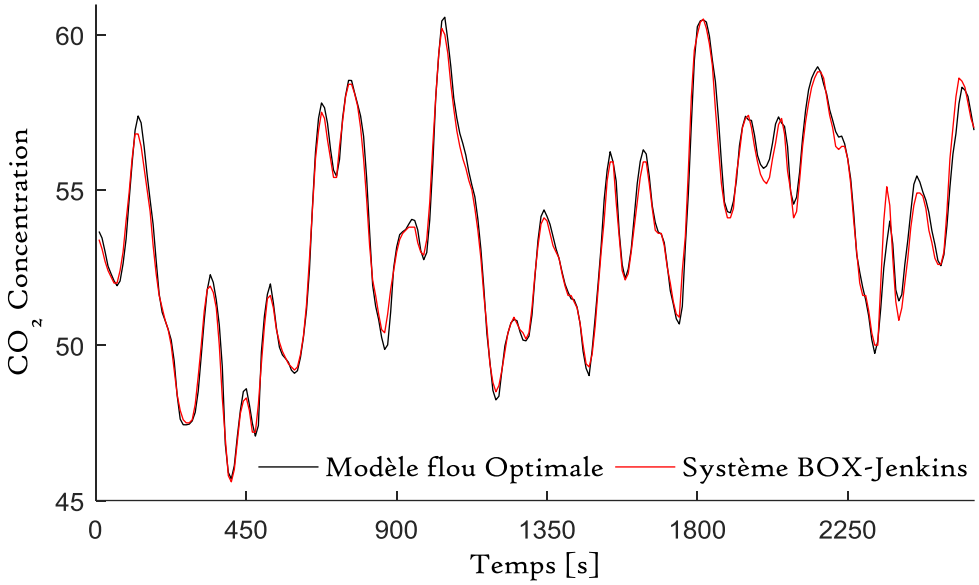


Figure III. 14 : Performance d'identification du four à gaz de BOX-JENKINS

La figure (III.14) montre la performance de la structure floue optimale à identifier le système du four à gaz de Box-Jenkins avec un MSE égal à 0,12373. Les facteurs d'échelle relatifs trouvés pour cette structure sont donnés par les valeurs suivantes : $G_{y(k-1)} = 0{,}33974$, $G_{u(k-4)} = 0{,}13028$ et $G_{y(k)} = 9{,}2852$. De cette figure, nous pouvons remarquer la similarité de la réponse du système flou optimal et celle du système réel. Ce qui prouve le succès de l'approche proposée.

La figure (III.15) montre les partitions d'entrée-sortie. Conformément à l'objectif de l'approche proposée, les partitions d'entrée-sortie sont sémantiquement faciles à interpréter. Grâce à l'utilisation d'intervalles pour ajuster les centres des fonctions d'appartenance et le taux de chevauchement entre elles, l'algorithme PSO a obtenu des partitions floues avec des fonctions d'appartenance distinguables, respectant la signification des termes linguistiques associés et couvrant l'ensemble des univers du discours des variable du modèle flou. Ainsi, toutes ces partitions respectent tous les critères d'intérprétabilité sémantique annoncée précédemment. Le tableau (III.4) illustre la base des règles floues optimales obtenue par l'approche proposée.

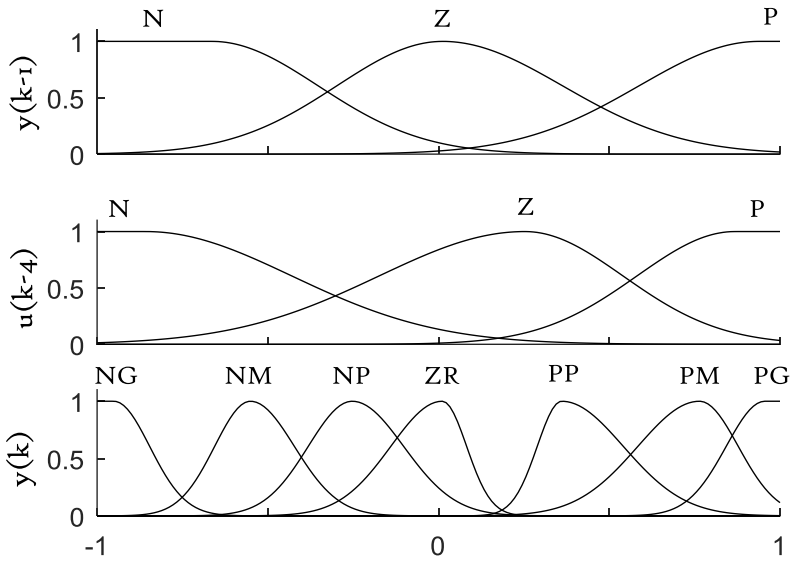


Figure III. 15 : Partitions floues optimales des entrées-sortie

Tableau III. 4 : Règles floues optimales (box-Jenkins)

		y(k-1)		
	y(k)	**N**	**Z**	**P**
u(k-4)	**N**	3	2	1
	Z	5	4	3
	P	7	6	5

III.9.2 Commande en temps réel

Dans ce deuxième test, nous abordons le problème du commande en temps réel. Nous utiliserons le pendule inversé Feedback 33-200(figure (III.16)) pour examiner l'efficacité de notre approche. Le système introduit est un test de contrôle habituellement utilisé pour examiner l'efficacité des techniques de commande [153]. En plus d'être très instable et non linéaire avec des variables multiples et fortement couplées, ce modèle tient compte des contraintes sur le mouvement rectiligne du chariot, qui est limité par des capteurs de fin de course. Par conséquent, le but de la boucle de contrôle est de stabiliser le pendule sur sa position verticale tout en maintenant le chariot éloigné des capteurs.

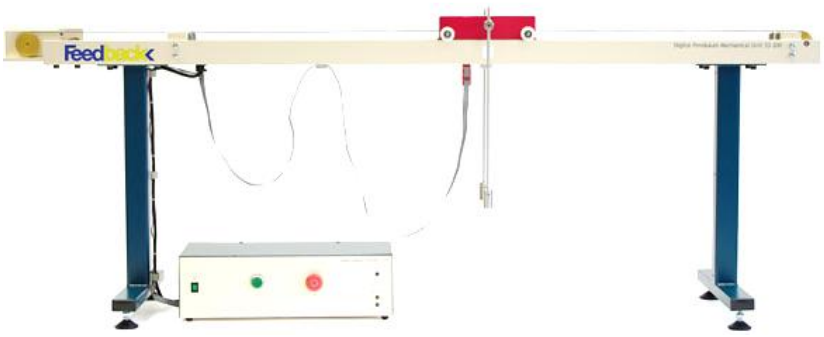


Figure III. 16 : Structure matérielle du pendule inversé Feedback 33-200

Le modèle mathématique descriptif du système chariot-pendule est exprimé par l'équation (III.11) dont les paramètres sont donnés dans le tableau (III.5). Les variables $(x, \dot{x}, \theta, \dot{\theta})$ correspondent respectivement à la position et la vitesse du chariot, l'angle et la vitesse angulaire du pendule.

$$\begin{cases} F = (M+m)\,\ddot{x} + ml\cos(\theta)\,\ddot{\theta} - ml\sin(\theta)\,\dot{\theta}^2 \\ 0 = (ml^2 + I)\,\ddot{\theta} + ml\ddot{x}\cos(\theta) + d\dot{\theta} - mgl\sin(\theta) \end{cases} \quad \text{(III.11)}$$

Tableau III. 5 : Paramètres du pendule inversé

M	*Masse du chariot*	2.4 *kg*
m	*Masse du pendule*	0.23 *kg*
l	*Longueur du pendule*	0.36 *m*
I	*Moment d'inértie*	0.099 $kg.m^2$
g	*Acceleration gravitationnelle*	9.81 m/s^2
d	*Coefficient de frottement du pendule*	0.005 *Nms /rad*
b	*Coefficient de frottement du Chariot*	0.00005 *N /m*
Te	*Période d'échantionnage*	*0.001 s*
Région de stabilisation	$\theta \in [-0.2\,, 0.2]$ *rad*	

La structure la plus intuitive du contrôleur flou pour commander ce système se compose de quatre entrées et d'une sortie. Dans ce cas, si nous utilisons seulement trois termes pour chaque variable d'entrée, il nous faudra alors 81 règles, ce qui rendra le processus d'apprentissage très gourmand en termes de temps de calcul en raison du nombre important de paramètres et de règles à ajuster. Dans [154], les auteurs ont proposé une mise en cascade de deux contrôleurs flous pour stabiliser le pendule inversé. Alors que, dans [155], les auteurs ont suggéré une structure parallèle de deux contrôleurs flous. Ces deux stratégies semblent meilleures que la première, mais il reste difficile à ajuster simultanément deux contrôleurs flous. Dans d'autres travaux, les auteurs proposent de nombreuses hybridations pour le système flou avec d'autres techniques de commande : la commande optimale [156], la commande en mode glissant [157], la commande backstepping [158]... Ces hybridations obtiennent des performances de contrôle honorables, mais restent complexes à réaliser.

Pour concilier la précision et la complexité du contrôleur flou à utiliser, nous suggérons une structure simple utilisant seulement deux entrées (E_1, E_2) et une seule sortie (F). La première entrée sera une combinaison de l'erreur rectiligne et de sa variation, tandis que la seconde sera une combinaison de l'angle de rotation et de sa vitesse telle que définie par l'équation (III.12) [159-160]. Ces combinaisons permettent une supervision globale du mouvement du chariot et du pendule simultanément, en lisant directement le plan de leur

dynamique respective formée par l'erreur et sa dérivée. Par conséquent, pour le contrôleur flou, le système est stable si : Seulement si, les dérivées et les erreurs sont égales à zéro, ce qui est le but recherché.

$$\begin{cases} E_1 = G_1 e_x + G_2 \dot{e}_x \\ E_2 = G_3 \theta + G_4 \dot{\theta} \end{cases} \quad \text{(III.12)}$$

L'entrée E_1 est exprimée par 3 fonctions d'appartenance, tandis que l'entrée E_2 et la sortie F sont exprimées par 5 fonctions. Par conséquent, 15 règles floues sont nécessaires. Pour la conception de ce contrôleur flou, seules les conclusions des règles floues et les quatre facteurs d'échelles (G_1, G_2, G_3, G_4) sont codées dans la structure des particules, alors que les fonctions d'appartenance sont uniformément réparties sur leur l'univers de discours.

L'apprentissage s'effectue sur 500 itérations. Il est basé sur la fonction objectif exprimée par l'équation (III.13), qui est une agrégation de l'intégrale du carré de l'erreur de la position linéaire et de l'angle de rotation. Cette fonction objectif est formulée de sorte à mesurer l'efficacité du contrôleur flou à faire imposer une trajectoire optimale au chariot tout en maintenant le pendule dans sa position verticale.

$$ISE = \sum_{k=1}^{K} (e_x)^2 + \sum_{k=1}^{K} (\theta)^2 \quad \text{(III.13)}$$

Le tableau (III.6) illustre les 15 règles floues optimales tandis que la Figure (III.17) montre les performances en régulation du contrôleur. On constate que le contrôleur flou obtenu régule facilement la position du chariot et du pendule sur les segments constants de la trajectoire. Cependant, pour les transitions, le contrôleur flou coordonne les deux mouvements du chariot et du pendule pour suivre la trajectoire. Il provoque lui-même un déséquilibre du pendule en déplaçant le chariot dans la direction opposée à la trajectoire souhaitée. Cette initiative lui permet de préparer simultanément la régulation du chariot et du pendule tout au long des transitions. Ce comportement souligne l'avantage de l'apprentissage automatique des systèmes flous, leur permettant de découvrir et d'acquérir de nouvelles connaissances grâce à une série de tests.

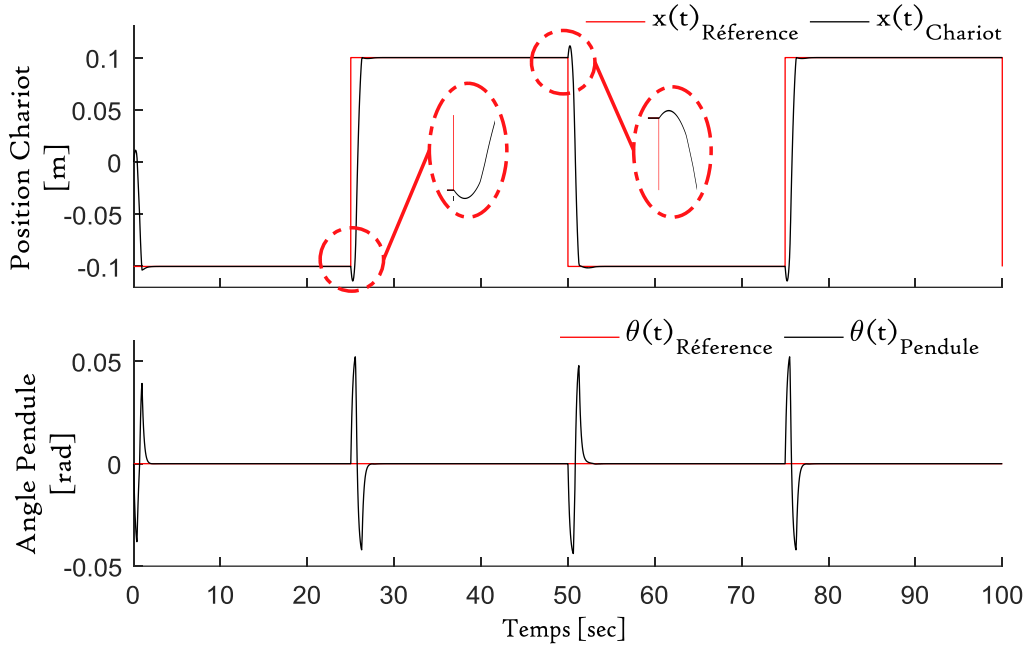
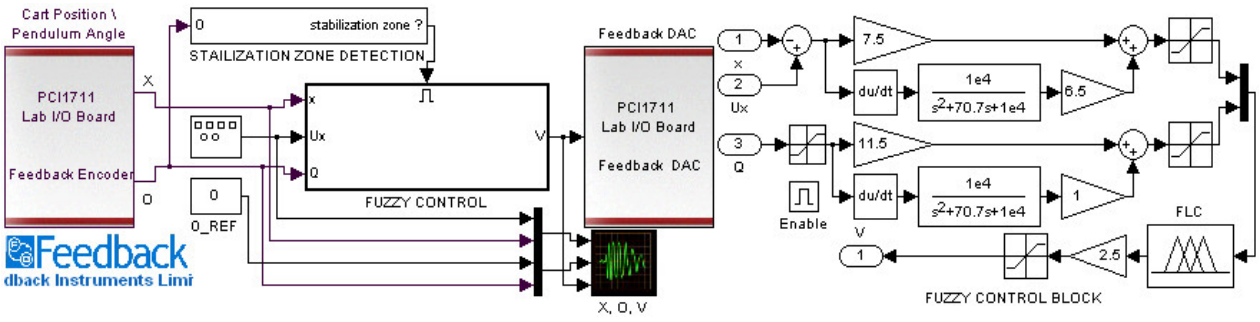


Figure III. 17 : Simulation théorique du contrôleur flou avec une trajectoire carrée

Tableau III. 6 : Règles floues optimales (Pendule Inversé)

E_2 \ E_1	N	Z	P
NL	5	4	4
NM	5	5	2
ZR	5	3	1
PM	4	2	1
PL	1	4	3

La Figure (III.18-a) montre l'implémentation dans l'environnement Matlab du contrôleur flou optimal où, deux blocs sont utilisés pour interfacer le modèle Simulink et le système réel. La Figure (III.18-b) montre la structure de la boucle de commande floue. Comme le système réel ne peut fournir que deux mesures x et θ, les vitesses $\dot{x}$ et $\dot{\theta}$ sont calculées analytiquement.



(a) (b)

Figure III. 18 : Structure logiciel de la commande floue

Dans ce premier test expérimental, nous avons reproduit le même essai théorique précèdent en utilisant le même signal de référence carré. Les performances en temps réel sont données dans la Figure (III.19). Ces résultats sont obtenus après avoir réajusté les facteurs

d'échelle des entrées pour adapter le régulateur flou aux paramètres du système en temps réel et les valeurs obtenues sont comme suit : $G_1=7.5$, $G_2=6.5$, $G_3=11.5$, $G_4=1$, $G_u=2.5$.

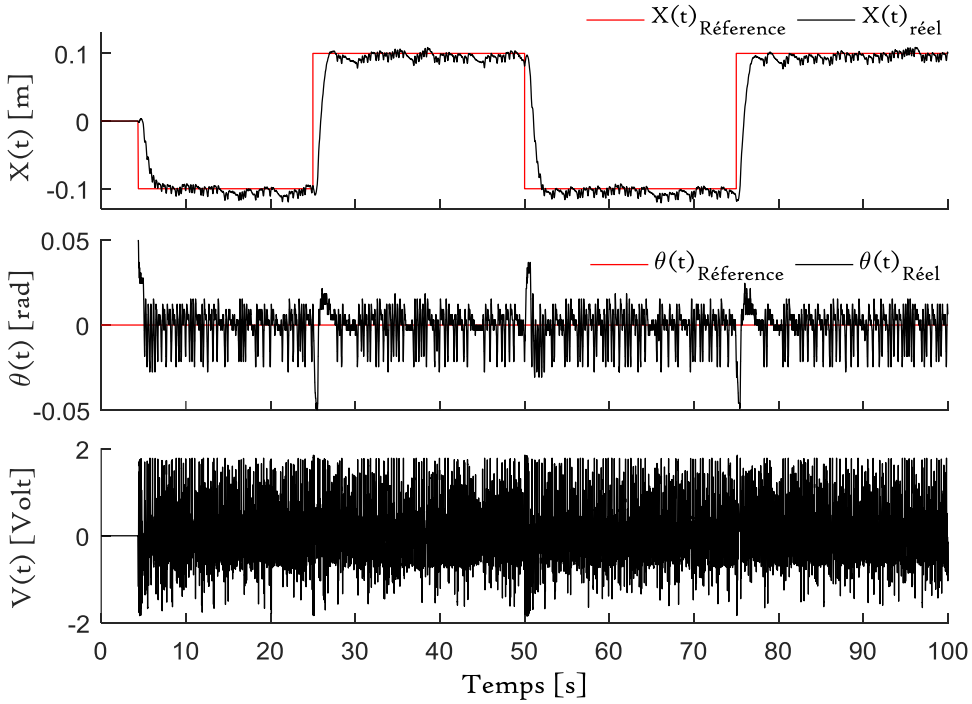


Figure III. 19 : Résultats de la commande floue en temps réel avec la trajectoire carrée

A partir de la figure (III.19), on peut remarquer que le contrôleur flou exprime les mêmes capacités à contrôler la position du chariot et du pendule avec une précision respectable en temps réel. Grâce à la stratégie adoptée basée sur la lecture du plan de la dynamique de chaque mouvement, l'apprentissage automatique a produit des conclusions floues en accord avec l'objectif initialement prévu : faire converger les erreurs et leurs dérivés vers zéro. Par conséquent, le système de pendule inversé revient toujours à la position souhaitée.

Comme mentionné précédemment, le contrôleur flou est conçu pour stabiliser le pendule dans sa position verticale, mais pas pour le redresser. C'est pourquoi nous avons placé le pendule manuellement dans la zone de stabilisation. Lorsque le pendule se retrouve dans cette zone de stabilisation, le contrôleur flou le maintien en position verticale pendant que le chariot suit la trajectoire imposée.

Les prochains tests visent à vérifier l'efficacité et la robustesse du contrôleur flou obtenu lorsque la trajectoire change. Par conséquent, de nombreux signaux ont été définis comme des références pour le déplacement du chariot. Chacun d'entre eux présente un changement, soit de forme, soit d'amplitude, soit de fréquence, soit des trois simultanément.

Les Figures (III.20) et (III.21) montrent les performances du contrôleur flou pour différentes formes de la trajectoire. Dans la première, une forme sinusoïdale avec (0,06 Hz) est appliquée, tandis que dans la seconde, une forme en dents de scie avec la même fréquence est utilisée. Pour les deux trajectoires, le contrôleur flou optimal a pu conduire le chariot sur la trajectoire souhaitée tout en stabilisant l'angle du pendule.

