ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE À L'OBTENTION
DE LA MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M. ING.

PAR
NADIA BENAHMED

OPTIMISATION DE RÉSEAUX DE NEURONES POUR LA RECONNAISSANCE DE CHIFFRES MANUSCRITS ISOLÉS : SÉLECTION ET PONDÉRATION DES PRIMITIVES PAR ALGORITHMES GÉNÉTIQUES

MONTRÉAL, LE 1ER MARS 2002

OPTIMISATION DE RÉSEAUX DE NEURONES POUR LA RECONNAISSANCE DE CHIFFRES MANUSCRITS ISOLÉS : SÉLECTION ET PONDÉRATION DES PRIMITIVES PAR ALGORITHMES GÉNÉTIQUES

Nadia Benahmed

SOMMAIRE

La sélection des primitives est une étape importante dans tout système de reconnaissance de formes. Cette sélection des primitives est considérée comme un problème d'optimisation combinatoire et a fait l'objet de recherche dans de nombreuses disciplines. L'objectif principal de la sélection des primitives est de réduire le nombre de celles-ci en éliminant les primitives redondantes et non pertinentes du système de reconnaissance. Le second objectif de cette sélection de primitives est aussi de maintenir et/ou améliorer la performance du classifieur utilisé par le système de reconnaissance. Les algorithmes génétiques sont utilisés pour résoudre ce type de problème de sélection des primitives dans la reconnaissance de chiffres isolés.
Les résultats obtenus lors de la sélection des primitives ont permis de réduire la complexité du réseau de neurones utilisé. Le nombre de primitives a été réduit de 25% par rapport à l'ensemble des primitives extraites du système de reconnaissance de chiffres manuscrits isolés tout en maintenant la performance en taux de reconnaissance du système.

OPTIMIZATION OF THE NEURAL NETWORKS ON ISOLATED HANDWRITTEN DIGITS: SELECTION AND WEIGHT FEATURES BY GENETIC ALGORITHMS

Nadia Benahmed

ABSTRACT

Feature selection is a significant stage in all pattern recognition systems. Feature selection can be considered as a global combinatorial optimization problem and is an active research subject. The first goal through the task of feature selection process is to reduce the number of features by throwing out the redundant and irrelevant features from the set of features. The second objective is also to maintain and/or to increase the performance of the classifier used by the recognition system. The genetic algorithms are used to solve the problem of feature selection for the recognition of the isolated handwritten digits.

The results obtained allow reducing the complexity of the neural networks used. The number of features was decrease by 25% compared to the subset of features that are extracted from the recognition system on the isolated handwritten digits while maintaining the performance.

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

- M. Robert Sabourin, directeur de mémoire
  Département de génie de la production automatisée à
  l'École de Technologie Supérieure

- M. Ching Y. Suen, codirecteur
  CENPARMI, Centre for Pattern Recognition and Machine Intelligence à
  l'Université de Concordia

- M. Tony Wong, professeur
  Département de génie de la production automatisée à
  l'École de Technologie Supérieure

- M. Mohamed Cheriet, professeur
  Département de génie de la production automatisée à
  l'École de Technologie Supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET UN PUBLIC

LE 4 FEVRIER 2002

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AVANT-PROPOS ET REMERCIEMENTS

La réalisation de ce mémoire a été rendu possible grâce à des bourses du CRSNG (Conseil de Recherches en Sciences Naturelles et en Génie du Canada) et de l'ETS (École de Technologie Supérieure).

Je remercie sincèrement mon directeur de mémoire Robert Sabourin professeur au département de génie de la production automatisée à l'ÉTS (École de Technologie Supérieure) et mon co-directeur Ching Y.Suen directeur du CENPARMI (Centre for Pattern Recognition and Machine Intelligence) à l'université Concordia, sans l'initiative desquels ce projet n'aurait pas été possible. Je tiens à leur exprimer toute ma reconnaissance pour leur dévouement, la confiance qu'ils m'ont accordé, leur rigueur et la qualité des commentaires et suggestions dont ils m'ont fait part.

