POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Optimisation par algorithmes de clustering de la construction automatique de bases de connaissances floues
Auteur: Author:	Aleksander Przybylo
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Przybylo, A. (2005). Optimisation par algorithmes de clustering de la construction automatique de bases de connaissances floues [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7671/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7671/
Directeurs de recherche: Advisors:	Marek Balazinski, & Luc Baron
Programme: Program:	Unspecified

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

OPTIMISATION PAR ALGORITHMES DE CLUSTERING DE LA CONSTRUCTION AUTOMATIQUE DE BASES DE CONNAISSANCES FLOUES

ALEKSANDER PRZYBYLO

DÉPARTEMENT DE GÉNIE MÉCANIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE MÉCANIQUE)

SEPTEMBRE 2005

© Aleksander Przybylo, 2005

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-16841-7
Our file Notre référence
ISBN: 978-0-494-16841-7

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

OPTIMISATION PAR ALGORITHMES DE CLUSTERING DE LA CONSTRUCTION AUTOMATIQUE DE BASES DE CONNAISSANCES FLOUES

présenté par : PRZYBYLO Aleksander

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. MASCLE Christian, Doct. ès sciences, président

M. BALAZINSKI Marek, Ph.D., membre et directeur de recherche

M. BARON Luc, Ph.D., membre et codirecteur de recherche

M. GALINIER Philippe, Doct., membre

REMERCIEMENTS

Je tiens à remercier tout d'abord mes directeurs de recherche : les Professeurs Marek Balazinski et Luc Baron. Je remercie également mon collègue Dr. Sofiane Achiche pour son soutien continu et inconditionnel.

Le support financier du Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG/NSERC) sous les subventions RGPIN-203618, RGPIN-105518 et STPGP-269579 est également grandement apprécié.

Dans le monde parascolaire, mes remerciements vont à Stéphanie qui a su me changer les idées quand il le fallait et à mes parents qui m'ont toujours appuyé dans mes études.

RÉSUMÉ

Le présent mémoire de maîtrise présente plusieurs applications des algorithmes de clustering à la génération automatique de bases de connaissances floues à l'aide d'algorithmes génétiques. Les quatre principaux objectifs des recherches présentées dans ce mémoire sont : la réduction du temps d'exécution, l'accroissement de la robustesse, l'amélioration de la performance et l'accroissement de l'autonomie.

La génération automatique de bases de connaissances floues à l'aide de techniques telles que les algorithmes génétiques a tendance à être fortement dépendante de la qualité des données d'apprentissage. Tout d'abord, des échantillons de données trop grands peuvent mener à d'importants temps de calcul, alors que des échantillons de données plus petits pourraient décrire le problème tout aussi bien. Pour palier à ce problème, on propose de compresser les données à l'aide d'algorithmes de clustering en éliminant les informations similaires et redondantes. Différents algorithmes de clustering sont comparés et la validation des résultats à travers une surface 3D synthétique montre que la compression de l'échantillon de données à l'aide d'algorithmes de clustering à 5% de sa taille permet d'accélérer le processus d'apprentissage de 94%.

De plus, la présence de bruit et d'outliers peut détériorer la qualité des résultats de l'algorithme d'apprentissage. Les algorithmes de clustering permettent le filtrage de données, rendant ainsi la génération de bases de connaissances floues plus précise. Lorsque la quantité d'outliers dans l'échantillon de données est importante, les algorithmes de clustering peuvent rendre les résultats plus stables et augmenter la

performance des bases de connaissances floues obtenues de jusqu'à 2.4% dans le cas d'un échantillon contenant 10% d'outliers, selon les tests effectués sur une surface 3D synthétique.

Les troisième et quatrième objectifs sont accomplis en palliant à un problème bien connu entourant la génération automatique de bases de connaissances floues à l'aide d'algorithmes génétiques, soit la recherche d'un nombre de sous-ensembles flous optimal pour chaque prémisse. Certains algorithmes génétiques couramment utilisés, emploient une méthode multi-objectif combinant la minimisation de l'erreur et la simplification du modèle (base de connaissances floues). Ce travail présente des solutions basées sur l'analyse des clusters et sur les indices de validation pour le nombre de clusters afin de prédéfinir les nombres de sous-ensembles flous. Deux indices de validation ainsi que la combinaison de l'un d'entre eux et de la méthode multi-objectif sont comparés à la méthode multi-objectif originale. La validation à travers des données synthétiques et expérimentales montre des améliorations considérables dans la précision de la prédiction effectuée avec l'usage des nombres de sous-ensembles flous prédéfinis. Sur l'ensemble de données synthétique, une amélioration moyenne de 15.4% (maximale de 21%) a été obtenue avec la technique Silhouette. Également, sur l'ensemble de données expérimentales, une amélioration de 28% a été obtenue. De plus, l'usage d'un nombre de sous-ensembles flous prédéfini permet de contourner la nécessité de préanalyser les données d'entraînement. Une complexité optimale peut être déterminée automatiquement par le processus de clustering, éliminant ainsi la nécessité de supervision humaine.

ABSTRACT

This master's thesis presents several applications of clustering algorithms for the automatic generation of fuzzy knowledge bases using genetic algorithms. Four main objectives are targeted: execution time, robustness, performance and autonomy.

Automatic knowledge base generation techniques such as genetic algorithms tend to be highly dependent on the quality and size of the learning data. First of all, large data sets can lead to unnecessary time loss, when smaller data sets could describe the problem as well. To address this issue, data are compressed by reducing similar and redundant information. Different clustering algorithms are compared and the validation of the results through a synthetic 3D surface shows that when compressing the data to 5% of its original size, clustering algorithms accelerate the learning process by up to 94%.

Second of all, the presence of noise and outliers can lead the learning algorithm to degenerate. Clustering techniques allow the filtering of the data, thus making the generation of fuzzy knowledge bases more accurate. When the learning data contains a large amount of outliers, clustering algorithms can make the results more stable and improve the fitness of the resulting fuzzy knowledge bases by 2.4% in presence of 10% of outliers, based on the tests performed on a synthetic 3D surface.

The third and fourth objectives are accomplished by addressing a well known issue surrounding fuzzy knowledge base generation using genetic algorithms: finding an optimal number of fuzzy sets for each premise. Some of the current genetic algorithm

methods for the automatic generation of fuzzy knowledge bases use a multi-objective method combining error minimization and simplification. This work proposes solutions based on cluster analysis and validation indices for the numbers of clusters used to predefine the numbers of fuzzy sets. Two different validation indices as well as a combination of one of them and of the multi-objective method are compared to the original multi-objective method. The validation with synthetic and experimental data shows considerable improvement in terms of prediction accuracy for fuzzy knowledge bases obtained with predefined numbers of fuzzy sets. On the synthetic data set, an average improvement of 15.4% (and a maximum of 21%) was obtained with the Silhouette technique. On the experimental data set, an improvement of 28% was obtained with the same technique. Moreover, the use of a predefined number of fuzzy sets removes the necessity to preanalyze the training data. An optimal complexity can be determined automatically by the clustering process, thus removing the need of human supervision.

TABLE DES MATIÈRES

Remerciements iv

Résumé v

Abstract vii

Table des matières ix

Liste des annexes xv

Liste des figures xvi

Liste des tableaux xix

Liste des sigles et abréviations xx

Introduction 1

Révue de la littérature 5

CHAPITRE 1 - Notions générales 7

1.1 Logique floue 7

1.1.1 Historique 7

1.1.2 Notions de base 8

1.1.3 Bases de connaissances floues 10

1.1.4 Moteurs d'inférence 12

1.1.5 Méthodes de défuzzification 17

1.2 Algorithmes génétiques 17

1.2.1 Historique 17

1.2.2 Notions générales 18

1.2.3 Codage des individus 19

1.2.4 Indice de performance 19

1.2.5 Population initiale 20

1.2.6 Opérateurs 20

1.2.6.1 Reproduction 20

1.2.6.2 Mutation 21

1.2.6.3 Sélection naturelle 21

1.3 Algorithmes de clustering 22

1.3.1 Historique 22

1.3.2 Notions générales 23

1.3.2.1 Algorithmes de clustering hiérarchiques 23

1.3.2.2 Algorithmes de clustering à partitionnement 25

1.3.3 Distribution initiale 26

1.3.4 Critère d'optimisation 28

1.3.5 Fonctions de distance/similarité 29

1.3.6 Mesure centrale 29

1.4 Intégration des trois domaines de l'intelligence artificielle 30

1.4.1 Intégration de la logique floue et des algorithmes génétiques 30

1.4.2 Intégration des algorithmes de clustering 30

CHAPITRE 2 - Synthèse des articles 33

2.1 Prétraitement de données brutes 33

2.2 Prédéfinition des nombres de sous-ensembles flous 36

CHAPITRE 3 - Influence of Clustering Pre-processing on Genetically Generated Fuzzy Knowledge Bases 39

Abstract 39

3.1 Introduction 40

3.2 Fuzzy Decision Support System 41

3.2.1 Theoretical specificities 41

3.2.2 FDSS Learning Paradigm 42

3.3 Automatic Generation of FKBs 43

3.3.1 Coding 44

3.3.2 Multi-Crossover Mechanisms 45

3.3.2.1 Premises/Conclusion Crossover 45

3.3.2.2 Fuzzy Rules Crossover 46

3.3.3 Fuzzy Set Reducer 46

3.3.4 Mutation 47

3.3.5 Natural Selection 47

3.3.6 Performance Criterion of the RBCGA 47

3.4 Clustering 48

3.4.1 The Hierarchical Algorithm 48

3.4.2 The Partitioning Algorithms 49

3.4.3 Central Measure 50

3.4.4 Distance/Similarity Functions 50

3.4.5 Optimal Number of Clusters 51

3.5 Validation Results 52

3.5.1 Time Reduction 56

3.5.2 Noise Influence 57

3.5.3 Outliers Influence 59

3.6 Conclusion 63

Acknowledgment 63

References 64

CHAPITRE 4 - Nombre de clusters dans un ensemble de données 67

4.1 Méthodes usuelles 67

4.2 Méthodes automatiques 68

CHAPITRE 5 - Predefining Numbers of Fuzzy Sets for genetically generated Fuzzy Knowledge Bases using Clustering Techniques 70

Abstract 70

5.1 Introduction 71

5.2 Fuzzy Decision Support System 72

5.3 Automatic Generation of FKBs 73

5.3.1 Coding 74

5.3.2 Multi-Crossover Mechanisms 74

5.3.2.1 Premises/Conclusion Crossover 74

5.3.2.2 Fuzzy Rules Crossover 75

5.3.3 Fuzzy Set Reducer 75

5.3.4 Mutation 76

5.3.5 Natural Selection 76

5.3.6 Performance Criterion of the RBCGA 77

5.4 Clustering 78

5.4.1 The *k*-means Algorithm 78

5.4.2 Number of Clusters 79

5.4.3 Global Algorithm Overview 81

5.5 Validation Results 82

5.5.1 3D Surface Data 82

5.5.2 Experimental Data 84

5.6 Conclusion 90

5.7 Acknowledgment 90

References 91

Discussion des résultats 95

Conclusion 97

Références 100

Annexes 104

LISTE DES ANNEXES

Annexe A - Manuel du logiciel GFS Cluster 104

A1 - Interface usager 104

A2 - Fenêtres de dialogue pour le clustering et pour le nombre de clusters 108

A3 - Format des fichiers d'importation et d'exportation 111

LISTE DES FIGURES

Figure 1.1 – Fonction d'appartenance en logique floue 9

Figure 1.2 – Exemple de base de connaissances floues 11

Figure 1.3 – Exemple de base de connaissances floues sous forme graphique 12

Figure 1.4 – Moteur d'inférence de Mamdani avec des entrées non floues 14

Figure 1.5 – Moteur d'inférence de Mamdani avec des entrées floues 14

Figure 1.6 – Moteur d'inférence de Larsen avec des entrées non floues 15

Figure 1.7 – Moteur d'inférence de Larsen avec des entrées floues 16

Figure 1.8 – Reproduction par croisement des chromosomes 21

Figure 1.9 – Opération de mutation 21

Figure 1.10 – Vue d'ensemble du fonctionnement des algorithmes génétiques 22

Figure 1.11 – Fonctionnement des algorithmes de clustering hiérarchiques 24

Figure 1.12 – Analogie entre les sous-ensembles flous et les clusters 32

Figure 3.1 – Graphic Interface of the FDSS Fuzzy-Flou 43

Figure 3.2 – Genetic learning paradigm 44

Figure 3.3 – Blended crossover α (BLX-α) 46

Figure 3.4 – Simple crossover 46

Figure 3.5 – Graphic Interface of the Clustering module 53

Figure 3.6 – Theoretical surface in 3D form 54

Figure 3.7 – Theoretical surface in colormap form 55

Figure 3.8 – Surface obtained with Gaussian noise of 0.1 standard deviation and 20% outlier probability 55

Figure 3.9 – Time reduction of the learning process with the use of clustering algorithms 56

Figure 3.10 – Influence of noise on the fitness of automatically generated FKBs with and without the use of clustering algorithms 58

Figure 3.11 – Influence of outliers on the fitness of automatically generated FKBs with and without the use of clustering algorithms 60

Figure 3.12 – Surface generated from the FKB obtained without clustering pre-processing on a data set with Gaussian noise of 0.1 standard deviation and 20% outlier probability 62

Figure 3.13 – Surface generated from the FKB obtained with clustering pre-processing on a data set with Gaussian noise of 0.1 standard deviation and 20% outlier probability 62

Figure 4.1 – Courbe de l'erreur de reproduction en fonction du nombre de clusters 67

Figure 5.1 – Graphic user interface of FDSS Fuzzy-Flou 72

Figure 5.2 – Genetic learning paradigm 73

Figure 5.3 – Simple crossover 75

Figure 5.4 – Screenshot of the clustering software 79

Figure 5.5 – Validation errors on synthetic data 83

Figure 5.6 – The selected signal features versus the used up portion of tool life from the training data (the symbols are defined above) 85

Figure 5.7 – Mean validation errors on experimental data 86

Figure 5.8 – Mean complexity of the FKBs 87

Figure 5.9 – FKBs obtained with the MO method (left) and with predefined numbers of fuzzy sets with the Silhouette technique (right) 88

Figure 5.10 – Predicted tool wear versus experimental data on four different tools 89

Figure A.1 – Capture d'écran de l'interface graphique du logiciel GFS Cluster 105

Figure A.2 – Fenêtre de dialogue pour le clustering 108

Figure A.3 – Fenêtre de dialogue pour le nombre de clusters 111

Figure A.4 – Exemple de fichier d'importation en format ASCII 112

LISTE DES TABLEAUX

Tableau 1.1 – Quelques opérations usuelles en logique floue 8

Tableau 1.2 – Opérations dans les moteurs d'inférence 11

Tableau 1.3 – Correspondance entre les éléments des algorithmes génétiques et ceux de la logique floue 28

Table 3.1 – Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of noise..... 58

Table 3.2 – Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of outliers..... 60

Table 5.1 – Summary of the validation errors on theoretical data..... 84

LISTE DES SIGLES ET ABRÉVIATIONS

Abréviations

AE	émissions acoustiques (*Acoustic Emissions*)
AG	algorithme génétique
AGNES	imbrication agglomérative (*Agglomerative Nesting*)
BCF	base de connaissances floues
BLX	croisement mixé (*Blended Crossover*)
CHpF	Calinski-Harabasz pseudo-F
COG	centre de gravité (*Centre of Gravity*)
CRI	règle de composition d'inférence (*Compositional Rule of Inference*)
DIANA	Analyse divisive (*Divisive Analysis*)
FDSS	système d'aide à la décision flou (*Fuzzy Decision Support System*)
FKB	base de connaissances floues (*Fuzzy Knowledge Base*)
GA	algorithme Génétique (*Genetic Algorithm*)
GFS	Geno-Flou Suite
HART	Hatrigan
HART + MO	combinaison des méthodes Hartigan et multi-objectif

MAX	maximum
MIN	minimum
MO	multi-objectif (*multi-objective*)
MOM	moyenne des maximums (*Mean of Maxima*)
PC	ordinateur personnel (*Personal Computer*)
PROD	produit
RBCGA	algorithme génétique hybride réel-binaire (*Real Binary-like Coded Genetic Algorithm*)
RMS	moyenne quadratique (*Root Mean Square*)
Var	variance

Sigles

a_i	dissimilarité moyenne entre l'objet i et tous les autres objets dans le même cluster
b_i	le plus petit $d_{i,j}$
B_k	somme des distances au carré inter-cluster
$d_{i,j}$	dissimilarité moyenne entre l'objet i et tous les objets du cluster j
E	erreur de réproduction
F_f	force d'avance (*Feed Force*)
k	nombre de clusters

k_{opt}	nombre de clusters optimal
L	étendue totale des données d'entraînement
n	taille de l'échantillon d'entraînement
p_c	probabilité de croisement de chromosomes
p_r	probabilité de réduction de complexité
p_m	probabilité de mutation
P	taille de la population
W_k	somme des distances au carré intra-cluster

Sigles grecques

Δ	erreur
ΔT	portion de la durée de vie écoulée
ϕ	indice de performance
μ	moyenne
μ_A	fonction d'appartenance du sous-ensemble flou A
σ	écart type
ω	poids

Opérateurs

also	connection de phrase

Σ	sommation
$\wedge$	minimum
$\vee$	maximum
$\circ$	opérateur d'inférence composée
*	opérateur produit

INTRODUCTION

L'intelligence artificielle est un domaine qui aux yeux de plusieurs est encore réservé à la science fiction. La seule juxtaposition des mots « intelligence » et « artificielle » peut sembler absurde. Pourtant, les recherches dans ce domaine font partie non seulement des plus intensives mais également des plus fructueuses. Une multitude de familles d'algorithmes aussi nombreux que divers cherchent à imiter, avec différents degrés de réussite, le fonctionnement du cerveau humain, de l'évolution de l'espèce, etc.

Ce mémoire se penche sur trois domaines différents de l'intelligence artificielle, afin de les intégrer et d'en exploiter au mieux les capacités. Tout d'abord, la logique floue est exploitée comme méthode de raisonnement. Ensuite, les algorithmes génétiques (AGs) sont utilisés comme outil d'apprentissage. Finalement, les méthodes de reconnaissance de patron par algorithmes de clustering servent à optimiser le fonctionnement des AGs et sont à la base des travaux présentés dans ce mémoire.

Les objectifs du travail présenté dans ce mémoire sont de rendre la génération des BCFs par les AGs :

- plus rapide en compressant les données redondantes;
- plus robuste en filtrant le bruit et les outliers dans les données brutes;
- plus performante et autonome en préanalysant les données d'apprentissage.

Tous ces objectifs sont accomplis par l'usage des algorithmes de clustering.

Le contenu du mémoire est composé de deux articles soumis à des journaux scientifiques et le premier des deux est déjà en cours de publication lors de la rédaction de ce mémoire. Le premier chapitre est divisé en trois sections, chacun traitant des notions de base de la logique floue, des AGs et des algorithmes de clustering. Ce chapitre a pour but de familiariser le lecteur avec le jargon technique attaché à chacun de ces trois domaines et de vulgariser certaines notions de base. Aussi, pour chacun de ces domaines, un court historique et les fondements théoriques sont présentés.

Le deuxième chapitre présente une synthèse en français des articles présentés aux chapitres trois et cinq.

Le troisième chapitre est constitué intégralement du premier article de journal intitulé « Influence of Clustering Pre-processing on Genetically Generated Fuzzy Knowledge Bases. » Il adresse les deux premiers objectifs et contient une description de l'application des algorithmes de clustering à l'apprentissage automatique de BCFs en se penchant sur leurs capacités de filtrage et de compression. On cherche à réduire le temps d'apprentissage en compressant les données brutes tout en conservant au maximum les informations qui y sont contenues. De plus, on montre que non seulement il est possible de réduire le temps d'apprentissage nécessaire mais aussi d'améliorer simultanément la performance des BCFs obtenues en réduisant l'effet du bruit et des données suspectes (outliers).

Le quatrième chapitre sert de lien entre les deux articles. Il présente une des limites de certains algorithmes de clustering présentés dans le troisième chapitre, notamment la

nécessité d'une connaissance à priori concernant le nombre de clusters dans les données brutes. On introduit ensuite les différentes techniques qui permettent de pallier à ce problème ainsi que leur application à l'apprentissage automatique.

Le cinquième et dernier chapitre contient le deuxième article intitulé « Predefining numbers of fuzzy sets for genetically generated fuzzy knowledge bases using clustering techniques, » qui adresse le troisième objectif de ce mémoire. Dans cet article, on cherche à optimiser l'apprentissage automatique des BCFs en analysant les données d'apprentissage afin de fixer certains paramètres nécessaires au fonctionnement des AGs. On analyse deux méthodes de détermination du nombre de clusters dans un ensemble de données afin de prédéfinir le nombre de sous-ensembles flous pour chacune des prémisses. Une comparaison avec la méthode par multi-objectif souvent employée lors de l'apprentissage automatique par AGs est également incluse. Ce chapitre contient également une application des méthodes décrites sur un ensemble de données expérimentales prélevées lors de tests d'usure des outils de coupe.

Un manuel d'utilisation du logiciel GFS Cluster, développé pour fins d'analyses contenues dans ce mémoire, est présenté en annexe.

L'ensemble de la recherche décrite dans ce mémoire, s'inscrit dans le contexte d'un projet de recherche intitulé Géno-Flou. Lors de la rédaction de ce mémoire un groupe de deux professeurs, un stagiaire post-doctoral, deux étudiants au doctorat et deux étudiants à la maîtrise font partie de l'équipe. Le but de ce projet est la création d'une suite logicielle à trois modules : prétraitement de données, apprentissage automatique et aide à

la décision. Les travaux décrits dans ce mémoire s'occupent surtout des deux premiers modules.

RÉVUE DE LA LITTÉRATURE

La recherche présentée dans ce mémoire porte sur l'optimisation des bases de connaissances floues à l'aide des algorithmes génétiques, plus particulièrement des RBCGA. La documentation sur ces algorithmes se trouve dans les multiples articles publiés par S. Achiche, M. Balazinski et L. Baron. Les détails sur l'algorithme en tant que tel, peuvent être trouvés dans [2]. Une description plus générale sur l'application des algorithmes génétiques à la création des BCFs peut être trouvée dans [1] et [6], alors que [3] et [4] portent plutôt sur certaines particularités des RBCGA exploitées dans l'optimisation par algorithmes de clustering présentée dans ce mémoire. Pour ce qui est des ouvrages généraux sur les algorithmes génétiques et la logique floue, [12] et [33] respectivement ont été principalement consultés.

Les algorithmes de clustering étudiées sont décrits dans [13] pour ce qui est de AGNES et de k-medoids et dans [17] pour ce qui est de k-means. Étant donné que le principal sujet de recherche de ce mémoire est le domaine des algorithmes de clustering, une multitude d'ouvrages généraux ont été consultés afin de retracer l'évolution et de découvrir la diversité de ces algorithmes. Les ouvrages en question sont les suivants : [8], [14], [16], [17], [21] et [22]. Parmi cette liste, les travaux de J. A. Hartigan ([16] et [17]) sont une référence incontournable dans le domaine.

Finalement, pour ce qui est de la recherche du nombre de clusters dans un ensemble de données brutes, plusieurs méthodes ont été testées et comparées. Dans la plupart des cas,

les documents consultés sont des articles de journaux ou de conférences publiés par les développeurs mêmes des méthodes en question. Ainsi, la méthode CHpF développée par Calinski et Harabasz est décrite dans [8], la méthode de Hartigan dans [17], celle de Krzanowski et Lai dans [24] et finalement la méthode de calcul de silhouette développée par Kaufman et Rousseeuw dans [22]. Mentionnons également que d'autres méthodes ont été récemment proposées mais n'ont pas été étudiées dans ce mémoire, telles que *Gap* proposée par Tibshirani et al. [31] et *Jump* développée par Sugar et James [30]. Cette dernière référence fournit également une description exhaustive et des résultats expérimentaux des autres méthodes citées auparavant.

CHAPITRE 1

NOTIONS GÉNÉRALES

Ce chapitre contient les notions générales sur les trois domaines de l'intelligence artificielle étudiés dans ce mémoire. La première section se penche sur la logique floue, la seconde sur les AGs et la troisième sur les algorithmes de clustering.

1.1 Logique floue

1.1.1 Historique

La logique floue a été introduite en 1965 par Lofti Zadeh, professeur de l'Université de Berkeley. Elle représente une alternative à la logique dite « conventionnelle » introduite par Aristote, où un élément doit absolument appartenir ou non à un ensemble. En contre partie, la logique floue, plus flexible que celle énoncée précédemment, permet d'affirmer qu'un élément peut appartenir à un ensemble avec un certain degré d'appartenance. En 1973 Zadeh a également introduit la notion de variables linguistiques dans un article publié dans *IEEE Transactions on Systems, Man and Cybernetics* [33].

La résistance de la communauté scientifique face à la logique floue était considérable. Ceci s'explique en grande partie par le fait que l'incertitude, le fondement même de la logique floue, a toujours été un élément que l'on cherchait à éviter dans les modèles proposés. Plutôt que chercher à développer des modèles qui assumaient leurs propres

imprécisions, on s'évertuait à développer des modèles parfaits qui tiennent compte d'absolument tout. Ceci explique pourquoi 25 années se sont écoulées avant que l'on commence à prendre les travaux de L. Zadeh au sérieux et à chercher des applications concrètes.

La première application pratique basée sur la logique floue a été réalisée par F.L. Smith & Co. en 1980 au Danemark et elle portait sur le contrôle de fours à ciment. La logique floue a commencé à connaître une popularité grandissante en 1987, avant d'atteindre son apogée en 1990. Le Japon a dominé le marché mondial durant ces années. Il a mis en pratique les notions de logique floue pour contrôler le système de freinage dans le métro de Tokyo. De nos jours, la logique floue trouve ses applications dans une multitude de domaines divers, tels que l'imagerie vidéo, les électroménagers etc.