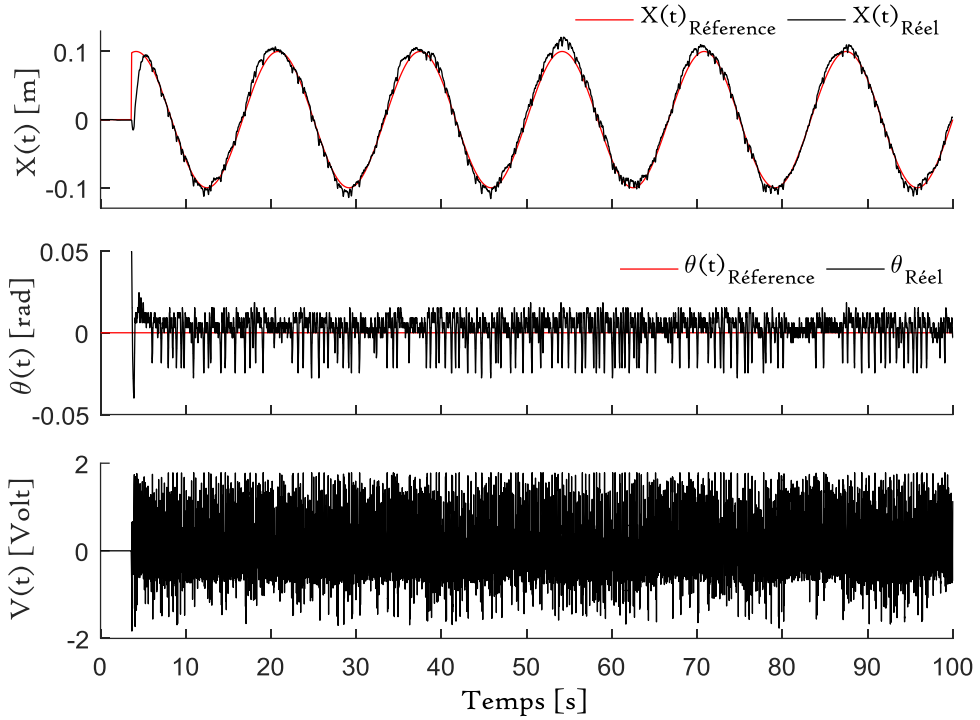


Figure III. 20 : Résultats de la commande floue en temps réel avec une trajectoire sinusoïdale

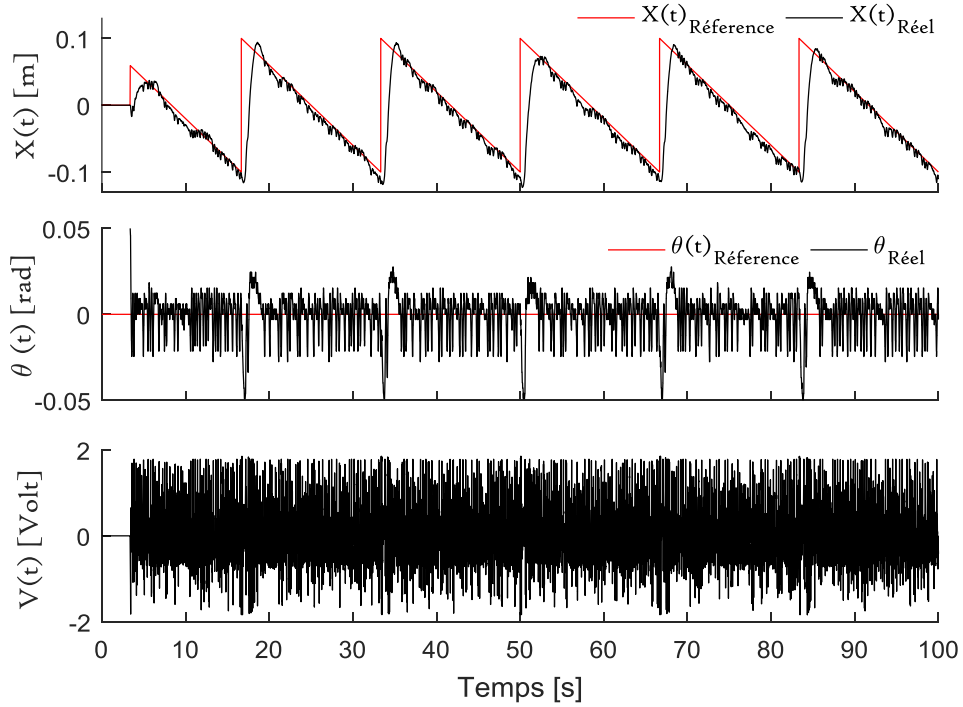


Figure III. 21 : Résultats de la commande floue en temps réel avec une trajectoire dents de scie

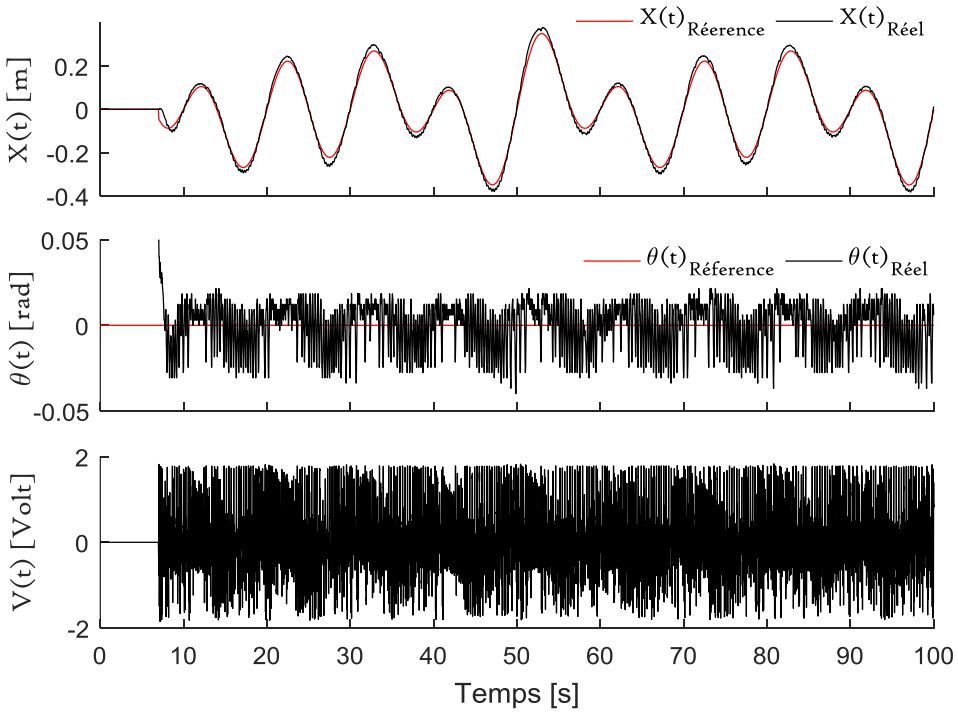


Figure III. 22 : Résultats de la commande floue en temps réel avec une trajectoire composée

Dans la Figure (III.22), la trajectoire utilisée est différente, elle est une composition de signaux sinusoïdaux. L'onde résultante est donnée par l'équation (III.14). Elle est composée de trois sinusoïdes de fréquences et d'amplitudes différentes provoquant ainsi différentes accélérations et différentes magnitudes pour le déplacement du chariot sur le rail, ce qui nous permettra de vérifier la réactivité du contrôleur flou.

$$y_k^{ref} = 0.1\ (2sin(0.2\pi k) + sin(0.12\pi k) + sin(0.08\pi k)) \qquad \text{(III.14)}$$

Toutes ces figures (Figure III.20 - III.22) prouvent la capacité du contrôleur flou optimisé à conduire le chariot avec différentes trajectoires, tout en stabilisant le pendule à sa position verticale, même s'il n'a jamais été soumis à la trajectoire auparavant. D'autre part, ces résultats confirment l'efficacité de l'approche proposée dans la recherche des conclusions les plus appropriées pour chaque problème auquel le système flou de Mamdani peut être confronté. En même temps, ils démontrent l'amélioration de l'algorithme PSO grâce à l'approche proposée, en rendant le mécanisme du PSO non restrictif au type de codage utilisé pour représenter les particules. Sans oublier, la possibilité et surtout la facilité de régler automatiquement les systèmes flous de Mamdani sans aucune connaissance préalable.

III.10 Conclusion

Durant ce chapitre, nous avons abordé la conception automatique du système flou de type Mamdani par des algorithmes d'optimisation mono-objectif. Après avoir survolé l'état de l'art sur l'utilisation des algorithmes mono-objectif, nous avons présenté une nouvelle proposition permettant de concevoir le système flou de type Mamdani de façon complétement automatique et sans aucune connaissance à priori.

L'approche de conception proposée a été validée à travers deux tests. Dans le premier test, une étude comparative a été réalisée avec de nombreux travaux portant sur l'identification floue, où notre approche a montré des performances très honorables. Le second test nous a permis de vérifier l'applicabilité du modèle flou obtenu après l'apprentissage, par l'approche proposée, sur des problèmes d'ingénierie en temps réel.

Chapitre IV

Conception multi-objectif des SIFs

IV.1 Introduction

Arriver à ce stade, tous les éléments du problème de conception des systèmes flous ont été annoncés. Bien que la démarche mono-objectif reste encore d'actualité et soit un moyen de conception de ces systèmes, elle reste toutefois loin de satisfaire toutes les recommandations et d'assurer les exigences attendues des systèmes flous vis-à-vis des différents domaines. Le nombre d'objectif à atteindre simultanément et surtout leurs contradictions constituent les principales entraves de cette démarche. Surmonter ces limitations a été justement la raison d'être d'une seconde classe de méthode de conception qui gouverne de nos jours le domaine de conception des systèmes flous. Il s'agit de la conception multi-objectif à laquelle ce présent chapitre sera entièrement consacré.

Dans ce chapitre, nous allons suivre le même ordre d'idées que dans le précèdent c'est-à-dire, nous débuterons ce chapitre par un état de l'art sur la conception automatique multi-objectif avant d'introduire pour chaque section une application illustrative sur son concept.

IV.2 Conception des SIF par algorithmes multi-objectif

La conception des systèmes flous a connu un nouveau souffle avec l'usage des algorithmes multi-objectif. Cette classe d'algorithme a su décupler le potentiel de l'apprentissage automatique des systèmes flous en fournissant la possibilité de réconcilier les différents aspects que requiert la conception de tels systèmes. L'introduction des heuristiques multi-objectif dans ce champ a donné naissance à un nouvel axe de recherche plus connu sous l'appellation de *MOEFS* (*Multi-Objective Evolutionary Fuzzy System*).

Les *MOEFSs* sont de plus en plus utilisés et le nombre de contributions ne cesse pas de grandir d'année en année au point où, il est vraiment très difficile de les citées ou de les enumérées toutes dans un seul chapitre. C'est la raison pour laquelle nous avons préferé plutôt de présenter une arboressance récapitulative, qui sera décortiquée au cours de ce chapitre. Cette taxonomie illustrée dans la figure (IV-1), englobe toutes les classes et les catégories que comporte ce domaine. Elle est considérée par de nombreux spécialites comme la vue d'ensemble sur l'etat de l'art de la conception multi-objectif des systèmes flous [161].

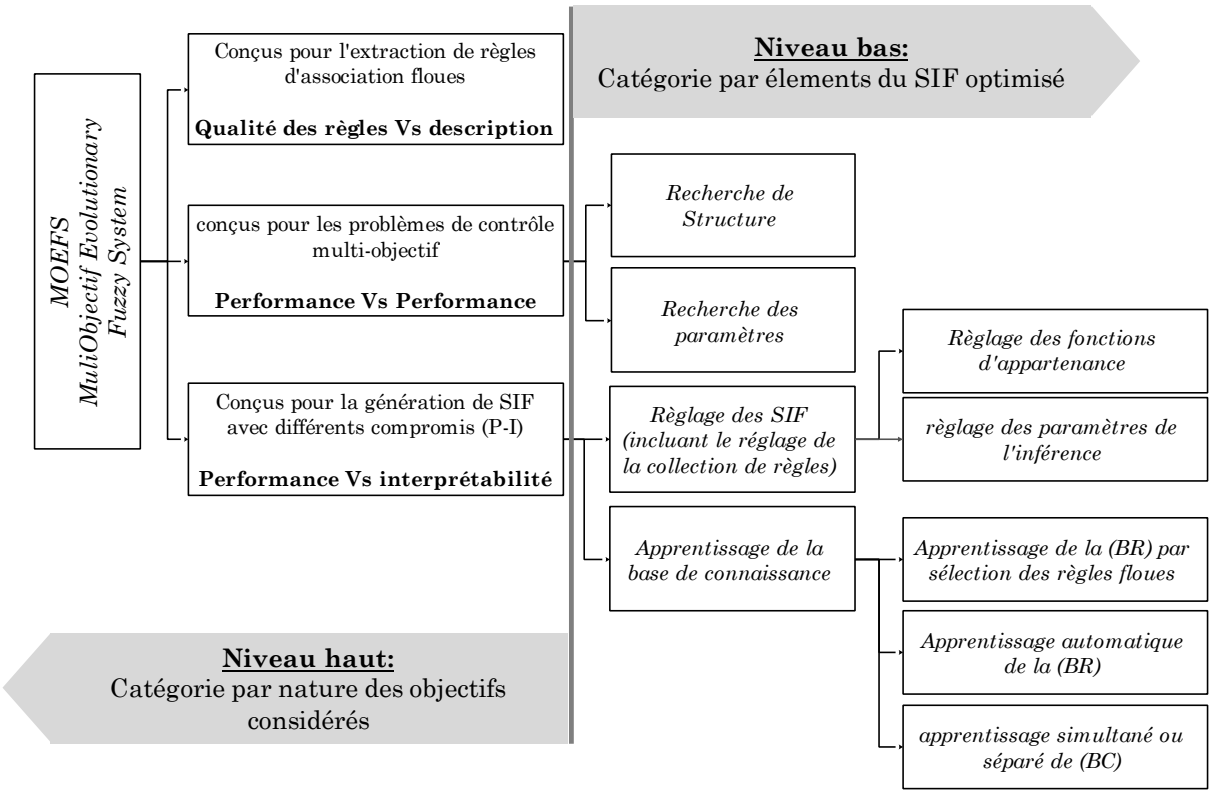


Figure IV. 1: Taxonomie des méthodes de conception multi-objectif des systèmes flous

La taxonomie se presente sous forme pyramidale à deux niveaux, où chaque niveau comporte plusieurs catégories et/ou sous-catégories. Le premier niveau (*Niveau haut*) présente

une classification suivant la nature des objectifs recherchés, tandis que le second niveau (*Niveau bas*) présente une classification selon les éléments optimisés du système flou durant la phase d'apprentissage. Cette différenciation entre la nature des objectifs et les éléments du système flou à optimiser est très importante, car elle définit le type de l'espace de recherche sur lequel opéreront les algorithmes d'optimisation.

IV.2.1 Compromis *Performance & Performance*

Cette première classe d'approches est la plus simple à formuler et la plus simple à assimiler également. Elle est principalement rencontrée dans les problèmes de commande des systèmes non-linéaires, complexes et/ou mal définis mathématiquement. Le principal objectif pour cette classe est d'obtenir la meilleure commande possible. Dans sa forme simplifiée, le problème de commande consiste à ramener le système de son état initial à un état final le plus rapidement possible et surtout avec un minimum d'énergie [162]. Bien que les deux objectifs cités sont de la même nature, ils ne sont pas pour autant concordants. En effet, la rapidité d'un système est inversement proportionnelle à sa consomation énérgitique.

Dans la litérature, on peut trouver d'autres formulations et d'autres considérations pour cette classe de conception. Certains auteurs formulent la performance du système comme un meilleur compromis entre ses performances en régime transitoire et celles en régime permanant [163]. Dans ce cas par exemple, l'énergie n'est pas considérée comme critère à optimiser. Toutes ces formulations même differentes en terme d'objectifs optimisés, restent néaumoins complémentaires et non exclusives.

La conception du système flou dans cette classe d'approche se scinde également en deux sous catégories. La première adresse la conception du point de vue purement paramétrique, c'est-à-dire, que l'algorithme se focalise sur la recherche des paramètres du contrôleur flou [164-166]. La seconde catégorie par contre, adresse le problème du point de vue structurel où, l'algorithme se focalise sur la recherche de la structure du contrôleur comme : le nombre de termes linguistiques à utiliser pour chaque partition des entrées/sorties, le nombre de règles floues que doit contenir le contrôlleur…..[167-169]. Dans certains travaux, on peut même trouver une combinaison de ces deux sous-catégories, où l'algorithme multi-objectif recherche à la fois la structure et les paramètres du contrôleur flou [170]. Dans cette catégorie de travaux, on rencontre un concept mentionné frequement dans la conception du système flou à savoir la « *Sélection des règles floues* ». Cette dernière constitue la façon la plus redondante et simple consernant l'optimisation structurelle des systèmes flous. Ici, soit

les poids des règles sont directement intégrés au codage des chromosomes via des chaines binaires, soit les poids sont déduit après une évaluation des règles floues en éliminant les moins importantes [171-173].

La démarche standard de conception dans cette catégorie (Performance Vs Performance) consiste à procéder par essai-erreur. C'est-à-dire, que l'algorithme d'optimisation observe l'ereur produite par la boucle de régulation et en fonction de sa valeur il ajuste soit les paramètres, soit la structure du contrôleur flou. Toutefois, dans d'autres travaux on peut trouver une autre façon de procéder. En se basant sur une commande déjà opérationnelle, les entrées et les sorties de l'organe de commande sont récoltées et enregistrées durant sont fonctionnement. Ces données collectées servirons ensuite à l'apprentissage du contrôleur flou. Dans ce cas le contrôleur résultant reprend la même dynamique que celle de la commande originale avec plus de tollérence.

IV.2.2 Compromis Qualité des règles & Precision

Cette seconde classe de conception est principalement dédiée au problème référé dans la littérature par le nom générique de « *Règles associatives* » [174-177], dont la classification ou la fouille de base de données font partie intégrante. Pour ce compromis, la conception vise à obtenir des systèmes flous capables d'extraire de façon automatique des connaissances dans une collection quelconque de données, sous forme d'une collection de règles floues. La conception des systèmes flous pour l'exploration de ces bases de données se déroule selon deux voies différentes. La première est dite : *Prédictive*, ici l'algorithme dispose d'une base de données bien étictée pour l'apprentissage. Durant la phase de conception, l'algoritme se focalise sur l'amélioration de l'habilité du système flou pour la prédiction

Dans la voie dite : *Discriptive*, l'algorithme se focalise sur l'amélioration de la capacité discriptive du système flou. Partant d'une spécification particulière de la part du concepteur, l'algorithme se focalise sur la formulation de règles spécifiques qui décrivent les liens entre les spécifications du concepteur et les données de sorties. Dans ce cas, la discription reste acceptable du moment que le système flou arrive à nous faire comprendre ces liens. On peut trouver également des travaux dans la littérature qui combinent ces deux approches. Ces travaux s'inscrivent dans une catégorie dans la littérature dite : *découverte de sous-groupe de règles floues intéressantes* [178-179]. Cette catégorie permet un équilibre entre la capacité de description et la capacité de prédiction du système flou face à des bases données parfois très conséquentes et complexes.

Ces deux performances (discriptive et prédictive) sont conjointement utilisées avec un autre objectif désigné par *: Qualité des règles floues*. Ainsi, dans cette classe aussi de méthodes on se ramène à deux objectifs au minimum. Le premier désigne la qualité des règles floues exprimé par : *l'importance des règles floues générées, leurs transparences, leurs cohérences, leurs tailles,…* Le second objectif quant à lui, designe la précision du système flou exprimé par le taux d'erreur que produit le système soit en terme de prédiction ou de discription.

IV.2.3 Compromis *Précision & Intérprétabilité*

Le compromis entre la précision et l'interprétabilié est probablement la classe la plus aimbigüe et la plus complexe des trois. Chaque concepteur a pratiquement une vision differente de ce compromis et plus exactement de l'interprétabilité du système flou. L'introduction de cette dernière notion à la conception des systèmes flous est relativement récente, en effet elle a été introduite pour la toute première flois par H. Ishibushi et son équipe [180] dans les années 90. La difficulté de travailler sur cette classe de compromis réside dans le fait que ses deux objectifs (précision & Interprétabilité) sont fortement contradictoires.