Je remercie Luiz Oliveira pour son assistance à la réalisation de ce mémoire et Patrick Maupin pour avoir corrigé et revu mes textes. Un grand merci à tous les membres du LIVIA (Laboratoire en imagerie, vision et intelligence artificielle) pour leur enthousiasme et leur bonne humeur quotidienne. Ce fut un plaisir de les côtoyer.
Merci également à Isabelle Paré, de l'ÉTS, pour ses encouragements soutenus.

Enfin, je remercie particulièrement mon mari, Mourad, pour sa compréhension et son support durant mes études, mes parents, ma sœur, Fatma, mes frères, Kamel et Abdeldjelil, pour leur soutien moral malgré la distance qui nous sépare, sans oublier ma famille et tous mes amis d'ici et d'ailleurs.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

LISTE DES FIGURES

Page

LISTE DES GRAPHIQUES

Page

LISTE DES SYMBOLES

f	fonction d'activation du réseau de neurones.
w	poids synaptique.
Z	vecteur de primitives.
$P(C_i\|Z)$	probabilité *a posteriori.*
$P(Z\|C_i)$	probabilité *a priori.*
C_k	classe d'appartenance.
η	constante d'apprentissage
d_j	valeur désirée du neurone j.
δ	Signal d'erreur du neurone j.
E_p	erreur quadratique.
N_c	nombre entier de neurones cachés.
N_s	nombre de neurones en sortie.
N_e	nombre d'entrées du réseau de neurones.
C^i	le $i^{ème}$ chromosome.
c^i_k	le $k^{ème}$ gène du $i^{ème}$ chromosome.
P	population de l'AG.
p_{cross}	probabilité de croisement.
p_{mut}	probabilité de mutation.
taille_pop	nombre de chromosomes dans la population.
nbre_gen	nombre de génération.
L	nombre de gènes dans un chromosome.
fitness	fonction d'adaptation pour la sélection des primitives.
$fitness_{pond}$	fonction d'adaptation pour la pondération des primitives.
f_1	taux d'erreur calculé sur la base lors de la sélection.
f_2	nombre de primitives sélectionnées.
f_3	taux de reconnaissance sur la base lors de la pondération.
$d(C^1,C^2)$	distance de hamming calculée entre C^1 et C^2.
S_i	sensibilité de la $i^{ème}$ entrée du réseau de neurons.

LISTE DES ABRÉVIATIONS ET DES SIGLES

AGS	Algorithme génétique simple
AGI	Algorithme génétique itératif
RNA	Réseau de neurones artificiel
MLP	Multi layer perceptron : réseau de neurones multicouches
SE	Squared error : erreur quadratique
NIST SD	National Institute of Standards and Technology, Special Database
LIVIA	Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle

INTRODUCTION

La reconnaissance de l'écriture manuscrite n'est pas un sujet nouveau. Il remonte à plus d'une trentaine d'années. La reconnaissance de l'écriture manuscrite s'avère un problème extrêmement complexe qui n'a pas de solution satisfaisante à ce jour. Ainsi rapidement, les chercheurs ont restreint leurs études à des problèmes particuliers en liaison avec des applications bien définies. Parmi celles-ci on cite les applications de la reconnaissance de l'écriture manuscrite : le tri automatique du courrier postal, le traitement automatique de dossiers administratifs, des formulaires d'enquêtes, ou encore l'enregistrement des chèques bancaires.

Dans le domaine de la reconnaissance de l'écriture, les primitives peuvent être décrites comme un moyen permettant de distinguer un objet (mot, lettre, chiffre) d'une classe d'un autre objet (mot, lettre, chiffre) d'une autre classe. Dès lors, il est nécessaire de définir des primitives significatives lors du développement d'un système de reconnaissance. Les primitives sont généralement définies par expérience ou par intuition. Plusieurs primitives peuvent être extraites. La représentation des primitives utilisée est une représentation vectorielle. La taille du vecteur peut être large si un grand nombre de primitives est extrait. Il a été observé que certaines primitives extraites sont non pertinentes ou redondantes au système de reconnaissance. La sélection de primitives pertinentes est par contre un problème complexe et fait l'objet de nombreuses recherches.