1.1.2 Notions de base

La différence fondamentale entre la logique floue et la logique conventionnelle est illustrée par les équations (1.1) et (1.2). On remarque que dans la logique conventionnelle, seules deux valeurs sont possibles (0 ou 1), autrement dit, un élément peut appartenir ou pas à un sous-ensemble donnée. Dans la logique floue, une infinité de valeurs comprises entre 0 et 1 sont possibles, donc un élément peut appartenir à un sous-ensemble à un certain degré.

Logique conventionnelle :

$$\mu_A(x) = \begin{cases} 1 & \text{si} \quad x \in A \\ 0 & \text{si} \quad x \notin A \end{cases} \text{ ou } \mu_A : X \rightarrow \{0,1\} \tag{1.1}$$

Logique floue :

$$\mu_A : X \rightarrow [0,1] \tag{1.2}$$

Dans les équations (1.1) et (1.2), $\mu_A(x)$ représente le degré d'appartenance de l'élément x dans le sous-ensemble A de l'ensemble global X.

La figure 1.1 présente un exemple de classification de la vitesse d'une voiture en trois sous-ensembles flous. On remarque que selon ce modèle une vitesse ne change pas brusquement de classe en croissant d'un kilomètre par heure. Cette transition se fait graduellement, ce qui est plus naturel. Ainsi, la vitesse d'une voiture qui roule à $v = 35$ km/h est à 25% petite ($\mu_{petite} = 0.25$), à 75% moyenne ($\mu_{moyenne} = 0.75$) et à 0% grande ($\mu_{grande} = 0.0$).

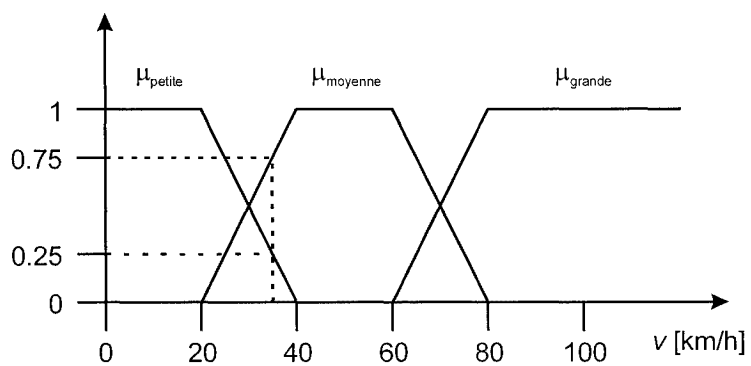

Figure 1.1 – Fonction d'appartenance en logique floue

Suivant cette logique, les opérations sur les sous-ensembles ont été étendues afin de pouvoir s'appliquer à la logique floue. Le tableau 1.1 fournit quelques exemples d'opérations usuelles s'appliquant à la logique floue.

Tableau 1.1 – Quelques opérations usuelles en logique floue

Opération	Fonction d'appartenance
Égalité	$\forall x \in X, \mu_A(x) = \mu_B(x)$
Inclusion	$\forall x \in X, \mu_A(x) \leq \mu_B(x)$
Union	$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$
Intersection	$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$
Complément	$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

1.1.3 Bases de connaissances floues

Afin d'obtenir un système d'aide à la décision basé sur la logique floue, il faut disposer d'un moyen de traiter les différentes informations dont on dispose. Tout d'abord, une BCF est nécessaire afin de pouvoir modéliser les informations accumulées et fournir une ou plusieurs conclusions. De telles BCFs sont usuellement construites par des experts connaissant tous les aspects du phénomène modélisé. Une fois que l'expert ait transmis son savoir dans la BCF, en théorie, sa présence n'est plus nécessaire car la machine peut désormais simuler son raisonnement. Les connaissances sont transmises tout d'abord sous forme de prémisses et de conclusions subdivisées en sous-ensembles flous et ensuite sous forme de règles reliant les prémisses aux conclusions tel que décrit par l'équation (1.3).

$$R^k_{\text{MISO}} : \text{si } x_1 \text{ est } X^k_1 \text{ et } x_2 \text{ est } X^k_2 \text{ et} \ldots \text{ et } x_n \text{ est } X^k_n \text{ alors } y \text{ est } Y_k \tag{1.3}$$

où R désigne une règle, x et X désignent respectivement une prémisse et un sous-ensemble flou de cette prémisse, y et Y désignent respectivement la conclusion et un sous-ensemble flou de la conclusion, k est le nombre de règles, n est le nombre de

prémisses et où MISO (Multiple Input Single Output) indique qu'il s'agît d'une BCF à plusieurs entrées et une sortie. Le présent mémoire ne traite que de ce type de BCFs.

La figure 1.2 présente un exemple classique de BCF : à partir de la vitesse d'un véhicule et de sa distance de la ligne d'arrêt, on décide de l'intensité de freinage que l'on doit appliquer.

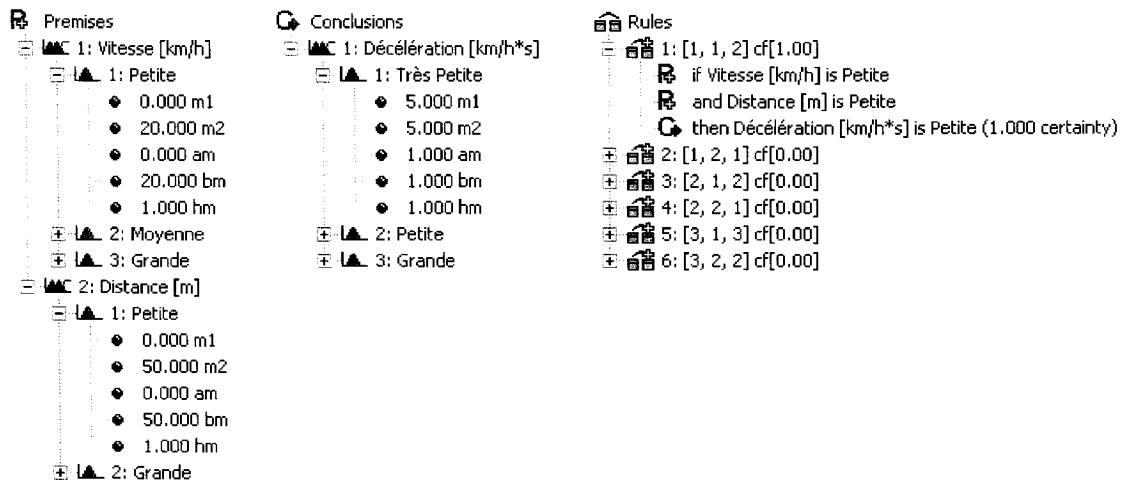

Figure 1.2 – Exemple de base de connaissances floues

Sur la figure 1.2, on distingue bien trois catégories : 1) les prémisses (*premises*) étant la vitesse et la distance, 2) la conclusion étant la décélération et 3) les règles (*rules*). Par exemple, la règle #5 indique [3, 1, 3], c'est-à-dire que lorsque la vitesse est grande (3) et que la distance est petite (1), la décélération doit être grande (3).

La figure 1.3 présente la même BCF sous forme graphique où l'on peut voir les tailles de toutes les prémisses et conclusions. Les observations présentées sur cette figure sont les suivantes : v = 32 km/h et d = 60 m. En se basant sur la BCF fournie et le moteur

d'inférence choisi (voir la sous-section suivante), le système d'aide à la décision flou conseille d'appliquer une décélération de 9.3762 km/h*s.

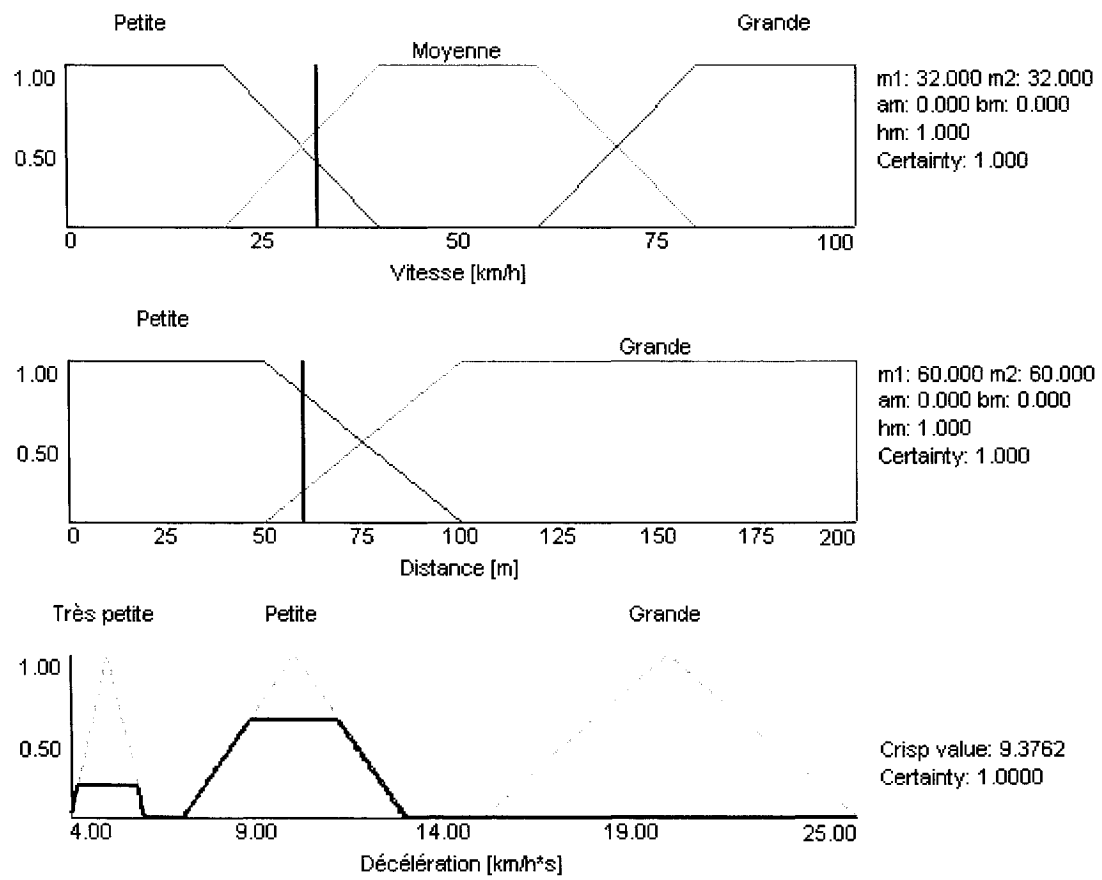

Figure 1.3 – Exemple de base de connaissances floues sous forme graphique

1.1.4 Moteurs d'inférence

Afin de tirer des résultats des différentes entrées nous avons besoin d'un mécanisme permettant de faire la composition des règles mises en jeu. Ceci est assuré par un moteur d'inférence. L'inférence se fait en deux étapes : la composition et l'implication. La première s'applique aux fonctions d'appartenance des prémisses et des entrées et fournit

des valeurs précises sur le degré d'influence de chaque règle pour chacune des prémisses. La deuxième étape permet d'obtenir le degré d'influence global de chaque règle à partir des valeurs obtenues précédemment. Ces valeurs sont ensuite appliquées aux fonctions d'appartenance des conclusions. Le tableau 1.2 résume les opérations utilisées à ces deux étapes par les deux moteurs d'inférence les plus populaires, celui de Mamdani et celui de Larsen.

Tableau 1.2 – Opérations dans les moteurs d'inférence

	Composition	**Implication**
Mamdani	MAX-MIN	MIN
Larsen	MAX-MIN	PROD

Les figures 1.4 et 1.5 illustrent une inférence selon la méthode de Mamdani pour une BCF à deux prémisses et à deux règles. La figure 1.4 illustre le cas avec des entrées non floues et la figure 1.5 le cas avec des entrées floues. Dans les deux cas, les variables *u* et *v* sont des prémisses, la variable *w* est la conclusion, les α_i sont les résultats de la composition et les C_i' sont les conclusions ayant pour fonctions d'appartenance $\mu_{Ci'}$.

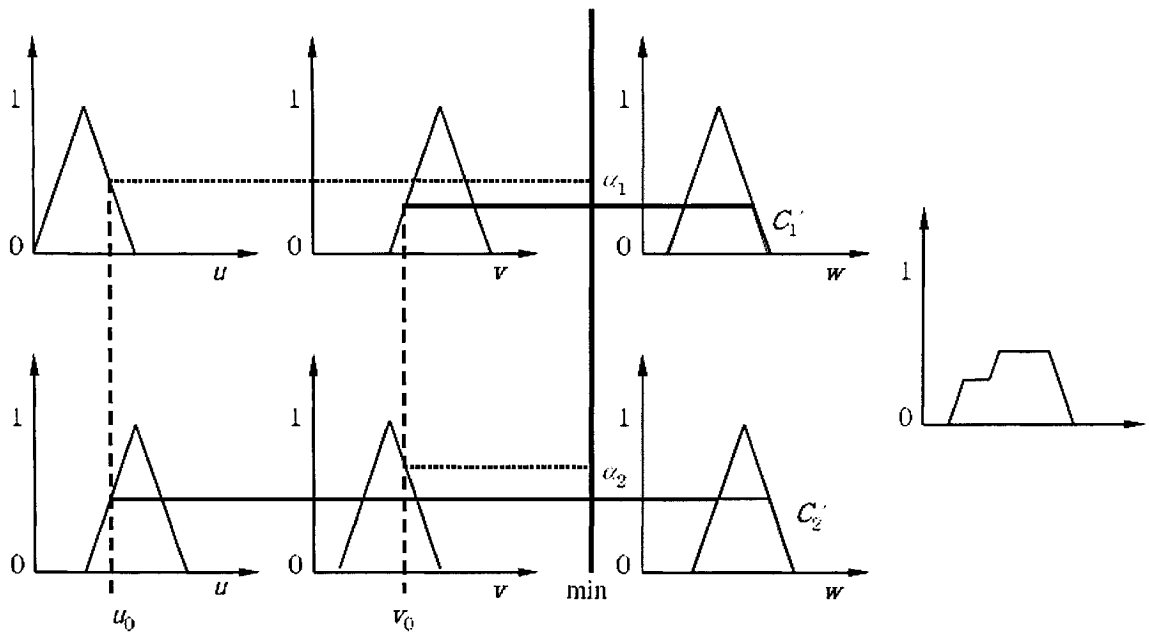

Figure 1.4 – Moteur d'inférence de Mamdani avec des entrées non floues

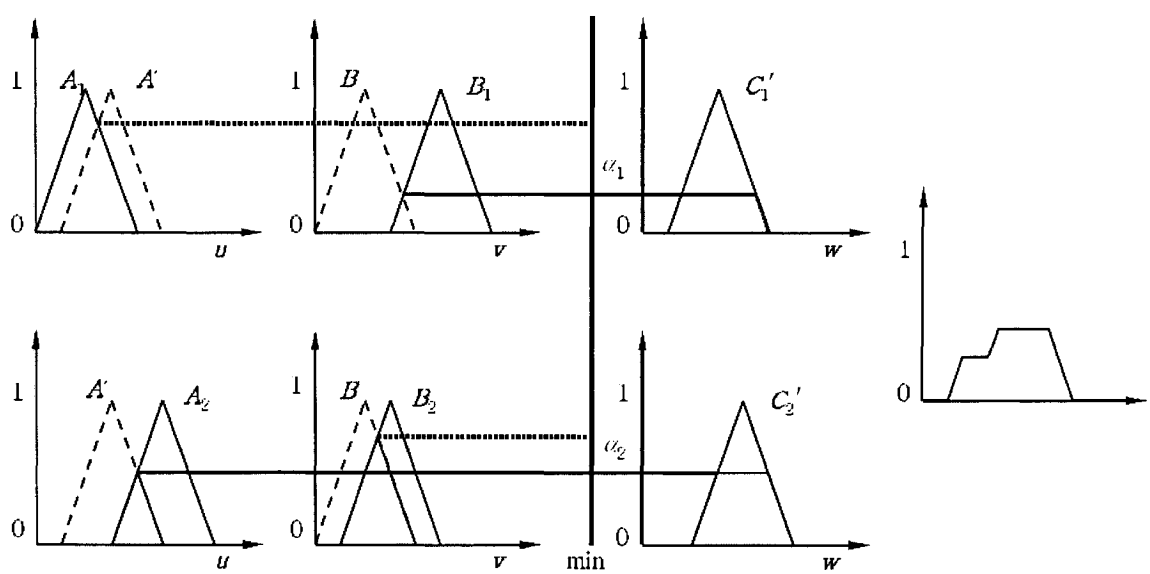

Figure 1.5 – Moteur d'inférence de Mamdani avec des entrées floues

On remarque que tout d'abord on effectue une opération MAX-MIN aux fonctions des prémisses et celles des entrées afin d'obtenir α_1 et α_2 et ensuite une opération MIN aux valeurs obtenues et aux C'.

En général, la méthode de Mamdani se résume comme suit :

$$\mu_{C'}(w) = \bigvee_{i=1}^{n}\left[\alpha_i \wedge \mu_{C_i}(w)\right] = \bigvee_{i=1}^{n} \mu_{C_i'}(w) \tag{1.4}$$

Les figures 1.6 et 1.7 illustrent la méthode de Larsen pour cette même BCF. Les significations des variables sont les mêmes qu'aux figures 1.4 et 1.5.

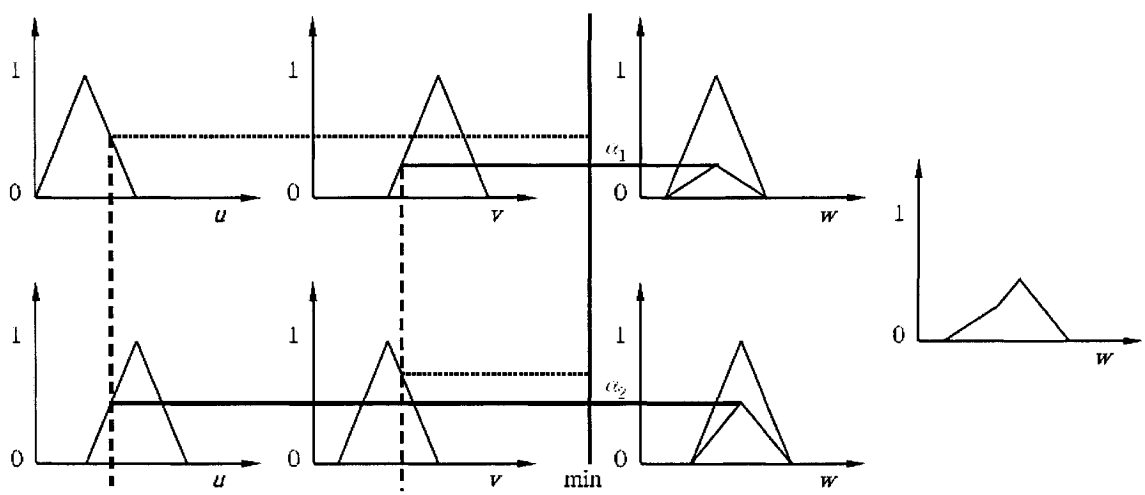

Figure 1.6 – Moteur d'inférence de Larsen avec des entrées non floues

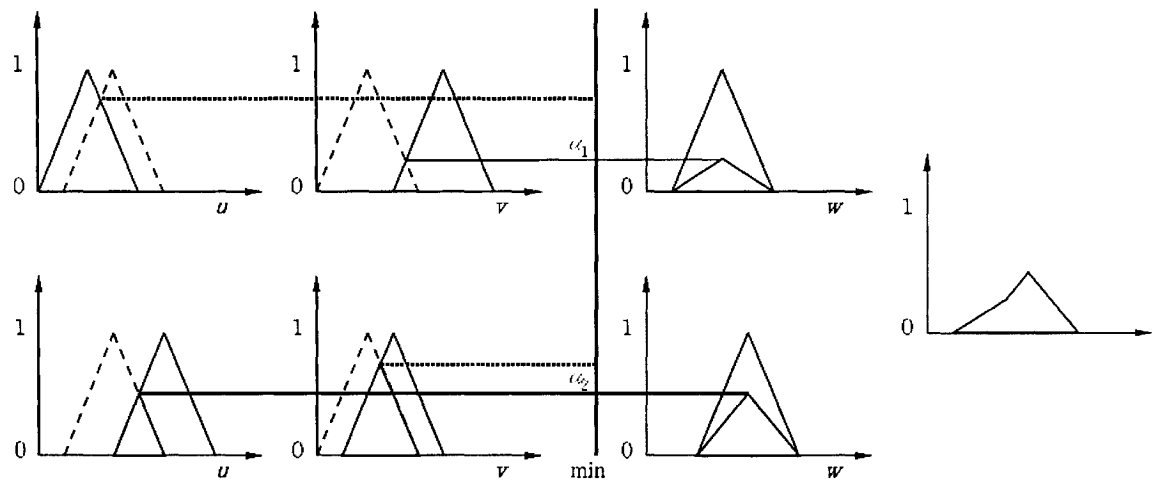

Figure 1.7 – Moteur d'inférence de Larsen avec des entrées floues

On remarque ici les effets de remplacer l'opérateur d'implication MIN par PROD : à la place de retrouver des fonctions d'appartenance trapézoïdales après section des fonctions d'appartenance triangulaires, on obtient des fonctions d'appartenance triangulaires par mise à échelle.

La méthode de Larsen se résume comme suit :

$$\mu_{C'}(w) = \bigvee_{i=1}^{n}\left[\alpha_i \circ \mu_{C_i}(w)\right] = \bigvee_{i=1}^{n} \mu_{C_i'}(w) \tag{1.5}$$

Dans les deux méthodes les α_i sont obtenus à l'aide de l'équation suivante :

$$\alpha_i = \min\left[\max_u\left(\mu_{A'}(u) \wedge \mu_{A_i}(u)\right), \max_v\left(\mu_{B'}(v) \wedge \mu_{B_i}(v)\right)\right] \tag{1.6}$$

Les fonctions finales affichées à droite dans les quatre figures sont obtenues avec une opération MAX, mais celle-ci ne fait pas partie des moteurs d'inférence en tant que tels. Pour terminer l'opération, il faut encore effectuer une défuzzification qui est décrite dans la sous-section suivante.

1.1.5 Méthodes de défuzzification

Afin d'obtenir une valeur numérique après que l'étape d'agrégation se soit produite, il faut utiliser une méthode de défuzzification. Plusieurs méthodes de défuzzification existent, mais la plus utilisée est celle du centre de masse.

Le centre de masse est la méthode de défuzzification privilégiée, car c'est celle qui respecte le mieux « l'allure » de la courbe obtenue par agrégation. En effet, cette méthode donne une importance à tout l'intervalle de réponse.

La méthode du maximum, consiste à repérer les points où le degré d'appartenance est le plus élevé. Si un seul point est trouvé, ce point constituera la valeur numérique finale. D'autres méthodes de défuzzification existent, telles que la sélection du point le plus à gauche ou le plus à droite, ou encore la moyenne de tous ces points (*Mean of Maxima*).

Il est important de mentionner que la défuzzification signifie une perte d'information et aucune des méthodes ne peut s'avérer meilleure que les autres. Il faut être en mesure de choisir la méthode de défuzzification la mieux adaptée à l'application développée.

1.2 Algorithmes génétiques

1.2.1 Historique

Les algorithmes génétiques ont été développés par John Holland, ses collègues et ses étudiants à l'Université du Michigan. Ses travaux ont commencé au début des années 60, mais ne sont arrivé à un premier aboutissement qu'en 1975 avec la publication de *Adaptation in natural and artificial system* [19]. Deux objectifs principaux étaient

poursuivis : (1) abstraire et expliquer rigoureusement le procédé adaptatif des systèmes naturels et (2) faire le design d'un logiciel de systèmes artificiels qui retient les mécanismes importants des systèmes naturels [12]. Le principal point d'intérêt qui a poussé les recherches dans le domaine des AGs était la robustesse. On voulait éviter la nécessité de recommencer le design d'un système à chaque fois que celui-ci change d'environnement. Les systèmes artificiels sont encore loin de la robustesse, de la flexibilité et de l'adaptabilité des systèmes naturels et c'est cet écart que les AGs cherchent à réduire.

1.2.2 Notions générales

Les algorithmes génétiques sont des méthodes d'optimisation qui commencent avec une population d'individus (dans le cas de ce mémoire, des BCFs) et performent des croisements de code génétique afin de tendre vers un optimum en un nombre donné de générations. Cette méthode simule le mécanisme de croisement de chromosomes naturel et se base sur les théories de l'évolution de l'espèce énoncées par Darwin. Il existe plusieurs types d'algorithmes génétiques classés selon quatre critères principaux [1] :

- le codage des individus (chromosome);
- l'indice de performance;
- la population initiale;
- l'ensemble des opérateurs de reproduction, de mutation et de sélection naturelle.

1.2.3 Codage des individus

Les AGs utilisés dans les travaux décrits dans ce mémoire, les RBCGAs (Real/Binary-Like Coded Genetic Algorithms), ont été développés par Sofiane Achiche et al. [2]. Il s'agît d'une méthode hybride combinant le codage binaire et réel. Le codage des sous-ensembles flous sur les prémisses et sur les conclusions se fait avec des nombres réels alors que le codage des règles se fait en binaire.

1.2.4 Indice de performance

Les RBCGAs utilisent une méthode multi-objectif pour converger vers la solution finale. L'indice de performance global ϕ est une combinaison de deux objectifs souvent contradictoires. D'une part, on cherche un modèle qui reproduit le mieux possible les données d'entraînement qui lui sont fournies. Cet objectif est caractérisé par l'indice de performance ϕ_1 qui représente l'erreur RMS évaluée sur l'échantillon d'entraînement.

$$\varphi_1 = \frac{L - \Delta_{\text{RMS}}}{L} \times 100 \tag{1.7}$$

où L est l'étendue totale des données d'entraînement et Δ_{RMS} est défini par :

$$\Delta_{\text{RMS}} = \sqrt{\sum_{i=1}^{N} \frac{(\text{RBCGA}_{\text{sortie}} - \text{données}_{\text{sortie}})^2}{N}} \tag{1.8}$$

où N est la taille de l'échantillon d'entraînement.