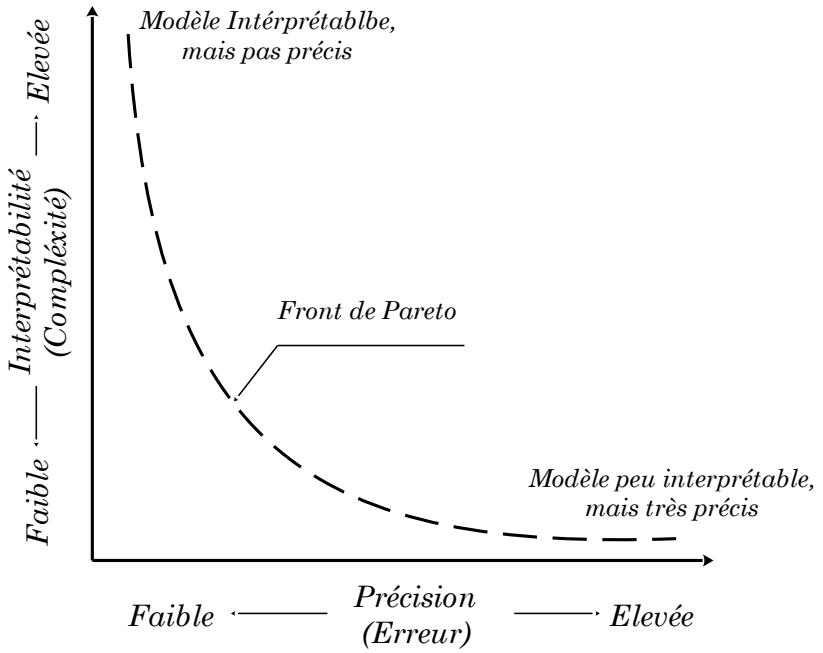


Figure IV. 2 : Compromis Intérprétabilité Vs Précisions

Comme le montre la figure (IV-2) représentative de cette classe, un système flou facilement interprétable est forcement non précis, et inversement un système flou précis est forcément difficilement inérprétable ou compréhensible. D'autre part, la formulation de cette notion d'intérprétabilité est en elle-même un challenge qui demeur jusqu'à présent une

question ouverte qui n'est pas toute à fait résolue definitivement. On trouve dans la littérature plusiseurs travaux consacrés à la question de définir des critères numériques pour exprimer cette notion [107][181-184], si certains aspects de cette notion sont plus ou moins faciles à quantifier et à mésurer par des critères numériques d'autres, par contre, sont encore difficiles à formuler et à estimer correctement.

Bien que cette classe est présentée souvent comme une classe indépendante des deux premières, elle reste cependant étroitement liée à elles. En effet, sélectionner des règles importantes pour un contrôleur flou ou mesurer la qualité des règles pour un classifieur flou, est une forme de l'interprétabilité. Ce lien avec les autres classes de conception du système flou est la seconde raison de la difficulté de travailler sur ce compromis.

IV.3 Interprétabilité des systèmes flous

Récement, la notion d'interprétabilité a reçu un intérêt considérable dans le domaine de conception des systèmes flous. Cependant d'un domaine à un autre ou plus exactement d'une application à une autre, cette notion d'interpretabilité elle-même peut prendre differents sens et aspects. En effet, chaque concepteur a sa propre perception de cette notion et de ce fait, par fois, la cohérence et la completude de la base de règles sont considérées comme étant suffisantes pour affirmer l'interprétabilité du système flou, ce qui est souvent le cas dans les problèmes de commande. D'autre fois par contre, les concepteurs ont des exigences plus complexes. Non seulement ils observent la cohérence et la completude de la base de règle, mais aussi le nombre de règles, la taille de chaque règle en plus de leur aspect sémantique et la répartition des fonctions d'appartenance ….. Ce genre d'exigence est souvent rencontré dans les applications où le système flou est utilisé pour l'extraction de connaissances.

En raison de cette discordance dans l'appréhension de cette notion d'interprétabilité, des efforts ont été menés pour unifier la définition de cette notion. Dans la littérature, on trouve plusieurs points de vue à ce propos justement. Pour certains auteurs, il existe deux niveaux d'interprétabilité pour un système flou: *un niveau bas*, et *un niveau haut* comme illustré dans la figure (IV-3). Le premier niveau (bas) répertorie les exigences simples à vérifier et à mesurer tandis que le second (haut) regroupe les aspects qui sont beaucoup plus complexes soit à verifier ou soit à mesurer. Selon cette répartition et selon les critères utilisés pour estimer l'interprétabilité du système flou, on parle soit d'une interprétabilité faible (critère de niveau bas) ou soit d'une interprétabilité forte (critère de niveau haut) [185-186].

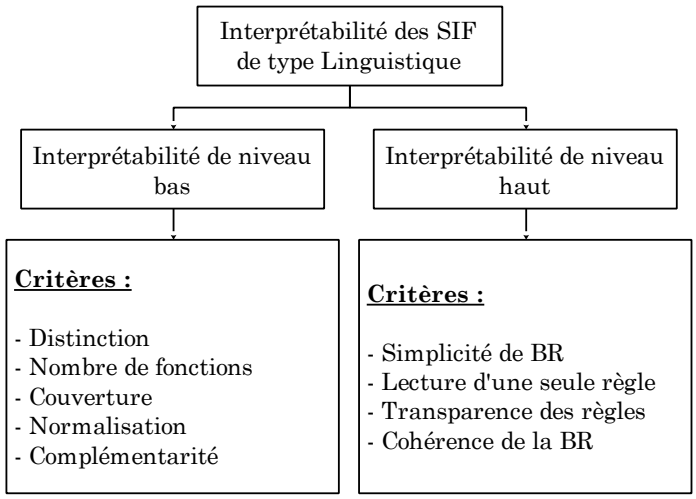


Figure IV. 3 : Classification des critères de mesure d'intérprétabilité selon deux niveaux

D'autres travaux [187] ont tenté d'apporter plus de clarité sur cette notion, en répartissant les critères numériques de mesure d'interprétabilité selon que ces derniers soient adressés aux partitions floues ou à la base de règles floues. Ensuite, selon que ces critères soient destinés à mésurer l'aspect complexcité ou l'aspect sémantique. Dans la figure (IV-4), cette classification est présentée en quatre quadrants, chacun désigne un aspect sur une seule partie des composantes du système flou.

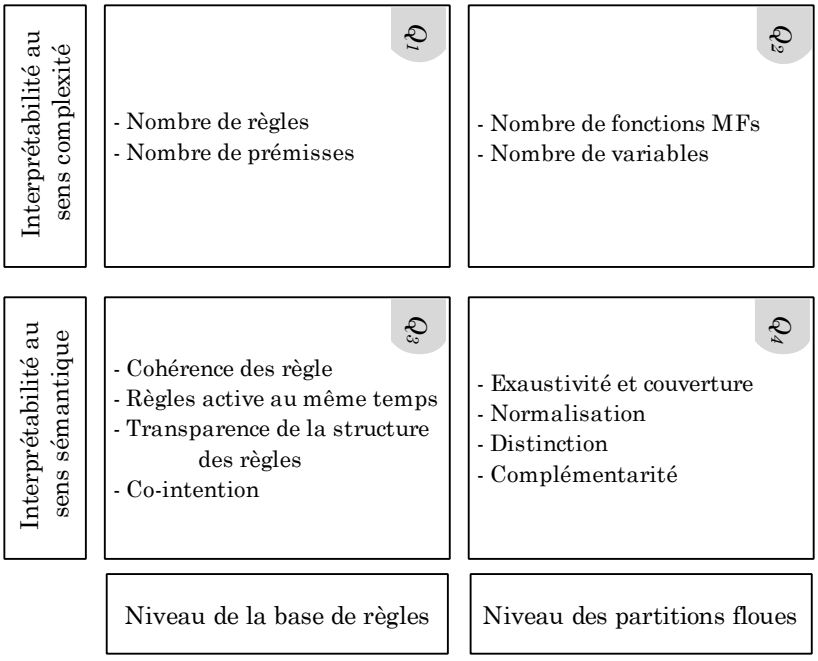


Figure IV. 4 : Classification des critères d'intérprétabilité selon les composantes du SIF

Durant les dernières années et en vue de simplifier la définition de cette notion, la litérature converge plus vers la considération de deux grandes familles d'interprétabilité. La première est relative à la compléxicité du système flou et tient compte de la base de règles et

des partitions floues. La seconde est relative à l'aspect sémantique du système flou. Ici aussi les deux parties du système flou sont considérées au meme degré d'importance.

IV.3.1 Interprétabilité au sens sémantique

Le modèle flou obtenu est considéré sémantiquement interprétable si, d'abord la sémantique de l'univers du discours est bien respectée. Autrement dit, les fonctions d'appartenance sont bien distinguables, respectent le sens sémantique de leurs termes et recouvrent tout l'univers de discours (*Section I.10.1, pp.26*)... D'autre part, la collection des règles floues obtenues à la fin de la conception sont surtout cohérentes. C'est-à-dire, elles ne sont pas contradictoires (*Section I.10.2 , pp.27*) [188-191].

IV.3.2 Interprétabilité au sens complexité

Le modèle flou est considéré interprétable au sens de la complexicité si d'abord, ce dernier comporte un minimum de règles floues. Ensuite, s'il n'utilise pas un nombre très elevé de prémisses par règle c'est-à-dire, si la taille des règles floues est aussi minimale. Ajouter à cela le nombre de fonctions d'appartenance décrivant les partitions des entrées/sorties qui est aussi un critère de complexicité également [192-194].

IV.4. Conception de contrôleur flou

Afin d'illustrer un peu plus clairement le concept de la première classe d'approche de conception (Performance Vs Performance), prenons le cas de la commande du système donné par l'equation (IV.1) [69]. Pour cette première application, nous adopterons la stratégie de conception basée sur essai-erreur. Comme le système considéré ici, est moyenement complexe, nous définirons la structure du système flou à priori. Ainis, le problème sera plutôt un réglage automatique au lieu d'un apprentissage automatique (voir figure III-2 *pp*.58).

$$y_k = \frac{1}{2}\left(\left(u_{k-1} - u_{k-1}^3\right) + \frac{y_{k-1}}{1 - y_{k-1}^2}\right) \qquad \text{(IV. 1)}$$

Le contrôleur est un PI-flou qui consiste à utiliser 5 termes linguistiques pour les deux entrées (Erreur : $e(k)$, variation de l'erreur $\Delta e(k)$) et pour la variation de la commande ($\Delta u(k)$), ce qui correspend à un ensemble de 25 règles floues. Dans ce problème, l'algorithme d'optimisation va chercher les valeurs optimales : des gains d'entrées/sortie ($G_{e(k)}, G_{\Delta e(k)}, G_{\Delta u(k)}$), des paramètres des fonctions d'appartenance et des conclusions des règles floues. Pour concretiser ceci, nous adopterons une structure des chromosmes similaire à celle présentée en (Figure III-10 *pp.67*). L'algorithme d'optimisation choisi pour cette

application est le NSGA-II avec une population de 20 individus pour des probabilités de croisement et de mutation respectivement égales à 0.85 et 0.05, executé sur 1000 générations.

Les objectifs à minimiser dans ce cas sont exprimés par l'equation (IV.6). Le première objectif correspond à l'intégrale de l'erreur absolue (*IAE*) mesurant la précision du contrôleur flou à commander le système, tandis que le second correspond à l'énergie consommée. *k* représente les instants d'échantillonnage (Te = 0.01s), *N* le nombre total d'echantillons, *u(k)* la commande du système et *e(k)* l'ecart entre la sortie du système et la consigne.

$$\begin{cases} Obj_1 = \sum_{k=1}^{N} |e(k)| \\ Obj_2 = \sum_{k=1}^{N} u(k)^2 \end{cases} \quad \text{(IV.6)}$$

La figure (IV.5) présente les résultats de simulation avec l'algorithme NSGA-2. La première figure (IV.5-a) correspond au front de pareto obtenu à la fin du processus d'optimisation, la seconde figure (IV.5-b) présente la distribution des solutions de ce dernier selon chaque fonction objectif, la troisième figure (IV.5-c) illustre l'évolution de la distance minimale $\delta(t)$ entre les individus du front et le point idéal de l'espace objectif, la quatrième figure (IV.5-d) quant à elle présente l'évolution de la dispersion des solutions $\Delta(t)$ sur le front de pareto au fil des générations et la dernière figure (IV.5-e) montre la distance hypercube de chaque individu du front de Pareto final.

Dans cette figure (IV.5) on peu voir l'évolution du processus d'optimisation de l'algorithme d'apprentissage. A travers le graphique de $\delta(t)$ on peut constater la performance de l'algorithme à trouver le minimum des deux objectifs en tentant de s'approcher le plus du point idéal. Entre temps, l'algorithme tente sans cesse de maximiser la dispersion des solutions sur le front de pareto. Ce qui a conduit à une certaine uniformité au niveau de la distribution des solutions sur le front.

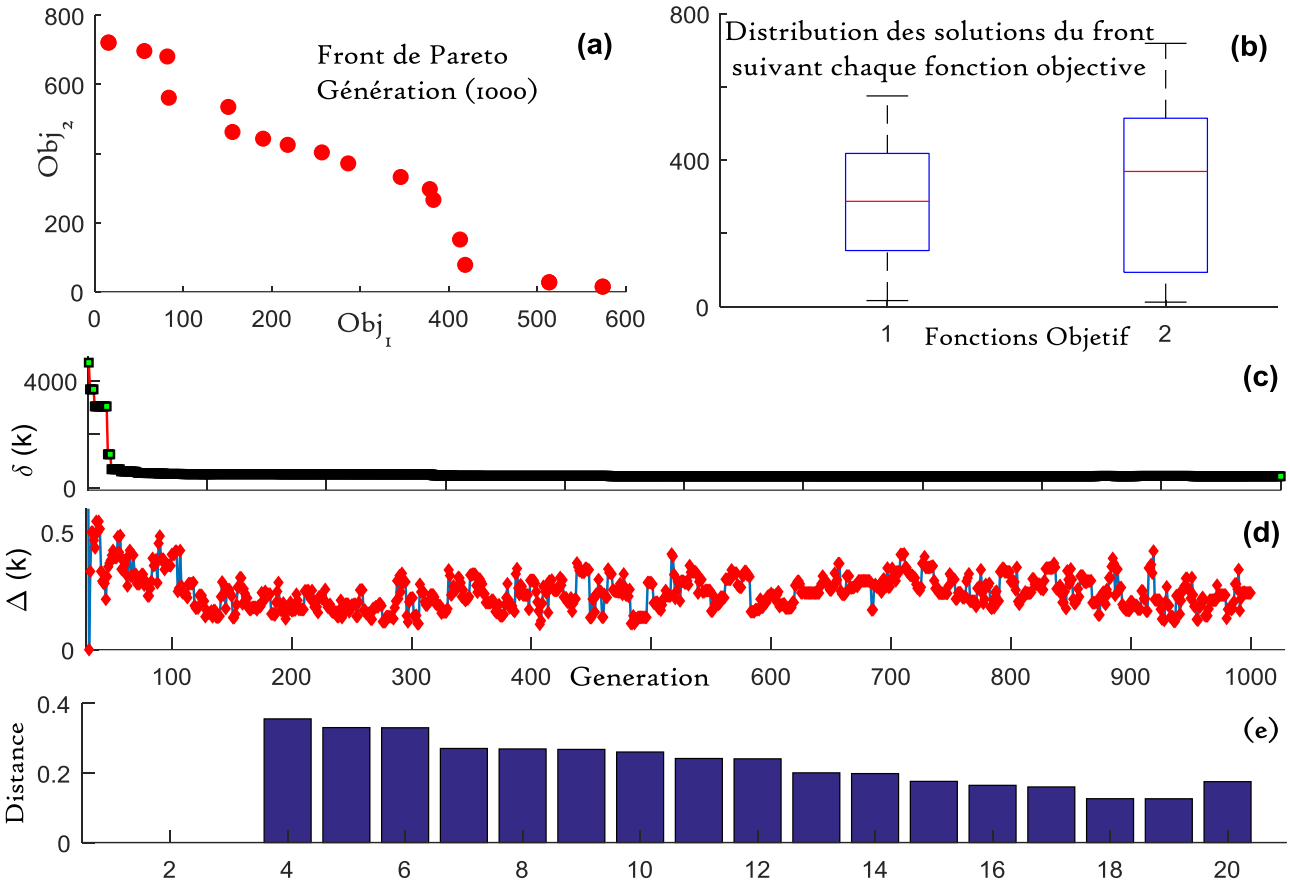


Figure IV. 5 : Résultats de simulation pour le problème de commande

Mantenant que toutes les solutions alternatives du problème sont identifiées, il faut choisir une d'entre-elles [195-196]. Pour cela, les méthodes de décision multicritères peuvent êtres utilisées comme expliquer précédemment dans (Section II.8 et la figure II.13, *pp*.53). Il très important à noté ici, que ce n'est pas une obligation de confondre les objectifs de l'optimisation et les critères de décision lors de la mise en place de ces techniques, il est toujours possible de définir d'autres critères supplémentaires pour le choix de la solution a adopté telques: le temps de réponse et le dépassement pour effectuer le choix sur le front de Pareto.

Cependant, dans le cas de notre problème le choix est un peu simple. Vu que nous avons deux valeurs indicatrices de la performance de chaque solution correpondant aux objectifs définis pour l'algorithme NSGA-2. En préférant la précision (Obj_1) à l'énergie (Obj_2) nous avons choisi la solution offrant le plus de précision.

La figure (IV.6) présente la performance du contôleur flou correspondant à la solution choisie sur le front de Pareto. Le tableau (IV.1) illustre la base de règles floues, et les facteurs d'échèlles du contrôleur en plus de la valeur des deux fonctions objectifs.

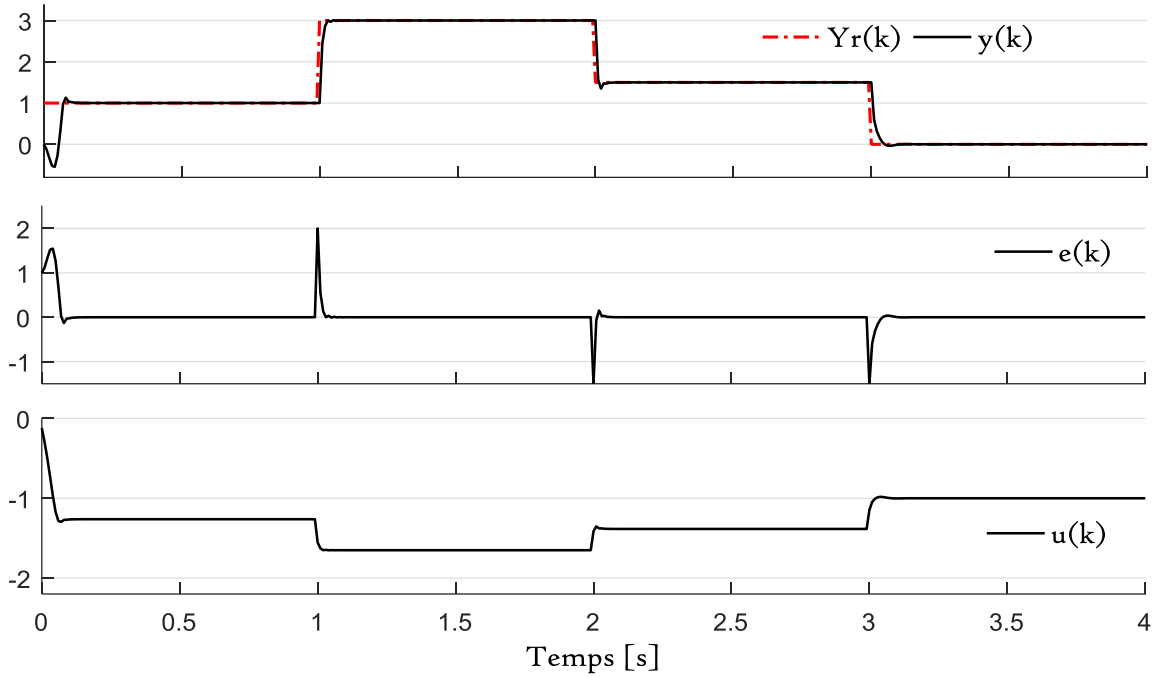


Figure IV. 6 : Performance du contrôleur flou sélectionné

A partir de la figure (IV.6) on peut voir facilement que la technique de conception a pu offrir une solution à ce problème de commande, en ajustant simultanément les gains des entrées/sortie, les paramètres des fonctions d'appartenance et la base des règles floues.