Nous proposons dans le cadre de ce travail une méthode d'optimisation pour la sélection et la pondération des primitives d'un système de reconnaissance de chiffres manuscrits isolés. Cette étude que nous avons mené est une phase exploratoire du projet de thèse de doctorat intitulé « Reconnaissance des chiffres manuscrits : Application à la lecture automatique des chèques bancaires ».

L'objectif de ce travail est d'orienter une partie du projet de recherche qui consiste à sélectionner des primitives pertinentes. Cette sélection des primitives permet d'optimiser le système de reconnaissance de l'écriture manuscrite. La méthode

d'optimisation utilisée est basée sur la théorie des algorithmes génétiques. Dans ce travail, l'optimisation du système de reconnaissance est multicritères. Le premier critère consiste à diminuer le nombre d'entrées (primitives) du réseau de neurones, alors que le second critère doit améliorer ou maintenir la performance du système de reconnaissance. Lors de l'application des algorithmes génétiques, les entrées du réseau de neurones sont diminuées. Le problème qui se pose est d'effectuer l'apprentissage du réseau pour chaque solution trouvée. Pour éviter de relancer l'apprentissage du réseau et pour minimiser le temps de calcul, la technique d'analyse de sensibilité est donc appréciable dans ce cas de figure. Cette technique consiste à remplacer les entrées ou les primitives supprimées du réseau de neurones par la moyenne calculée de l'ensemble des primitives de toute la base d'apprentissage.

Une seconde étape s'ajoute à ce travail qui est la pondération des primitives. La pondération des primitives consiste à classer les primitives d'après leur importance respective. Elle permet d'identifier les primitives les plus importantes du système de reconnaissance. Cette approche est très délicate et peut ne pas aboutir à une solution car le réseau de neurones a déjà généré des poids de chaque entrée lors de l'apprentissage. Cependant, nous avons suggéré d'effectuer la pondération des primitives une fois que la sélection est terminée. Cela consiste à récupérer les primitives sélectionnées par l'algorithme génétique utilisant la technique de sensibilité et à relancer l'apprentissage du réseau avec cette solution. Ainsi la matrice de poids est générée à nouveau avec un nombre réduit d'entrées du réseau. La pondération des primitives par l'algorithme génétique est donc effectuée sur ce nouveau réseau de neurones.

Dans ce mémoire, nous présentons une revue bibliographique sur la reconnaissance de caractères, les réseaux de neurones et les algorithmes génétiques en général. Nous exposons ensuite la démarche suivie pour l'optimisation du système de reconnaissance de chiffres isolés. Nous illustrons les résultats des expériences réalisées que nous analysons.

Le chapitre 1 présente l'état de l'art dans le domaine de la reconnaissance de caractères où nous décrivons les étapes nécessaires au développement d'un système de reconnaissance de caractères. Une revue de littérature sur des méthodes de sélection des primitives est élaborée.

Le chapitre 2 est consacré à l'état de l'art des réseaux de neurones, entre autre le MLP adopté dans notre projet. Une description de la méthode d'apprentissage utilisée est aussi présentée.

Le chapitre 3 est une revue de littérature détaillée sur les algorithmes génétiques. Dans cette partie, nous expliquons les différentes étapes de l'algorithme génétique, les opérateurs de reproduction (croisement et mutation) et le choix des paramètres. Nous discutons également l'approche algorithme génétique simple versus l'algorithme génétique itératif.

Le chapitre 4 décrit les étapes de la reconnaissance appliquées dans notre système. En premier, nous décrivons la base de données. En second lieu, nous abordons la phase d'extraction de primitives suivie par la sélection de ces dernières où nous décrivons la technique de l'analyse de sensibilité utilisée. Ensuite nous entamons la pondération des primitives suivie par la partie classification. Enfin, la méthodologie adoptée dans le projet est illustrée.

Le chapitre 5 présente les expériences effectuées dans le cadre de ce projet et les résultats obtenus.