D'autre part, on cherche à simplifier les BCFs afin d'obtenir des modèles avec une meilleure capacité de généralisation. La complexité d'une BCF, évaluée par le nombre de règles actives, est représentée par l'indice de performance ϕ_2.

Ainsi, l'indice de performance global est défini par une somme pondérée de ϕ_1 et de ϕ_2 basée sur le poids ω_0 associé à ϕ_1.

$$\phi = \omega_0\phi_1 + (1 - \omega_0)\phi_2 \tag{1.9}$$

1.2.5 Population initiale

La population initiale est créée de manière aléatoire. Tout d'abord, les bornes pour chaque prémisse et pour chaque conclusion sont définies selon les limites supérieures et inférieures des données d'entraînement. Ensuite, pour chaque prémisse et conclusion de chaque individu, un nombre prédéfini de sous-ensembles flous est créé de manière aléatoire à l'intérieur de ces limites. Pour ce qui est de la taille de la population, elle peut être définie par l'usager. En général, on règle la taille de la population à 100 individus.

1.2.6 Opérateurs

1.2.6.1 Reproduction

La reproduction des individus s'effectue par simple croisement des chromosomes des meilleurs individus, basé sur leur habilité à survivre à la sélection naturelle. Ainsi, deux individus (un père et une mère) sont choisis aléatoirement parmi les meilleurs et leurs chromosomes sont croisés afin d'obtenir les chromosomes des enfants (une fille et un garçon) tel qu'illustré à la figure 1.8. Ceci est répété jusqu'à l'obtention du double de la population initiale.

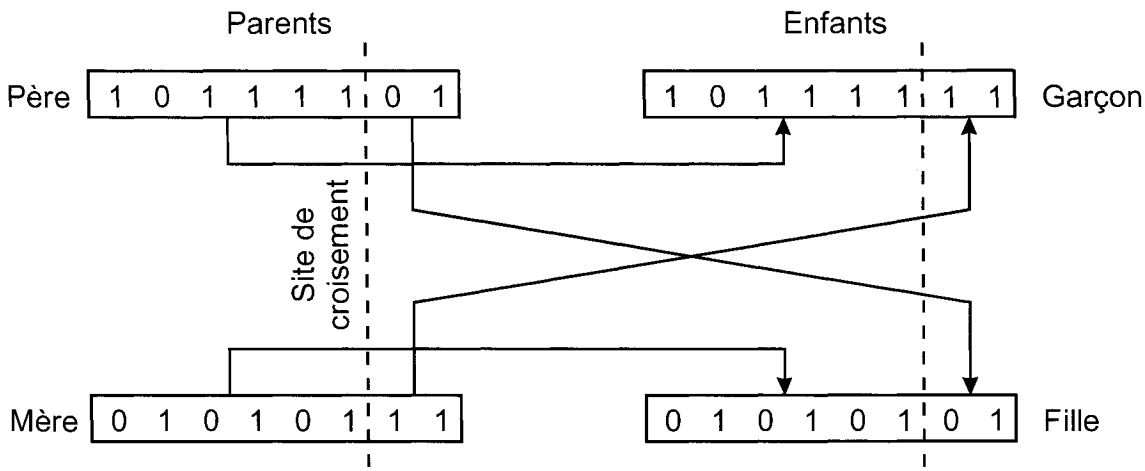

Figure 1.8 – Reproduction par croisement des chromosomes

1.2.6.2 Mutation

La mutation consiste à inverser un gène d'un chromosome choisi aléatoirement. La mutation permet d'explorer des solutions nouvelles et potentiellement meilleures en empêchant la convergence prématurée. La figure 1.9 illustre ce mécanisme.

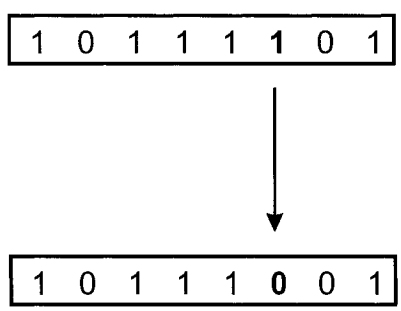

Figure 1.9 – Opération de mutation

1.2.6.3 Sélection naturelle

La sélection naturelle consiste à éliminer les individus les plus faibles en se basant sur l'indice de performance énoncé dans la section 1.2.4. Le mécanisme de sélection naturelle appliqué dans les RBCGAs a été grandement simplifié par rapport au

mécanisme naturel. Notamment, on ne considère pas l'âge des individus, ce qui permet à un individu de rester dans la population active durant toutes les générations. Également, la taille de la population reste constante, ainsi, à chaque génération, la population double après reproduction et revient à sa taille initiale après élimination de la moitié la plus faible.

La figure 1.10 illustre le fonctionnement de l'algorithme dans son ensemble.

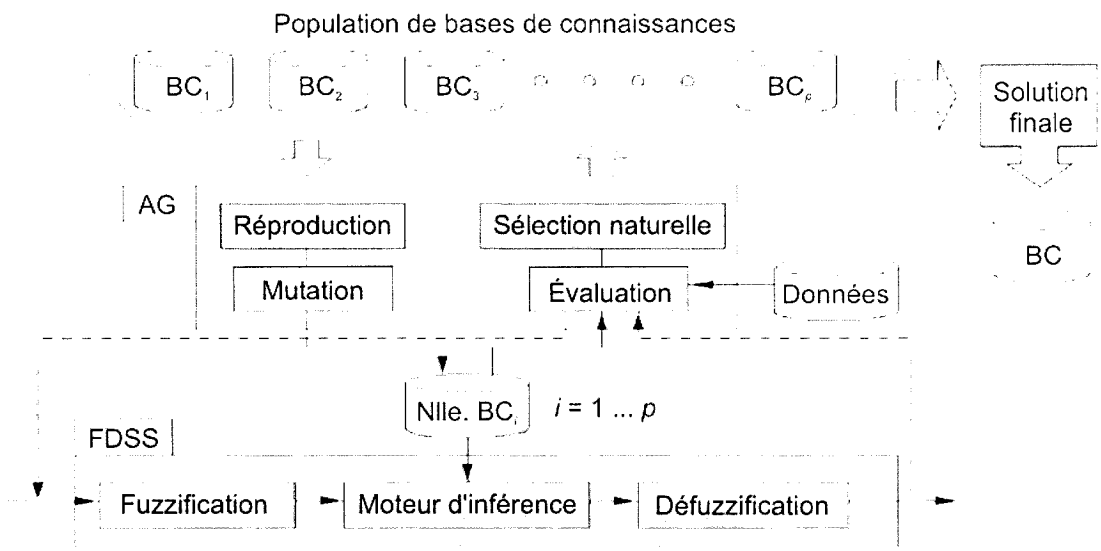

Figure 1.10 – Vue d'ensemble du fonctionnement des algorithmes génétiques

1.3 Algorithmes de clustering

1.3.1 Historique

Les biologistes et les zoologues étaient les premiers à s'intéresser aux méthodes de clustering, domaine qu'ils appelaient taxonomie. Les travaux de classification de plantes et d'animaux de Carl Linnaeus publiés dans Genera Plantarum en 1737 sont probablement les mieux connus des premiers travaux sur les méthodes de clustering. Les

algorithmes de clustering sont maintenant utilisés dans une multitude de domaines allant de l'imagerie numérique au marketing.

L'algorithme de clustering le plus utilisé de nos jours, le k-means, a été développé dans les alentours de 1960. Il est difficile de retracer précisément ses origines car il en existe une quantité impressionnante de versions. Plus d'une centaine d'articles scientifiques mentionnent cet algorithme dans leurs résumés chaque année.

1.3.2 Notions générales

Il existe deux grandes familles d'algorithmes de clustering : les algorithmes hiérarchiques et les algorithmes à partitionnement. Trois algorithmes (un hiérarchique et deux à partitionnement) ont été implantés dans le logiciel *GFS Cluster* développé dans le cadre de ce mémoire.

1.3.2.1 Algorithmes de clustering hiérarchiques

Les algorithmes hiérarchiques créent une hiérarchie dans la classification des objets. Ainsi, selon le nombre de clusters que l'on veut obtenir, on n'a qu'à choisir un niveau approprié dans la hiérarchie. Ces algorithmes peuvent être subdivisés selon plusieurs critères.

Tout d'abord, ils peuvent être agglomératifs ou divisifs. Les algorithmes agglomératifs commencent en créant un cluster pour chaque objet et regroupent les clusters selon leur similarité jusqu'à obtenir un seul ou le nombre voulu de clusters. Les algorithmes divisifs, par contre, font le chemin inverse. Ils commencent avec tous les objets dans un

seul cluster et subdivisent successivement les clusters selon la distance entre les objets y contenus. Ces deux méthodes sont illustrées à la figure 1.11.

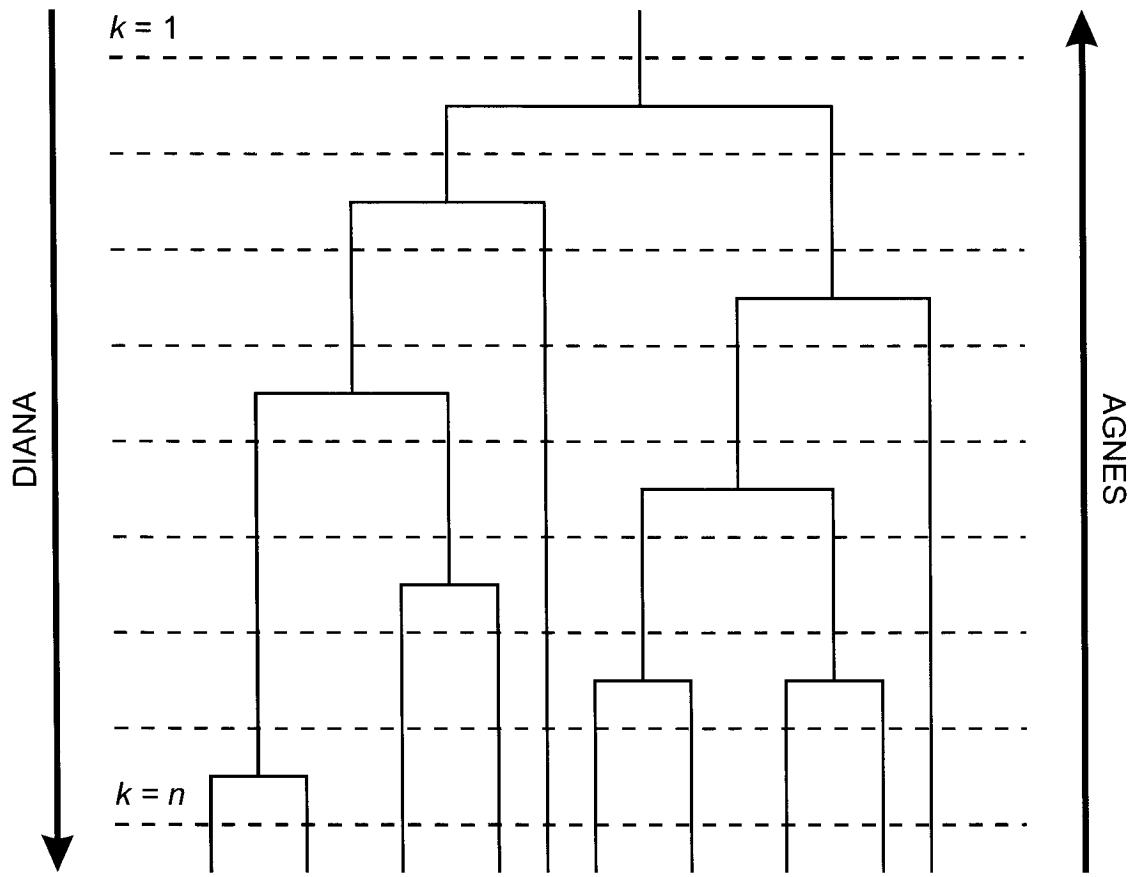

Figure 1.11 – Fonctionnement des algorithmes de clustering hiérarchiques

À la figure 1.11, DIANA désigne l'algorithme divisif (DIvisive ANAlysis), AGNES désigne l'algorithme agglomératif (AGglomerative NESting), k est le nombre de clusters et n est la taille de l'échantillon de données.

Le deuxième plus important critère de classification est la nature des liens entre les clusters qui peuvent être simples, multiples ou complets. Il s'agît ici du nombre d'objets qui sont considérés lors de l'évaluation de la similarité entre deux clusters. La plus

simple méthode consiste à calculer la distance euclidienne entre les centroïdes des deux clusters. Toutefois, il est également possible de considérer les points les plus distants ou les plus proches, voire même tous les points de chaque cluster.

Ces algorithmes ont le bénéfice d'être déterministes. En effet, pour un échantillon donné, les clusters trouvés seront toujours identiques. Également il est très facile d'évaluer le temps de calcul, car le nombre d'opérations est parfaitement défini.

Toutefois le principal inconvénient des ces algorithmes est leur rigidité. En effet, le chemin se fait toujours à sens unique et une fois qu'un lien est créé (dans le cas des algorithmes agglomératifs) il ne peut être défait. Il est donc très important de faire attention à bien choisir la fonction de similarité et le type de lien afin que l'agglomération se fasse de manière appropriée.

Dans ce mémoire, l'algorithme hiérarchique le plus populaire est étudié : l'algorithme AGNES. Il s'agît d'un algorithme agglomeratif à lien simple développé par Kaufman et Rousseeuw [22].

1.3.2.2 Algorithmes de clustering à partitionnement

Les algorithmes de partitionnement requièrent la connaissance à priori du nombre de clusters k à obtenir. À partir d'une distribution initiale quelconque dans les k clusters, l'algorithme cherche à optimiser l'agrégation des objets dans les clusters en suivant un certain critère d'optimisation. Suite à cette description, deux questions se posent instantanément. Premièrement, quelle est la distribution initiale et ensuite quel est le critère d'optimisation? À cela s'ajoute une multitude d'autres paramètres tels que la

fonction de distance/similarité, la mesure centrale des clusters et ainsi de suite. Les deux algorithmes de clustering à partitionnement les plus populaires, le *k*-means et le *k*-medians, sont étudiés dans ce mémoire et implantés dans le logiciel *GFS Cluster*.

Le fonctionnement de base d'un algorithme de partitionnement, tel que le *k*-means, est le suivant :

- générer une distribution initiale (voir ci-dessous);
- trouver le centroïde le plus proche pour chaque objet;
- déplacer l'objet dans un autre cluster, soit lorsqu'un cluster plus proche est trouvé, soit lorsque le cluster le plus proche est trouvé, selon la version de l'algorithme;
- recalculer les centroïdes soit lorsque tous les objets ont été parcourus, soit à chaque fois qu'un objet est déplacé, selon la version de l'algorithme;
- arrêter l'algorithme lorsque tous les objets sont dans le cluster le plus proche.

1.3.3 Distribution initiale

Il existe plusieurs choix de distribution initiale. Le logiciel *GFS Cluster* propose sept distributions différentes :

- aléatoire : un nombre égal (si possible) d'objets est placé aléatoirement dans chaque cluster;

- séquentielle : un nombre égal (si possible) d'objets est placé séquentiellement (selon l'ordre dans l'échantillon de données) dans chaque cluster;
- centroïdes aléatoires : k centroïdes sont définis alétoirement parmi l'échantillon de données et les objets sont placés dans les clusters dont les centroïdes sont les plus proches;
- centroïdes séquentiels : k centroïdes sont définis séquentiellement parmi l'échantillon de données (un pas est d'abord calculé afin de balayer l'échantillon au complet) et les objets sont placés dans les clusters dont les centroïdes sont les plus proches;
- centroïdes éloignés : pour chaque objet, on calcule la somme des distances à tous les autres objets. Le premier centroïde est défini par l'objet avec la plus grande somme des distances. Ensuite, pour tous les objets restants, la somme des distances au premier centroïde est calculée. Le second centroïde est encore une fois celui qui possède la plus grande somme des distances. Les centroïdes restants sont définis par les objets qui possèdent la plus grande somme des distances à tous les centroïdes déjà définis. Finalement, les objets restants sont placés dans les clusters dont les centroïdes sont les plus proches;
- MaxMin : pour chaque objet, on calcule la somme des distances à tous les autres objets. Le premier centroïde est défini par l'objet avec la plus grande somme des distances. Ensuite, pour tous les objets restants, la somme des distances au premier centroïde est calculée. Le second centroïde est encore une fois celui qui

possède la plus grande somme des distances. Les centroïdes restants sont définis par les objets qui possèdent la plus grande distance minimale aux centroïdes déjà définis. Finalement, les objets restants sont placés dans les clusters dont les centroïdes sont les plus proches;

- AGNES : la distribution initiale est obtenue par l'algorithme hiérarchique AGNES décrit à la section 1.3.2.1.

Les deux premières méthodes offrent l'avantage de placer un nombre égal d'objets dans chaque cluster, ce qui est très pratique dans le cas de données contenant des outliers car ces derniers sont « absorbés » dans les clusters contenant d'autres objets. Pour les autres méthodes, on peut toujours utiliser d'autres tactiques afin d'obtenir des distributions uniformes. Par exemple, on peut définir une capacité maximale de chaque cluster. De la même manière, on peut définir une capacité minimale, afin que certains clusters ne se vident pas jusqu'à ne contenir que des outliers. L'article contenu au chapitre 3 met particulièrement l'emphase sur la sensibilité des différentes techniques aux outliers et au bruit.

1.3.4 Critère d'optimisation

Le critère d'optimisation dans les algorithmes de partitionnement étudiés dans ce mémoire est l'erreur de reproduction (voir éq. 1.10).

$$E = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{c}_i\| \tag{1.10}$$

où E est l'erreur de reproduction, $\boldsymbol{x}_i$ sont les positions des objets de l'échantillon de données, $\boldsymbol{c}_i$ sont les positions des centroïdes des clusters auxquels les objets $\boldsymbol{x}_i$ appartiennent et n est la taille de l'échantillon.

1.3.5 Fonctions de distance/similarité

Les fonctions de distance/similarité font objet de recherches très approfondies dans le domaine de l'analyse de clusters. Celles implantées dans le logiciel *GFS Cluster* sont : Euclidienne, Minkowski [15], Canberra [28] et somme harmonique [10]. Des méthodes par corrélation telles que celles de Pearson [25] et de Kendall's [23] sont également implantées. Elles offrent une robustesse beaucoup plus importante contre les outliers mais exigent des calculs considérables. Les détails de ces méthodes sont omis de ce mémoire car seulement la distance Euclidienne est étudiée. Les références fournies offrent toutefois des descriptions et des analyses approfondies.

1.3.6 Mesure centrale

Pour ce qui est de la mesure centrale, deux choix sont privilégiés : la moyenne (k-means) et la médiane (k-medians). La médiane requiert plus de calculs, surtout lorsqu'on se trouve dans une dimension supérieure à deux mais offre une robustesse accrue aux outliers. Ceci est dû au fait que la distance d'un outlier potentiel n'influence pas la position de la médiane. Un autre avantage des médianes réside dans le fait que les centroïdes sont presque toujours des objets de l'échantillon initial (sauf dans le cas de médianes multiples où on calcule généralement la moyenne des médianes).

1.4 Intégration des trois domaines de l'intelligence artificielle

Le principal intérêt de ce mémoire réside non dans l'utilisation d'un des trois domaines de l'intelligence artificielle exposés dans cette section, mais dans la façon que ceux-ci sont intégrés afin de répondre aux objectifs ciblés.

1.4.1 Intégration de la logique floue et des algorithmes génétiques

Tout d'abord l'intégration de la logique floue et des algorithmes génétiques dans les RBCGA [2] a principalement fait objet des recherches de Sofiane Achiche. Ces algorithmes créent tout d'abord des BCFs de manière aléatoire afin de les optimiser ensuite grâce aux mécanismes des algorithmes génétiques décrits dans la section 1.2. Le tableau 1.3 présente les associations entre les éléments des AGs et ceux de la logique floue.

Tableau 1.3 – Correspondance entre les éléments des algorithmes génétiques et ceux de la logique floue

Algorithmes génétiques	Logique floue
Population	Ensemble de BCFs
Individu	Une BCF
Code génétique	Ensemble (1) de sous ensembles flous des prémisses et des conclusions et (2) de règles
Performance	Habilité d'une BCF à reproduire l'ensemble de données d'apprentissage ou à généraliser sur un ensemble de données de validation

1.4.2 Intégration des algorithmes de clustering

L'intégration des algorithmes génétiques, de la logique floue et des algorithmes de clustering se fait à deux niveaux différents. Tout d'abord, les données sont prétraitées

afin de réduire leur taille ainsi que la quantité de bruit et de données suspectes (c.f. Chapitre 3). Dans ce cas-ci, l'intégration se fait de manière séquentielle et il n'y a pas de véritable travail coopératif entre les algorithmes employés. Le travail coopératif commence lorsqu'on utilise les algorithmes de clustering afin de déterminer le nombre de sous-ensembles flous dans chaque prémisse (c.f. Chapitre 5).

L'essentiel du travail des algorithmes de clustering réside dans l'aide à la création de la population initiale. Tout d'abord l'ensemble de données d'apprentissage est analysé avec des techniques de détermination du nombre de clusters (c.f. Chapitre 4). Pour cela, les différentes prémisses et conclusions sont découplées et les données sont traitées comme plusieurs ensembles de données à une dimension. Chacun de ces ensembles de données est ensuite traité séparément et le nombre de clusters pour chacun d'entre eux est trouvé. Les clusters trouvés sont également utilisés dans la création des sous-ensembles flous : les sommets de ceux-ci sont placés aux positions déterminées par les centroïdes des clusters.

Ainsi, la supposition sur laquelle est basée l'intégration des algorithmes de clustering avec les algorithmes génétiques et la logique floue est que la complexité d'un problème (nombre de règles de la BCF) est reliée au nombre de clusters dans l'ensemble d'entraînement. Cette supposition est soutenue par les résultats de validation présentés dans le chapitre 5 et obtenus sur des ensembles de données synthétiques ainsi qu'expérimentales.

La figure 1.12 présente l'analogie entre les sous-ensembles flous d'un problème à une prémisse et une conclusion et les positions des centroïdes des clusters.

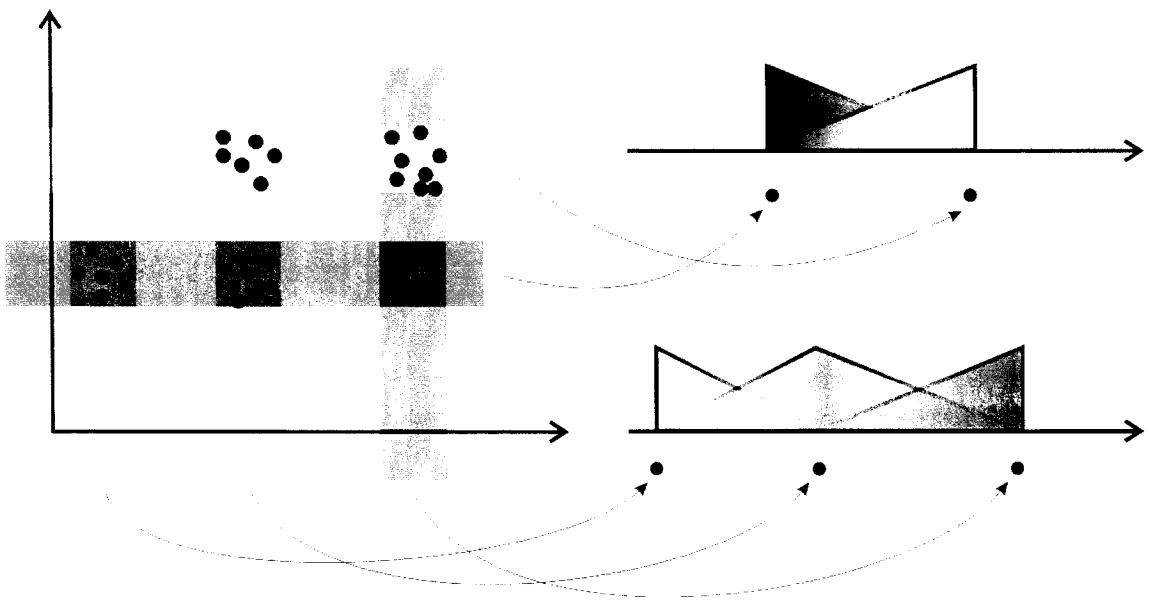

Figure 1.12 – Analogie entre les sous-ensembles flous et les clusters

CHAPITRE 2

SYNTHÈSE DES ARTICLES

Le contenu de ce chapitre résume en langue française les articles de journaux présentés aux chapitres trois et cinq. Le premier article porte sur le prétraitement des données brutes et est intitulé « Influence of Clustering Pre-processing on genetically generated Fuzzy Knowledge Bases » alors que le deuxième article porte sur la prédéfinition des nombres de sous-ensembles flous et est intitulé « Predefining Numbers of Fuzzy Sets For genetically generated Fuzzy Knowledge Bases using Clustering Techniques. »

2.1 Prétraitement de données brutes

La génération automatique de bases de connaissances floues à l'aide de techniques telles que les algorithmes génétiques a tendance à être fortement dépendante de la qualité des données d'apprentissage. Tout d'abord, des échantillons de données trop grands peuvent mener à d'importantes pertes de temps, alors que des échantillons de données plus petits pourraient décrire le problème tout aussi bien. Pour palier à ce problème, les données sont compressées en réduisant la quantité d'informations similaires et redondantes.

Trois algorithmes de clustering ont été testés par rapport à leurs capacités de compression et de filtrage de bruit et d'outliers. Tout d'abord, l'algorithme AGNES (AGglomerative NESting) a été sélectionné parmi les algorithmes hiérarchiques. Cet algorithme crée une hiérarchie de distributions en partant d'une distribution où chaque

objet constitue un cluster séparé. Ensuite, on procède en combinant les clusters les plus proches jusqu'à ce que le nombre de clusters voulu soit atteint. L'algorithme AGNES est déterministe et il est facile de prévoir son temps d'exécution avec exactitude. Toutefois, il soufre de plusieurs inconvénients dus à sa rigidité.