Tableau IV. 1 : Base de règles floues

$e(t)$ \ $\Delta e(t)$	NG	N	Z	P	PG
NG	*N*	*N*	*P*	*N*	*NG*
N	*PG*	*PG*	*N*	*NG*	*Z*
Z	*Z*	*PG*	*N*	*N*	*NG*
P	*P*	*P*	*N*	*NG*	*PG*
PG	Z	PG	N	Z	P

Gains des entrées/sortie

$D_{e(t)}$	$\Delta_{e(t)}$	$\Delta_{u(t)}$
0.087916	***0.13749***	***1.0421***
Obj 1 : IAE		***Obj 2 : Energie***
16.3235		***718.5703***

IV.4.1 Conception du contrôleur flou avec un minimum de règles

Dans l'application précédente, la conception du conrôleur flou a été adréssée du point de vue purement paramètrique qui est la formulation la plus simple. L'algorithme d'optimisation s'est focalisé sur la rechercher des pramètres du contrôleur flou sans toucher à sa structure qui est définie au préalable. Dans cette seconde application, nous allons intégrer la structure du contrôleur flou ou plus exactement le nombre de règles qu'il contient.

Comme nous l'avons avancer précedement, il existe deux façons principales de procéder dans ce sens. La première méthode, se base sur une évaluation des règles floues avant de décider quelle règle à retenir et quelle règle à supprimer. Dans [197] [118], nous avons proposé une démarche basée sur la mésure du degré d'activation des règles durant la simulation du contrôleur flou pour enfin décider quelles sont les règles importantes à retenir et quelles sont celles inutiles qu'on peut suprimer. Le principe de cette approche est très simple, il suffit de fixer à priori un seuil de tolérence (τ), puis toutes les règles ayant un degré d'activation inférieur à ce seuil seront considérées comme inutiles et donc suprimées. Bien que cette démarche est simple et intuitive, elle reste toutefois sensible à la valeur de (τ) qui est difficile à fixer à priori.

La seconde méthode par contre repose sur le codage des règles ou de leurs poids dans le chromosome lui-même. Soit au même titre que les autres paramètres, soit on considérant une hiérarchisation dans ces paramètres. Pour notre cas, nous allons utilisé la première approche c'est-à-dire que les poids seront codés par une chaine binaire au même titre que les paramètres des fonctions d'appartenance, les conlusions de ces règles floues et les gains des entrées/sortie comme illustré dans la figure (IV.7). Alors si le poid lié à la conclution de la règles (*i*) est égal à 1, la règles est retenue. Dans le cas contraire, la règles est éliminée.

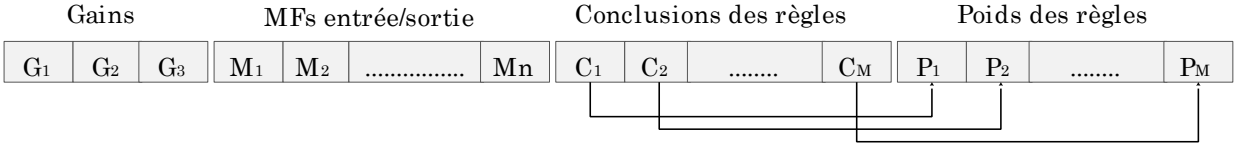


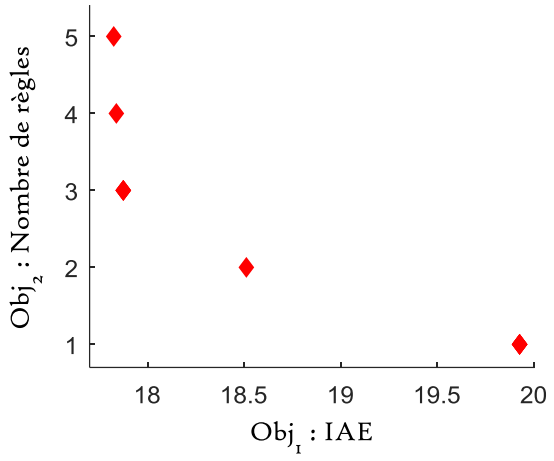
Figure IV. 7: Structure du chromosome avec les poids des règles floues

Pour la simulation de cette approche, nous allons reprendre les mêmes conditions que dans l'application précédente. De même, nous allons garder la première fonction objectif exprimant la précision du contrôleur flou (*IAE*). Cependant, la seconde fonction objectif sera le nombre de règles floues selectionnées qui est donnée par l'équation (IV.9).

$$Obj_2 = \sum_{i=1}^{N_R} \left(P_i / P_i \neq 0 \right) \qquad \text{(IV.9)}$$

Avec : (N_R) est le nombre total de règles codées dans le chromosome, *Pi* : poid de la règle (*i*).

Les résulats de simulation après 1000 générations sont illustrés dans la figure (IV.8) qui comporte le front de Pareto obtenu et des valeurs numériques des objectifs pour chaque solution du front.



Solutions du Front de Pareto

Obj_1	Obj_2
17.8248	5
17.8384	4
17.8743	3
18.5111	2
19.9250	1

Figure IV. 8 : Front de Pareto obtenu et valeurs des objectifs

Là aussi, pour le choix de la solution à implémenter on a préfererer la précision à la complexcité. La solution sélectionnée est celles correspondante aux valeur des objectifs mensionnée en couleur grise dans la liste de la figure (IV.8). Les performances du contrôleur flou associé à cette solution sont illustrées dans la figure (IV.9). La base de règles et les gains d'entrées/sortie sont aussi donnés par le tableau (IV.2).

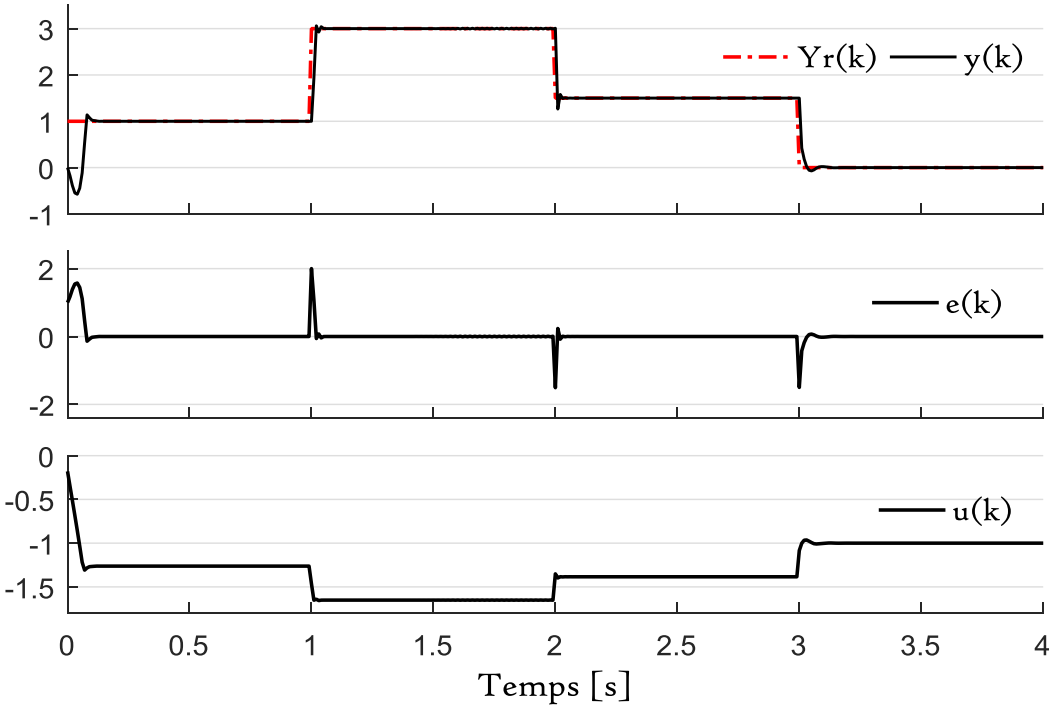


Figure IV. 9 : Performance de la solution sélectionnée du front de Pareto

Tableau IV. 2 : Base de règles floues

$e(t)$ \ $\Delta e(t)$	NG	N	Z	P	PG
NG	***Z***				
N	***PG***				
Z					
P					
PG	**Z**		**Z**	**Z**	

Gains des entrées/sortie

$D_{e(t)}$	$\Delta_{e(t)}$	$\Delta_{u(t)}$
0.50131	***0.084811***	***2.0695***

A partir de la figure (IV-9), on peut voir que le concept de sélection de règles floues est un moyen efficace pour réduire la complexcité du contrôleur flou. Malgré cette réduction du nombre de règles, le contrôleur flou assure tout de même un niveau de performance très similaire au premier cas où toutes les règles ont été considérées. Ceci peut être expliquer par le fait que les règles éliminées sont effectivement inutiles ou qu'elles ont moins d'effet sur la performance du contrôleur flou.

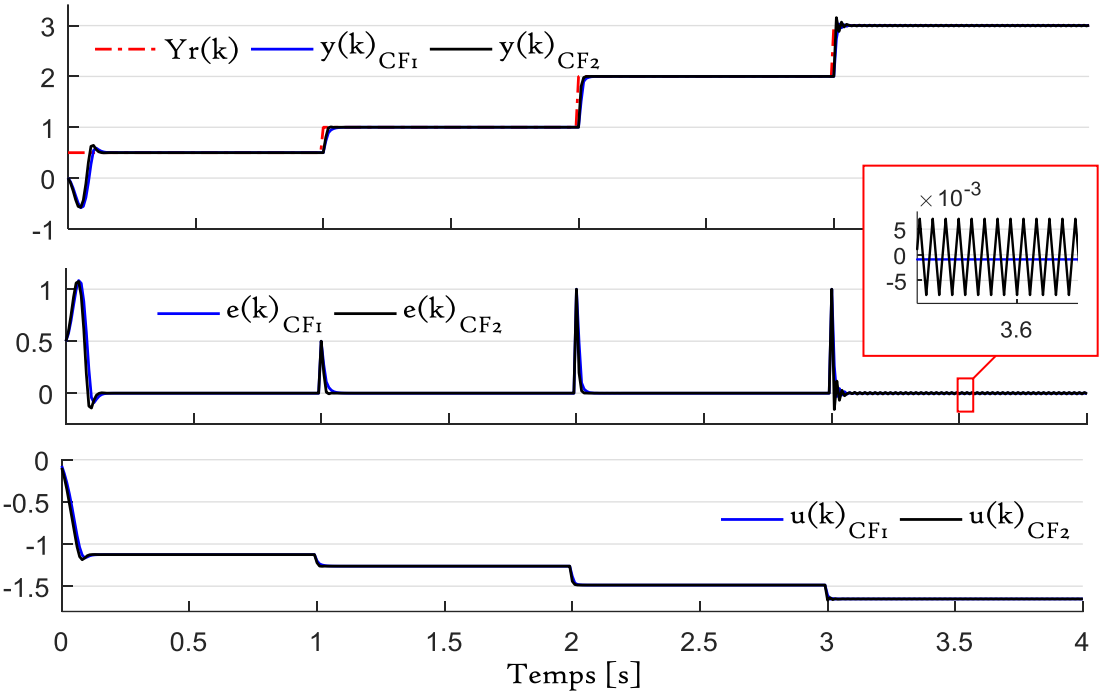


Figure IV. 10 : Performance du contrôleur flou avec et sans sélection de règles

Pour s'assurer que l'algorithme a éliminé seulement les règles inutiles, les deux contrôleurs ont été testés sur une autre trajectoire complètement nouvelle qu'aucun d'entre eux n'a jamais vue auparavant (durant la procédure d'apprentissage). La figure (IV.10) montre les résultats de simulation des deux contrôleurs : en couleur bleue, le contrôleur avec toutes les 25 règles floues (Table IV.1), et en couleur noire, le contrôleur flou après élimination des règles inutiles (tabe IV.2). Effectivement, la sélection des règles n'a pas

affecté la performance du contrôleur flou, ce qui confirme bien que les règles éliminées ne sont pas concrêtement très importantes.

IV.5. Conception de classifieur flou

Pour cette section, nous allons exposer un cas d'étude bien connu dans la littérature afin de mieux illustrer le principe de la classe de conception dédiée au compromis (Qualité des règles - Performance). Il s'agit de la classification des fleurs Iris [198-200] selon trois espèces differentes nommées : *la Setosa, la Versicolor* et *la Verginica* (Figure IV.11*).* Pour faire cette classification on se basera sur quatre attributs phisiologiques et qui sont: *les longueurs des Sépales (SL)* et *leurs largeurs (SW)*, en plus *des longueurs des Pétales (PL)* et *leurs largeurs (PW).* Dans cette application, nous souhaitons concevoir un classifieur flou pour cette base de données Iris qui comprend un ensemble de 50 exemples pour chaque type de fleur, ce qui fait un total de 150 exemples répertoriés et correspondent à 150 fleurs.

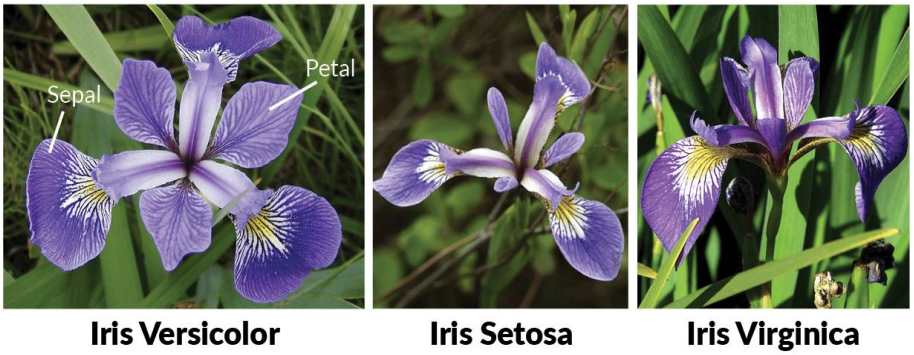


Figure IV. 11 : Classes de la base de données des fleurs iris

Le classifieur flou à concevoir est donc constitué de quatres entrées dont chacune est liée à un attribut, et une seule sortie indiquant la classe de la fleur décrite par les attributs. Toutes les partitions d'entrées/sortie sont constituées de trois fonctions d'appartenance ce qui induit un total de 81 règles. Ainsi le codage de toutes ces règles implique l'utilisation de chromosmes de taille très conséquente qui peut être nuisible à l'algorithme d'optimisation. Pour palier à cette contrainte, nous n'allons pas coder toutes ces règles, mais juste quelqu'unes qui vont être construites aléatoirement par l'algorithme via ses propres opérateurs, et donc la taille du chromosme est à priori limitée. Sur les 81 règles floues possibles, nous allons juste coder 20 règles au maximum dans chaque chromosme. L'algorithme se chargera alors de trouver les meilleures configurations pour ces règles floues, tout en éliminant les règles considérées inultiles et/ou redondantes .

Pour atteindre cet objectif, les règles codées dans les chromomoses ne seront pas toutes utilisées. La base de règles du classifieur flou est construite au fure et à mesure via une procedure d'analyse et de selection. Cette procédure est incorporée dans le processus de décodage du chromosme en vue d'améliorer la consistance de la base de règles ainsi que sa cohérence également, en éliminant toute redondance et tout conflit entre les règles. Cette procedure (figure (IV.12)) vise à obtenir des règles interprétables, en réduisant leur nombre d'abord (minimum de règles) et en augmentant leurs lisibilités et leurs compréhensions en réduisant le nombre de prémisses dans chaque règle.

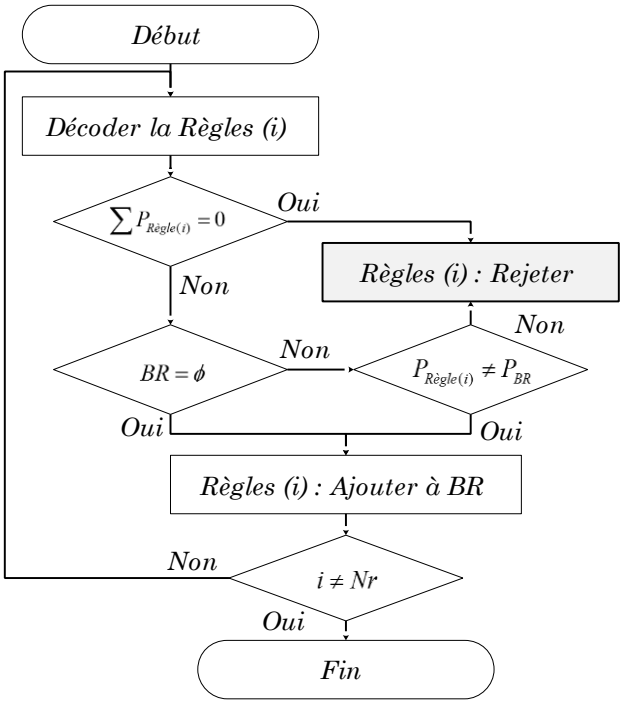


Figure IV. 12 : Algorithme de traitement des règles floues

Dans cet algorithme (Figure IV.12), *BR* correspond à la base de règles définitive du classifieur flou, *Nr* désigne le nombre total de règles codées par le chromosome (*Nr* = 20), $P_{règle}$ *et* P_{BR} désignent respectivement la prémisse de la règle décodée et l'ensemble des prémisses des règles déjà admises dans la base du classifieur flou. Pour chaque règle floue, une fois décodée elle est vérifiée par l'algorithme. Les règles qui ne sont pas accéptées se voient attribuer un poids égal à zéro (*éliminée*). Celles par contre qui ne sont pas redondantes et qui ne sont pas conflictuelles sont admises. La figure (IV.13) illustre le codage du chromosome en plus de son interprétation.

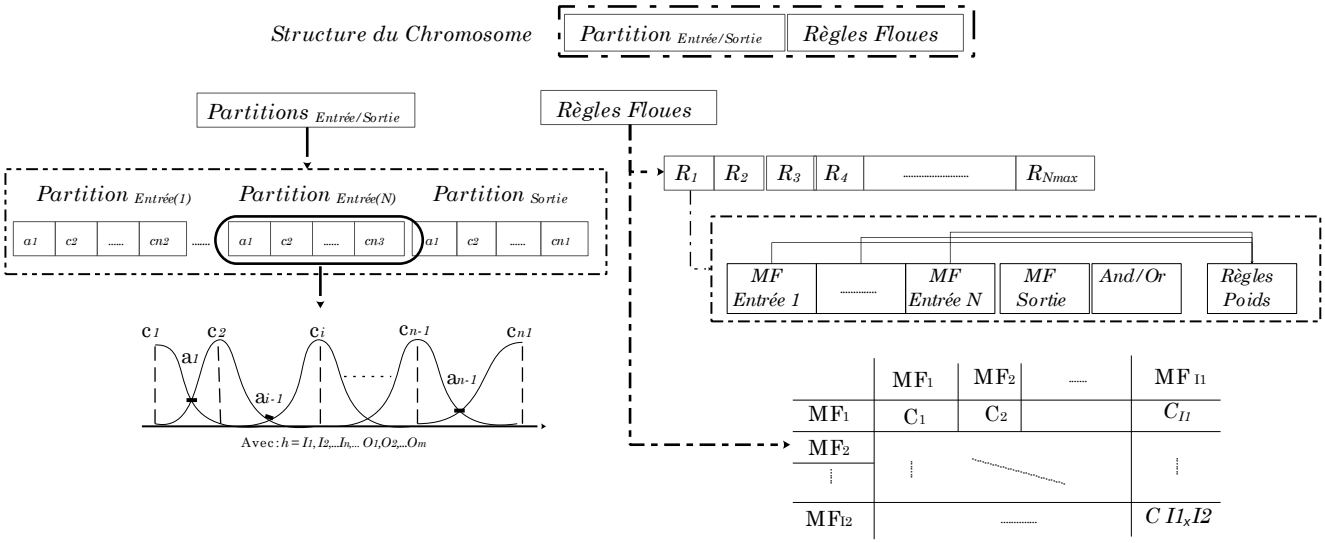


Figure IV. 13 : Structure et interprétation des chromosomes

Dans cette application nous garderons les mêmes conditions de simulation que celles de l'application précédente à l'exception de la taille de la population, qui est de 32 individus maintenant. Les objectifs d'optimisation sont aussi différents, dans ce genre d'application nous souhaitons d'une part obtenir un classifieur ayant un maximum d'habilité de prédicition (minimum d'erreur), et d'autre part obtenir une structure simple à comprendre donc un minimum de complexcité et un maximum de transparence pout les règles. Ces deux objectifs sont formulés par l'équation (IV.8)

$$\begin{cases} Obj_1 = \dfrac{1}{Nd}\sum_{k=1}^{Nd}\left(y^{Réference} - y^{Classifier}\right)^2 \\ Obj2 = Nra \times \left(\sum_{j=1}^{Nra}\sum_{i=1}^{Np}\left(P_{ij} \neq 0\right)\right) \end{cases} \qquad \text{(IV.8)}$$

Les paramètres *Nra, Np, Pij* corrrespondent respectivement au nombre de règles floues admises dans la base du classifieur, le nombre d'attributs ou d'entrées et finalement la prémisse relative à l'entrée (*i*) dans la règles (*j*). Cette formulation des deux objectifs incitera l'algorithme à observer d'abord la précision du classifieur en réduisant l'écart entre la réponse de ce dernier et les données de références. D'autre part, l'algorithme tentera d'aboutir à une base de règles comportant un minimum de règles floues et un minimum de prémisses, ce qui simplifirait considérablement la compréhension de ces dernières et donc de l'expertise réalisée sur la base de données.