Le chapitre 6 est consacré à l'analyse des résultats présentés dans le chapitre précédent.

Nous finirons ce rapport par une conclusion et des recommandations et des suggestions sur les travaux futurs dans ce domaine de recherche.

CHAPITRE 1

LA RECONNAISSANCE DES CARACTÈRES

1.1 Introduction

Depuis toujours, l'homme est fasciné par les automates. Les premiers automates amusaient et effectuaient des tâches répétitives. De nos jours ils sont dotés d'organes sensoriels leur permettant de voir, d'entendre et même de goûter. Parmi les automates les plus perfectionnés, on retrouve ceux qui sont capable de déchiffrer l'écriture humaine. Ce domaine de l'écriture reste toujours à explorer étant donnée sa complexité et sa diversité. On peut distinguer deux axes de recherche : la reconnaissance des caractères imprimés et la reconnaissance des caractères manuscrits.

Dans le cas de la reconnaissance de l'écriture manuscrite, il existe deux champs d'applications suivant le mode de saisie. Si le mode de saisie est dynamique, on parle de reconnaissance en temps réel appelée *en ligne* ou *on-line*. Dans le cas de la saisie statique on parle alors de reconnaissance en temps différé appelée *hors-ligne* ou *off-line* [1,9]. La reconnaissance hors-ligne s'applique une fois que l'écriture est sur un support papier qui est numérisé puis enregistré sur un format d'image. Cette image contient des pixels soit de type binaire (pixels noirs et blancs) soit de type entier (image niveaux de gris). Cette reconnaissance hors-ligne regroupe deux thèmes : la reconnaissance de caractères manuscrits isolés (numériques ou alphanumériques) et la reconnaissance de mots. Les applications actuelles visent de plus en plus la lecture automatique de documents manuscrits du type chèques bancaires ou courrier postal. Cas études portent sur la reconnaissance de caractères manuscrits provenant de la segmentation de chiffres comme par exemple le montant numérique d'un chèque ou le code postal d'une adresse ou de la segmentation en lettres minuscules et/ou majuscules dans le cas par exemple du montant littéral d'un chèque ou du nom de la ville figurant sur une adresse. Ces caractères isolés présentent de fortes variations principalement provoquées par la position de la lettre dans le mot.

En général, l'objectif de la reconnaissance de l'écriture manuscrite est de développer un système qui se rapproche le plus de l'être humain dans sa capacité de lire. Cependant, cette reconnaissance de l'écriture consiste à extraire d'une forme inconnue (mot, lettres, chiffres) une description plus simple et à établir sur celle-ci une décision. Cette décision est effectuée généralement en mesurant la ressemblance d'une forme inconnue avec un ensemble de références stockées en mémoire et décrites dans une représentation analogue. Les références sont obtenues lors d'une phase antérieure qualifiée d'apprentissage. Cette phase est très importante dans tout système de reconnaissance de l'écriture. Autrement dit c'est un passage de l'espace observable vers un espace de décision d'appartenance à une classe.

La construction d'un système de reconnaissance de l'écriture comprend plusieurs étapes distinctes. Les étapes qui sont décrites par Duda et al [2] sont représentées par la figure 1:

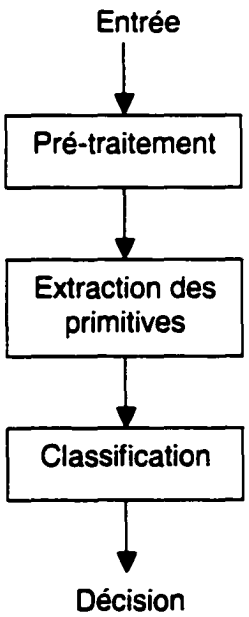


Figure 1 - Les étapes de la reconnaissance de formes

L'entrée du système acquière soit une forme ou un signal par un capteur (appareil photo, scanner, magnétophone, etc). Cette entrée peut être une image scannée ou un

signal vocal qui sera stocké dans un fichier. Par la suite plusieurs traitements sont opérés sur ces images et ces fichiers. Le but de ces pré-traitements est d'éliminer les phénomènes qui provoquent une dégradation des performances du système, de réduire les bruits de quantification (binarisation) et de préserver la connexité des composantes connexes dans l'image. Le résultat de cette phase permettra d'extraire ou de mettre en évidence des particularités locales ou globales. Cette étape permet d'engendrer pour chaque image, un vecteur de primitives qui sert comme entrée au module responsable de la classification.