D'autre part, deux algorithmes à partitionnement ont été choisis : le *k*-means et le *k*-medoids. L'algorithme *k*-means part avec une distribution quelconque et cherche à l'optimiser en analysant son voisinage (distributions obtenues par échange d'un objet entre deux clusters). Il utilise le centre de masse comme mesure centrale, alors que le *k*-medoids utilise plutôt la médiane. Les deux algorithmes ont été testés sous deux formes différentes : Meilleur Possible (MP) et Premier Meilleur (PM). La différence entre ces deux versions est que la version PM est une version plus vorace : plutôt que chercher la meilleure alternative dans le voisinage, elle accepte la première qui offre un avantage par rapport à la situation présente. Les algorithmes à partitionnement sont plus flexibles que les algorithmes hiérarchiques, car les objets peuvent être facilement échangés entre les clusters. Dans le cas de AGNES par contre, une fois un lien entre deux clusters créé, celui-ci ne peut plus être défait.

Les trois algorithmes ont été testés sur un ensemble de données synthétiques définies par :

$$f(x, y) = 3x^2 y - y^3 \tag{2.1}$$

dans l'intervalle [0; 1] × [0; 1].

Plusieurs ensembles de données avec des bruits Gaussiens et des quantités d'outliers variables ont été générés aléatoirement. Pour chaque combinaison des deux variables testée, dix échantillons de 1000 objets ont été générés et à partir des résultats obtenus, les moyennes et les écarts types ont été calculés. Les données ont ensuite été compressées à 5% de leur taille initiale (50 objets).

Pour ce qui est de la réduction de temps d'exécution, on a obtenu des gains de 94% avec les algorithmes *k*-means (PM et MP), de 75% avec *k*-medoids (MP) et de 47% avec AGNES.

Du point de vue de bruit, les algorithmes testés n'ont pas présenté d'amélioration significative. En général, les AGs sont des algorithmes très robustes contre le bruit et l'usage des algorithmes clustering ne fait que diminuer la quantité d'informations contenues dans l'échantillon de données brutes et donc baisser la qualité des BCFs obtenues. L'algorithme *k*-medoids présente un intérêt du point de vue de la stabilité seulement pour des bruits Gaussiens dont l'écart type dépasse 0.4, mais en moyenne les résultats sont toujours inférieurs à ceux obtenus sans usage des algorithmes de clustering.

C'est lors du filtrage des outliers que le véritable intérêt des algorithmes de clustering se fait sentir. En effet, l'algorithme *k*-medoids réussit à éliminer les outliers de l'échantillon d'apprentissage jusqu'à des probabilités d'occurrence dépassant les 20%. Le gain en performance atteint 2.4% pour un échantillon contenant 10% d'outliers et 0.8% pour 20% d'outliers. De plus, les résultats sont plus stables. Les autres algorithmes par contre

ne sont pas aussi robustes contre les outliers (voir 1.3.6) surtout AGNES qui produit des BCFs avec un critère de performance aussi bas que 42.69%.

2.2 Prédéfinition des nombres de sous-ensembles flous

Le second article adresse un problème bien connu entourant la génération automatique de bases de connaissances floues à l'aide des algorithmes génétiques, soit la recherche d'un nombre de sous-ensembles flous optimal pour chaque prémisse. Certains algorithmes génétiques emploient une méthode multi-objectif combinant la minimisation de l'erreur et la simplification du modèle. Ce travail présente des solutions alternatives, basées sur l'analyse des clusters et sur les indices de validation pour le nombre de clusters afin de prédéfinir les nombres de sous-ensembles flous.

Deux indices de validation : Silhouette et Hartigan ainsi que la combinaison de Hartigan et de la méthode multi-objectif sont comparés à la méthode multi-objectif. Les détails de ces méthodes sont présentés au quatrième chapitre.

L'algorithme d'apprentissage global a été divisé en deux étapes. Tout d'abord, les données sont analysées par le logiciel de clustering afin de déterminer les nombres de clusters optimaux (k_{opt}) pour l'échantillon de données traité. Chaque dimension est traitée séparément et une fois les distributions optimales obtenues, les nombres de clusters sont directement utilisés afin de créer les nombres de sous-ensembles flous appropriés pour chaque prémisse. De plus, les positions des clusters sont utilisées afin de placer les sommets des sous-ensembles flous aux endroits appropriés.

La deuxième étape consiste à exécuter les algorithmes génétiques avec certains paramètres prédéfinis. En effet, le mécanisme de mutation est presque désactivé ($p_m = 0.01$) afin de diminuer la probabilité que deux sous-ensembles flous se superposent. Le mécanisme de réduction de sous-ensembles flous, quant à lui, est totalement désactivé ($p_r = 0.0$) car le nombre optimal de sous-ensembles flous est supposé obtenu par le clustering.

La validation à travers des données synthétiques (éq. 2.1 dans l'intervalle [-3; 3] × [-3; 3]) et expérimentales montre des améliorations considérables dans la précision de la prédiction effectuée avec l'usage des nombres de sous-ensembles flous prédéfinis. Sur l'ensemble de données synthétique, une amélioration moyenne de 15.4% (maximale de 21%) a été obtenue avec la technique Silhouette.

Pour ce qui est de l'ensemble de données expérimentales, il s'agît d'une application de prédiction de l'usure des outils. Quatre signaux ont été prélevés (force de coupe, force d'avance, force de refoulement et émissions acoustiques) et à partir de ces signaux plusieurs caractéristiques ont été calculées. L'usure pour chaque outil a été approximée comme la portion de sa durée de vie totale déjà écoulée. En tout 9 outils ont été usés sur lesquels un total de 94 opérations a été effectué.

Une amélioration maximale de 28% a été obtenue sur la performance des BCFs avec l'usage de l'indice Silhouette. De plus, l'usage de nombres de sous-ensembles flous prédéfinis permet de contourner la nécessité de préanalyser les données d'entraînement.

Une complexité optimale peut être déterminée automatiquement par le processus de clustering, enlevant ainsi la nécessité de supervision humaine.

CHAPITRE 3

INFLUENCE OF CLUSTERING PRE-PROCESSING ON GENETICALLY GENERATED FUZZY KNOWLEDGE BASES

Publié à *Computer Assisted Mechanics and Engineering Sciences*, Vol. 12-2, Warsaw, Poland, 2005.

Abstract

Automatic knowledge base generation using techniques such as genetic algorithms tend to be highly dependent on the quality and size of the learning data. First of all, large data sets can lead to unnecessary time loss, when smaller data sets could describe the problem as well. Second of all, the presence of noise and outliers can cause the learning algorithm to degenerate. Clustering techniques allow compressing and filtering the data, thus making the generation of fuzzy knowledge bases faster and more accurate. Different clustering algorithms are compared and the validation of the results through a theoretical 3D surface shows that when compressing the data to 5% of its original size, clustering algorithms accelerate the learning process by up to 94%. Moreover, when the learning data contains noise and/or a large amount of outliers, clustering algorithms can make the results more stable and improve the fitness of the obtained FKBs.

3.1 Introduction

Machine learning applications such as Genetic Algorithms (GA) [1] allow automatic building of knowledge bases on phenomena without any theoretical knowledge nor expert assistance, based only on a set of learning data. The quality of the obtained knowledge bases is highly dependant on that of the data set that is presented to the learning algorithm. Hence, learning data sets containing a large amount of noise and/or outliers will result in a loss of fitness of the knowledge base, since the learning algorithm will try to model the noise/outliers instead of the real phenomenon. Moreover, lack of knowledge about the phenomenon often leads to "over-representation," meaning collecting a very large data set when a much smaller one would be sufficient. Such large data sets will inevitably cause a considerable time loss which can be problematic for applications that need a fast and accurate response.

This paper presents a study of the robustness in terms of Gaussian noise and outlier probability of the Real Binary-Like Coded Genetic Algorithm (RBCGA) used to generate Fuzzy Knowledge Bases (FKB) and proposes a technique based on clustering algorithms to improve the fitness of the obtained FKBs and reduce the time needed to perform the learning process. First, short presentations of the Fuzzy Decision Support System (FDSS) and RBCGAs describe the main aspects and specificities of these techniques and the software that was used. A description of the clustering techniques and validation results follow in the next two sections.

3.2 Fuzzy Decision Support System

A rule-based approach to decision making using fuzzy logic techniques may consider imprecise vague language as a set of rules linking a finite number of conclusions. The knowledge base of such systems consists of two components: a linguistic terms base and a fuzzy rules base [3]. The former is divided into two parts: the fuzzy premises (or inputs) and the fuzzy conclusions (or outputs).

3.2.1 Theoretical specificities

In this paper we consider FKBs of multiple inputs and one single output (MISO). Moreover, we consider only general overlapping triangular fuzzy sets on the premises and sharp-symmetric triangular fuzzy sets on the conclusion. The representation of such imprecise knowledge by means of fuzzy linguistic terms makes it possible to carry out quantitative processing in the course of inference that is used for handling uncertain (imprecise) knowledge. This is often called approximate reasoning [21]. This knowledge, expressed by (k = 1, 2, …, K) finite heuristic fuzzy rules of the type MISO, may be written in the form:

$$R^k_{MISO}: \text{if } x_1 \text{ is } X^k_1 \text{ and } x_2 \text{ is } X^k_2 \text{ and} \ldots \text{and } x_N \text{ is } X^k_N \text{ then y is } Y_k, \qquad (3.1)$$

where $\{X^k_i\}^N_{i=1}$ denote values of linguistic variables $\{x_i\}^N_{i=1}$ (conditions) defined in the following universe of discourse $\{X_i\}^N_{i=1}$; and Y^k stands for the value of the independent linguistic variable y (conclusion) in the universe of discourse Y. The global relation aggregating all rules from j = 1 to J is given as

$$R = also_{k=1}^{K}\left(R_{\mathrm{MISO}}^{k}\right). \tag{3.2}$$

where the sentence connective *also* denotes any t- or s-norm (e.g., MIN ($\wedge$) or MAX ($\vee$) operators) or averages. For a given set of fuzzy inputs $\{X_i'\}_1^N$ (or observations), the fuzzy output Y' (or conclusion) may be expressed symbolically as:

$$Y' = \left(X_1', X_2', \ldots, X_N'\right) \circ R, \tag{3.3}$$

where $\circ$ denotes a compositional rule of inference (CRI), e.g., the MAX-MIN (MAX-$\wedge$) or MAX-PROD (MAX-*). Alternatively, the CRI of Eq. 3.3 is easily computed as

$$Y' = X_N' \circ \ldots \circ \left(X_2' \circ \left(X_1' \circ R\right)\right). \tag{3.4}$$

The CRI mechanisms allow us to obtain different conclusions represented as the membership function Y'. In FDSS Fuzzy-Flou, there are three defuzzification methods: the centre of gravity (COG); the mean of maxima (MOM); and the height method. All the results presented in this paper are obtained using the Σ-MAX*-*-* CRI (MAX-PROD) and COG as defuzzification.

3.2.2 FDSS Learning Paradigm

In general, FDSS requires a knowledge base in order to support the decision-making process of end-users. The FKB can be created manually by a human expert or automatically learned from a set of sampled data. In this paper the automatic learning process of the FDSS knowledge base is automatic. The learning process is aimed at producing knowledge bases that are manageable by either a human expert or a computer. The FKBs must accurately reproduce the set of learned data, while interpolating or

extrapolating fair conclusions in other situations. A minimalist approach is implemented through an automatic reduction of fuzzy rules and sets on the premises, whenever the approximation error is not penalized too greatly by this reduction.

Fig. 3.1 shows a screen printout of the FDSS Fuzzy-Flou software used as a validation tool for the genetically generated FKBs. It was developed at École Polytechnique de Montréal (Canada) and the Silesian University of Technology (Poland). For more information on the Fuzzy-Flou software, please refer to [3].

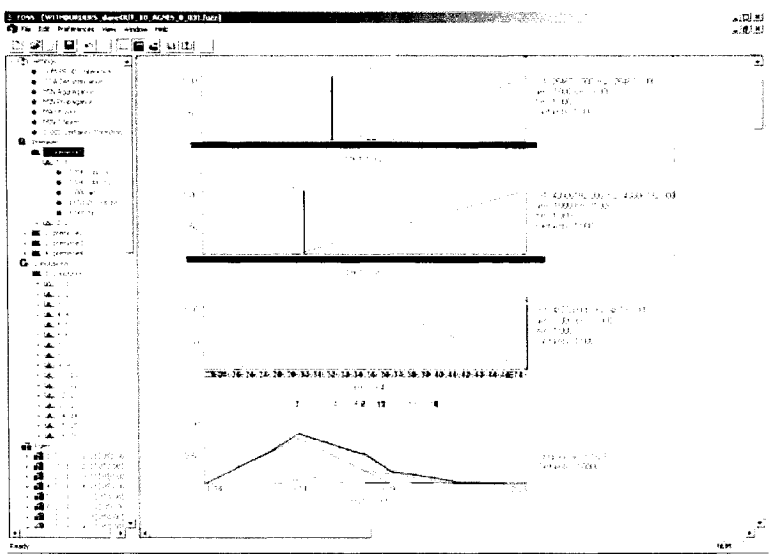

Figure 3.1 – Graphic Interface of the FDSS Fuzzy-Flou

3.3 Automatic Generation of FKBs

GAs are powerful stochastic optimization techniques based on the analogy of the mechanics of biological genetics and imitate the Darwinian survival of the fittest approach [1]. As shown in Fig. 3.2, each individual of a population is a potential FDSS Fuzzy-Flou knowledge base. Fig. 3.2 presents the encoding/decoding scheme as well as

the four basic operations, i.e.: reproduction, mutation, evaluation and natural selection, of the developed GA learning software. This method uses iterative improvement of individuals at each generation to converge toward multiple optima simultaneously. When the number of unknown parameters increases, GAs exhibit only a polynomial increase of the complexity [6][16], while the other optimization techniques show an exponential increase. The RBCGA developed by the authors [1] is a combination of a real coded genetic algorithm (RCGA) and a binary coded genetic algorithm (BCGA).

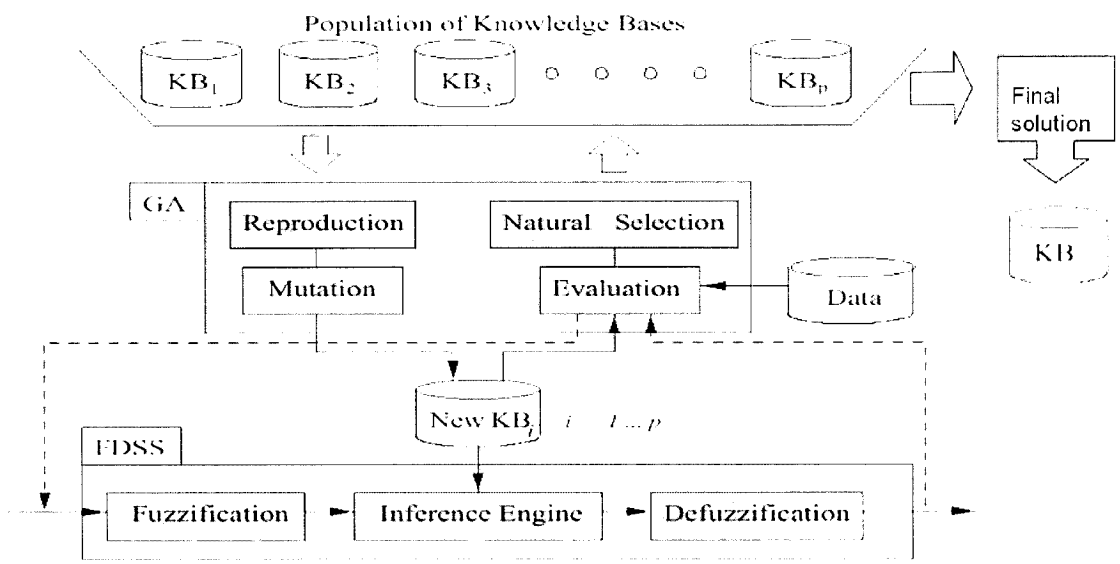

Figure 3.2 – Genetic learning paradigm

3.3.1 Coding

The *genotype* of an FKB is the coding of its parameters into chromosomes and corresponds to several independent sets of real numbers and a set of integers. The genotype contains the following items:

- Input/Output Premises: A set of real numbers; for the sake of coding simplicity, only non-symmetrical-overlapping triangular fuzzy sets are considered for the premises and symmetrical triangular fuzzy sets for the conclusion.
- Fuzzy Rules: A set of integer numbers; the genotype of fuzzy rules contains information about all the possible combinations of connecting one fuzzy set on each premise to a fuzzy set on the conclusion.

3.3.2 Multi-Crossover Mechanisms

The evolution of a population of FKBs at each generation is achieved by the reproduction of the "best" individuals, based on their abilities to survive natural selection. Reproduction is performed by crossover of the genotype of the parents to obtain the genotype of an offspring, using a multi-crossover which is composed of a premises/conclusion crossover and a fuzzy rules crossover. These mechanisms are governed by the initiating probability p_c.

3.3.2.1 Premises/Conclusion Crossover

The mechanism used is called blending crossover α (BLX-α) [19], where α determines the exploitation/exploration level of the offspring (see Fig. 3.3). The parameter α is set to 1.0 for the first third of the generations (exploration) to 0.5 for the second third (relaxed exploitation) and finally to 0.1 for the last third of the evolution (exploitation). These changes in exploitation/exploration balance, help avoiding premature convergence [2].

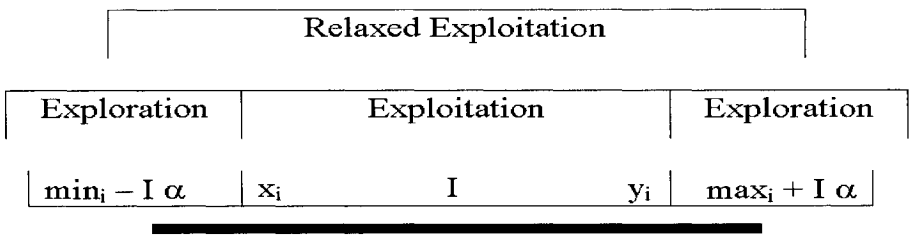

Figure 3.3 – Blended crossover α (BLX-α)

3.3.2.2 Fuzzy Rules Crossover

Since the part of the genotype representing the fuzzy rule base is composed of integer numbers, the crossover on this part of the genotype is done by a simple crossover. The operation is performed by exchanging the end part of the sets (containing the fuzzy rules) of the parents at a randomly selected crossover site, as shown in Fig. 3.4.

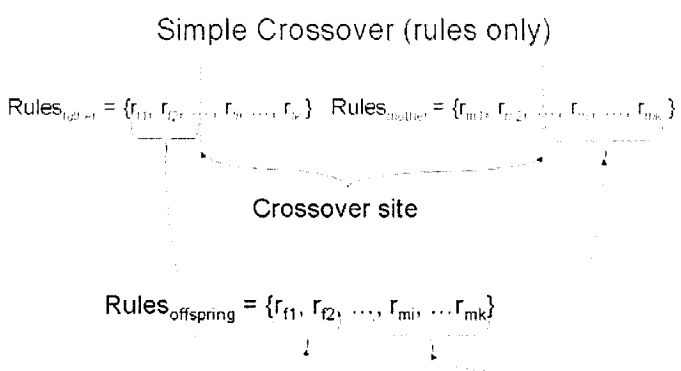

Figure 3.4 – Simple crossover

3.3.3 Fuzzy Set Reducer

This mechanism is set to reduce the complexity of the FKBs by selecting a summit on each premise and erasing it from the respective sets together with the corresponding fuzzy rules. This mechanism is governed by the initiating probability p_r.

3.3.4 Mutation

Mutation is a random alteration of a new member's genotype in the population. Mutation makes it possible to seek completely new solutions without any control on the direction, as opposed to gradient-based optimization techniques. The probability p_m governs the occurrence of this mechanism. In this paper, uniform mutation [11], similar to BLX-α with $\alpha = 0$, is applied.

3.3.5 Natural Selection

Natural selection is performed on the population by keeping the "most promising" individuals, based on their fitness. The first generation begins with P FKBs and the same number is generated by crossover and mutation. To keep the population constant, we apply natural selection on the $2P$ FKBs by ordering them according to the performance criterion and keeping the P first FKBs. In this paper the population size is set to 100 and the number of generations is set to 500.

3.3.6 Performance Criterion of the RBCGA

The performance criterion allows one to compute the rating of each FKB. This performance rating is used by the RBCGA in order to perform the natural selection. Here, the performance criterion is the accuracy level of an FKB in reproducing the outputs of the learning data (approximation error). The approximation error Δ_{RMS} is measured using the RMS error method:

$$\Delta_{\mathrm{RMS}} = \sqrt{\sum_{i=1}^{N} \frac{(\mathrm{RBCGA}_{\mathrm{output}} - \mathrm{data}_{\mathrm{output}})^2}{n}} \tag{3.5}$$

where, n represents the size of the learning data. The fitness value is evaluated as a percentage of L, the output length base of the conclusion, i.e.,

$$\phi_{\mathrm{RMS}} = \frac{L - \Delta_{\mathrm{RMS}}}{L} \times 100 \tag{3.6}$$

3.4 Clustering

"Clustering consists of partitioning a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often similarity or proximity for some defined distance measure." [20] Among all the various clustering techniques, the three most popular were selected: one hierarchical method, AGglomerative NESting (AGNES) [13], and two partitioning methods (*k*-means [17] and *k*-medoids [13]).

3.4.1 The Hierarchical Algorithm

Since most of the hierarchical methods are relatively time consuming, only the simplest one, AGNES, has been tested. It starts with each object in its own cluster and merges the clusters closest to each other, thus creating larger clusters. A step by step description of the algorithm follows.

- Place each object in its own cluster;
- Calculate the distances between each pair of clusters;
- Merge the pair of clusters with the smallest distance;

- Repeat steps 2 and 3 until the desired number of clusters is attained.

The AGNES algorithm is deterministic. Its execution time depends only on the size of the data and the number of clusters desired. It does not depend on the way the data is ordered. On the other hand, it suffers from several handicaps, mostly due to its rigidity. The most important one is the fact that once two clusters are combined, that combination cannot be undone. Clusters cannot be separated and objects cannot be swapped between clusters. This isn't the case of the partitioning algorithms.

3.4.2 The Partitioning Algorithms

The partitioning algorithms start with an arbitrary distribution in the desired number of clusters and then examine its neighbourhood (similar distributions with one object swapped between two clusters). The Best Possible (BP) versions search for the best similar distribution whereas the First Better (FB) versions perform the swap as soon as a better one is found. Finally, the algorithms stop when the distribution is better than its entire neighbourhood. The two partitioning methods, *k*-means and *k*-medoids, differ only by the central measure of the clusters. Their step by step description is as follows:

- Create as many clusters as needed
- Place about as many objects in each cluster
- Compute the centroids/medoids of each cluster
- For each object search for clusters with centroids/medoids closer to the object than its actual cluster's centroid/medoid

- Perform the swap when
    - the closest cluster is found (BP)
    - a closer cluster is found (FB)
- Stop when each object's distance to its own cluster is minimum

The algorithm converges towards near-optimal solutions, meaning that it does not necessarily find the global optimum of the chosen objective function, i.e. minimum sum of distances from the objects to their respective cluster centers. In all of its variations, the algorithms only find a local optimum.

3.4.3 Central Measure

In the case of the partitioning methods, two different central measures were tested, medoids and means. The medoids require more heavy computations than the means but they also offer a strongly improved robustness against outliers, since, contrarily to the mean, the distance of an outlier to the rest of the objects has no effect on the value of the medoid.

3.4.4 Distance/Similarity Functions

All of the algorithms have been tested with the Euclidean distance function in order to evaluate the similarity between two objects or clusters. In order not to "penalize" variables with small variances, values for each variable can be normalized with mean

equal to 0 and variance equal to 1. Also, a much simpler technique is to normalize within the range from 0 to 1. Nevertheless, this approach is very sensitive to outliers.

Other distance functions such as Minkowski [9], Canberra [18] and Harmonically Summed [12] have been implemented but they showed no serious improvement in the obtained results. Also the correlation ranks such as Pearson's [15] and Kendall's [14] have been implemented. They offer a much improved robustness against outliers but the necessity of computing the correlations at each step represents a very important time cost.

3.4.5 Optimal Number of Clusters

The Calinski-Harabasz pseudo-F (CHpF) [5] (see eq. 3.7) statistic has been implemented and can be used to obtain an estimate of the optimal number of clusters for each individual distribution.

$$CHpF = \frac{\frac{B_k - W_k}{W_k}}{\frac{k-1}{n-k}} \tag{3.7}$$

where:

- n is the number of objects
- k is the number of clusters
- B_k is the sum of squared distances from the clusters centroids to the overall data centroid

- W_k is the sum of squared distances from the objects to their respective clusters' centroids

One way to obtain the optimal number of clusters is to run a hierarchical algorithm and compute the CHpF statistic at each step. Once the optimal CHpF statistic is obtained, a partitioning method can be run to optimize the distribution at the appropriate level in the hierarchy.

In the tests described in the following section this technique has not been used systematically, since it would have been necessary to perform it for each individual data set and the time required to perform the necessary operations would have been too important. It was only used to roughly estimate a global compromise between the compression ratio and the optimal number of clusters for all the data sets.

3.5 Validation Results

Three different sets of tests have been performed. The first one only shows the time reduction when either one of the three algorithms are used. The second one shows the robustness of the GAs and clustering algorithms in terms of noise and the third one in terms of outliers. In all of these tests a theoretical 3D surface has been used (see eq. 3.8).

$$f(x,y) = 3x^2 y - y^3 \tag{3.8}$$

For the purpose of the studies presented in this section a clustering module was developed by the authors and used to obtain all the results. Figure 3.5 shows a screen printout of the aforementioned module.