La procedure d'analyse et de selection de règles floues combinée à un codage des fonctions d'appartenance comme celui présenté dans le chapitre précédent, assure une interprétabilté sémantique des règles floues. En effet, avec ces deux mécanismes le respect de la sémantique des règles est assuré, d'une part nous aurons des partitions floues qui seront

facilement lisibles (*Figure I.18. , pp.27*) et d'autre part nous aurons des règles floues qui seront consistantes, non conflictuelles, et non redondantes. Le processus de conception de ce clasifieur par l'algorithme multi-objectif est entièrement résumé dans la figure (IV.14).

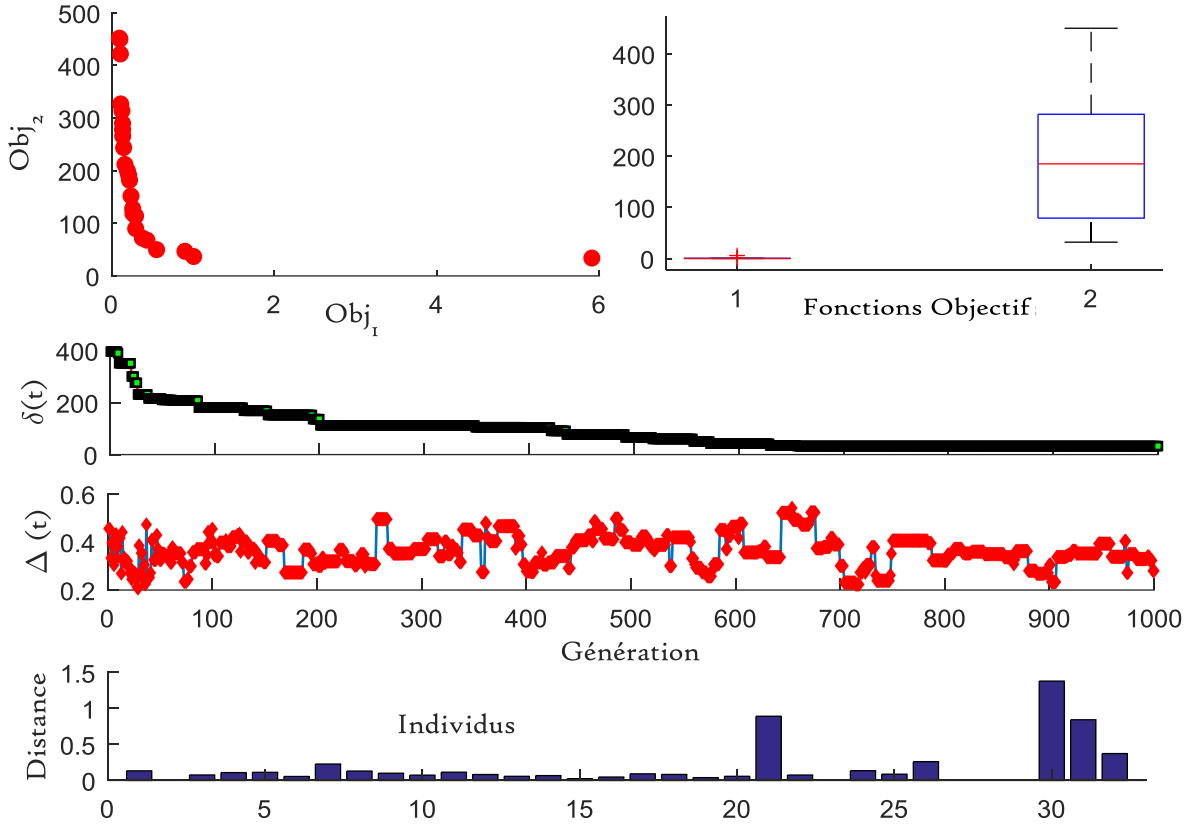


Figure IV. 14 : Résultats de simulation pour le problème de classification

La figure (IV.15) présente à la fois la description des partitions floues du classifieur en plus des deux phases d'apprentissage et de test. La procedure de codage des partitions floues adoptée a permis l'obtention de fonctions d'appartenance distinguables et facilement lisibles ce qui a assuré le premier aspect de l'interprétabilité sémantique du classifieur relatif aux partitions floues. L'introduction de l'algorithme de la figure (IV.12) a significativement augmenté d'avantage la qualité de la base des règles floues comme illustré dans la figure (IV.16). Du point de vue précision, le classifieur sélectionné est assez bon du moment qu'il a réussi à apprendre les liens entre les entrées et la sortie. Cela peut êre observé à travers les valeurs atteintes pour le premier objectif durant phase d'apprentissage (qui est de l'ordre de *0.91377*) et durant la phase de test (*MSE=2.99*) sur de nouvelles données qu'il n'a jamais vues durant la phase d'apprentissage.

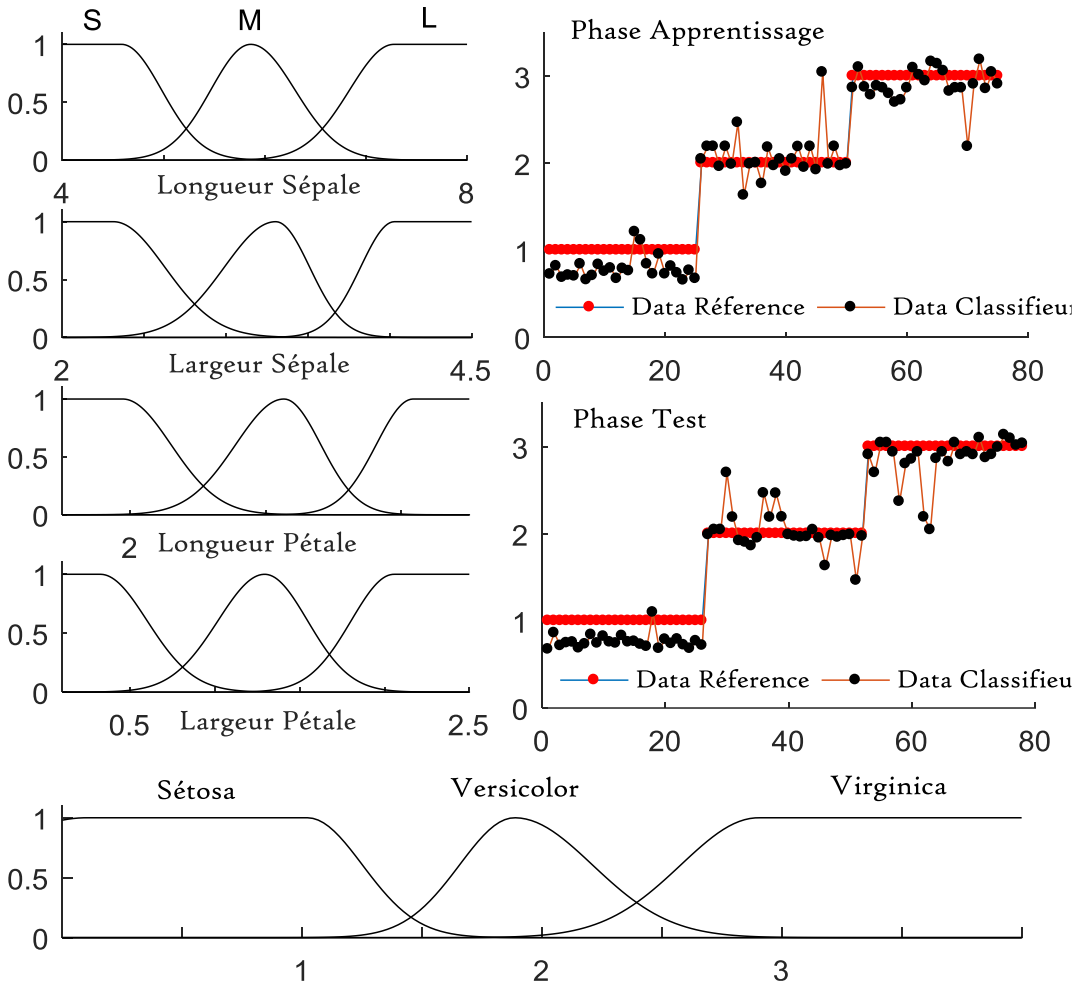
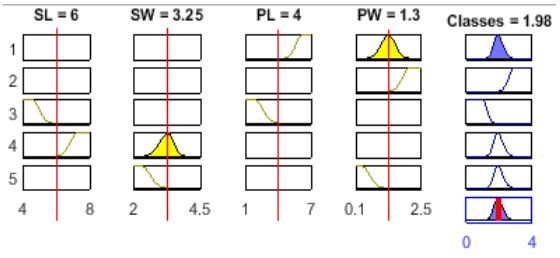


Figure IV. 15 : Performance du classifieur sélectionné

Base de règles floues sous forme graphique



Base de règles floues sous forme linguistique

1. ***If*** *(PL is L)* ***or*** *(PW is M)* ***then*** *(Classes is versicolor)*
2. ***If*** *(PW is L)* ***then*** *(Classes is virginica)*
3. ***If*** *(SL is S)* ***and*** *(PL is S)* ***then*** *(Classes is setosa)*
4. ***If*** *(SL is L)* ***and*** *(SW is M)* ***then*** *(Classes is versicolor)*
5. ***If*** *(SW is S)* ***and*** *(PW is S)* ***then*** *(Classes is versicolor)*

Figure IV. 16 : Table optimale de règles du classifieur flou

IV.6. Conception de systèmes flous Interprétables

Dans cette dernière section, nous adresserons principalement le compromis précision-intérprétabilité. Il est donc question de concevoir un système flou précis et surtout humainement compréhensible. L'application qu'on se propose d'étudier est l'identification du système du four à gaz de Box-Jenkins déjà présenté dans le chapitre 3 (*section III.9.1, pp.70*). Dans cette application nous allons considérer la même configuration pour le système flou, c'est-à-dire : deux entrées et une sortie et chacune d'entre-elles est associée un gain.

Notre objectif dans cette application sera d'obtenir: 1- des partitions floues sémantiquement interprétables (*Couverture, normalisation*, …) ; 2- un minimum de termes linquistiques ; 3- une base de règles simple et facilement compréhenssible ; 4- un jeu minimal de paramètres. Pour améliorer la qualité du système flou nous n'allons pas simplement agir sur les poids des règles floues, mais nous allons plutôt coder différents maillages pour l'espace des entrées, ce qui va certainement produire un nombre de règles différent. D'autre part nous allons coder le nombre de fonctions et leur forme. Finalement, et en vue d'augmenter la coopération et la cohérence des règles floues, nous allons également coder les opérateurs du moteur d'inférence (*conjonction, implication, aggrégation et défuzzification*) [201].

De cette manière l'algorithme d'optimisation accède pratiquement à toutes les composantes du système flou, et à l'exception du nombre d'entrées/sortie, rien n'est fixé ou choisi préalablement. Chose qui devra normalement permettre une meilleure exploitation du potentiel de la machine lors de la phase d'apprentissage. La figure (IV.17) illustre le codage de toutes ces composantes en plus de l'interprétation de chaque élement.

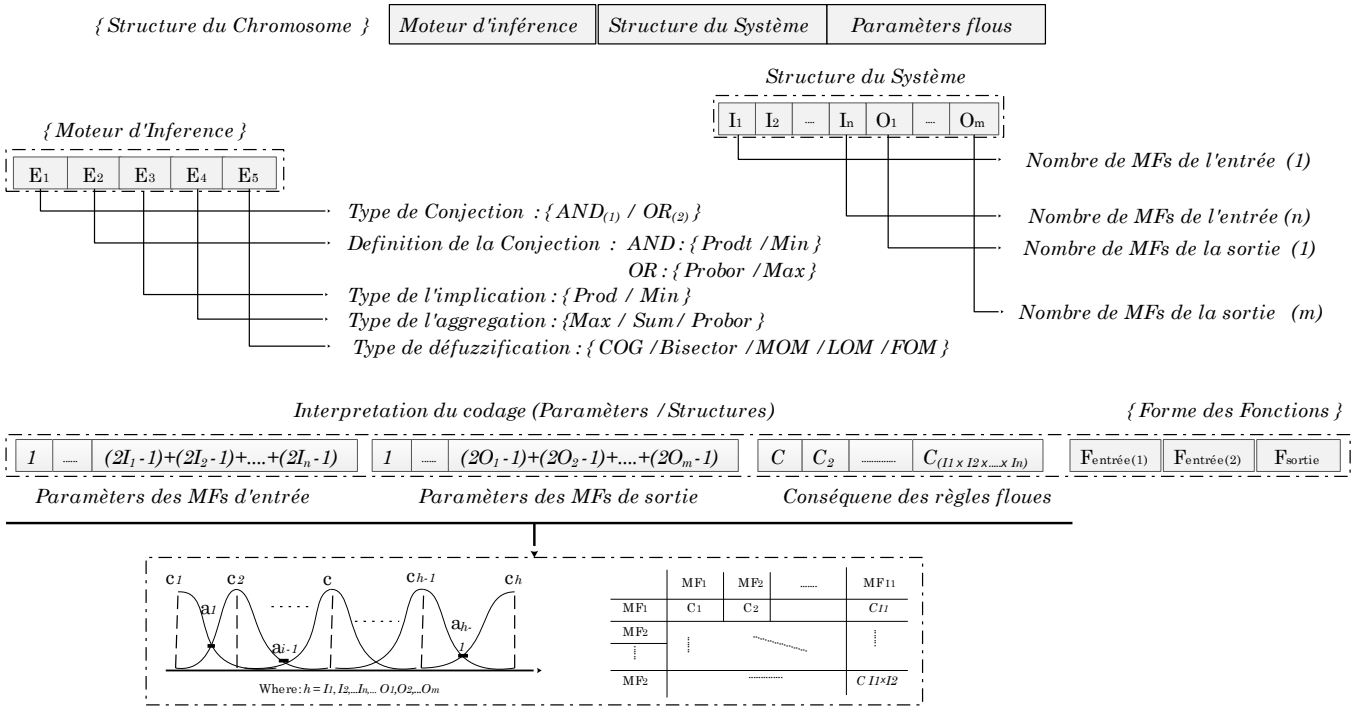


Figure IV. 17 : Codage et interprétation des differents composantes du chromosome

Le chromosme est constitué d'un codage mixte des paramètres. Des chaines rèelles pour les fonctions d'apartenance et les gains des entrèes/sorties ; des chaines entières pour les formes des fonctions, les opérateurs flous et le nombre de fonctions par partition.

En plus de ce codage, nous avons également défini deux fonctions objectif permetant d'exprimer notre volonté d'orienter la conception vers le compromis entre la précision du système flou et son interprétabilité. Ainsi, la première fonction est l'erreur quadratique moyenne (MSE) et la seconde est l'expression de l'aspect interprétable du système flou. Ces deux fonctions objectif sont données par l'équation (IV.9)

$$\begin{cases} Obj_1 = \dfrac{1}{Nd}\sum_{k=1}^{Nd}\left(y^{Système} - y^{Flou}\right)^2 \\ Obj_2 = \dfrac{N_R^2 \times P_{Forme}}{\sum_{i=1}^{Nd}\sum_{j=1}^{N_R}\mu_{Rj}(i)} \end{cases} \quad \text{(IV.9)}$$

Pour le second objectif, N_R : correspond au nombre de règles que contient le système flou défini par le maillage utilisé, P_{Forme} : représente le nombre de paramètres que requiert la défintition de chaque règle selon la forme des fonctions d'appartenance selectionnée pour les entrèes/sortie. $\mu_{Rj}(i)$ quant à lui, représente le degré d'activation de la règle (j) pour les entrées (i). Le numérateur de cette expression vise principalement à obtenir une structure minimale du système flou tandis que le denominateur oriente l'algorithme vers des règles floues très signaficatives et de ne pas seulement se contenter de réduire leur nombre.

Durant cette application, l'algorithme d'optimisation multi-objectif est executé pour une population de 60 individus. La population est initialement générée de façon completement aléatoire où le maillage varie entre 2 et 9 fonctions d'appartenance pour chaque partition floue. Les résulats de simulation sont illustrés par la figure récapitulative (IV.18),

Parmi toutes les solutions constituant le front de Pareto, nous avons choisi une solution à adopter comme modèle pour ce système. La figure (IV.19) présente les partitions floues correspondantes au système flou codé par cette solution. La structure contient deux fonctions d'appartenance pour l'entrée (*y(k-1*)), trois fonctions pour la seconde entrée (*u(k-4)*) et quatre fonctions pour la sortie (*y(k)*) associées respectivement aux gains $G_{y(k-1)}$ = 0.11868, $G_{u(k-4)}$ = 0.207926 et $G_{y(k)}$=14.99. Cela induit un total de 6 règles floues. Ces dernières sont illustrées dans le tableau (IV.3) et les définitions des opérateurs flous sont données dans le tableau (IV.4)

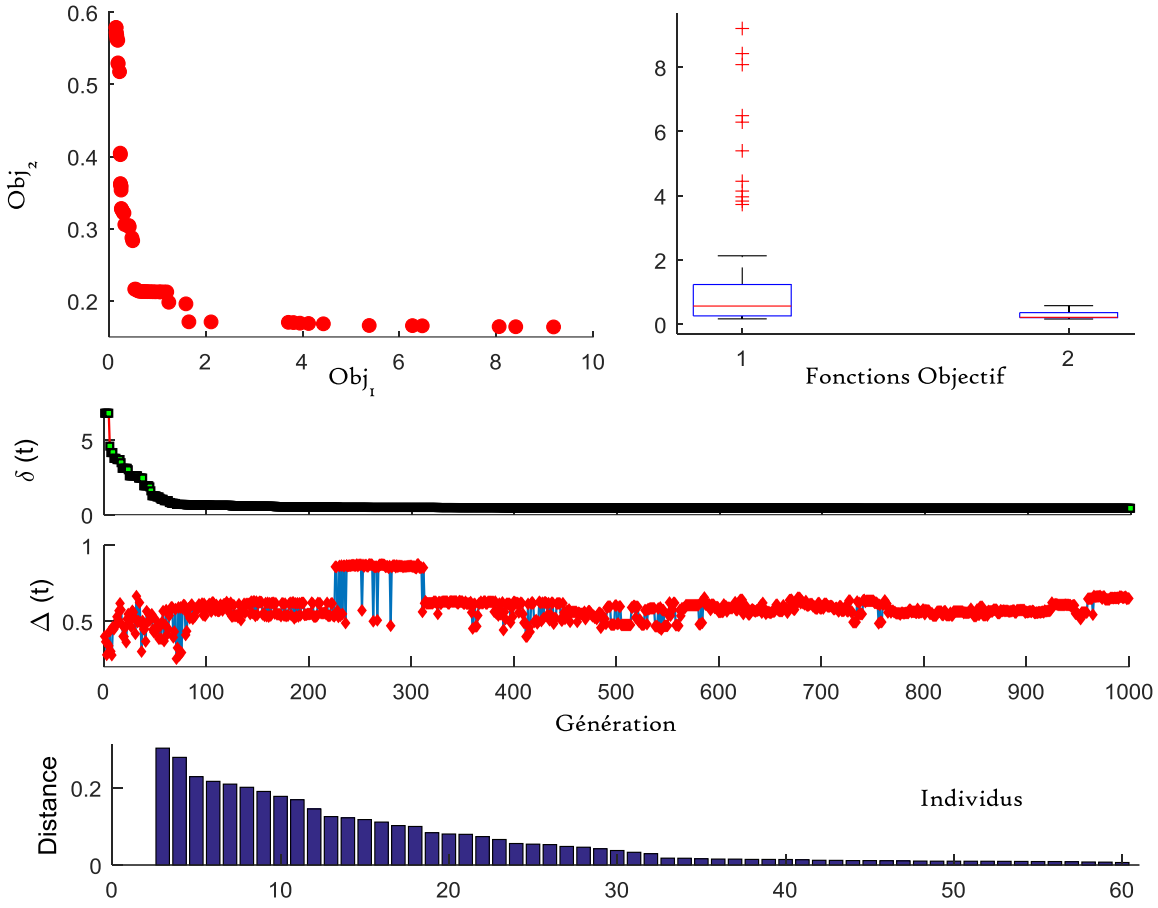


Figure IV. 18 : Evolution du processus d'optimisation avec compromis precision-interprétabilité

Tableau IV. 3 : Base de règles floues

1.	**If** (y(k-1) is N) **Or** (u(k-4) is N) **then** (y(k) is PP)	
2.	**If** (y(k-1) is N) **And** (u(k-4) is Z) **then** (y(k) is NG)	
3.	**If** (y(k-1) is N) **Or** (u(k-4) is P) **then** (y(k) is NG)	
4.	**If** (y(k-1) is P) **Or** (u(k-4) is N) **then** (y(k) is PG)	
5.	**If** (y(k-1) is P) **And** (u(k-4) is Z) **then** (y(k) is NP)	
6.	**If** (y(k-1) is P) **Or** (u(k-4) is P) **then** (y(k) is PP)	

Tableau IV. 4: Moteur d'inférence

SIF Type : Mamdani	
Opér (And) : Minimum	
Opér (OR) : Probabiliste	
Opér (Implication) : Produit	
Opér (Aggrégation) : Probabisliste	
Opér (Défuzzification) : Centre de gravité	

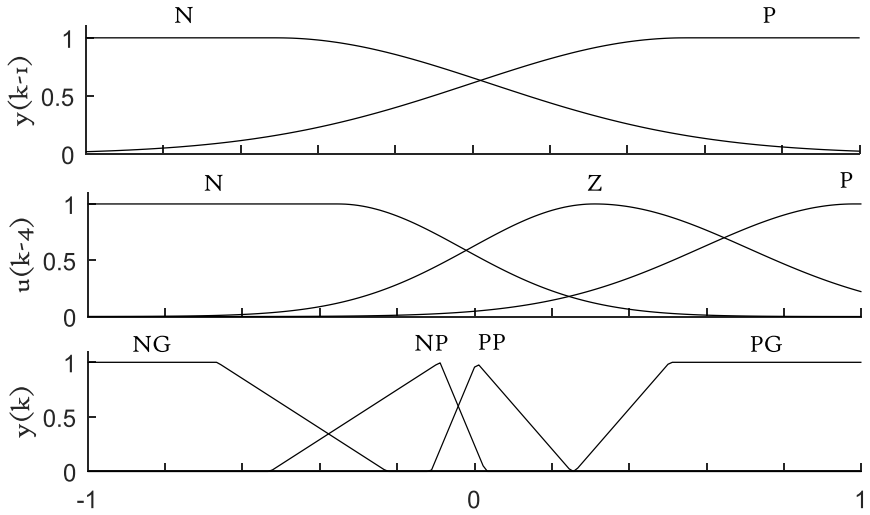


Figure IV. 19 : Partitions floues des entrées/sortie

La figure (IV.20) présente les résultats de simulation correspondant à cette solution. La figure est partagée en quatre quadrants. Les deux premiers en haut corréspondent aux résultats de simulation du système flou lors de la phase d'apprentissage et de test. Les deux derniers en bas donnent les erreurs produites durant chacune de ces deux phases.

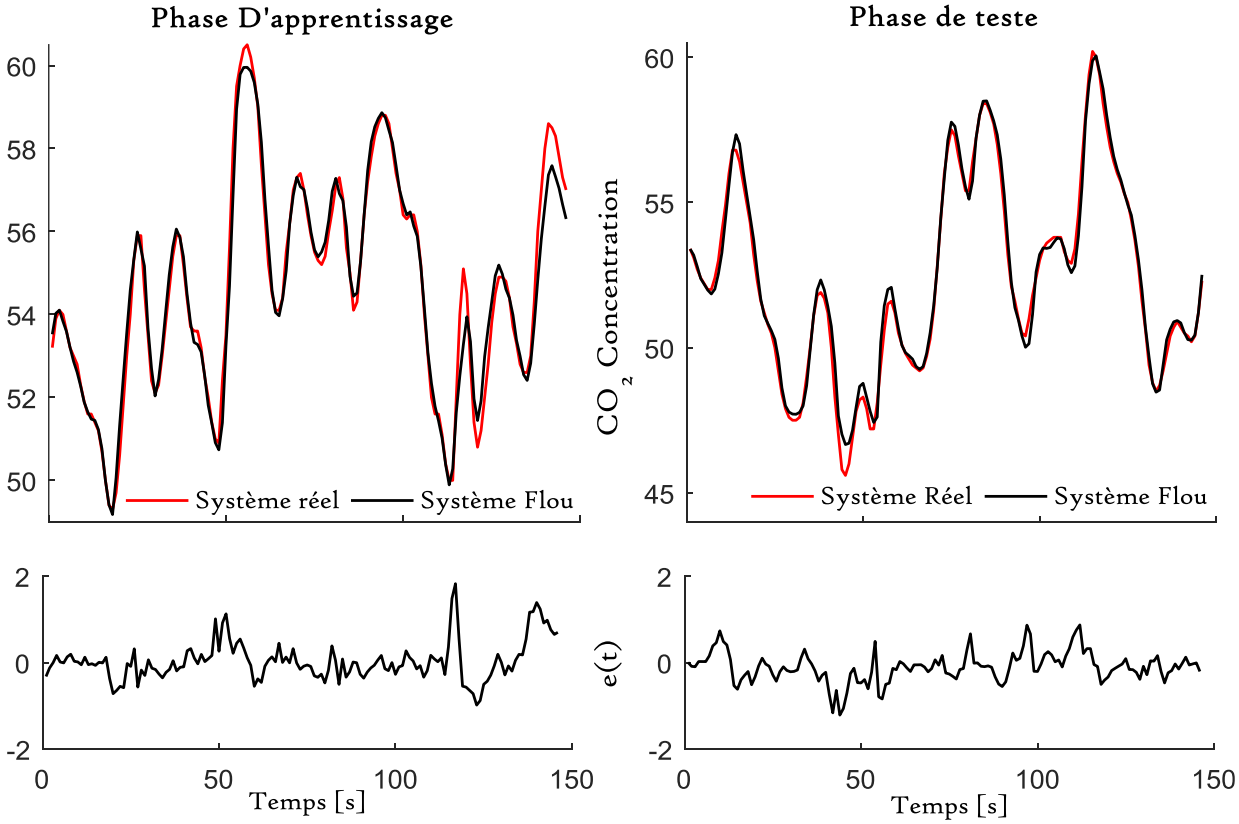


Figure IV. 20 : Performance du système flou

A partir de cette figure (IV.20), on remarque que le système flou a bien appris le lien entre les entrées et la sortie du système réel durant la phase d'apprentissage. Le graphe de la phase de test prouve que le système flou arrive à bien prédire la sortie du système réel, ce qui confirme bien la validité du processus d'identification du four à gaz.

Maintenant, en comparant ce résultat avec ceux du tableau (III.2, *pp72*) on peut facilement voir l'intérêt de cette approche. En effet sur toute la série de données temporelles, ce système flou arrive à aproximer la dynamique du système réel (MSE = 0.1696) avec seulement 6 règles floues, ce qui très significative par rapport aux modèles linguistiques du même tableau. Ce résultat est bien en accord avec l'hypothèse établie au départ de cette application à savoir : l'ajustement du moteur d'inférence améliore bien l'efficacité des règles floues en augmentant leurs coopérations. Par cette méthode de codage des différents paramètres du système flou, on a pu obtenir une meilleur précision pour le modèle flou final

même si ce dernier ne comporte que quelques règles floues. D'autre part, l'approche permet d'agir sur les quatre quadrants définissant la notion d'interprétabilité.

IV.7 Conclusion

Durant ce chapitre, nous avons étudié le problème de conception des systèmes flous par les algorithmes multi-objectif. Nous avons d'abord présenté l'état de l'art sur ce sujet avant de détailler le principe de chaque classe de conception. Afin d'assurer une bonne assimimlation de ce problème de conception dit : multi-objectif, nous avons présenté quelques applications illustratives. Pour les deux premières applications, les approches illustrées sont tirées de la littérature, par contre pour les deux dernières applications, nous avons présenté des approches developpées durant cette thèse.

Les différents résultats de simulation, confirment le potentiel des approches proposées. En effet, les algorithmes d'apprentissage dévellopés ont réussi à construire des systèmes flous intéréssants. Ce succès met en valeur les mecanismes proposés et les différentes définitions formulées à chaque application pour la mesure des aspects de l'interprétabilité (sémantique et complexcité).

Conclusion générale

Le manuscrit de cette thèse adresse l'un des problèmes les plus récurrents dans le domaine de l'apprentissage automatique et de l'ingénierie. Il s'agit de la conception des systèmes flous linguistiques du type Mamdani par les méta-heuristiques. Cette problématique a été traitée de façon graduelle et progressive pour une meilleure assimilation, en abordant à chaque fois une partie de ce problème.

Dans un premier temps, le système flou a été présenté dans sa forme basique (*type-1*) avec ses différentes restrictions et conditions de conception, avant d'introduire l'outil souvent utilisé pour le concevoir automatiquement à savoir : les méta-heuristiques d'optimisation. L'importance de ces algorithmes pour le problème de conception nous a conduit à l'étude de l'un des algorithmes les plus sollicité dans ce contexte durant les dernières années, à savoir l'algorithme PSO. Ceci, nous a permet de proposer une variante de cet algorithme qui améliore ses performances de recherche. La proposition repose essentiellement sur un partage plus efficace des informations au sein d'une population multi-essaims conduisant à un meilleur pilotage des deux mécanismes d'exploration et d'exploitation dans l'espace de recherche. Les résultats obtenus par cette proposition sont très prometteurs comparé à ceux des autres propositions de la même famille.

Une fois que le système flou et l'outil de prédilection de sa conception ont été présentés, le manuscrit s'est étalé sur la conception automatique des systèmes flous. Ce dernier point a été traité en deux phases. Dans la première, les approches basées sur les algorithmes mono-objectif ont été présentées avec leurs différentes variantes populaires dans la littérature avant d'introduire notre contribution dans ce sens. Cette dernière vise principalement à améliorer la capacité d'apprentissage du système flou linguistique de type Mamdani par l'algorithme PSO.

Cette proposition apporte un mécanisme supplémentaire à l'algorithme PSO lui permettant d'investiguer d'avantage l'espace de recherche en lui offrant la possibilité de s'extraire facilement des optimums locaux lors de l'optimisation des règles floues. Le but étant toujours de trouver la meilleure base de règles floues avec les paramètres des fonctions d'appartenance et les gains d'entrées/sortie. À la place d'utiliser uniquement le mécanisme intrinsèque du PSO, cette approche adopte en plus un seuillage dynamique et auto-ajustable pour le changement des règles floues. Les résultats obtenus ont montrés un potentiel très honorable, tant pour les applications théoriques que pour les applications d'ingénierie en temps réel.

Pour la deuxième phase de traitement de cette problématique de conception, le manuscrit introduit les approches basées sur les algorithmes multi-objectif, en illustrant leurs différentes familles et les objectifs que regroupe chacune d'entre-elles. Bien que cette classe de conception soit probablement la plus prometteuse, elle reste néanmoins la plus difficile à mettre en œuvre. La présence d'objectifs complètement contradictoires n'est pas le seul problème rencontré. La difficulté à formuler certains objectifs si essentiels aux systèmes flous linguistiques de type Mamdani est actuellement l'autre entrave à cette démarche de conception. Dans ce même sens, nous avons évoqué la notion d'intérprétabilité qui reste encore un problème non résolut dans la littérature. Toutefois, nous avons présenté différentes démarches de conception qui permet de cerner au mieux les concepts de la conception multi-objectif. Certaines de ces démarches sont nos propres propositions développées durant cette thèse qui s'avèrent prometteuse.

En conclusion, et loin de toutes les contraintes avancées dans cette thèse quant à la position de la littérature vis-à-vis de ce problème tel que : *le nombre d'objectifs, la formulation de certaines notions*, le problème de conception des systèmes flous est intrinsèquement un problème difficile même du simple point de vue algorithmique ou informatique. En effet, il fait partie d'une classe de problème particulière. Pour cette classe, il est très facile de vérifier la solution une fois obtenue, mais pas facile justement à la trouvée. Et rien que cette caractéristique suffit à souligné la difficulté de procéder à la conception automatique de tels systèmes.

Cependant, nous pouvons dire que les algorithmes d'optimisation multi-objectif sont effet un outil puissant pour la résolution de ce problème de conception, vu que les objectifs optimisés sont souvent incommensurables. Toutefois, ces algorithmes nécessitent une bonne formulation des objectifs à optimiser selon le domaine d'application visé et un choix judicieux

pour la méthode à utiliser pour représenter les différents paramètres du système flou, car cela conditionne fortement la performance de ces algorithmes

Perspectives

Plusieurs questions ne sont pas discutées dans ce manuscrit, qui peuvent faire une très bonne suite à ce travail. Il serait par exemple très intéressant d'étudier la même problématique pour les systèmes flous d'ordre supérieur et de réfléchir sur la possibilités de réduire le temps de calcul pendant leurs conception.

Il serait également intéressant de se focaliser sur l'étude et l'amélioration des algorithmes de conception particulièrement ceux multi-objectif. En réduisant leur sensibilité à la dimension du problème (*nombre d'objectif supérieur à 2*).

Par ailleurs, l'étude des moyens d'intégration des systèmes flous dans les champs scientifiques récents comme l'exploration de très grande base de données (Big Data)…s'avère aussi une problématique intéressante.

Références

[1] Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). Deep blue. *Artificial intelligence*, 134(1-2), 57-83. *https://doi.org/10.1016/S0004-3702(01)00129-1.*

[2] Wang, F. Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., & Yang, L. (2016). Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2), 113-120. *10.1109/JAS.2016.7471613.*

[3] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115-133. *https://doi.org/10.1007/BF02478259.*

[4] Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3), 338-353. *https://doi.org/10.1016/S0019-9958(65)90241-X.*

[5] Mamdani, E. H. (1974, December). Application of fuzzy algorithms for control of simple dynamic plant. *In Proceedings of the institution of electrical engineers* (Vol. 121, No. 12, pp. 1585-1588). IET. *https://doi.org/10.1049/piee.1974.0328.*

[6] Sugeno, M. (1999). On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *IEEE Transactions on Fuzzy systems*, 7(2), 201-224. *https://doi.org/10.1109/91.755401.*

[7] Jang, J. S. R. (1991, July). Fuzzy modeling using generalized neural networks and kalman filter algorithm. *In AAAI* (Vol. 91, pp. 762-767).

[8] Jang, J. S. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3), 665-685. *https://doi.org/10.1109/21.256541.*

[9] Herrera, F. (2005). Genetic fuzzy systems: Status, critical considerations and future directions. *International Journal of Computational Intelligence Research*, 1(1), 59-67. *https://doi.org/10.5019/j.ijcir.2005.23.*

[10] Castillo, O., & Melin, P. (2012). Soft computing for control of non-linear dynamical systems (Vol. 63). *Physica*.

[11] De Cock, M., & Kerre, E. E. (2004). Fuzzy modifiers based on fuzzy relations. *Information Sciences*, 160(1-4), 173-199. *https://doi.org/10.1016/j.ins.2003.09.002.*

[12] Grisales Palacio, V. H. (2007). Modélisation et commande floues de type Takagi-Sugeno appliquées à un bioprocédé de traitement des eaux usées (*Doctoral dissertation, Uniandes*).

[13] Wu, Y., Zhang, B., Lu, J., & Du, K. L. (2011). Fuzzy logic and neuro-fuzzy systems: A systematic introduction. *International Journal of Artificial Intelligence and Expert Systems*, 2(2), 47-80.

[14] Van Leekwijck, W., & Kerre, E. E. (1999). Defuzzification: criteria and classification. *Fuzzy sets and systems*, 108(2), 159-178. *https://doi.org/10.1016/S0165-0114(97)00337-0.*

[15] Naaz, S., Alam, A., & Biswas, R. (2011). Effect of different defuzzification methods in a fuzzy based load balancing application. *International Journal of Computer Science Issues (IJCSI)*, 8(5), 261.

[16] Van Leekwijck, W., & Kerre, E. E. (1999). Defuzzification: criteria and classification. *Fuzzy sets and systems*, 108(2), 159-178. *https://doi.org/10.1016/S0165-0114(97)00337-0.*

[17] Yadav, S. K. (2015). DC motor position control using fuzzy proportional-derivative controllers with different defuzzification methods. *IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)* e-ISSN, 10(1), 2278-1676. *https://doi.org/10.9790/1676-10133747.*

[18] Liu, X. (2007). Parameterized defuzzification with maximum entropy weighting function—another view of the weighting function expectation method. *Mathematical and Computer Modelling*, 45(1-2), 177-188. *https://doi.org/10.1016/j.mcm.2006.04.014.*

[19] Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, 7(1), 1-13. *https://doi.org/10.1016/S0020-7373(75)80002-2.*

[20] Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, (1), 116-132. *https://doi.org/10.1109/ TSMC. 1985. 6313399.*

[21] Tsukamoto, Y. (1975). An approach to fuzzy reasoning method., M. M. Gupta (Ed), *in Advances in Fuzzy Set Theory and Applications*. pp. 407-428.

[22] Carlsson, C., & Fullér, R. (2001). Optimization under fuzzy if–then rules. *Fuzzy sets and systems*, 119(1), 111-120. *https://doi.org/10.1016/S0165-0114(98)00465-5.*

[23] Abraham, A. (2001, June). Neuro fuzzy systems: State-of-the-art modeling techniques. In *International Work-Conference on Artificial Neural Networks* (pp. 269-276). Springer, Berlin, Heidelberg. *https://doi.org/10.1007/3-540-45720-8_30.*

[24] Nauck, D. D., & Nürnberger, A. (2013). Neuro-fuzzy systems: A short historical review. *In Computational intelligence in intelligent data analysis* (pp. 91-109). Springer, Berlin, Heidelberg. *https://doi.org/10.1007/978-3-642-32378-2_7.*
[25] Lin, C. T., & Lee, C. S. G. (1991). Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on computers*, 40(12), 1320-1336. *https://doi.org/10.1109/12.106218*.
[26] Tung, W. L., & Quek, C. (2004). Falcon: neural fuzzy control and decision systems using FKP and PFKP clustering algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B (Cybernetics), 34(1), 686-695. *https://doi.org/10.1109/TSMCB.2003.809227.*
[27] Berenji, H. R., & Khedkar, P. (1992). Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on neural networks*, 3(5), 724-740. *https://doi.org/10.1109/72.159061.*
[28] Berenji, H. R. (1996, September). Fuzzy Q-learning for generalization of reinforcement learning. *In Proceedings of IEEE 5th International Fuzzy Systems* (Vol. 3, pp. 2208-2214). IEEE. *https://doi.org/10.1109/FUZZY.1996.553542.*
[29] Nauck, D., & Kruse, R. (1999). Neuro-fuzzy systems for function approximation. *Fuzzy sets and systems*, 101(2), 261-271. *https://doi.org/10.1016/S0165-0114(98)00169-9.*
[30] Nürnberger, A., Nauck, D., & Kruse, R. (1999). Neuro-fuzzy control based on the NEFCON-model: recent developments. *Soft Computing*, 2(4), 168-182. *https://doi.org/10.1007/s005000050050.*
[31] Nauck, D. D. (2003). Fuzzy data analysis with NEFCLASS. *International journal of approximate reasoning*, 32(2-3), 103-130. *https://doi.org/10.1016/S0888-613X(02)00079-8.*
[32] Yousefi, J., & Hamilton-Wright, A. (2019). Input Value Skewness and Class Label Confusion in the NEFCLASS Neuro-Fuzzy System. *In Computational Intelligence* (pp. 179-196). Springer, Cham. *https://doi.org/10.1007/978-3-319-99283-9_9.*
[33] Jang, J. S. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3), 665-685. *https://doi.org/10.1109/21.256541.*
[34] Jang, R. (1992). Neuro-Fuzzy Modelling: Architectures. Analysis and Applications. *PhD Thesis, University of California, Berkley*.
[35] Simanihuruk, T., Hartono, H., Abdullah, D., Erliana, C. I., Napitupulu, D., Ongko, E., ... & Sukiman, A. S. A. (2018). Hesitant Fuzzy Linguistic Term Sets with Fuzzy Grid Partition in Determining the Best Lecturer. *International Journal of Engineering & Technology*, 7(2.3), 59-62. *https://doi.org/10.14419/ijet.v7i2.3.12322.*
[36] Malekizadeh, M., Karami, H., Karimi, M., Moshari, A., & Sanjari, M. J. (2020). Short-term load forecast using ensemble neuro-fuzzy model. *Energy*, 196, 117127. *https://doi.org/10.1016/j.energy.2020.117127.*
[37] Wu, Y., Zhang, B., Lu, J., & Du, K. L. (2011). Fuzzy logic and neuro-fuzzy systems: A systematic introduction. *International Journal of Artificial Intelligence and Expert Systems*, 2(2), 47-80.
[38] Piegat, A. (2013). Fuzzy modeling and control (Vol. 69). *Physica*.
[39] Ali, O. A. M., Ali, A. Y., & Sumait, B. S. (2015). Comparison between the effects of different types of membership functions on fuzzy logic controller performance. *International Journal of Emerging Engineering Research and Technology*, *3(3)*.,76-83.
[40] Touil, S., & Attous, D. B. (2013). Effect of different membership functions on fuzzy power system stabilizer for synchronous machine connected to infinite bus. *International Journal of Computer Applications*, 71(7). *https://doi.org/10.1007/s13198-015-0344-8.*
[41] Kim, S., Lee, M., & Lee, J. (2017). A study of fuzzy membership functions for dependence decision-making in security robot system. *Neural Computing and Applications*, 28(1), 155-164. *https://doi.org/10.1007/s00521-015-2044-3.*
[42] MacVicar-Whelan, P. J. (1976). Fuzzy sets for man-machine interaction. *International Journal of Man-Machine Studies*, 8(6), 687-697. *https://doi.org/10.1016/S0020-7373(76)80030-2.*
[43] Guillaume, S. (2001). Designing fuzzy inference systems from data: An interpretability-oriented review. *IEEE Transactions on fuzzy systems*, 9(3), 426-443. *https://doi.org/10.1109/91.928739.*
[44] Yen, J., & Wang, L. (1999). Simplifying fuzzy rule-based models using orthogonal transformation methods. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B (Cybernetics), 29(1), 13-24. *https://doi.org/10.1109/3477.740162.*
[45] Dubois, D., Prade, H., & Ughetto, L. (1997). Checking the coherence and redundancy of fuzzy knowledge bases. *IEEE Transactions on Fuzzy Systems*, 5(3), 398-417. *https://doi.org/10.1109/91.618276.*
[46] Collette. Y, et Siarry. P,. (2002). Optimisation Multiobjectif, *Edition EYROLLES, Paris. France*.
[47] Brownlee, J. (2011). Clever algorithms: nature-inspired programming recipes. *Jason Brownlee*.
[48] Holland, J. (1975). Adaptation In Natural and Artificial Systems. *University of Michigan Press, Ann Arbor*.
[49] Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. *In Simulated annealing: Theory and applications* (pp. 7-15). Springer, Dordrecht. *https://doi.org/10.1007/978-94-015-7744-1_2.*
[50] Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information sciences*, 179(13), 2232-2248. *https://doi.org/10.1016/j.ins.2009.03.004.*