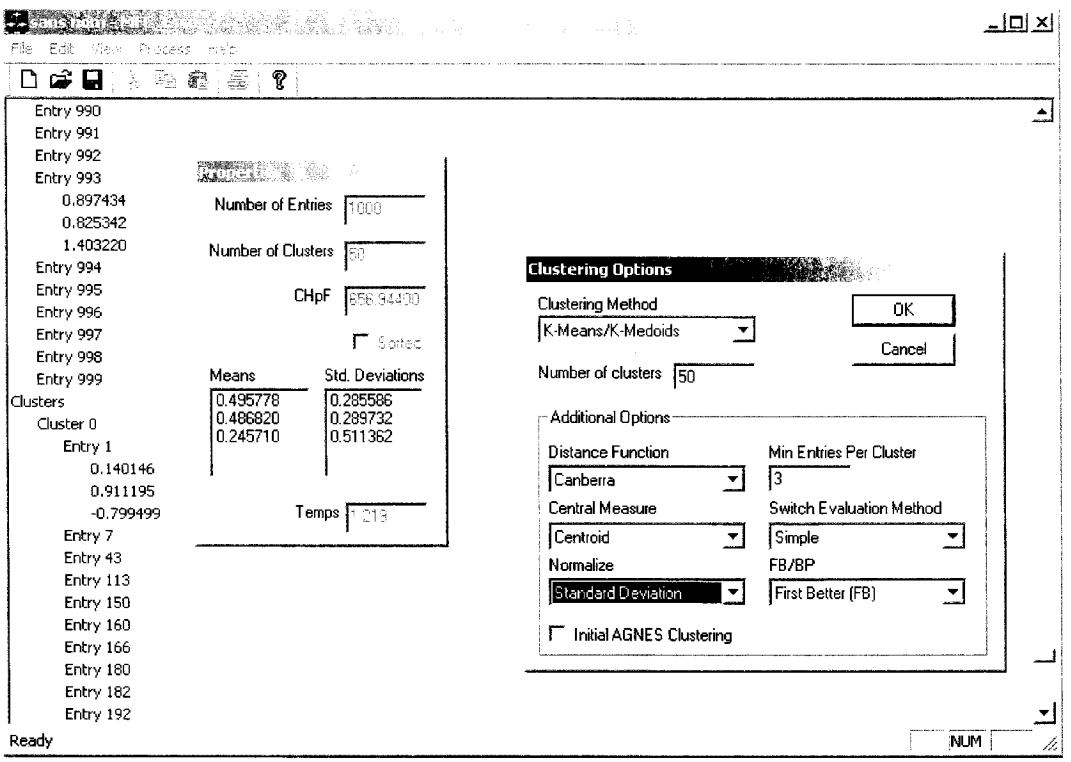

Figure 3.5 – Graphic Interface of the Clustering module

Numerous data sets with variable Gaussian noise and outlier probability have been generated randomly. Outliers have been generated by replacing a random object with a random value in the interval of two times the total range of the data set. Data sets for each parameter combination have been generated ten times and the clustering and learning processes have been run separately. The final results presented in this paper are means (μ) of each set of ten runs. Standard deviations (σ) are also included in the tables to show the stability (or the lack of it) of the different algorithms.

The size of the initial data sets was chosen in order to obtain reasonably pronounced execution times. Data sets of 1000 objects gave palpable results. As for the size of the compressed data sets, it was set to 50 (see section 3.4.5). The performance of each FKB

is obtained by the method described in section 3.3.6 except that the $data_{output}$ in eq. 3.5 is replaced by the actual theoretical values consisting of a uniformly distributed set of 1000 objects. Fig. 3.6, Fig. 3.7 and Fig. 3.8 present 3D and colormap representations of the theoretical data set and of the data set obtained for Gaussian noise with 0.1 standard deviation and 20% outlier probability. The outliers in Fig. 3.8 have been isolated from the rest of the learning data and represented as crosses in order to highlight their large quantity.

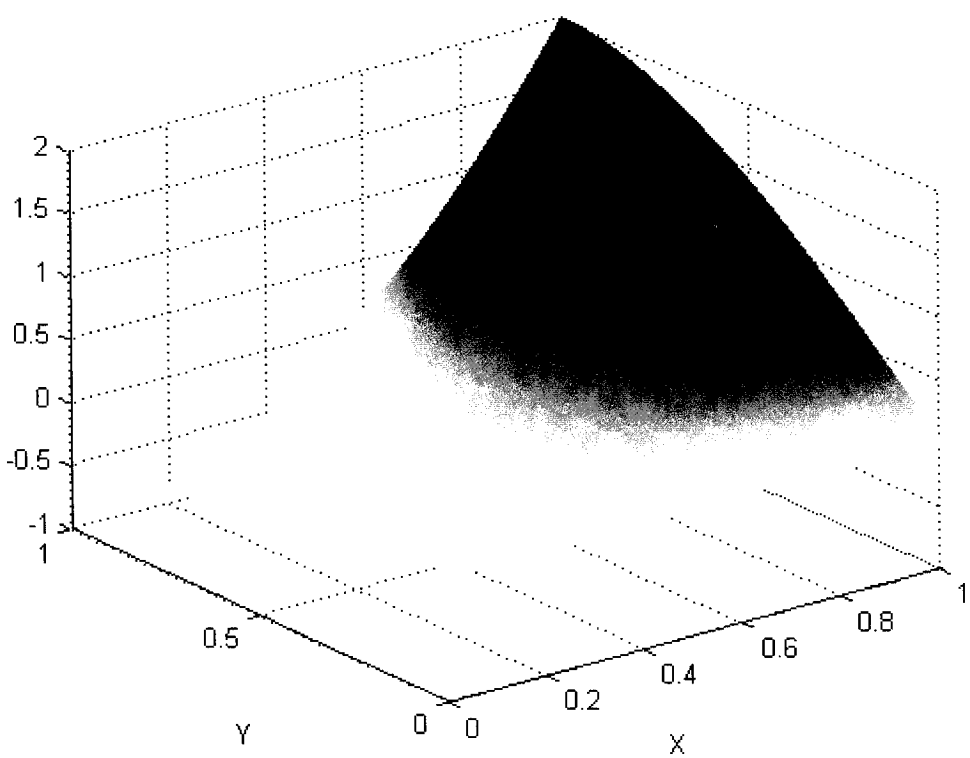

Figure 3.6 – Theoretical surface in 3D form

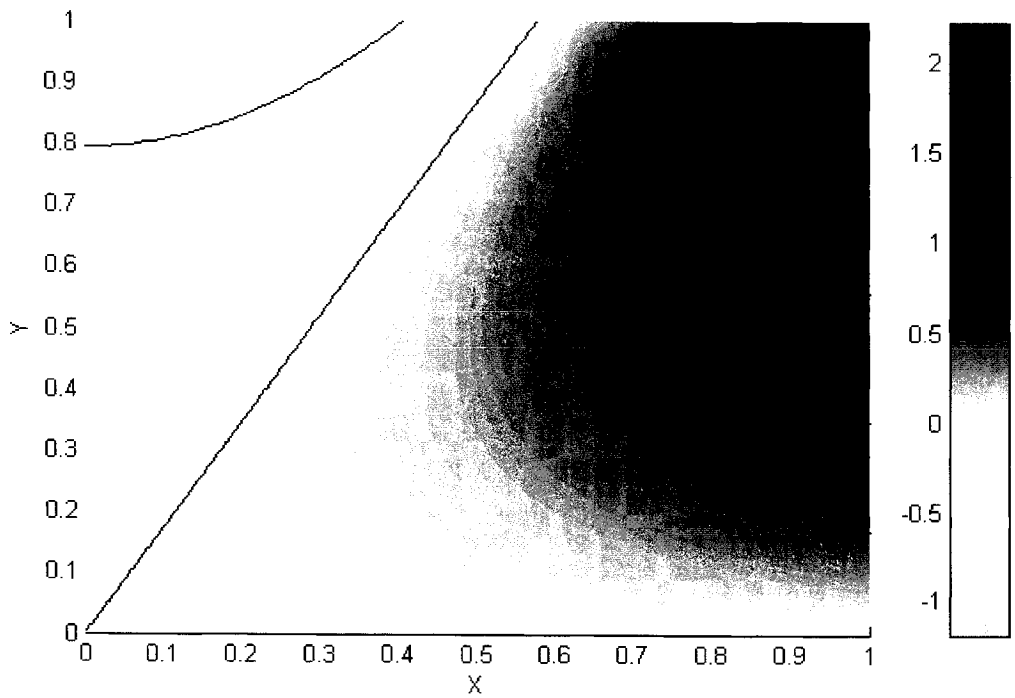

Figure 3.7 – Theoretical surface in colormap form

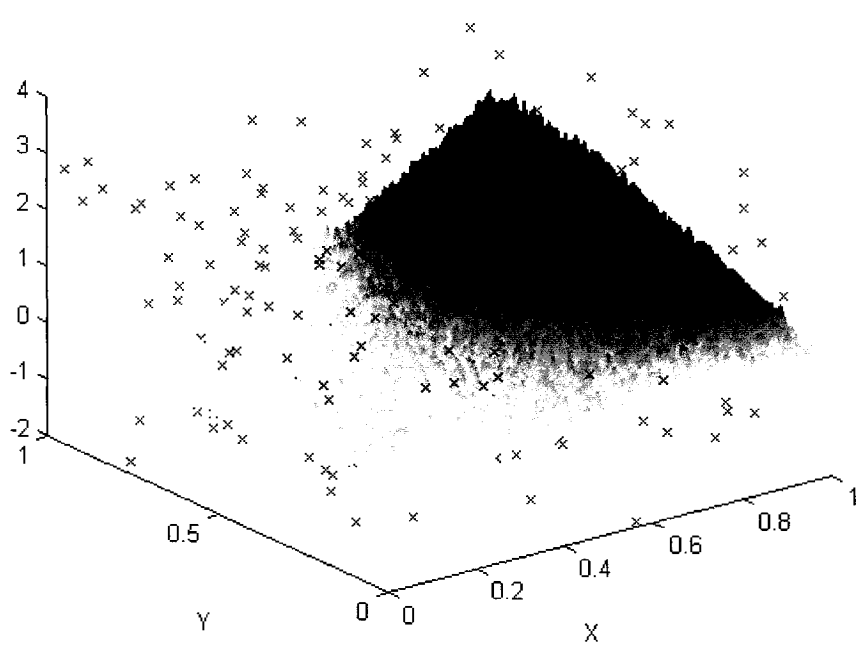

Figure 3.8 – Surface obtained with Gaussian noise of 0.1 standard deviation and 20% outlier probability (the crosses represent the outliers isolated from the rest of the learning data)

It is noteworthy that:

- the RBCGAs have been run with $p_c = 0.92$, $p_r = 0.08$ and $p_m = 0.02$ (see sections 3.3.2, 3.3.3 and 3.3.4);
- the tests have been performed on a Pentium IV 3.00 GHz PC.

3.5.1 Time Reduction

Fig. 3.9 shows the time necessary to perform the learning process with and without the aid of the clustering algorithms.

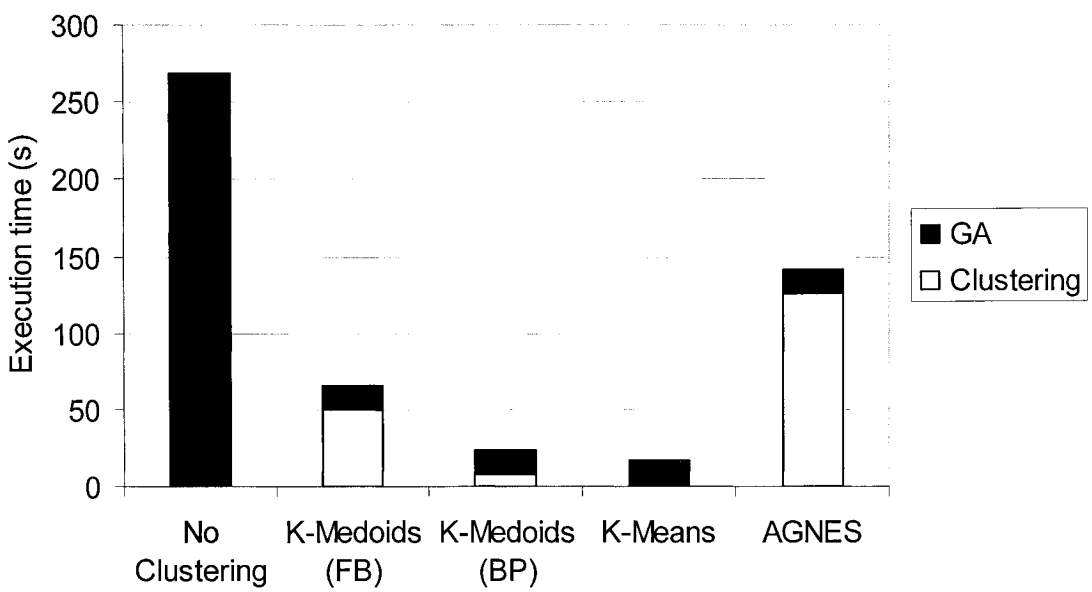

Figure 3.9 – Time reduction of the learning process with the use of clustering algorithms

The black parts of the columns represent the time needed to perform the RBCGAs, whereas the white parts represent the time needed to perform the clustering processes. For example, in the case of the *k*-means algorithm a 94% time gain is obtained. The

other algorithms offer time gains of 47% (AGNES), 75% (*k*-medoids FB) and 91% (*k*-medoids BP). The longer execution time for the FB version over the BP version of the *k*-medoids algorithm is explained by the fact that more exploration steps must be performed in order to achieve the final distribution. The results presented in the rest of the paper are obtained with the FB version which in most of the cases gave slightly better results.

3.5.2 Noise Influence

Tests on various noise levels have shown that the RBCGAs are very robust and stable. Even tests with very noisy data, with standard deviation equal to double of the original data range, lead to over 90% fitness of the obtained FKBs. No clustering algorithms tested in this study gave a significant improvement on the final fitness. Fig. 3.10 and Table 3.1 show the results for a theoretical data set as well as for data sets with three different levels of Gaussian noise.

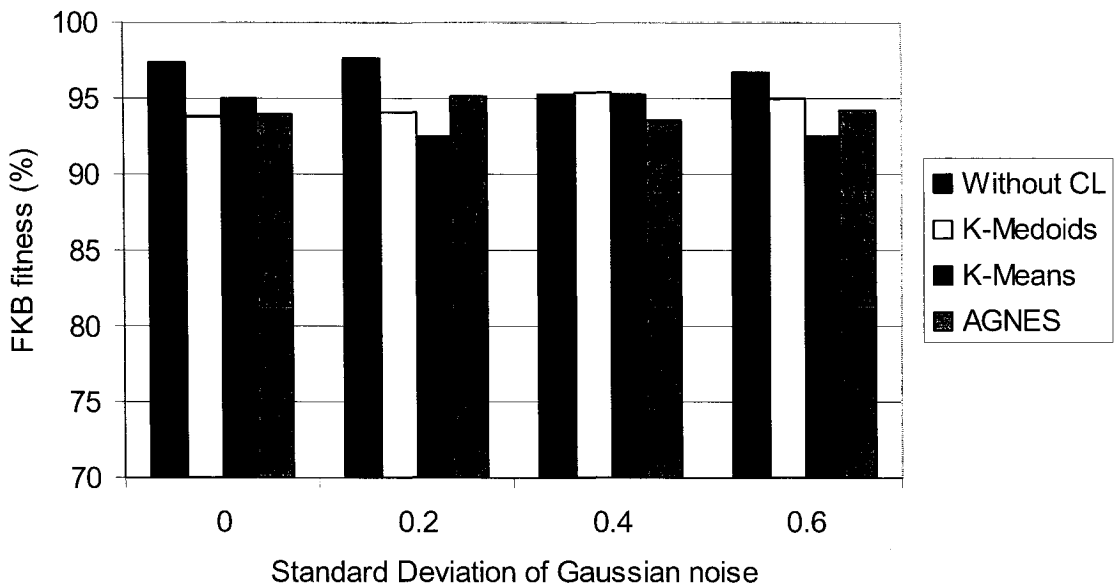

Figure 3.10 – Influence of noise on the fitness of automatically generated FKBs with and without the use of clustering algorithms

Table 3.1 – Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of noise

σ of Gaussian noise	0.0		0.2		0.4		0.6	
Fitness	μ	σ	μ	σ	μ	σ	μ	σ
No Clustering	97.39	0.31	97.67	0.63	95.31	1.30	96.71	1.46
AGNES	93.92	2.88	95.15	1.76	93.55	2.32	94.25	3.01
k-means	94.97	1.99	92.51	5.05	95.32	2.17	92.47	3.75
k-medoids	93.83	1.72	94.05	1.17	95.36	1.31	94.97	0.81

Results in Table 3.1 show that in case of noisy data, although the *k*-medoids algorithm does not always give the best results fitness-wise, it is the most stable one. When the standard deviation of Gaussian noise is larger than 0.4, data sets pre-processed with this algorithm give slightly less fit but more stable results than those obtained with no clustering pre-processing.

It is noteworthy that the standard deviations for a given parameter combination are influenced not by the lack of consistency of the genetic algorithms but by the fact that each data set is composed of different data samples. Several tests on the same data set actually give close to zero standard deviation on the obtained fitnesses.

3.5.3 Outliers Influence

At section 3.5.2 it has been shown that the clustering algorithms offer hardly any improvement on the fitness of the obtained FKBs in the case of noisy data. However, in the case of outlier probabilities, the *k*-medoids algorithm allows the RBCGA to produce FKBs with approximately 95% fitness for data sets with between 10% and 15% of outlier probabilities. In order to "absorb" the outliers in clusters containing other objects, a minimum number of objects per cluster can be forced in the case of the partitioning algorithms. This avoids outliers forming individual clusters but also makes the algorithm more constrained. Fig. 3.11 and Table 3.2 present the fitnesses of FKBs obtained from data sets containing various amounts of outliers and 0.1 standard deviation Gaussian noise.

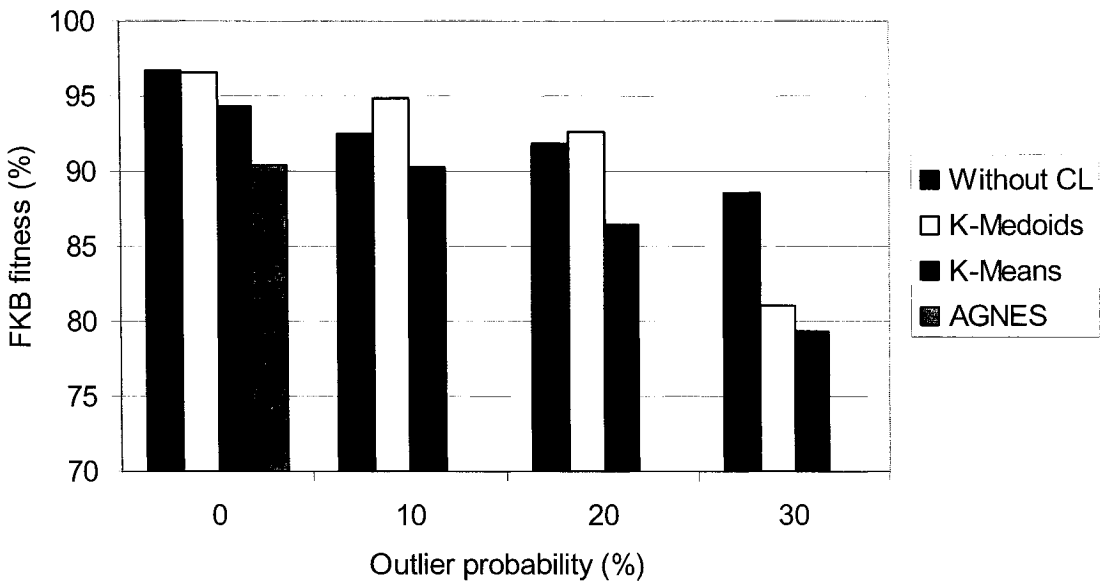

Figure 3.11 – Influence of outliers on the fitness of automatically generated FKBs with and without the use of clustering algorithms

Table 3.2 – Fitness of automatically generated FKBs with and without the use of clustering algorithms under different amount of outliers

Outliers	0%		10%		20%		30%	
Fitness	μ	σ	μ	σ	μ	σ	μ	σ
No Clustering	96.78	0.82	92.48	1.46	91.80	0.69	88.56	1.68
AGNES	90.35	8.21	57.67	3.60	41.40	20.26	42.69	10.34
k-means	94.40	2.67	90.32	4.78	86.45	2.95	79.30	4.72
k-medoids	96.59	0.66	94.88	1.11	92.60	0.65	81.07	6.25

Several conclusions can be drawn from Fig. 10 and Table 2:

- without any clustering pre-processing, the fitness declines proportionally to the amount of outliers. It falls below 90% for data sets with between 20% and 30% outlier probability;

- the fitness of the FKBs for data sets with up to 20% outlier probability pre-processed with the *k*-medoids algorithm is improved by 2.4% for 10% of outliers and by 0.8% for 20% of outliers. It declines abruptly when more than 20% of outliers are present in the learning data, because the algorithm starts forming clusters around outliers exclusively, thus increasing their relative importance;
- until the aforementioned breakdown point of 20% of outliers, the *k*-medoids algorithm also gives the most stable results;
- the *k*-means algorithm leads to a considerable fitness loss but with a regular behaviour;
- the AGNES algorithm is totally unstable and inapplicable for data sets containing outliers (the columns are not visible on Fig. 10, because the fitness of the generated FKBs falls under 70%);
- for data sets containing over 20% of outliers, the RBCGA alone give the best and the most stable results. For such extreme cases, other filtering techniques should be used to pre-process the data.

Fig. 3.12 and Fig. 3.13 show colormap representations of the surfaces generated from the FKBs obtained from a 0.1 standard deviation Gaussian noise and 20% outlier probability data set, with (Fig. 3.13) and without (Fig. 3.12) clustering pre-processing. The clustering algorithm used to produce the results illustrated on Fig. 3.13 is the *k*-medoids algorithm.

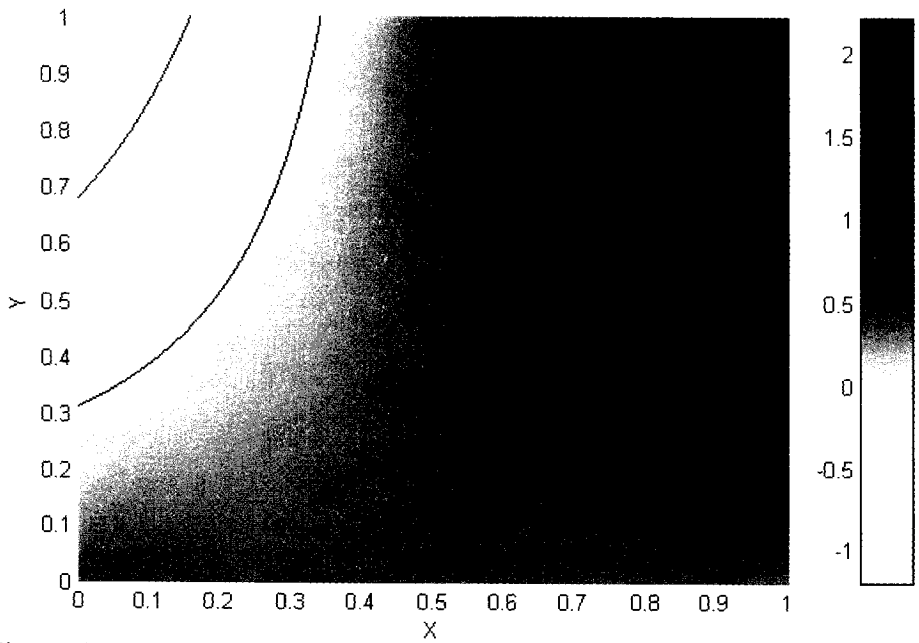

Figure 3.12 – Surface generated from the FKB obtained without clustering pre-processing on a data set with Gaussian noise of 0.1 standard deviation and 20% outlier probability

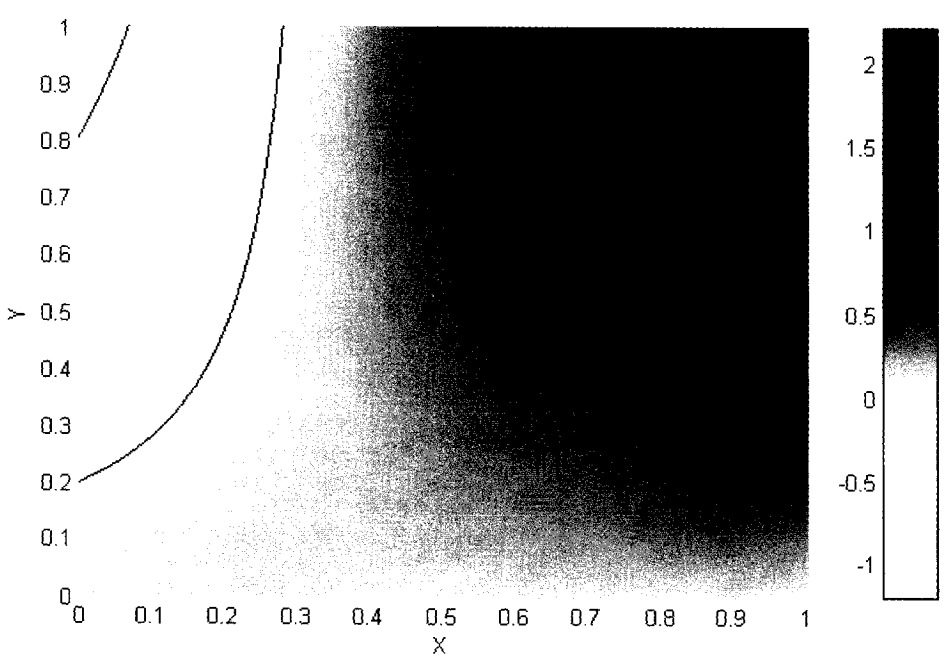

Figure 3.13 – Surface generated from the FKB obtained with clustering pre-processing on a data set with Gaussian noise of 0.1 standard deviation and 20% outlier probability

The fitness of the FKB illustrated on Fig. 3.13 is 1% stronger than that of the FKB illustrated on Fig. 3.12. These figures can also be compared with Fig. 3.7 in order to see how Fig. 3.13 approaches the theoretical results more accurately.