[51] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4), 28-39. *https://doi.org/10.1109/MCI.2006.329691.*
[52] Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. *In International symposium on stochastic algorithms* (pp. 169-178). Springer, Berlin, Heidelberg. *https://doi.org/10.1007/978-3-642-04944-6_14.*
[53] Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, 27(4), 1053-1073. *https://doi.org/10.1007/s00521-015-1920-1.*
[54] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. *In Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). IEEE. *10.1109/ICNN.1995.488968.*
[55] Goldberg, D.E,. Algorithmes génétiques : Exploration, optimisation et apprentissage automatique. *Addison-Wesley. France*, 1994.
[56] Man, K. F., Tang, K. S., & Kwong, S. (1996). Genetic algorithms: concepts and applications [in engineering design]. *IEEE transactions on Industrial Electronics*, 43(5), 519-534. *https://doi.org/10.1109/41.538609.*
[57] Wu, C. J., & Lin, G. Y. (1999, October). Design of fuzzy logic controllers using genetic algorithms. *In IEEE SMC'99 Conference Proceedings*. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028) (Vol. 6, pp. 104-109). IEEE. *https://doi.org/10.1109/ICSMC.1999.816466.*
[58] Deep, K., & Thakur, M. (2007). A new crossover operator for real coded genetic algorithms. *Applied mathematics and computation*, 188(1), 895-911. *https://doi.org/10.1016/j.amc.2006.10.047.*
[59] Michalewicz, Z. (2013). Genetic algorithms+ data structures= evolution programs. *Springer Science & Business Media*.
[60] Deep, K., & Thakur, M. (2007). A new mutation operator for real coded genetic algorithms. *Applied mathematics and Computation*, 193(1), 211-230. *https://doi.org/10.1016/j.amc.2007.03.046.*
[61] Medina, A. J. R., Pulido, G. T., & Ramírez-Torres, J. G. (2009, October). A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers. *In IJCCI* (pp. 152-159).
[62] Lin, L., & Gen, M. (2009). Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. *Soft Computing*, 13(2), 157-168.
[63] Chatterjee, A., & Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & operations research*, 33(3), 859-871. *https://doi.org/10.1016/j.cor.2004.08.012.*
[64] Eberhart, R. C., & Shi, Y. (2001, May). Tracking and optimizing dynamic systems with particle swarms. *In Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)* (Vol. 1, pp. 94-100). IEEE. *https://doi.org/10.1109/CEC.2001.934376.*
[65] Mancer, N., Mahdad, B., Srairi, K., Hamed, M., & Hadji, B. (2015). Optimal Coordination of Directional Overcurrent Relays Using PSO-TVAC. *Energy Procedia*, 74, 1239-1247. *https://doi.org/10.1016/j.egypro.2015.07.768.*
[66] Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on evolutionary computation*, 8(3), 240-255. *https://doi.org/10.1109/TEVC.2004.826071.*
[67] Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1), 58-73. *https://doi.org/10.1109/4235.985692.*
[68] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In 1998 IEEE international conference on evolutionary computation proceedings. *IEEE world congress on computational intelligence (Cat. No. 98TH8360) (pp. 69-73). IEEE*. *https://doi.org/10.1109/ICEC.1998.699146.*
[69] Cavuslu, M. A., Karakuzu, C., & Karakaya, F. (2012). Neural identification of dynamic systems on FPGA with improved PSO learning. *Applied Soft Computing*, 12(9), 2707-2718. *https://doi.org/10.1016/j.asoc.2012.03.022.*
[70] Pant, M., Thangaraj, R., & Abraham, A. (2007). A new PSO algorithm with crossover operator for global optimization problems. *In Innovations in Hybrid Intelligent Systems* (pp. 215-222). Springer, Berlin, Heidelberg. *https://doi.org/10.1007/978-3-540-74972-1_29.*
[71] Liang, J. J. and Suganthan, P. N. 2005. Dynamic multiswarm particle swarm optimizer. *Proceedings 2005 IEESwarm IntelligenceSymposium, 2005. SIS 2005., Pasadena, CA, USA*, 205, pp 124-129. *https://doi.org/10.1109/SIS.2005.1501611.*
[72] Clerc, M. (2012). Standard particle swarm optimisation - from 2006 to 2011 (2012-09-23 version). Particle Swarm Central [Online]. Available: http://www.particleswarm.info.
[73] Gong, Y. J., & Zhang, J. (2013, July). Small-world particle swarm optimization with topology adaptation. *In Proceedings of the 15th annual conference on Genetic and evolutionary computation* (pp. 25-32). *https://doi.org/10.1145/2463372.2463381.*

[74] Matsushita, H., & Nishio, Y. (2009, October). Network-structured particle swarm optimizer with small-world topology. *In Proc. of Int. Symposium on Nonlinear Theory and its Applications* (pp. 372-375). *https://doi.org/10.34385/proc.43.B2L-B3.*
[75] Liu, Y., Zhao, Q., Shao, Z., Shang, Z., & Sui, C. (2009, September). Particle swarm optimizer based on dynamic neighborhood topology. *In International Conference on Intelligent Computing* (pp. 794-803). Springer, Berlin, Heidelberg. *https://doi.org/10.1007/978-3-642-04020-7_85.*
[76] Xiao, R., Chen, W., & Chen, T. (2012). Modeling of ant colony's labor division for the multi-project scheduling problem and its solution by PSO. *Journal of Computational and Theoretical Nanoscience*, 9(2), 223-232. *https://doi.org/10.1166/jctn.2012.2016.*
[77] Zhu, Z., Zhou, J., Ji, Z., & Shi, Y. H. (2011). DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm. *IEEE Transactions on Evolutionary Computation*, 15(5), 643-658. *https://doi.org/10.1109/TEVC.2011.2160399.*
[78] Zhang, C., Ning, J., Lu, S., Ouyang, D., & Ding, T. (2009). A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. *Operations Research Letters*, 37(2), 117-122. *10.1016/j.orl.2008.12.008.*
[79] Fan, S. K. S., Liang, Y. C., & Zahara, E. (2004). Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions. *Engineering optimization*, 36(4), 401-418. *https://doi.org/10.1080/03052150410001685 21.*
[80] Garg, H. (2016). A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, 274, 292-305. *10.1016/j.amc.2015.11.001.*
[81] Jie, J., Zeng, J., & Han, C. (2006, August). Adaptive particle swarm optimization with feedback control of diversity. *In International Conference on Intelligent Computing* (pp. 81-92). Springer, Berlin, Heidelberg. *10.1007/11816102_9.*
[82] Han, F., & Liu, Q. (2014). A diversity-guided hybrid particle swarm optimization based on gradient search. *Neurocomputing*, 137, 234-240. *https://doi.org/10.1016/j.neucom.2013.03.074.*
[83] Ding, J., Liu, J., Chowdhury, K. R., Zhang, W., Hu, Q., & Lei, J. (2014). A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization. *Neurocomputing*, 137, 261-267. *https://doi.org/10.1016/j.neucom.2013.03.075.*
[84] Douglas, A. E. (1994). Symbiotic Interactions: Oxford Science Pub-lications. *Oxford University Press, Oxford.*
[85] Niu, B., Zhu, Y., He, X., & Wu, H. (2007). MCPSO: A multi-swarm cooperative particle swarm optimizer. *Applied Mathematics and computation*, 185(2), 1050-1062. *https://doi.org/10.1016/j.amc.2006.07.026.*
[86] Zhao, X., Liu, Z., & Yang, X. (2014). A multi-swarm cooperative multistage perturbation guiding particle swarm optimizer. *Applied Soft Computing*, 22, 77-93. *https://doi.org/10.1016/j.asoc.2014.04.042.*
[87] Jie, J., Zhang, J., Zheng, H., & Hou, B. (2016). Formalized model and analysis of mixed swarm based cooperative particle swarm optimization. *Neurocomputing*, 174, 542-552. *https://doi.org/10.1016/j.neucom.2015.08.065.*
[88] Kacimi, M. A., Guenounou, O., & Haddoufi, L. (2020, June). A Multi-Level Environment for the Improvement of the Multi-Swarm Cooperative PSO Algorithm. *In 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)* (pp. 1-6). IEEE. *https://doi.org/10.1109/ICECCE49384.2020.9179309.*
[89] Shi, Y., & Eberhart, R. C. (1999, July). Empirical study of particle swarm optimization. *In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (Vol. 3, pp. 1945-1950). IEEE. *10.1109/CEC.1999.785511.*
[90] Kennedy, J., & Mendes, R. (2002, May). Population structure and particle swarm performance. *In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1671-1676). IEEE. *https://doi.org/10.1109/CEC.2002.1004493.*
[91] Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE transactions on evolutionary computation*, 8(3), 204-210. *https://doi.org/10.1109/TEVC.2004.826074.*
[92] Qu, B. Y., Suganthan, P. N., & Das, S. (2012). A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, 17(3), 387-402. *https://doi.org/10.1109/TEVC.2012.2203138.*
[93] Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, 10(3), 281-295. *https://doi.org/10.1109/TEVC.2005.857610.*
[94] Zhao, S. Z., Liang, J. J., Suganthan, P. N., & Tasgetiren, M. F. (2008, June). Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. *In 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)* (pp. 3845-3852). IEEE. *https://doi.org/10.1109/CEC.2008.4631320.*

[95] Cheng, R., & Jin, Y. (2015). A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, 291, 43-60. *https://doi.org/10.1016/j.ins.2014.08.039.*
[96] Tang, K., Yáo, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M., & Yang, Z. (2007). Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. *Nature inspired computation and applications laboratory, USTC*, China, 24, 1-18.
[97] Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. (Vol. 16). *John Wiley & Sons*.
[98] O. GUENOUNOU. Méthodologie de conception de contrôleur intélligents par l'approche génétique-application à un bioprocédé. *thèse de doctorat, univiersité de Toulouse III- Paul Sabatier, france*, 2009.
[99] Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *In: Proceedings of the 1st International Conference on Genetic*. pp. 93–100.
[100] Fourman, M. P. (1985, July). Compaction of symbolic layout using genetic algorithms. *In Proceedings of the 1st international conference on genetic algorithms* (pp. 141-153).
[101] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2), 182-197. *https://doi.org/10.1109/4235.996017.*
[102] Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, 103. *https://doi.org/10.3929/ethz-a-004284029.*
[103] Roy, B. (1968). Classement et choix en présence de points de vue multiples. *Revue française d'informatique et de recherche opérationnelle*, 2(8), 57-75.
[104] Brans, J. P., Vincke, P., & Mareschal, B. (1986). How to select and how to rank projects: The ROMETHEE method. *European journal of operational research*, 24(2), 228-238. *https://doi.org/10.1016/0377-2217(86)90044-5.*
[105] Saaty, T. L. (1990). Multicretaria decision making series—the analytic hierarchy process: planning, priority setting, ressource allocation. *University of Pittsburgh*.
[106] Moussaoui, F., Cherrared, M., Kacimi, M. A., & Belarbi, R. (2018). A genetic algorithm to optimize consistency ratio in AHP method for energy performance assessment of residential buildings—Application of top-down and bottom-up approaches in Algerian case study. *Sustainable Cities and Society*, 42, 622-636. *https://doi.org/10.1016/j.scs.2017.08.008.*
[107] Ojha, V., Abraham, A., & Snášel, V. (2019). Heuristic design of fuzzy inference systems: A review of three decades of research. *Engineering Applications of Artificial Intelligence*, 85, 845-864. *https://doi.org/10.1016/j.engappai.2019.08.010.*
[108] Cord, O. (2001). Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases (Vol. 19). *World Scientific*.
[109] Herrera, F. (2008). Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1), 27-46. *https://doi.org/10.1007/s12065-007-0001-5.*
[110] Karr, C. (1991). Genetic algorithms for fuzzy controllers. *Ai Expert*, 6(2), 26-33.
[111] Casillas, J., Cordón, O., Del Jesus, M. J., & Herrera, F. (2005). Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems*, 13(1), 13-29. *https://doi.org/10.1109/TFUZZ.2004.839670.*
[112] Alcalá, R., Alcalá-Fdez, J., & Herrera, F. (2007). A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. *IEEE Transactions on Fuzzy Systems*, 15(4), 616-635. *https://doi.org/10.1109/TFUZZ.2006.889880.*
[113] Alcalá-Fdez, J., Herrera, F., Márquez, F., & Peregrín, A. (2007). Increasing fuzzy rules cooperation based on evolutionary adaptive inference systems. *International Journal of Intelligent Systems*, 22(9), 1035-1064. *https://doi.org/10.1002/int.20237.*
[114] Crockett, K. A., Bandar, Z., Fowdar, J., & O'Shea, J. (2006). Genetic tuning of fuzzy inference within fuzzy classifier systems. *Expert Systems*, 23(2), 63-82. *https://doi.org/10.1111/j.1468-0394.2006.00325.x.*
[115] Crockett, K., Bandar, Z., & Mclean, D. (2007, July). On the optimization of T-norm parameters within fuzzy decision trees. *In 2007 IEEE International Fuzzy Systems Conference* (pp. 1-6). IEEE *https://doi.org/10.1109/FUZZY.2007.4295348.*
[116] Kim, D., Choi, Y. S., & Lee, S. Y. (2002). An accurate COG defuzzifier design using Lamarckian co-adaptation of learning and evolution. *Fuzzy sets and systems*, 130(2), 207-225. *https://doi.org/10.1016/S0165-0114(01)00167-1.*
[117] Ahmad bt., K. A., Othman, M., Mansor, A. R., & Bakar, M. N. A. (2017). A Fuzzy Learning Algorithm for Harumanis Maturity Classification. *In Proceedings of the International Conference on Computing,*

Mathematics and Statistics (iCMS 2015) (pp. 3-12). Springer, Singapore. https://doi.org/10.1007/978-981-10-2772-7_1.

[118] Kacimi M.A.(2014) , Conception des systems flous de type Mamdani par des algorithms génétiques multi-objectif. *Mémoire de magister. Université de Bejaia, algérie*.

[119] Sergienko, R., & Semenkin, E. (2013, June). Michigan and pittsburgh methods combination for fuzzy classifier design with coevolutionary algorithm. *In 2013 IEEE Congress on Evolutionary Computation* (pp. 3252-3259). IEEE. https://doi.org/10.1109/CEC.2013.6557968.

[120] Mirsaleh, M. R., & Meybodi, M. R. (2018). A Michigan memetic algorithm for solving the vertex coloring problem. *Journal of computational science*, 24, 389-401. https://doi.org/10.1016/j.jocs.2017.10.005.

[121] Michael, A. L., & Takagi, H. (1993). Dynamic control of genetic algorithms using fuzzy logic techniques. *In Proceedings of the Fifth International Conference on Genetic Algorithms* (IVGA'93), Urbana-Chamign, IL,pp.76-83,July 17-21-,1993.

[122] Yubazaki, N., Otani, M., Ashida, T., & Hirota, K. (1995, March). Dynamic fuzzy control method and its application to positioning of induction motor. *In Proceedings of 1995 IEEE International Conference on Fuzzy Systems*. (Vol. 3, pp. 1095-1102). IEEE. https://doi.org/10.1109/FUZZY.1995.409820.

[123] Peng, Q., Asif, A., Noor, M. Y., & Inam, A. (2020). A Simple Tuning Algorithm of Augmented Fuzzy Membership Functions. *IEEE Access*, 8, 35805-35814. https://doi.org/10.1109/ACCESS.2020.2974533.

[124] Harrag, A., & Messalti, S. (2019). IC-based variable step size neuro-fuzzy MPPT Improving PV system performances. *Energy Procedia*, 157, 362-374. https://doi.org/10.1016/j.egypro.2018.11.201.

[125] Jiao, W., Liu, G., & Liu, D. (2008, October). Elite particle swarm optimization with mutation. *In 2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing* (pp. 800-803). IEEE. https://doi.org/10.1109/ASC-ICSC.2008.4675471.

[126] Juang, C. F., & Chang, Y. C. (2011). Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments. *IEEE Transactions on Fuzzy Systems*, 19(2), 379-392. https://doi.org/10.1109/TFUZZ.2011.2104364.

[127] Sa-ngawong, N., & Ngamroo, I. (2015). Intelligent photovoltaic farms for robust frequency stabilization in multi-area interconnected power system based on PSO-based optimal Sugeno fuzzy logic control. *Renewable energy*, 74, 555-567. https://doi.org/10.1016/j.renene.2014.08.057.

[128] Herrera, I. E., Mandow, A., & García-Cerezo, A. (2018, June). Using Particle Swarm Optimization for Fuzzy Antecedent Parameter Identification in Active Suspension Control. *In 2018 26th Mediterranean Conference on Control and Automation (MED)* (pp. 1-9). IEEE. https://doi.org/10.1109/MED.2018.8442790.

[129] Rastegar, S., Araújo, R., & Mendes, J. (2017). Online identification of Takagi–Sugeno fuzzy models based on self-adaptive hierarchical particle swarm optimization algorithm. *Applied Mathematical Modelling*, 45, 606-620. https://doi.org/10.1016/j.apm.2017.01.019.

[130] Kacimi, M. A., Guenounou, O., Brikh, L., Yahiaoui, F., & Hadid, N. (2020). New mixed-coding PSO algorithm for a self-adaptive and automatic learning of Mamdani fuzzy rules. *Engineering Applications of Artificial Intelligence*, 89, 103417. https://doi.org/10.1016/j.engappai.2019.103417.

[131] Box, G.E.P., Jenkins, G.M., 1976. Time Series Analysis: Forecasting and Control, *second ed. Holden-Day, San Francisco, CA.*

[132] Zhao, L., Qian, F., Yang, Y., Zeng, Y., & Su, H. (2010). Automatically extracting T–S fuzzy models using cooperative random learning particle swarm optimization. *Applied soft computing*, 10(3), 938-944. https://doi.org/10.1016/j.asoc.2009.10.012.