3.6 Conclusion

This paper aims to highlight the advantages of clustering pre-processing for the improvement of automatic generation of FKBs using a GA. It has been shown that initial clustering can significantly reduce the time needed to complete the learning process, which can be of a very high interest in factory floor applications and on-line/real-time learning. The tests that have been performed show that, when compressing the data to 5% of its original size, clustering algorithms allow to accelerate the learning process by 75% and 94% respectively in the case of the *k*-medoids and *k*-means algorithms with only minimal loss on the fitness of the obtained FKBs. Moreover, when the learning data contains a large amount of outliers, the *k*-medoids algorithm can improve the fitness of the obtained FKBs for data sets containing as much as 20% of outliers. Finally, the *k*-medoids algorithm can also make the results more stable in the case of very noisy data sets.

Acknowledgment

Financial support from the Natural Sciences and Engineering Research Council of Canada under grants of RGPIN-203618, RGPIN-105518 and STPGP-269579 is gratefully acknowledged.

References

[1] S. Achiche, M. Balazinski, L. Baron. Real/Binary-Like Coded Genetic Algorithm to Automatically Generate Fuzzy Knowledge Bases. The 4th International Conference on Control and Automation, June 2003.

[2] S. Achiche, M. Balazinski, L. Baron. Multi-Combinative Strategy to Avoid Premature Convergence in Genetically-Generated Fuzzy Knowledge Bases. Journal of Theoretical and Applied Mechanics, Vol. 42, 3: 417-444, 2004.

[3] M. Balazinski, M. Bellerose, E. Czogala. Application of Fuzzy Logic Techniques to the Selection of Cutting Parameters in Machining Processes. International Journal for Fuzzy Sets and Systems, Vol. 61: 307-317, 1993.

[4] L. Baron., S. Achiche, M. Balazinski. Fuzzy Decisions System Knowledge Base Generation Using a Genetic Algorithm. International Journal of Approximate Reasoning, pp. 25-148, 2001.

[5] T. Calinski, J. Harabasz. A Dendrite Method for Cluster Analysis. Communications in Statistics, Vol. 3: 1-27, 1974.

[6] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, Vol. 6: 182-200, 2000.

[7] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Massachusetts, 1989.

[8] J. Han, M. Kambler. Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[9] H. Hancock. Development of the Minkowski Geometry of Numbers. the Macmillan Company, New York, 1939.

[10] J.A. Hartigan. Clustering Algorthms. John Wiley & Sons, New York, USA. 1975.

[11] F. Herrera, M. Lozano. Gradual Distributed Real-Coded Genetic Algorithms. IEEE Transactions on Evolutionary Computation, Vol. 4, pp. 43-63, 2000.

[12] M. De Hoon, S. Imoto, S. Miyano. the C Clustering Library. Human Genome Center, University of Tokyo, 2004

[13] L. Kauffman, P.J. Roussewuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, 1990.

[14] M. G. Kendall. Rank Correlation Methods. Griffin, 1962.

[15] E. L. Lehmann, H. J. M. D'abrera. Nonparametric: Statistical Methods Based on Ranks. Holden-Day, 1975.

[16] F. G. Lobo, D. E. Goldberg, M. Pelikan. Time Complexity of Genetic Algorithms Exponentially Scaled Problems. GECCO 2000: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 151-158, 2000.

[17] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1: 281-297, 1967.

[18] K. V. Mardia. J. T. Kent, J. M. Bibby, Multivariate Analysis. Academic Press, London, 1979.

[19] Z. Michalewicz. Genetic Algorithms + Data Structure = Evolution Programs. Springer, New York, 1992.

[20] Wikipedia, the Free Encyclopedia. http://www.wikipedia.org/. Consulted on January 27th 2005.

[21] L.A. Zadeh. Outline of new Approach to the Analysis of Complex Systems and Decisions Processes. IEEE Transactions of Systems, Man and Cybernetics, Vol. 3: 28-44, 1973.

CHAPITRE 4

NOMBRE DE CLUSTERS DANS UN ENSEMBLE DE DONNÉES

De nombreux travaux de recherche sur l'algorithme *k*-means concernent le nombre de clusters optimal k_{opt}. La faiblesse de cet algorithme réside dans le fait que le nombre de clusters doit être connu avant le lancement de l'algorithme.

4.1 Méthodes usuelles

Une des techniques les plus utilisées est de générer les clusters pour tous les *k* possibles et d'analyser la courbe de l'erreur de reproduction *E* en fonction de *k*. On choisit ensuite le nombre de clusters à l'endroit où son augmentation ne présente plus d'intérêt suffisant en termes de diminution de l'erreur de reproduction. La figure 4.1 présente une courbe typique de l'erreur de reproduction en fonction du nombre de clusters.

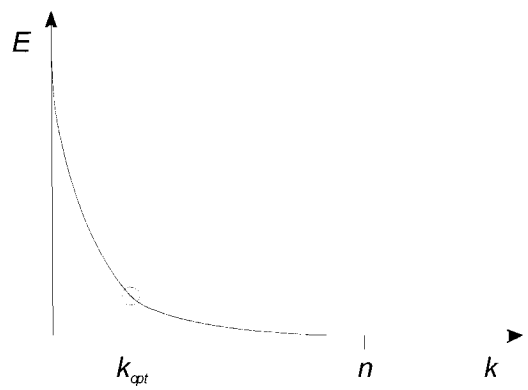

Figure 4.1 – Courbe de l'erreur de reproduction en fonction du nombre de clusters

Le désavantage évident de cette méthode est la nécessité de supervision humaine. Ce pourquoi, d'autres méthodes ont été développés depuis les années 1970, certaines d'entre elles même très récemment. Elles permettent de trouver k_{opt} automatiquement sans supervision humaine en se basant sur différents indices. Certaines de ces méthodes ne requièrent pas la génération de clusters pour tous les k possibles mais sont en général peu satisfaisantes [11]. Une revue de ces méthodes peut être trouvée dans [21].

4.2 Méthodes automatiques

Les quatre méthodes implantées dans le logiciel *GFS Cluster* exigent toutes la génération de clusters pour tous les k possibles. La première et la plus ancienne de ces méthodes, proposée par Calinski et Harabasz [8] se base sur l'indice *CHpF* (Calinski-Harabasz pseudo-F) défini par :

$$CHpF = \frac{\frac{B_k - W_k}{W_k}}{\frac{k-1}{n-k}} \tag{4.1}$$

où B_k est la somme des distances inter-clusters, W_k est la somme des distances intra-clusters, n est la taille de l'échantillon de données et k est le nombre de clusters.

La deuxième méthode, proposée par Hartigan [17], est basée sur la somme des distances intra-clusters (W_k). Cette méthode propose de garder la distribution avec le plus petit k telle que $Hart_k < 10$ (éq. 4.2)

$$Hart_k = \left(\frac{W_k}{W_{k+1}} - 1\right) \times (n - k - 1) \tag{4.2}$$

La troisième méthode, proposée par Krzanowski et Lai [24], cherche à maximiser KL_k :

$$KL_k = \left| \frac{\mathrm{DIFF}(k)}{\mathrm{DIFF}(k+1)} \right| \tag{4.3}$$

avec

$$\mathrm{DIFF}(k) = (k-1)^{2/p} W_{k-1} - k^{2/p} W_k \tag{4.4}$$

La dernière méthode, proposée par Kaufman et Rousseeuw [22], propose la distribution avec la plus grande Silhouette moyenne. La Silhouette d'un objet i est définie comme :

$$Silhouette_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{4.5}$$

où a_i est la dissimilarité moyenne entre l'objet i et tous les autres objets dans le même cluster et b_i est le plus petit $d_{i,j}$, où $d_{i,j}$ est la dissimilarité moyenne entre l'objet i et tous les objets du cluster j (j représente à tour de rôle tous les clusters auxquels i n'appartient pas).

D'autres méthodes ont été récemment proposées, telles que *Gap* proposée par Tibshirani et al. [31] et *Jump* développée par Sugar et James [30].

Le chapitre suivant présente une application de certaines de ces méthodes à la génération automatique des BCFs.

CHAPITRE 5

PREDEFINING NUMBERS OF FUZZY SETS FOR GENETICALLY GENERATED FUZZY KNOWLEDGE BASES USING CLUSTERING TECHNIQUES

Soumis à *International Journal of Approximate Reasoning.*

Journal affilié à l'organisation NAFIPS (North American Fuzzy Information Processing Society). Edition Elsevier Science.

Abstract

One of the numerous problems surrounding fuzzy knowledge base generation using genetic algorithms is finding an optimal number of fuzzy sets for each premise. Some genetic algorithm methods for the automatic generation of fuzzy knowledge bases (including the current algorithm developed by the authors) use a multi-objective method combining error minimization and simplification. This paper proposes solutions based on cluster analysis and validation indices for the numbers of clusters used in predefining the numbers of fuzzy sets. Two different validation indices as well as a combination of one of these and of the multi-objective method are compared to the original multi-objective method on both synthetic and experimental data. Results obtained with the proposed techniques showed a considerable improvement over the multi-objective method on both data sets.

Keywords: clustering, genetic algorithms, multi-objective optimization, fuzzy logic, fuzzy sets

5.1 Introduction

One of the numerous problems surrounding fuzzy knowledge base (FKB) generation using genetic algorithms is finding an optimal number of fuzzy sets for each premise. The simplest genetic algorithms start with a random distribution of fuzzy sets and optimize their location and shape, but cannot determine the number of fuzzy sets [11]. More "evolved" methods use a multi-objective (MO) method combining error minimization and simplification. The simplification of the knowledge bases is often performed by starting with an over estimated number of fuzzy sets and then randomly eliminating fuzzy sets one by one along with the corresponding fuzzy rules. Such is the case of the real binary-like coded genetic algorithm (RBCGA) developed by the authors [1].

The main drawback of the multi-objective method is that the final solutions often remain too complex. Moreover, since the evolutionary algorithm starts with very complex solutions, the computation time necessary to evaluate each solution is very high.

This paper describes a technique for finding an optimal number of fuzzy sets for each premise based on the *k*-means clustering algorithm. The first two sections present the Fuzzy Decision Support System (FDSS) and the main features and specificities of the RBCGA. A description of the clustering techniques and validation results follow in the last two sections. Finally, a conclusion summarizes the results.

5.2 Fuzzy Decision Support System

The rule-based approach to decision making is done using fuzzy logic techniques by applying the compositional rule of inference. This approach is used to handle imprecise knowledge and was developed in the sixties by L.A. Zadeh [12]. Figure 5.1 shows a screen printout of the FDSS Fuzzy-Flou software used as a validation tool for genetically generated FKBs.

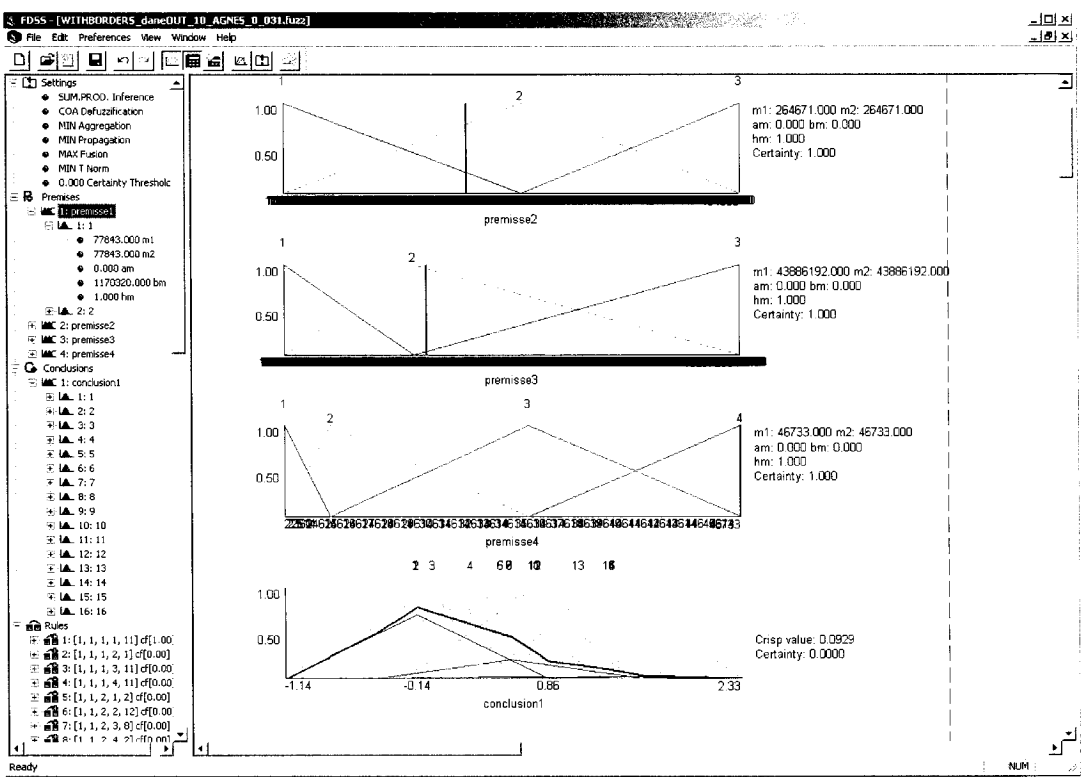

Figure 5.1 – Graphic user interface of FDSS Fuzzy-Flou

For more information on Fuzzy-Flou software, please refer to [3].

5.3 Automatic Generation of FKBs

GAs are powerful stochastic optimization techniques based on the analogy of the mechanics of biological genetics and imitate the Darwinian survival of the fittest approach [1]. As shown in figure 5.2, each individual of a population is a potential FDSS Fuzzy-Flou knowledge base. Figure 5.2 presents the encoding/decoding scheme as well as the four basic operations of the developed GA learning software; reproduction, mutation, evaluation and natural selection. The method uses iterative improvement of individuals in each generation to converge toward multiple optima simultaneously. The RBCGA developed by the authors [1] is a combination of a real coded genetic algorithm and a binary coded genetic algorithm.

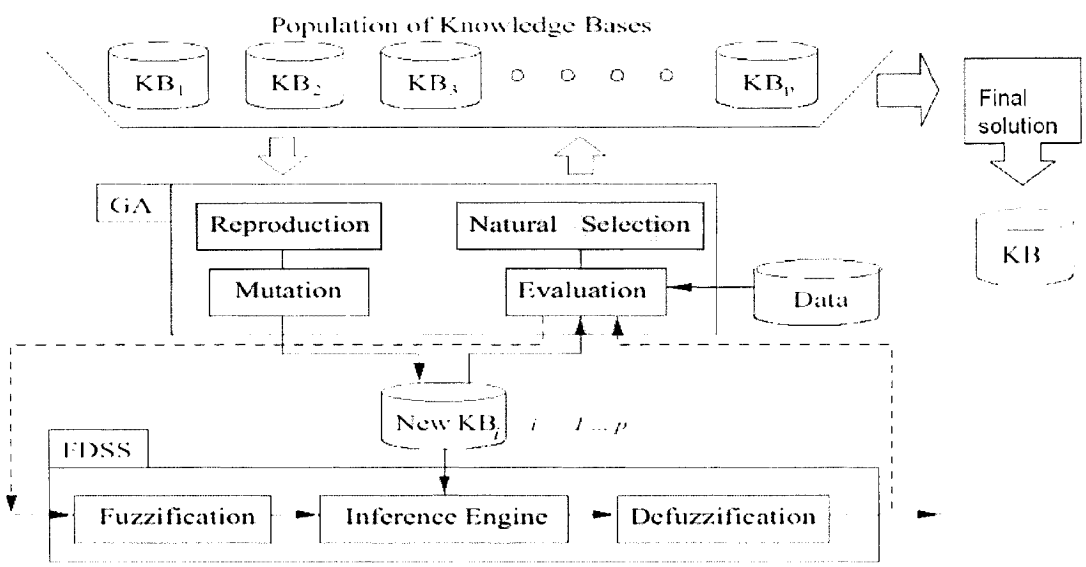

Figure 5.2 – Genetic learning paradigm

5.3.1 Coding

The *genotype* of an FKB is generated by coding its parameters into chromosomes and corresponds to several independent sets of real numbers and a set of integers. The genotype contains the following items:

- Input/Output Premises: A set of real numbers; for the sake of coding simplicity we consider only non-symmetrical-overlapping triangular fuzzy sets for premises and symmetrical triangular fuzzy sets for the conclusion.
- Fuzzy Rules: A set of integer numbers; the genotype of fuzzy rules contains information about all possible combinations for connections between one fuzzy set on each premise and a fuzzy set on the conclusion.

5.3.2 Multi-Crossover Mechanisms

The evolution of a population of FKBs for each generation is achieved by the reproduction of the "best" individuals, based on their abilities to survive natural selection. Reproduction is performed by crossover of the genotype of the parents to obtain the genotype of an offspring using a multi-crossover which is composed of a premises/conclusion crossover and a fuzzy rules crossover. These mechanisms are governed by the initiating probability p_c.

5.3.2.1 Premises/Conclusion Crossover

The mechanism used is called a blending crossover α (BLX-α) [19], where α determines the exploitation/exploration level of the offspring. The parameter α is set to

1.0 for the first third of the generations (exploration) to 0.5 for the second third (relaxed exploitation) and finally to 0.1 for the last third of the evolution (exploitation). These changes in exploitation/exploration balance, improve the behaviour of the RBCGA to avoid premature convergence [2] [3].

5.3.2.2 Fuzzy Rules Crossover

Since the part of the genotype representing the fuzzy rule base is composed of integer numbers, the crossover of this part of the genotype is done by a simple crossover. The operation is performed by exchanging the end part of the sets (containing the fuzzy rules) of the parents at a randomly selected crossover site as shown in figure 5.3.

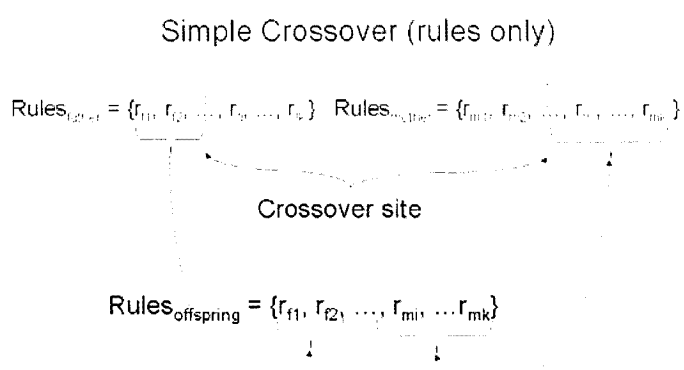

Figure 5.3 – Simple crossover

5.3.3 Fuzzy Set Reducer

The fuzzy set reducer mechanism is used to reduce the complexity of the FKBs by selecting a fuzzy set on each premise and erasing it from the respective sets together with the corresponding fuzzy rules. This mechanism is governed by the initiating probability p_r. In this paper, p_r is set to 0.15 while testing the multi-objective technique

and to 0.0 while using predefined numbers of fuzzy sets, since the optimal number of fuzzy sets is supposed to be reached in the initial population. The 0.15 value for p_r generated the most reliable results on the validation data used in section 5. When using the MO method, the initial number of fuzzy sets was set to 7. Such a conservative choice was made due to the assumption that we had no initial knowledge on the data.

5.3.4 Mutation

Mutation is a random alteration of an offspring's genotype in the population. Mutation makes it possible to seek completely new solutions without any control on the direction, as opposed to gradient-based optimization techniques. The probability p_m governs the occurrence of this mechanism. In this paper, uniform mutation is applied [11]. The probability p_m is set to 0.1 when testing the MO technique and to 0.01 when predefined numbers of fuzzy sets are used. Again, the 0.1 value for p_m was obtained after several tests on the validation data. The mutation occurrence probability has been lowered for predefined numbers of fuzzy sets to decrease the chance of having two fuzzy sets overlapping and causing one of them to be eliminated by the evolutionary algorithm.

5.3.5 Natural Selection

Natural selection is performed on the population by keeping the "most promising" individuals based on their fitness. The first generation begins with P randomly generated FKBs and the same number is generated by crossover and mutation. To keep the population constant, we apply natural selection on the $2P$ FKBs by ordering them according to the performance criterion and keeping the first P FKBs. In this paper the

population size is set to 100 and the number of generations is set to 200. Such conservative choices are necessary for comparison of the different methods described in the following sections.

5.3.6 Performance Criterion of the RBCGA

The performance criterion allows the computation of the rating of each FKB used by the RBCGA in order to perform natural selection. In this paper, the performance criterion is the combination of the accuracy level of an FKB (approximation error) in reproducing the outputs of the learning data and the simplicity of the FKB.

To begin, the approximation error Δ_{RMS} is measured using the RMS error method:

$$\Delta_{RMS} = \sqrt{\sum_{i=1}^{N} \frac{(\text{RBCGA}_{\text{output}} - \text{data}_{\text{output}})^2}{n}} \tag{5.1}$$

where n represents the size of the learning data. The RMS fitness value ϕ_1 is evaluated as a percentage of the output length base of the conclusion L, i.e.

$$\phi_1 = \frac{L - \Delta_{\text{RMS}}}{L} \times 100 \tag{5.2}$$

The second objective function ϕ_2 evaluates the complexity of an FKB based on the number of active rules.

The resulting objective function (eq. 5.3) combines these two contradictory objectives in a weighted sum of ϕ_1 and ϕ_2 based on the weight ω_0 associated with ϕ_1.

$$\phi = \omega_0\phi_1 + (1 - \omega_0)\phi_2 \tag{5.3}$$

The weight ω_0 is set to 0.9 in order to put the emphasis on accuracy rather than simplicity.

5.4 Clustering

5.4.1 The *k*-means Algorithm

The *k*-means algorithm [17] was selected for the computation of clusters as it is arguably the most popular clustering algorithm. The version implemented by the authors in the software used in this paper starts with a distribution based on the maximization of distance between clusters. First, the sum of distances between each object and all the other objects is computed. The object with the largest sum of distances constitutes the first cluster centroid. Next cluster centroid is the furthest object from the first centroid. The subsequent cluster centroids are those with the largest values for the minimum distance to the previous centroids. This method ensures very fast and accurate results in the case of outlier-free data. In presence of outliers, another initial distribution or clustering method should be used. In this paper an assumption has been made that the data contains no outliers.

Convergence is obtained by selecting an object and moving it to the first cluster closer than the one it belongs to. After each swapping, the centroids are recalculated. Such a method usually gives better results than searching for the closest centroid. The loop stops when each object belongs to the closest cluster.

The number of clusters k must be specified *a-priori* since the k-means algorithm can only swap objects between clusters. It cannot merge or divide existing clusters. This is why additional techniques must be used in order to find the optimal number of clusters (k_{opt}) for each data set.

Figure 5.4 shows a screenshot of the clustering software developed for optimization of FKB automatic generation using clustering techniques.

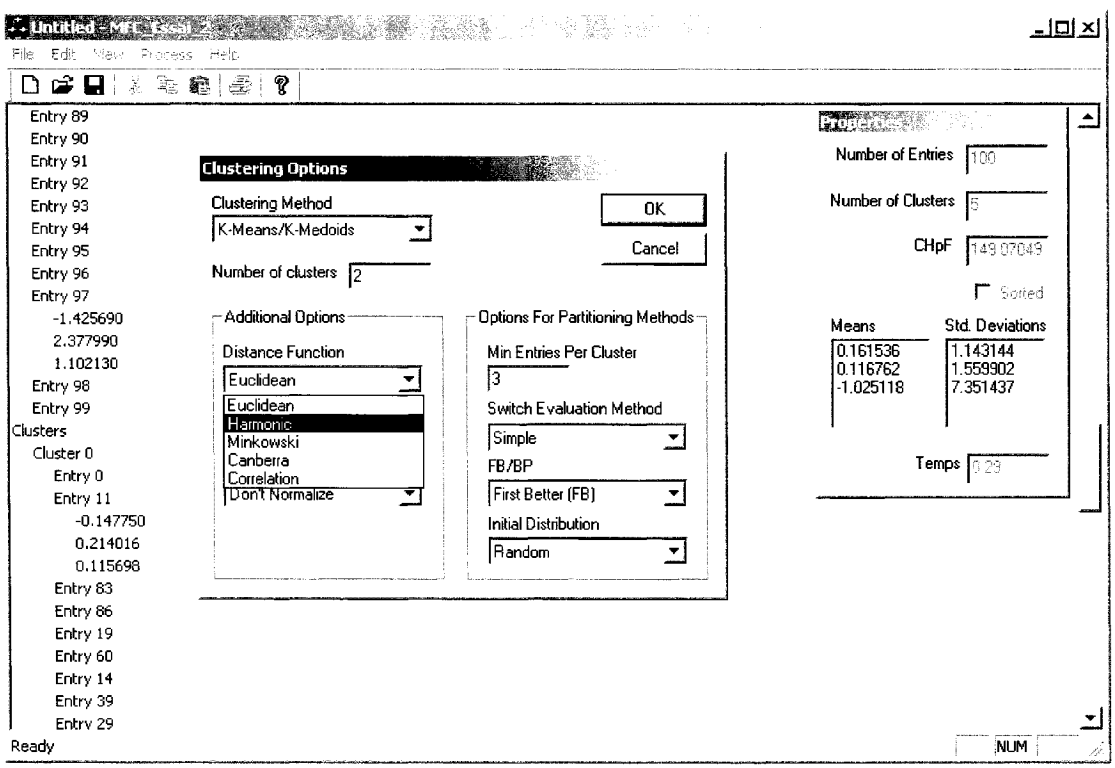

Figure 5.4 – Screenshot of the clustering software

5.4.2 Number of Clusters

Several techniques for finding the optimal number of clusters k_{opt} are available. Most of these techniques require the computation of clusters for every possible k and then

propose a distribution based on several parameters such as within-cluster similarity, between-cluster dissimilarity, etc. Such is the case for the five most popular indices suggested by Calinski and Harabasz [5], by Krzanowski and Lai [19], by Hartigan [13], by Kauffman and Rousseeuw [13] and by Tibshirani [26]. Sugar and James [25] provide quite extensive descriptions and experimental results for these techniques. Several methods have been proposed that do not require computation of clusters for every possible k, but none of them is completely satisfactory [9]. Jain and Dubes [16] provide a general overview of such methods.

In this paper, two different methods have been selected. The first method, proposed by Hartigan [13], is based on the within cluster sum of square distances (W_k). This method advises the distribution with the smallest k such that $Hart_k < 10$ (eq. 5.4)

$$Hart_k = \left(\frac{W_k}{W_{k+1}} - 1\right) \times (n - k - 1) \tag{5.4}$$

where n is the size of the data.

The Silhouette method proposed by Kaufman and Rousseeuw [13] advises the distribution with the largest average Silhouette. The Silhouette of an object i is defined by eq. 5.5.

$$Silhouette_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{5.5}$$

where a_i is the average dissimilarity between object i and all other objects in the cluster to which it belongs and b_i is the smallest $d_{i,j}$, where $d_{i,j}$ is the average distance of object i to all objects of cluster j to which it does not belong.

These two methods have been selected because they usually give relatively distinct results, whereas the other methods often give results close to the Silhouette method.

The results obtained with these techniques, as well as those obtained with the original multi-objective method, are compared in section 5. Since, the *Hart* index tends to overestimate the number of clusters in a data set, a combination of this technique with the MO method has also been included in the tests.

5.4.3 Global Algorithm Overview

The learning process has been divided into two stages. First, the data is analyzed by the clustering software in order to find k_{opt} for each variable. The different dimensions are treated independently as several sets of one-dimensional data. The distribution obtained for each premise is then memorized and used as input to the genetic algorithm software.

The second stage starts by creating the desired number of knowledge bases with the appropriate number of fuzzy sets, defined by the clustering process for each premise. The first 80% of the knowledge bases are created randomly and the remaining 20% are created using the centroid positions found by the clustering algorithm. The 80/20 proportion has been obtained from performance and stability tests on several data sets. It is noteworthy that the first and last clusters are ignored, since the first and last premises are defined by the limits of the data. This is necessary to preserve a valid range of the

premises in the knowledge bases. Due to this restriction, when any premise has only two fuzzy sets, the centroid positions for this particular premise are totally ignored. Another version of the algorithm uses all the centroid positions and adds two additional fuzzy sets for the limits, thus obtaining (k_{opt} + 2) fuzzy sets. Nevertheless, creating 2 additional fuzzy sets increases the complexity of the FKB and the results obtained with this version didn't give satisfactory results. Hence, they were omitted in the validation results in the next section.

What should also be mentioned is that no preliminary tests are required when predefining the number of fuzzy sets in order to find the optimal values for p_c, p_r and p_m for each new data set. This makes the automatic generation less fastidious and more reliable (see sections 5.3.3 and 5.3.4).

5.5 Validation Results

Two series of validation tests have been performed. The first one is based on a synthetic 3D surface and the second one on experimental 5D data obtained from a tool condition monitoring application.

5.5.1 3D Surface Data

The synthetic data was obtained from a 3D surface defined by eq. 5.6 in the interval of [−3; 3] × [−3; 3]. The output range for this input interval is [−54.0; 54.0].

$$f(x, y) = 3x^2 y - y^3 \tag{5.6}$$

Data samples containing 100 entries have been randomly generated in 9 Gaussian clusters with 10 different standard deviations in the interval of [0.05, 0.5]. A slight Gaussian noise (standard deviation of 1.0) has also been added. A set of 30 samples have been generated for each of the 10 cluster spreads (standard deviations), thus giving good estimates of performance and stability of the employed techniques.

The obtained FKBs were all tested on the same sample of 3721 uniformly distributed theoretical data without any noise. Figure 5.5 shows the resulting validation errors obtained with each technique.

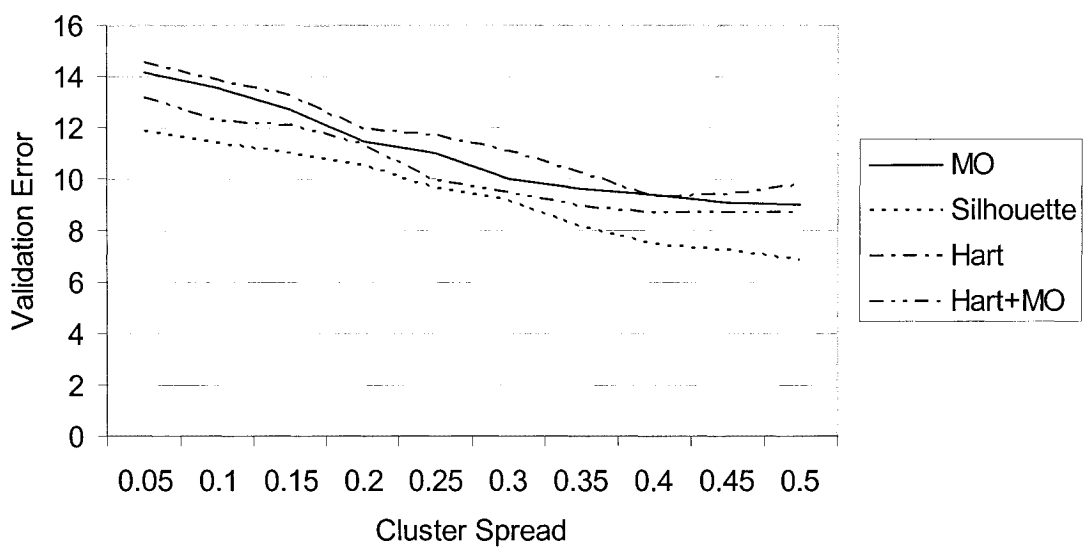

Figure 5.5 – Validation errors on synthetic data

Figure 5.5 demonstrates that both Silhouette and the combination of the Hartigan and the multi-objective technique (Hart + MO) give quite noticeable improvement on the validation error. The Hartigan technique alone gives slightly worse results. As previously mentioned, this can be explained by the fact that the Hartigan technique tends

to overestimate the number of clusters in a data set. This increases the complexity of the obtained FKBs causing overfitting of the training data and hence diminishing its generalization capabilities. A summary of the results is shown in Table 5.1.

Table 5.1 – Summary of the validation errors on theoretical data

Cluster Spread	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**
MO	14.14	12.69	10.99	9.58	9.10
Silhouette	11.85	10.98	9.65	8.12	7.23
Improvement over MO	16 %	14 %	12 %	15 %	21 %
Hart	14.53	13.25	11.68	10.22	9.35
Improvement over MO	-3 %	-4 %	-6 %	-7 %	-3 %
Hart + MO	13.13	12.09	9.91	8.91	8.70
Improvement over MO	7 %	5 %	10 %	7 %	4 %

Table 5.1 indicates a maximum improvement over the original MO method of 21% in the case of the method based on the Silhouette index and of 10% in the case of the method based on the combination of the Hartigan index and the MO method. An approximate 5% increase on the validation error was observed in the case of the method based on the Hartigan index alone.

5.5.2 Experimental Data

The experimental data were collected by Bombinski and Jemielniak [6] for tool wear monitoring. Four signals were collected (cutting force, feed force, passive force and acoustic emissions) out of which numerous signal features (SFs) were computed: average value, RMS, several distribution parameters etc. Based on their correlation with the tool wear, four SFs were selected for the learning process: variance of the feed force ($F_{f,Var}$), variance of the feed force from the 2nd seconds of cuts ($F_{f,Var,2nd}$), RMS of the

acoustic emissions from the 2nd seconds of cuts ($AE_{RMS,2nd}$) and the average value of the feed force ($F_{f,av}$). The tool wear has been approximated as the ratio of the cutting time as performed so far to the overall tool life (ΔT). In total 94 operations were performed during which 9 tools were worn out. For more precise information on how the data were collected and on the signal features that were chosen, please refer to [6]. Figure 5.6 shows the selected signal features versus the used up portion of the tool life from 4 different tools.

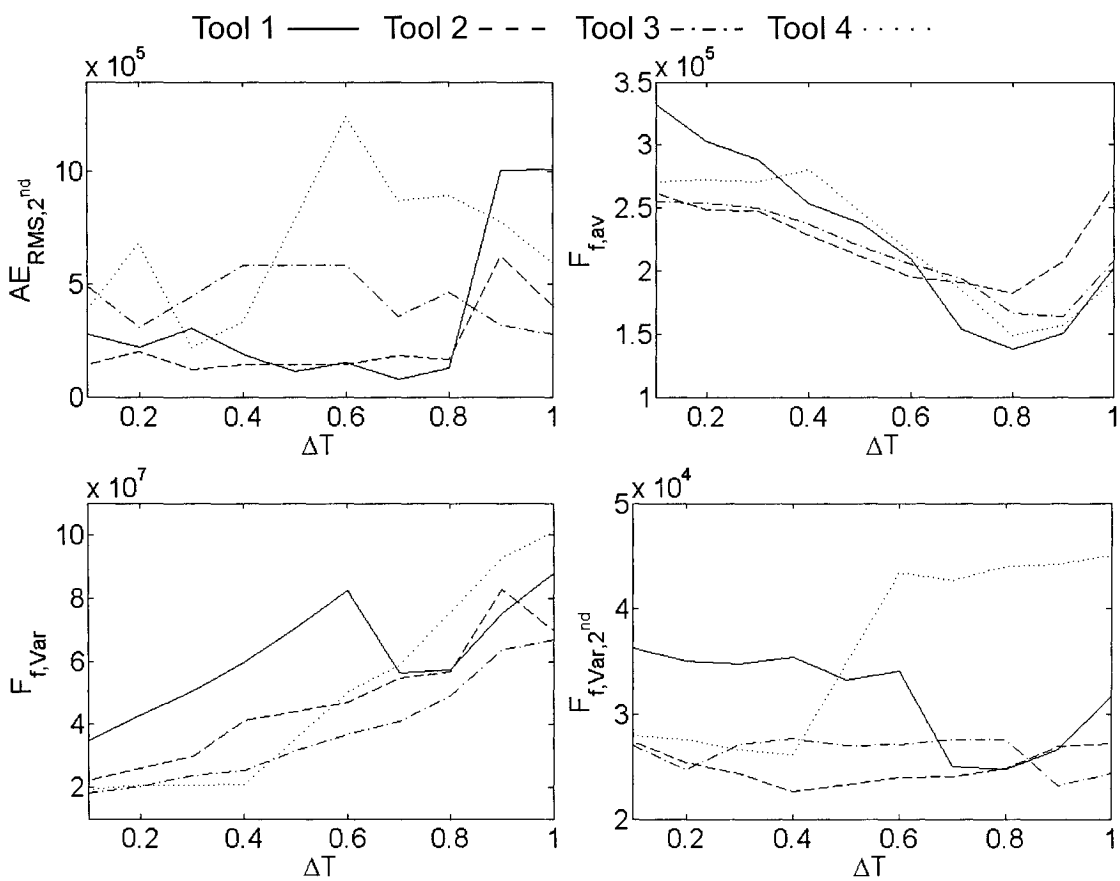

Figure 5.6 – The selected signal features versus the used up portion of tool life from the training data (the symbols are defined above)

Since the amount of data was quite small, a leave-one-out cross-validation technique was used. During each of the 9 training sessions data collected from one tool was hidden and later used for validation, whereas data collected from the 8 other tools were used for training. Figure 5.7 shows the mean validation errors for each technique.

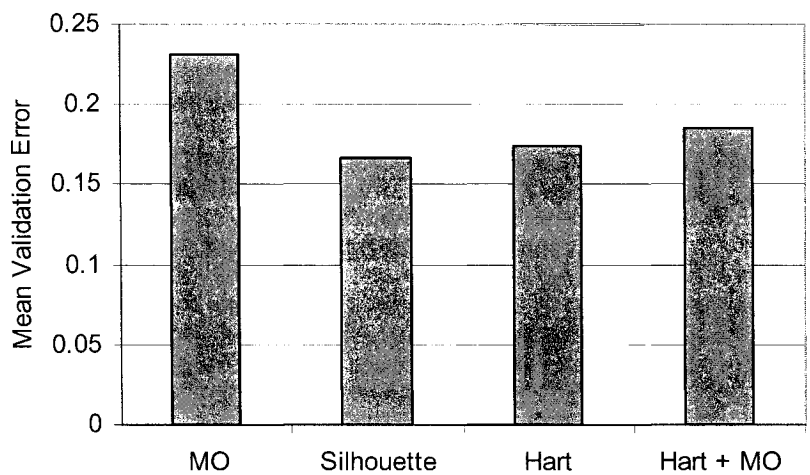

Figure 5.7 – Mean validation errors on experimental data

Figure 5.7 shows similar order of validation errors to that on figure 5.5. Again, the Silhouette and Hart + MO techniques gave clear improvement over the multi-objective method except that this time the Hart technique gave better results than the Hart + MO. It is noteworthy that a 28% improvement has been obtained. The contents of figure 5.7 can also be analyzed by their relationship with the complexity of the FKBs, i.e. the mean number of rules (the product of the numbers of fuzzy sets in the particular premises) which are presented on figure 5.8.

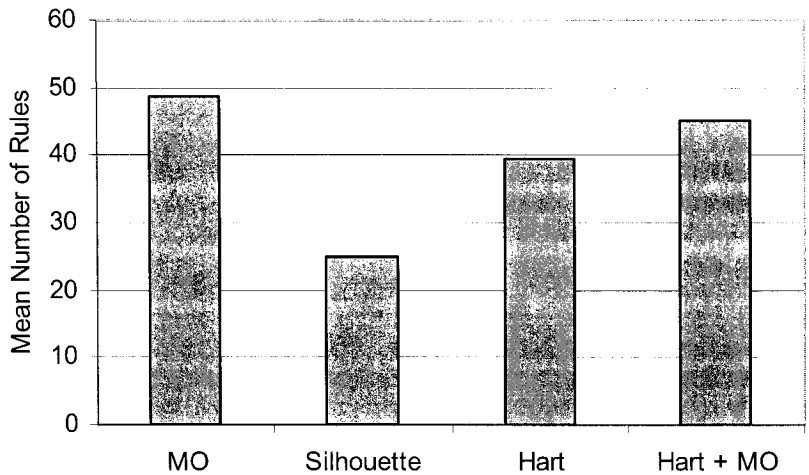

Figure 5.8 – Mean complexity of the FKBs

It can be seen on figures 5.7 and 5.8 that the prediction accuracy of the FKBs is directly linked to their complexity. The weaker performance of the FKBs obtained with the MO method can thus be explained by the inability of the MO method to accurately reach the optimal complexity for that particular data set.

Figure 5.9 shows representative examples of FKBs: one obtained using the MO method and one obtained with predefined numbers of fuzzy sets using the Silhouette technique.

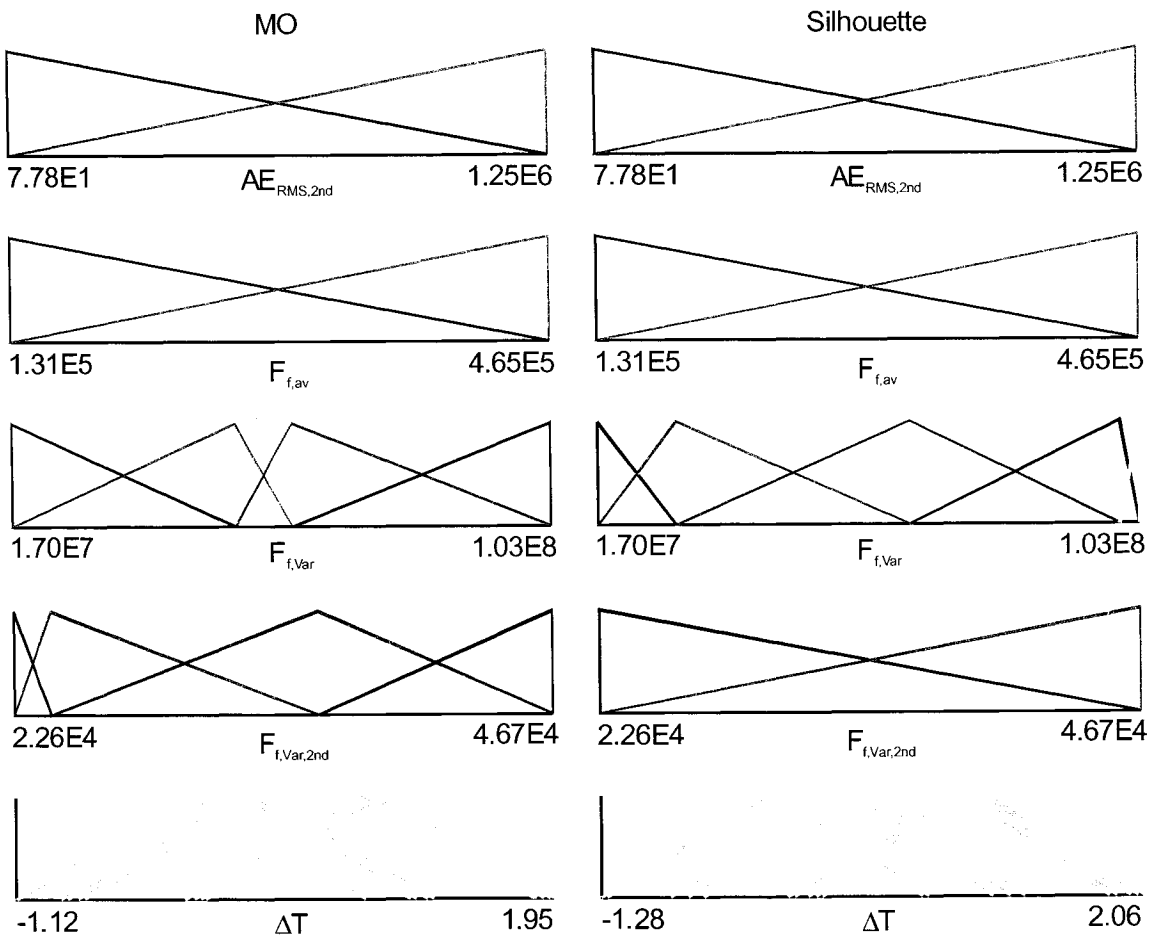

Figure 5.9 – FKBs obtained with the MO method (left) and with predefined numbers of fuzzy sets with the Silhouette technique (right)

Slight differences can be seen between the two knowledge bases. In these particular examples (not all FKBs obtained with the same method have the same complexity) the FKBs obtained with the MO method and with the Silhouette technique have respectively 64 and 40 rules.

Figure 5.10 shows the plots of the FKBs outputs obtained with all four tested methods versus the ΔT in the actual experimental data. On each plot a diagonal line has been included to represent ideal prediction. The more precise a prediction, the more the plots

fit to these diagonals. To obtain each of these plots, four different FKBs were used. For example to obtain the plots for the second tool, the FKB used was the one trained without the data obtained from that particular tool. This means that the FKBs are not reproducing data that they were trained with but predicting totally new data.

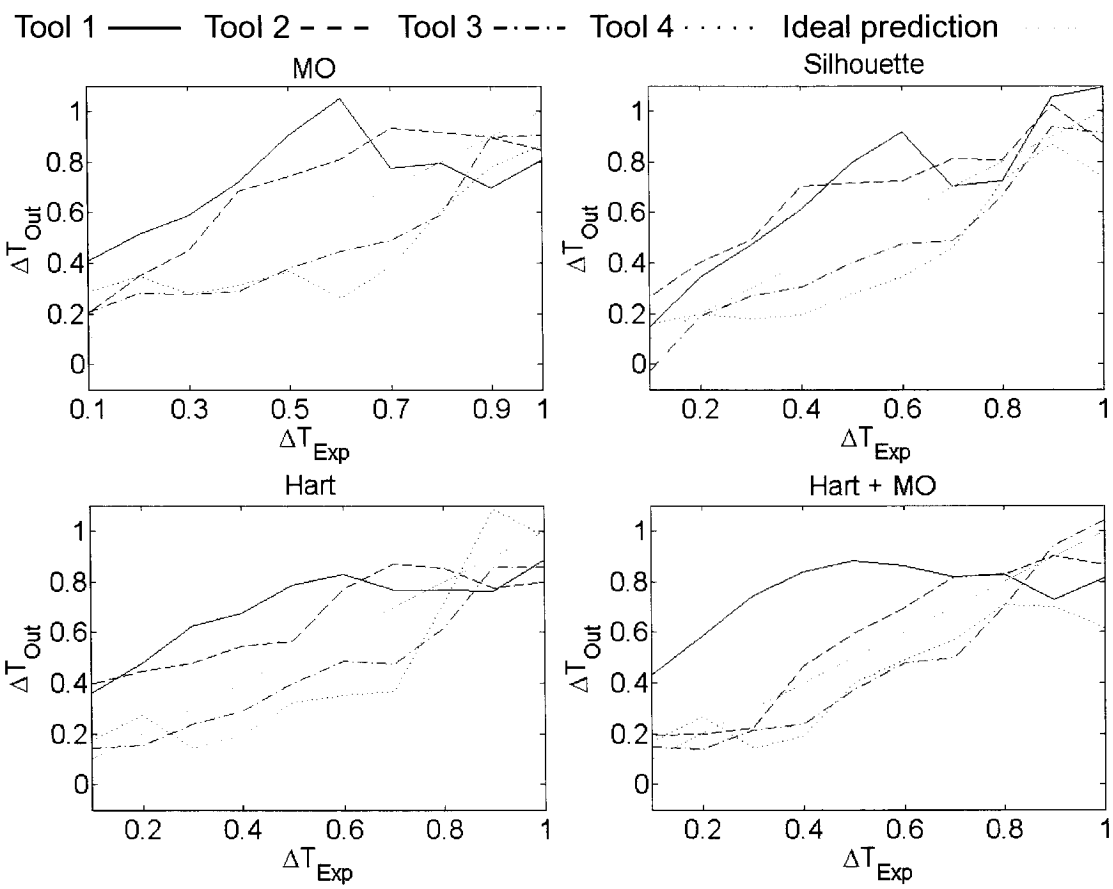

Figure 5.10 – Predicted tool wear versus experimental data on four different tools

On each of the four plots of figure 5.10, ΔT_{Exp} and ΔT_{Out} are respectively the experimental data and the output of the obtained FKBs.

Comparing the results obtained with the four methods, an important improvement in the results obtained with predefined numbers of fuzzy sets can be seen.

5.6 Conclusion

Two different method of predefining the numbers of fuzzy sets for automatic generation of FKBs using GAs were developed and compared with the initial MO method. A combination of one of these techniques and of the MO method was also analyzed. Validation with synthetic and experimental data shows considerable improvement in prediction accuracy for FKBs obtained with predefined numbers of fuzzy sets. On the synthetic data set, an average improvement of 15.4% (and a maximum of 21%) was obtained with the Silhouette technique. Also, on the experimental data set, an improvement of 28% was obtained with that same technique. Moreover, the use of predefined number of fuzzy sets removes the necessity to preanalyze the training data. An optimal complexity can be determined automatically by the clustering process, thus removing the need for human supervision.

5.7 Acknowledgment

Financial support from the Natural Sciences and Engineering Research Council of Canada under grants of RGPIN-203618, RGPIN-105518 and STPGP-269579 is gratefully acknowledged.

Authors would also like to thank Sebastian Bombinski and Krzysztof Jemielniak for providing us with very valuable experimental data.

References

[1] S. Achiche, M. Balazinski, L. Baron, Real/Binary-Like Coded Genetic Algorithm to Automatically Generate Fuzzy Knowledge Bases, The 4th International Conference on Control and Automation, Montreal, Canada, June 2003.

[2] S. Achiche, M. Balazinski, L. Baron, Multi-Combinative Strategy to Avoid Premature Convergence in Genetically-Generated Fuzzy Knowledge Bases, Journal of Theoretical and Applied Mechanics 42 (3) (2004) 417-444.

[3] S. Achiche, L. Baron, M. Balazinski, Scheduling Exploration/Exploitation Levels in Genetically-Generated Fussy Knowledge Bases, NAFIPS International Conference on Fuzzy Systems, Banff, Canada, June 2004.

[4] M. Balazinski, M. Bellerose, E. Czogala, Application of Fuzzy Logic Techniques to the Selection of Cutting Parameters in Machining Processes, International Journal for Fuzzy Sets and Systems 61 (1993) 307-317.

[5] L. Baron, S. Achiche, M. Balazinski, Fuzzy Decisions System Knowledge Base Generation Using a Genetic Algorithm, International Journal of Approximate Reasoning (2001) 25-148.

[6] S. Bombinski, K. Jemielniak, Hierarchical Strategies In Tool Wear Monitoring, International Conference on Advances in Production Engineering, Warsaw, Poland, 2004.

[7] T. Calinski, J. Harabasz, A Dendrite Method for Cluster Analysis, Communications in Statistics 3 (1974) 1-27.

[8] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2000) 182-200.

[9] S. Dudoit, J. Fridlyand, A prediction-based resampling method for estimating the number of clusters in a dataset, Genome Biology 3 (7) (2002) research0036.1-0036.21.

[10] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Massachusetts, 1989.

[11] X. Guo, Y. Zhou, D. Gong, Optimization of Fuzzy Sets of Fuzzy Control System Based on Hierarchical Genetic Algorithms, IEEE Region 10 Annual International Conference, Proceedings/TENCON, Vol. 3, Beijing, China, 2002, pp. 1463-1466.

[12] J. Han, M. Kambler, Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, 2001.

[13] J.A. Hartigan, Statistical Theory in Clustering, Journal of Classification 2 (1985) 63-76.

[14] F. Herrera, M. Lozano, Gradual Distributed Real-Coded Genetic Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 4, 2000, pp. 43-63.

[15] M. De Hoon, S. Imoto, S. Miyano, The C Clustering Library, Human Genome Center, University of Tokyo, 2004.

[16] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, 1988.

[17] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990.

[18] M. G. Kendall, Rank Correlation Methods, Griffin, 1962.