[133] Tong, R. M. (1980). The evaluation of fuzzy models derived from experimental data. *Fuzzy sets and systems*, 4(1), 1-12. https://doi.org/10.1016/0165-0114(80)90059-7.

[134] Pedrycz, W. (1984). An identification algorithm in fuzzy relational systems. *Fuzzy sets and systems*, 13(2), 153-167. https://doi.org/10.1016/0165-0114(84)90015-0.

[135] Takagi, T. Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*. (15). 116-132. https://doi.org/10.1109/TSMC.1985.6313399.

[136] Xu, C. W., & Lu, Y. Z. (1987). Fuzzy model identification and self-learning for dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(4), 683-689. https://doi.org/10.1109/TSMC.1987.289361.

[137] Sugeno, M., & Tanaka, K. (1991). Successive identification of a fuzzy model and its applications to prediction of a complex system. *Fuzzy sets and systems*, 42(3), 315-334. https://doi.org/10.1016/0165-0114(91)90110-C.

[138] Sugeno, M., & Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on fuzzy systems*, 1(1), 7-31.

[139] Wang, L., & Langari, R. (1996). Complex systems modeling via fuzzy logic. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1), 100-106. https://doi.org/10.1109/3477.484441.

[140] Kim, E., Park, M., Ji, S., Park, M., 1997. A new approach to fuzzy modeling. *IEEE Trans. Fuzzy Syst*. 5 (3), 328–337. https://doi.org/10.1109/91.618271.

[141] Farag, W.A., Quintana, V.H., Lambert-Torres, G., 1998. A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems. *IEEE Trans. Neural Netw*. 9 (5), 756–767. https://doi.org/10.1109/72.712150.

[142] Kang, S.J., Woo, C.H., Hwang, H.S., Woo, K.B., 2000. Evolutionary design of fuzzy rule base for nonlinear system modeling and control. *IEEE Trans. Fuzzy Syst*. 8 (1), 37–45. https://doi.org/10.1109/91.824766.

[143] Evsukoff, A., Branco, A.C., Galichet, S., 2002. Structure identification and parameter optimization for non-linear fuzzy modeling. *Fuzzy Sets and Systems*. 132 (2), 173–188. https://doi.org/10.1016/S0165-0114(02)00111-2.

[144] Khosla, A., Kumar, S., & Aggarwal, K. K. (2005, December). A framework for identification of fuzzy models through particle swarm optimization algorithm. *In 2005 Annual IEEE India Conference-Indicon* (pp. 388-391). IEEE. https://doi.org/10.1109/INDCON.2005.1590196.

[145] Bagis, A., 2008. Fuzzy rule base design using tabu search algorithm for nonlinear system modeling. *ISA Trans*. 47 (1), 32–44. https://doi.org/10.1016/j.isatra.2007.09.001.

[146] Zhao, L., Qian, F., Yang, Y., Zeng, Y., Su, H., 2010. Automatically extracting T–S fuzzy models using cooperative random learning particle swarm optimization. *Appl. Soft Comput*. 10 (3), 938–944. https://doi.org/10.1016/j.asoc.2009.10.012.

[147] Su, Z.G., Wang, P.H., Shen, J., Zhang, Y.F., Chen, L., 2012. Convenient T–S fuzzy model with enhanced performance using a novel swarm intelligent fuzzy clustering technique. *J. Process Control* 22 (1), 108–124. https://doi.org/10.1016/j.jprocont.2011.10.002.

[148] Wadhawan, S., Goel, G., & Kaushik, S. (2013). Data driven fuzzy modeling for Sugeno and Mamdani type fuzzy model using memetic algorithm. *Int. J. Information Technology and Computer Science*, 5(8), 24-37. https://doi.org/10.5815/IJITCS.2013.08.03.

[149] Habbi, H., Boudouaoui, Y., Karaboga, D., & Ozturk, C. (2015). Self-generated fuzzy systems design using artificial bee colony optimization. *Information Sciences*, 295, 145-159. https://doi.org/10.1016/j.ins.2014.10.008.

[150] Bagis, A., & Konar, M. (2016). Comparison of Sugeno and Mamdani fuzzy models optimized by artificial bee colony algorithm for nonlinear system modelling. *Transactions of the Institute of Measurement and Control*, 38(5), 579-592. https://doi.org/10.1177/0142331215591239.

[151] Konar, M., & Bagis, A. (2016). Performance comparison of particle swarm optimization, differential evolution and artificial bee colony algorithms for fuzzy modelling of nonlinear systems. *Elektronika ir Elektrotechnika*, 22(5), 8-13. https://doi.org/10.5755/j01.eie.22.5.16336.

[152] Turki, M., Sakly, A., 2017. Extracting T–S Fuzzy models using the Cuckoo search algorithm. *Comput. Intell. Neurosci*. UNSP 8942394. https://doi.org/10.1155/2017/8942394.

[153] Hamza, M. F., Yap, H. J., Choudhury, I. A., Isa, A. I., Zimit, A. Y., & Kumbasar, T. (2019). Current development on using Rotary Inverted Pendulum as a benchmark for testing linear and nonlinear control algorithms. *Mechanical Systems and Signal Processing*, 116, 347-369. https://doi.org/10.1016/j.ymssp.2018.06.054.

[154] Ding, Z., & Li, Z. (2014, June). A cascade fuzzy control system for inverted pendulum based on Mamdani-Sugeno type. *In 2014 9th IEEE Conference on Industrial Electronics and Applications* (pp. 792-797). IEEE. https://doi.org/10.1109/ICIEA.2014.6931270.

[155] Hamza, M., Zahid, Q., Tahir, F., & Khalid, Z. (2011, December). Real-time control of an inverted pendulum: A comparative study. *In 2011 Frontiers of Information Technology* (pp. 183-188). IEEE. https://doi.org/10.1109/FIT.2011.41.

[156] Luhao, W., & Zhanshi, S. (2010, December). LQR-Fuzzy control for double inverted pendulum. *In 2010 International Conference on Digital Manufacturing & Automation* (Vol. 1, pp. 900-903). IEEE. *https://doi.org/10.1109/ICDMA.2010.170.*

[157] Khanesar, M. A., Kaynak, O., Yin, S., & Gao, H. (2014). Adaptive indirect fuzzy sliding mode controller for networked control systems subject to time-varying network-induced time delay. *IEEE Transactions on Fuzzy Systems*, 23(1), 205-214. *https://doi.org/10.1109/TFUZZ.2014.2362549.*

[158] Liu, Y. J., Gao, Y., Tong, S., & Li, Y. (2015). Fuzzy approximation-based adaptive backstepping optimal control for a class of nonlinear discrete-time systems with dead-zone. *IEEE Transactions on Fuzzy Systems*, 24(1), 16-28. *https://doi.org/10.1109/TFUZZ.2015.2418000.*

[159] Bhangal, N.S., 2013. Design and performance of LQR and LQR based fuzzy controller for double inverted pendulum system. *J. Image Graph*. 1 (3), 143–146. *https://doi.org/10.12720/joig.1.3.143-146.*

[160] Bharali, J., & Buragohain, M. (2016, July). Design and performance analysis of fuzzy LQR; fuzzy PID and LQR controller for active suspension system using 3 degree of freedom quarter car model. *In 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)* (pp. 1-6). IEEE. *https://doi.org/10.1109/ICPEICES.2016.7853369.*

[161] Fazzolari, M., Alcala, R., Nojima, Y., Ishibuchi, H., & Herrera, F. (2012). A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE Transactions on Fuzzy systems*, 21(1), 45-65. *https://doi.org/10.1109/TFUZZ.2012.2201338.*

[162] Lewis, F. L., Vrabie, D., & Syrmos, V. L. (2012). Optimal control. *John Wiley & Sons*.

[163] Guenounou, O., Dahhou, B., & Chabour, F. (2020, June). Multi-objective optimization of fuzzy MPPT using improved strength Pareto evolutionary algorithm. *In 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)* (pp. 1-6). IEEE. *https://doi.org/10.1109/ICECCE49384.2020.9179236.*

[164] Abdel-Kader, F. M., El-Serafi, K. A., El-Saadawi, A. M., & Abdel-Rhman, A. S. (2021). A Fuzzy Self-Tuning Hybrid Force/Position Controller for Robotic Manipulator.(Dept. E). *MEJ. Mansoura Engineering Journal*, 27(3), 17-26. *https://doi.org/10.21608/BFEMU.2021.142702.*

[165] Yarlagadda, V., Radhika, G., Gnanendar, R., & Rajesh, K. (2021). Robust Fuzzy Logic Controller for DC Motor Stability Enhancement Under Load Disturbances. *In Intelligent Computing in Control and Communication* (pp. 79-90). Springer, Singapore. *https://doi.org/10.1007/978-981-15-8439-8_7.*

[166] Tripathi, S., Shrivastava, A., & Jana, K. C. (2020). Self-Tuning fuzzy controller for sun-tracker system using Gray Wolf Optimization (GWO) technique. *ISA transactions*, 101, 50-59. *https://doi.org/10.1016/j.isatra.2020.01.012.*

[167] Azizi, M., Ghasemi, S. A. M., Ejlali, R. G., & Talatahari, S. (2019). Optimal tuning of fuzzy parameters for structural motion control using multiverse optimizer. *The Structural Design of Tall and Special Buildings*, 28(13), e1652. *https://doi.org/10.1002/tal.1652.*

[168] Guenounou, O., Belmehdi, A., & Dahhou, B. (2009). Multi-objective optimization of TSK fuzzy models. *Expert systems with applications*, 36(4), 7416-7423. *https://doi.org/10.1016/j.eswa.2008.09.044.*

[169] Cococcioni, M., Corsini, G., Lazzerini, B., & Marcelloni, F. (2008). Solving the ocean color inverse problem by using evolutionary multi-objective optimization of neuro-fuzzy systems. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 12(5-6), 339-355. *https://doi.org/10.3233/KES-2008-125-604.*

[170] Fleming, P. J., & Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering: a survey. *Control engineering practice*, 10(11), 1223-1241. *https://doi.org/10.1016/S0967-0661(02)00081-3.*

[171] Ishibuchi, H., Murata, T., & Gen, M. (1998). Performance evaluation of fuzzy rule-based classification systems obtained by multi-objective genetic algorithms. *Computers & industrial engineering*, 35(3-4), 575-578. *https://doi.org/10.1016/S0360-8352(98)00162-4.*

[172] Gacto, M.J., Alcalá, R. & Herrera, F. A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems. *Appl Intell*. 36, 330–347 (2012). *https://doi.org/10.1007/s10489-010-0264-x.*

[173] Guenounou, O., Dahhou, B., & Chabour, F. (2015). TSK fuzzy model with minimal parameters. *Applied Soft Computing*, 30, 748-757. *https://doi.org/10.1016/j.asoc.2015.02.017.*

[174] Marín-Blázquez, J. G., & Shen, Q. (2002). From approximative to descriptive fuzzy classifiers. *IEEE Transactions on Fuzzy Systems*, 10(4), 484-497. *https://doi.org/10.1109/TFUZZ.2002.800687.*

[175] Hoffmann, F., Baesens, B., Mues, C., Van Gestel, T., & Vanthienen, J. (2007). Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms. *European journal of operational research*, 177(1), 540-555. *https://doi.org/10.1016/j.ejor.2005.09.044.*

[176] Chan, G. Y., Lee, C. S., & Heng, S. H. (2014). Defending against XML-related attacks in e-commerce applications with predictive fuzzy associative rules. *Applied Soft Computing*, 24, 142-157. *https://doi.org/10.1016/j.asoc.2014.06.053Get.*
[177] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3), 37-37. *https://doi.org/10.1609/aimag.v17i3.1230.*
[178] Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *AI magazine*, 13(3), 57-57. *https://doi.org/10.1609/aimag.v13i3.1011.*
[179] Klosgen., W., and Zytkow J. M., "Subgroup discovery," in Handbook of Data Mining and Knowledge Discovery. *New York: Addison-Wesley*, 2005, pp. 354–367.
[180] Ishibuchi, H., Murata, T., & Türkşen, I. B. (1997). Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy sets and systems*, 89(2), 135-150. *https://doi.org/10.1016/S0165-0114(96)00098-X.*
[181] Łapa, K., Cpałka, K., & Rutkowski, L. (2018). New aspects of interpretability of fuzzy systems for nonlinear modeling. *In Advances in Data Analysis with Computational Intelligence Methods* (pp. 225-264). Springer, Cham. *https://doi.org/10.1007/978-3-319-67946-4_9.*
[182] Cordón, O. (2011). A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *International journal of approximate reasoning*, 52(6), 894-913. *https://doi.org/10.1016/j.ijar.2011.03.004.*
[183] Ahmed, M. M., & Isa, N. A. M. (2017). Knowledge base to fuzzy information granule: A review from the interpretability-accuracy perspective. *Applied Soft Computing*, 54, 121-140. *https://doi.org/10.1016/j.asoc.2016.12.055.*
[184] Rey, M. I., Galende, M., Fuente, M. J., & Sainz-Palmero, G. I. (2017). Multi-objective based Fuzzy Rule Based Systems (FRBSs) for trade-off improvement in accuracy and interpretability: A rule relevance point of view. *Knowledge-Based Systems*, 127, 67-84. *https://doi.org/10.1016/j.knosys.2016.12.028.*
[185] Zhou, S. M., & Gan, J. Q. (2008). Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling. *Fuzzy sets and systems*, 159(23), 3091-3131. *https://doi.org/10.1016/j.fss.2008.05.016.*
[186] Alonso, J. M., Magdalena, L., & González-Rodríguez, G. (2009). Looking for a good fuzzy system interpretability index: An experimental approach. *International Journal of Approximate Reasoning*, 51(1), 115-134. *https://doi.org/10.1016/j.ijar.2009.09.004.*
[187] Gacto, M. J., Alcalá, R., & Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, 181(20), 4340-4360. *https://doi.org/10.1016/j.ins.2011.02.021.*
[188] Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems–achievements, new concepts and open issues. *Information sciences*, 251, 22-46. *https://doi.org/10.1016/j.ins.2013.07.002.*
[189] Pulkkinen, P., & Koivisto, H. (2009). A dynamically constrained multiobjective genetic fuzzy system for regression problems. *IEEE Transactions on Fuzzy Systems*, 18(1), 161-177. *https://doi.org/10.1109/TFUZZ.2009.2038712.*
[190] Liu, F., Quek, C., & Ng, G. S. (2007). A novel generic hebbian ordering-based fuzzy rule base reduction approach to Mamdani neuro-fuzzy system. *Neural Computation*, 19(6), 1656-1680. *https://doi.org/10.1162/neco.2007.19.6.1656.*
[191] Alonso, J. M., Magdalena, L., & Cordón, O. (2010, March). Embedding HILK in a three-objective evolutionary algorithm with the aim of modeling highly interpretable fuzzy rule-based classifiers. *In 2010 4th International Workshop on Genetic and Evolutionary Fuzzy Systems (GEFS)* (pp. 15-20). IEEE. *https://doi.org/10.1109/GEFS.2010.5454165.*
[192] Alonso, J. M., & Magdalena, L. (2011). HILK++: an interpretability-guided fuzzy modeling methodology for learning readable and comprehensible fuzzy rule-based classifiers. *Soft Computing*, 15(10), 1959-1980. *https://doi.org/10.1007/s00500-010-0628-5.*
[193] Gorzałczany, M. B., & Rudziński, F. (2012). Accuracy vs. interpretability of fuzzy rule-based classifiers: an evolutionary approach. *In Swarm and Evolutionary Computation* (pp. 222-230). Springer, Berlin, Heidelberg. *https://doi.org/10.1007/978-3-642-29353-5_26.*
[194] Guillaume, S., & Charnomordic, B. (2004). Generating an interpretable family of fuzzy partitions from data. *IEEE transactions on fuzzy systems*, 12(3), 324-335. *https://doi.org/10.1109/TFUZZ.2004.825979.*
[195] Ishibuchi, H., Nakashima, Y., & Nojima, Y. (2011, April). Double cross-validation for performance evaluation of multi-objective genetic fuzzy systems. *In 2011 IEEE 5th International Workshop on Genetic and Evolutionary Fuzzy Systems (GEFS)* (pp. 31-38). IEEE. *https://doi.org/10.1109/GEFS.2011.5949503.*
[196] Lamamra, K., Batat, F., & Mokhtari, F. (2020). A new technique with improved control quality of nonlinear systems using an optimized fuzzy logic controller. *Expert Systems with Applications*, 145, 113148. *https://doi.org/10.1016/j.eswa.2019.113148.*

[197] Kacimi, M.A,, & Guenounou, O. (2018). Multi-objective optimization of Mamdani Fuzzy Rule based System With Rule Minimization, *IC3E'18, Bouira, Algerie*

[198] Poojitha, V., Bhadauria, M., Jain, S., & Garg, A. (2016, January). A collocation of IRIS flower using neural network clustering tool in MATLAB. *In 2016 6th International Conference-Cloud System and Big Data Engineering (Confluence)* (pp. 53-58). IEEE. *https://doi.org/10.1109/CONFLUENCE.2016.7508047.*

[199] Pinto, J. P., Kelur, S., & Shetty, J. (2018, October). Iris Flower Species Identification Using Machine Learning Approach. *In 2018 4th International Conference for Convergence in Technology (I2CT)* (pp. 1-4). IEEE. *https://doi.org/10.1109/I2CT42659.2018.9057891.*

[200] Bora, D. J., Gupta, D., & Kumar, A. (2014). A comparative study between fuzzy clustering algorithm and hard clustering algorithm. *arXiv preprint arXiv*:1404.6059. *https://doi.org/10.14445/22312803/IJCTT-V10P119.*

[201] Márquez, A. A., Márquez, F. A., & Peregrín, A. (2010, July). A multi-objective evolutionary algorithm with an interpretability improvement mechanism for linguistic fuzzy systems with adaptive defuzzification. *In International Conference on Fuzzy Systems* (pp. 1-7). IEEE. *https://doi.org/10.1109/FUZZY.2010.5584294.*

Résumé

Le travail entrepris dans le cadre de cette thèse est consacré à l'étude de la conception automatique du système flou linguistique (Mamdani). Ce manuscrit, retrace la genèse du problème de conception de ces systèmes en parcourant toutes les familles d'approches rapportées dans la littérature. En même temps, il met en évidence les différentes complications, conditions et contraintes liées à ce problème. D'autre part, ce manuscrit introduit différentes propositions élaborées au cours de cette thèse qui sont présentées en contraste avec celles de la littérature. La première concerne l'optimisation, la seconde est orientée vers la conception mono-objectif et enfin, les dernières sont orientées vers la conception multi-objectif. Les résultats obtenus pour les différentes propositions, montrent un potentiel très prometteur à la fois pour les applications théoriques et les applications en temps réelles.

***Mots-Clés**: Systèmes flous de Mamdani; Optimisation Multi-objectif ; Apprentissage automatique.*

Abstract

The work undertaken in this thesis is devoted to the study of the automatic design of linguistic fuzzy system (Mamdani). This manuscript traces the genesis of the problem of designing these systems, by going through all the families of approaches reported in the literature. At the same time, it highlights the various complications, conditions and constraints related to this problem. On the other hand, this manuscript introduces different propositions elaborated during the course of this thesis, which are presented in contrast to those of the literature. The first concerns optimization, the second is oriented towards single-objective design, and the last ones are dedicated to multi-objective design. The obtained results show a very promising potential for both theoretical and real-time applications.

***Key-Words** : Mamdani fuzzy system; multi-objective optimization; Machine learning .*

ملخص

العمل الذي تمَّ في هذه الرسالة مخصص لدراسة التصميم الآلي للنظام الغامض اللغوي (Mamdani). تتتبع هذه المخطوطة نشأة مشكلة تصميم هذه الأنظمة، من خلال استعراض جميع عائلات الأساليب المذكورة والواردة في الأدبيات. في الوقت نفسه، يسلط الضوء على مختلف التعقيدات والقيود المتعلقة بهذه المشكلة. من ناحية أخرى، تقدم هذه المخطوطة اقتراحات مختلفة تمَّ تطويرها خلال مسار هذه الأطروحة. الاقتراح الأول مرتبط بمجال التفاؤل أو التحسين، والثاني موجه نحو التصميم أحادي الهدف، والاقتراحات الأخِيرة مخصص للتصميم متعدد الأهداف. تظهر النتائج المحصول عليها للإ قترحات المختلفة، إمكانيات واعدة للغاية لكلٍ من التطبيقات النظرية والتطبيقات الهندسية في الوقت الفِعلي.

***الكلمات المفتاحية:** الأنظمة* الغامضة *(Mamdani) ؛ التحسين متعدد الأهداف. التَعلم الآلـي.*