[19] W. J. Krzanowski, Y. T. Lai, A Criterion for determining the number of clusters in a data set, Biometrics 44 (1985) 22-34.

[20] E. L. Lehmann, H. J. M. D'abrera, Nonparametric: Statistical Methods Based on Ranks, Holden-Day, 1975.

[21] F. G. Lobo, D. E. Goldberg, M. Pelikan, Time Complexity of Genetic Algorithms Exponentially Scaled Problems, GECCO 2000: Proceedings of the Genetic and Evolutionary Computation Conference, Las Vegas, NV, USA, 2000, pp. 151-158.

[22] J. MacQueen, Some Methods for Classification and Analysis of Multivariate Observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, Berkeley, CA, USA, 1967, pp. 281-297.

[23] K. V. Mardia, J. T. Kent, J. M. Bibby, Multivariate Analysis, Academic Press, London, 1979.

[24] Z. Michalewicz, Genetic Algorithms + Data Structure = Evolution Programs, Springer, New York, 1992.

[25] C. A. Sugar, G. M. James, Finding the Number of Clusters in a Data Set: An Informative Theoretic Approach, Journal of the American Statistical Association 98 (2003) 750-763.

[26] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, Journal of the Royal Statistical Society, Series B 63 (2001) 411-423.

[27] L.A. Zadeh, Outline of new Approach to the Analysis of Complex Systems and Decisions Processes, IEEE Transactions of Systems, Man and Cybernetics, Vol. 3, 1973, pp. 28-44.

DISCUSSION DES RESULTATS

Tel que démontré dans les différents tests de validation, les méthodes employées améliorent la qualité des bases de connaissances floues (BCFs) dans la majorité des cas d'application tout en réduisant le temps d'exécution. Les résultats présentés au chapitre 3 indiquent que l'utilisation du prétraitement de données brutes par les algorithmes de clustering fournissent des BCFs plus performantes en présence de jusqu'à 20% d'outliers. Au dessus de cette limite, les BCFs obtenues sans prétraitement sont légèrement moins performantes. Afin de palier à ce problème, il serait préférable de réduire la quantité d'outliers à l'aide d'autres méthodes conçues spécialement au filtrage de données suspectes, telles que les méthodes statistiques ou d'analyse de densités.

Pour ce qui est du bruit, les résultats présentés au chapitre 3 indiquent que les mécanismes inclus dans les algorithmes génétiques sont très robustes et ne nécessitent pas de prétraitement. Toutefois, si on veut réduire le temps d'exécution à l'aide des algorithmes de clustering, il serait préférable de les combiner à un mécanisme de lissage de données brutes, afin d'éviter la perte de performance observée en présence du bruit.

En ce qui attrait aux méthodes de prédéfinition des nombres de sous-ensembles flous présentées au chapitre 5, il est à noter que, une fois les clusters obtenus, seuls les sous-ensembles flous sont utilisés lors de la création de la population initiale. Les règles par contre sont générées aléatoirement. Il serait intéressant de tester également l'impact de l'ajout de la correspondance entre les différents clusters en analysant les données brutes.

Par exemple, on pourrait chercher dans les données brutes, à quel cluster la conclusion a le plus de chance d'appartenir si les prémisses appartiennent à des clusters donnés. Ensuite, à partir de ces informations, on pourrait construire une base de règles afin d'obtenir une base de connaissance complète. Actuellement, étant donnée que les règles sont générées aléatoirement, il se peut qu'une base de connaissance construite à partir des clusters ait un indice de performance très faible dû à une mauvaise association des prémisses aux conclusions.

Finalement, il est à noter que le nombre d'algorithmes de clustering différents est presque infini et que les analyses présentées dans ce mémoire ne se basent que sur les trois algorithmes les plus populaires (AGNES, k-means et k-medoids). Il serait également intéressant de tester le comportement de quelques autres algorithmes de clustering (en particulier le fuzzy c-means basé sur la logique floue) dans différents les cas d'application présentés dans ce mémoire.

CONCLUSION

Le travail de recherche présenté dans ce mémoire vise à optimiser la générationautomatique des bases de connaissances floues (BCFs) à l'aide d'algorithmes génétiques (AGs). Cette optimisation se fait grâce à l'utilisation d'algorithmes de clustering à deux stades différents lors de la génération automatique. Le premier stade se fait sur les données brutes alors que le deuxième se fait lors de l'initialisation de la population initiale des AGs

Les objectifs visent l'amélioration du processus de création automatique sous quatre aspects différents. Tout d'abord, on réduit, par le prétraitement de données brutes, le temps perdu à cause des données redondantes dans l'échantillon de données d'apprentissage. La deuxième action accomplie par le prétraitement est le filtrage de données afin d'éliminer le bruit et les données suspectes (outliers). Finalement, les deux derniers objectifs sont d'accroître la performance des AGs en prédéfinissant le nombre des sous-ensembles flous et, par le fait même, de réduire la nécéssité de supervision humaine.

- Les divers algorithmes de clustering testés, en particulier *k*-means et *k*-medoids, ont permis d'obtenir des améliorations considérables tant du point de vue de temps d'exécution que de la robustesse, de la performance et de l'autonomie. Les résultats présentés au deuxième chapitre montrent que tout en réduisant le temps

d'exécution global de 75%, on obtient une amélioration de 2.4% sur la performance des BCFs obtenues avec un échantillon contenant 10% d'outliers.

- L'application des algorithmes de clustering vise également à rendre la génération automatique des BCFs plus performante et plus autonome en analysant l'échantillon de données d'apprentissage et en prédéfinissant le nombre de sous-ensembles flous. Dans ce cas-ci, l'usage des algorithmes de clustering permet d'accroître la performance des BCFs générées en désactivant dans les AGs le mécanisme de simplification et en concentrant ainsi le travail des AGs sur l'optimisation (précision des BCFs). De plus, on enlève la nécessité de préanalyser les données d'apprentissage afin de déterminer les paramètres optimaux régissant le croisement, la réduction et la mutation.

- Deux méthodes différentes de recherche du nombre de clusters dans un échantillon de données ont été comparées : Silhouette et Hartigan. Aussi, puisque la méthode Hartigan a tendance a surévaluer le nombre de clusters, une combinaison de cette méthode avec la méthode multi-objectif (MO) a été testée. Toutes ces méthodes ont été comparées à la méthode MO, ainsi, un total de quatre méthodes figure dans les résultats des tests de validation.

- Les tests de validation ont été effectués sur des ensembles de données synthétiques et expérimentales. Les données expérimentales ont été recueillies en vue de prédiction de l'usure des outils de coupe. Quatre signaux ont été mesurés sur lesquels une multitude de caractéristiques ont ensuite été calculées. Pour ce

qui est de l'ensemble synthétique, une amélioration moyenne de 15.4% et maximale de 21% ont été obtenues avec la méthode Silhouette. Une amélioration de 28% a été obtenue sur l'ensemble expérimental avec cette même technique.

- Le mémoire dans son ensemble souligne l'importance du prétraitement des données d'apprentissage afin d'obtenir des résultats fiables dans des délais plus courts. Également, les bénéfices d'un travail coopératif de différents domaines de l'intelligence artificielle ont été démontrés.

Les travaux futurs dans le cadre du projet Géno-Flou sauront sans doute accroître davantage la performance et la fiabilité de ce système d'aide à la décision. En particulier, un module de sélection des caractéristiques rendrait le prétraitement des données encore plus significatif. Dans le même ordre d'idées, il serait bénéfique d'implanter d'autres techniques de filtrage. En particulier, des modules supplémentaires de détection de données suspectes, de réduction de bruit et de gestion de données manquantes seraient grandement utiles.

RÉFÉRENCES

[1] Achiche, S., Balazinski, M., Baron, L. 2000. « Génération automatique par un algorithme génétique de bases de connaissances pour un système d'aide à la décision ». *International Conference on Integrated Design and Manufacturing in Mechanical Engineering*. Montréal, Canada.

[2] Achiche, S., Balazinski, M., Baron, L. 2003. « Real/Binary-Like Coded Genetic Algorithm to Automatically Generate Fuzzy Knowledge Bases ». *The 4th International Conference on Control and Automation*. Montreal, Canada.

[3] Achiche, S., Balazinski, M., Baron, L. 2004. « Multi-Combinative Strategy to Avoid Premature Convergence in Genetically-Generated Fuzzy Knowledge Bases ». *Journal of Theoretical and Applied Mechanics* 3:42. P. 417-444.

[4] Achiche, S., Baron, L., Balazinski, M. 2004. « Scheduling Exploration/Exploitation Levels in Genetically-Generated Fussy Knowledge Bases ». *NAFIPS International Conference on Fuzzy Systems*. Banff, Canada.

[5] Balazinski, M., Bellerose, M., Czogala, E. 1993. « Application of Fuzzy Logic Techniques to the Selection of Cutting Parameters in Machining Processes ». *International Journal for Fuzzy Sets and Systems* 61. P. 307-317.

[6] Baron, L., Achiche, S., Balazinski, M. 2001. « Fuzzy Decisions System Knowledge Base Generation Using a Genetic Algorithm ». *International Journal of Approximate Reasoning*. P. 25-148.

[7] Bombinski, S., Jemielniak K. 2004. « Hierarchical Strategies In Tool Wear Monitoring ». *International Conference on Advances in Production Engineering*. Warsaw, Poland.

[8] Calinski, T., Harabasz, J. 1974. « A Dendrite Method for Cluster Analysis ». *Communications in Statistics* 3. P. 1-27.

[9] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. 2000. « A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II ». *IEEE Transactions on Evolutionary Computation* 6. P. 182-200.

[10] De Hoon, M., Imoto, S., Miyano, S. 2004. *The C Clustering Library*. Human Genome Center, University of Tokyo. http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/cluster.pdf (Page consultée le 23 juin 2005).

[11] Dudoit, S., Fridlyand, J. 2002. « A Prediction-based Resampling Method for estimating the Number of Clusters in a Dataset ». *Genome Biology*. 3:7 research0036.1-0036.21.

[12] Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Massachusetts.

[13] Guo, X., Zhou, Y., Gong, D. 2002. « Optimization of Fuzzy Sets of Fuzzy Control System Based on Hierarchical Genetic Algorithms ». *IEEE Region 10 Annual International Conference, Proceedings/TENCON* 3. Beijing, China. P. 1463-1466.

[14] Han, J., Kambler, M. 2001. *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers.

[15] Hancock, H. 1939. *Development of the Minkowski Geometry of Numbers*. The Macmillan Company, New York.

[16] Hartigan, J.A. 1975. *Clustering Algorthms*. John Wiley & Sons. New York, USA.

[17] Hartigan, J. A. 1985. « Statistical Theory in Clustering ». *Journal of Classification*. 2. P. 63-76.

[18] Herrera, F., Lozano, M. 2000. « Gradual Distributed Real-Coded Genetic Algorithms ». *IEEE Transactions on Evolutionary Computation* 4. P. 43-63.

[19] Holland, J. H. 1975. *Adaptation in Natural and Artificial System*. The University of Michigan Press.

[20] De Hoon, M., Imoto, S., Miyano, S. 2004. *The C Clustering Library*, Human Genome Center, University of Tokyo.

[21] Jain, A.K., Dubes, R.C. 1988. *Algorithms for clustering data*, Prentice-Hall.

[22] Kaufman, L., Rousseeuw, P.J. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons.

[23] Kendall, M. G. 1962. *Rank Correlation Methods*. Griffin.

[24] Krzanowski, W. J., Lai, Y. T. 1985. « A Criterion for determining the Number of Clusters in a Data Set ». *Biometrics*. 44. P. 22-34.

[25] Lehmann, E. L., D'Abrera, H. J. M. 1975. *Nonparametric: Statistical Methods Based on Ranks*. Holden-Day.

[26] Lobo, F. G., Goldberg, D. E., Pelikan, M. 2000. « Time Complexity of Genetic Algorithms Exponentially Scaled Problems ». *GECCO 2000: Proceedings of the*

Genetic and Evolutionary Computation Conference. Las Vegas, NV, USA. P. 151-158.

[27] MacQueen, J. 1967. « Some Methods for Classification and Analysis of Multivariate Observations ». *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1. Berkeley, CA, USA. P. 281-297.

[28] Mardia, K. V., Kent, J. T., Bibby, J. M. 1979. *Multivariate Analysis*. Academic Press, London.

[29] Michalewicz, Z. 1992. « Genetic Algorithms + Data Structure = Evolution Programs ». Springer, New York.

[30] Sugar, C. A., James, G. M. 2003. « Finding the Number of Clusters in a Data Set: An Informative Theoretic Approach ». *Journal of the American Statistical Association*. 98. P. 750-763.

[31] Tibshirani, R., Walther, G., Hastie, T. 2001. « Estimating the Number of Clusters in a Data Set via the Gap Statistic ». *Journal of the Royal Statistical Society, Series B*. 63. P. 411-423.

[32] *Wikipedia, the Free Encyclopedia*. http://www.wikipedia.org/.
(Page consultée le 27 janvier 2005).

[33] Zadeh, L. A. 1973. « Outline of new Approach to the Analysis of Complex Systems and Decisions Processes ». *IEEE Transactions of Systems, Man and Cybernetics*. 3. P. 28-44.

ANNEXE A – MANUEL DU LOGICIEL GFS CLUSTER

Le logiciel GFS Cluster a été développé en Visual C++/MFC pour les fins d'analyse et de validation des algorithmes présentés dans ce mémoire. Il permet de générer des distributions d'objets dans des clusters à partir d'un ensemble de données brutes. Plusieurs algorithmes de clustering sont disponibles et, pour chacun d'entre eux, divers paramètres peuvent être ajustés. Le manuel est divisé en 3 sections : (A1) interface usager, (A2) fenêtres de dialogue pour le clustering et pour le nombre de clusters et (A3) format des fichiers d'importation et d'exportation.

A1 Interface usager

L'interface usager se divise en 4 parties : la fenêtre principale, le menu, la barre d'outils et la boite de propriétés. La figure A.1 présente une capture d'écran de cette interface. Tout d'abord, la fenêtre principale permet de parcourir les coordonnées des objets contenus dans l'échantillon de données et, si une opération de clustering a déjà été effectuée, les objets contenus dans chaque cluster. Afin d'accéder aux coordonnées d'un objet ou aux objets contenus dans un cluster, il suffit de double-cliquer sur l'objet ou sur le cluster voulu.

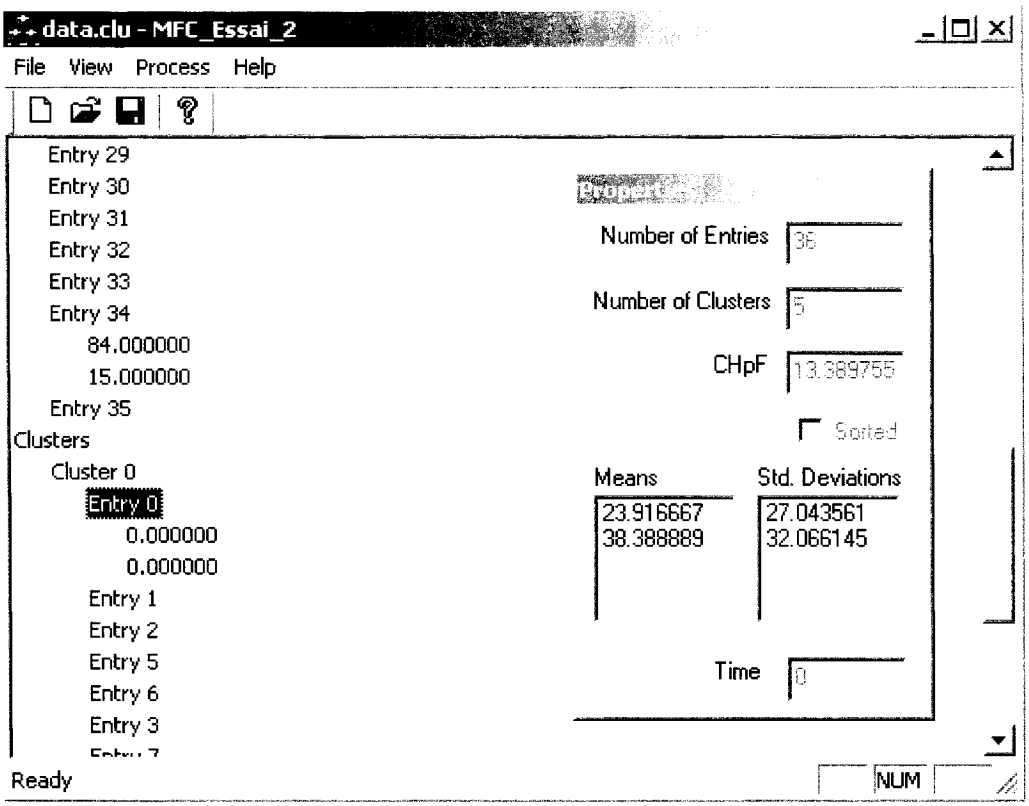

Figure A.1 – Capture d'écran de l'interface graphique du logiciel GFS Cluster

Ensuite, le menu contient toutes les fonctions accessibles dans le logiciel et la barre d'outils regroupe celles qui sont le plus souvent utilisées. La liste qui suit présente toutes les fonctions accessibles à partir du menu :

- File Ensemble des fonctions relatives aux fichiers
  - New Créer un nouveau fichier
  - Open... Ouvrir un fichier existant (données et clusters)
  - Save Enregistrer le fichier courant (données et clusters)
  - Save As... Enregistrer le fichier sous un nouveau nom

- Import... Importer un fichier ASCII contenant des données brutes
- Export... Exporter les clusters dans un fichier ASCII
- Exit Quitter le logiciel

- View
  - Toolbar Afficher/Cacher la barre d'outils
  - Status Bar Afficher/Cacher la barre de statut
  - Properties Toolbar Afficher/Cacher la boite de propriétés
- Process
  - Cluster... Générer les clusters
  - Batch Clustering... Générer les clusters sur plusieurs fichiers
  - Number of Clusters... Chercher le nombre de clusters optimal
  - Batch Number... Chercher le nombre de clusters optimal sur plusieurs fichiers
  - Sort Entries Trier les objets selon leurs coordonnées
- Help
  - About GFS Cluster... Afficher les renseignements sur la version du logiciel

La barre d'outils contient quatre icônes :

Créer un nouveau fichier

Ouvrir un fichier existant

Enregistrer le fichier courant

Afficher les renseignements sur la version du logiciel

Finalement, la boite de propriétés affiche certaines informations principales sur les objets chargés ainsi que sur les clusters. Voici la liste et description des champs contenus dans cette boite :

Number of Entries	Nombre d'objets contenus dans l'échantillon de données
Number of Clusters	Nombre de clusters
CHpF	L'indice Calinski-Harabasz pseudo-F (section 3.2)
Sorted	Données triées (case cochée) ou non (case non cochée)
Means	Les moyennes des coordonnées selon chaque dimension
Std. Deviations	Les écarts types des coordonnées selon chaque dimension
Time	Le temps d'exécution de l'opération de clustering

A2 Fenêtres de dialogue pour le clustering et pour le nombre de clusters

La figure A.2 présente la fenêtre de dialogue pour le clustering. Celle-ci offre plusieurs options quant aux différents algorithmes de clustering offerts.

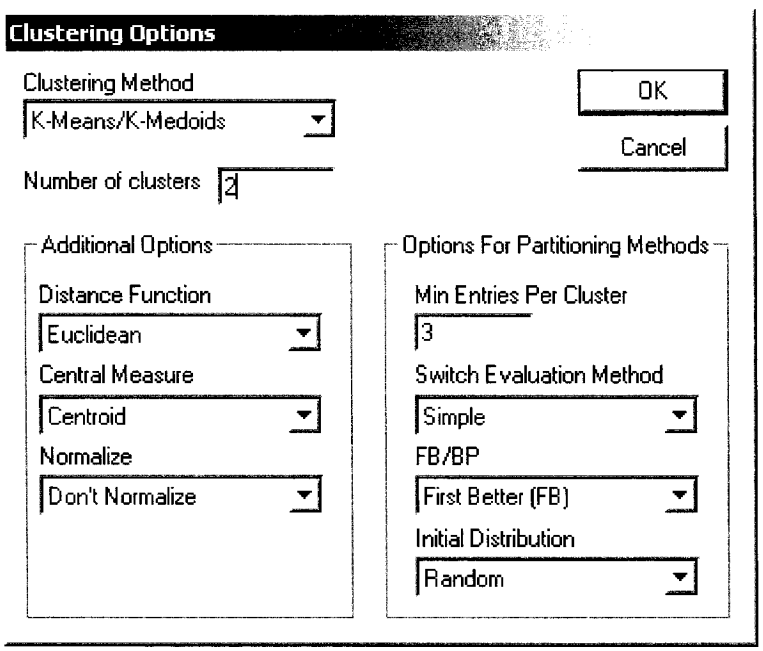

Figure A.2 – Fenêtre de dialogue pour le clustering

La liste qui suit présente les descriptions de chaque champ contenu dans la fenêtre de dialogue :

Clustering Method	Choix de l'algorithme de clustering (section 1.3.2)
AGNES	Algorithme hiérarchique AGNES
k-means/k-medoids	Algorithmes à partitionnement *k*-means et *k*-medoids
Number of Clusters	Nombre de clusters désiré
Distance Function	Choix de la fonction de distance (section 1.3.5)

Euclidean	Distance euclidienne
Harmonic	Somme harmonique
Minkowski	Distance de Minkowski généralisée
Canberra	Corrélation de Canberra
Correlation	Corrélation
Central Measure	Choix de la mesure centrale des clusters (section 1.3.6)
Centroid	Centroïde
Medoid	Medoïde
Normalize	Normalisation des données
Don't Nomrmalize	Aucune normalisation
Range	Normaliser à l'intérieur de [0, 1]
Standard Deviation	Soustraire la moyenne et diviser par l'écart type
Min Entries Per Cluster	Nombre d'objets par cluster minimal
Switch Evaluation Method	Type d'évaluation de voisinage
Simple	Évalue la distance de l'objet aux deux clusters
Advanced	Évalue l'erreur de reconstruction avant et après
FB/BP	Type de l'algorithme à partitionnement (section 2.1)

First Better (FB)	Procède à l'échange d'objet dès qu'une meilleure distribution est trouvée
Best Possible (BP)	Procède à l'échange d'objet dès que la meilleure distribution est trouvée
Initial Distribution	Distribution initiale (section 1.3.3)
Random	Distribution aléatoire
Sequential	Distribution séquentielle
Random Means	Centroïdes aléatoires
Sequential Means	Centroïdes séquentiels
Furthest Means	Centroïdes éloignés
MaxMin	Plus grande distance minimale
AGNES	Distribution initiale obtenue par l'algorithme hiérarchique AGNES

La figure A.3 présente la fenêtre de dialogue pour le nombre de clusters. Celle-ci est très similaire à celle présentée à la figure A.2. La seule différence est que le champ « Number of Clusters » est remplacé par « Validation Method » (section 4.2) qui offre les choix suivants :

CHpF	L'indice Calinski-Harabasz pseudo-F défini par (éq. 4.1)
Hart	L'indice Hartigan défini par (éq. 4.2)

KL	L'indice Krzanowski & Lai défini par (éq. 4.3)
Silhouette	L'indice Silhouette défini par (éq. 4.4)

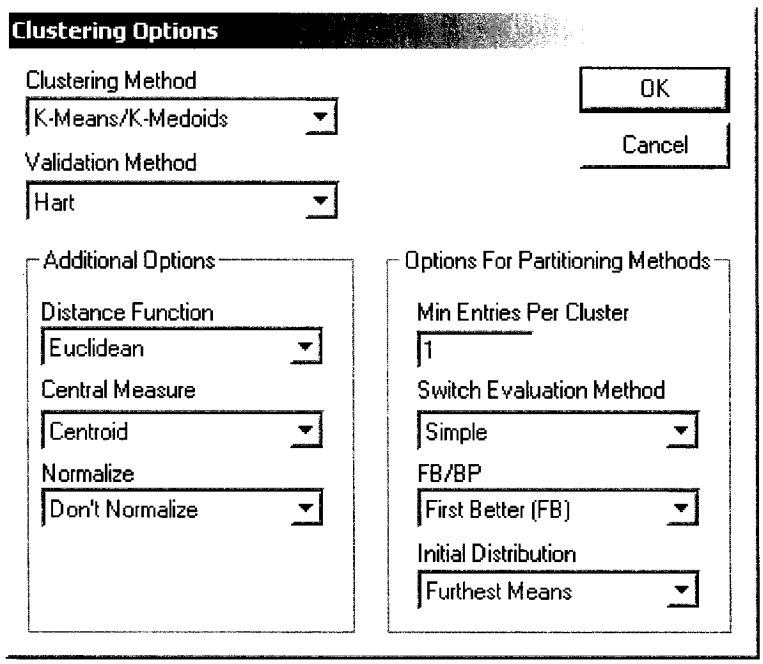

Figure A.3 – Fenêtre de dialogue pour le nombre de clusters

A3 Format des fichiers d'importation et d'exportation

Afin d'importer les données brutes dans le logiciel GFS Cluster, il faut d'abord les présenter dans un format que le logiciel peut reconnaître. Tout d'abord, il faut que le fichier soit dans le format ASCII. Ensuite, la première ligne doit contenir le nombre de variables de l'échantillon de données. Finalement, chaque ligne qui suit doit contenir les coordonnées de chaque objet séparées par des espaces ou des tabulations.

Pour ce qui est du format des fichiers d'exportation, il s'agît également de fichiers ASCII. Le contenu du fichier est simplement constitué des coordonnées du centroïde ou du médoïde (selon la mesure centrale choisie) de chaque cluster. Ces coordonnées sont

disposées une ligne par cluster et séparées par des tabulations. La figure A.4 présente un exemple de fichier d'importation en format ASCII.

```
3
0.154236      2.462378      10.116538
2.423744      7.328945      -2.248329
3.237982      -2.423876     -102.653842
45.636324     -21.231563    1001.125153
!             !             !
```

Figure A.4 – Exemple de fichier d'importation en format ASCII