Notre étude porte sur le thème de la reconnaissance hors-ligne des images binaires de caractères isolés binaires (pixels noirs et blancs). Ces caractères représentent les chiffres arabes de 0 à 9. Le fait que les chiffres soient isolés (segmentés) garantit une homogénéité de la forme. Cette étude s'est basée sur les travaux de Oliveira et al [3, 4].

Dans la section suivante nous abordons l'état de l'art de chaque phase de la reconnaissance à savoir : les pré-traitements, l'extraction des primitives et la classification.

1.2 Pré-traitements

L'objectif des pré-traitements est de faciliter la caractérisation de la forme (caractère, chiffre, mot) ou de l'entité à reconnaître soit en nettoyant l'image représentant la forme ou en réduisant la quantité d'information à traiter pour ne garder que les informations les plus significatives. Le nettoyage de l'image consiste essentiellement à éliminer les bruits résiduels issus de la binarisation. La réduction de la quantité d'information à traiter peut être obtenue à partir des opérations visant à ramener l'épaisseur du trait à un seul pixel (soit par squelettisation [6], soit par suivi de trait [6]) ou à partir d'extracteurs de contours supérieurs, inférieurs et/ou intérieurs.
Notons que certaines formes (caractères, chiffres, mots) sont inclinées ou penchées donc il est nécessaire de normaliser en pente cette forme afin de segmenter la forme (par exemple segmentation d'un mot en lettres). Cette normalisation consiste à corriger

la pente d'un mot ou à redresser l'inclinaison des lettres dans un mot afin de faciliter la segmentation.
La segmentation est une phase des pré-traitements. Son but est de localiser et d'extraire le plus précisément possible les informations à reconnaître. Elle comprend deux étapes : la binarisation et la localisation des informations à reconnaître. La binarisation consiste à réduire la quantité d'informations à traiter lorsque l'image est en niveaux de gris et la localisation des entités à reconnaître concerne des formes comme des lignes, les mots, les groupes de lettres ou les lettres. Le résultat obtenu lors de cette phase de pré-traitement permettra le passage vers la phase d'extraction.
Pour plus de détails, les ouvrages de Belaid et Belaid [5] et Abhijit et al [10] traitent de cette phase.

1.3 Extraction des primitives

L'objectif de l'extraction des caractéristiques dans le domaine de la reconnaissance consiste à exprimer les primitives sous une forme numérique ou symbolique appelée *codage*. Selon le cas, les valeurs de ces primitives peuvent être réelles, entières ou binaires. Le vecteur des n primitives constituées représente un point dans le nouvel espace à n dimensions. Cette étape de la reconnaissance consiste à extraire des caractéristiques permettant de décrire de façon non équivoque les formes appartenant à une même classe de caractères tout en les différenciant des autres classes.
Trier et al [6] ont précisé que la sélection et la définition des primitives lors de l'extraction est la partie la plus importante dans le système de reconnaissance, car le choix de celles-ci dépend du type de problème à résoudre.

Il existe différentes méthodes d'extraction de primitives citées dans la littérature. Trier et al [6] décrivent des méthodes d'extraction en fonction de la représentation du caractère. Cette extraction se fait sur des images en niveaux de gris, en binaire, en contour binaire et selon une représentation vectorielle. Les primitives sont classées en deux catégories [7] : les primitives structurelles (ou primitives locales) basées sur une représentation linéaire du caractère (décomposition du caractère en segments de droites et courbes, contours du caractère, squelette) et les primitives statistiques (ou

primitives globales) basées sur la distribution des points, les transformations et les mesures physiques faites sur le caractère (surface, moments, projections). Ces deux types peuvent être combinés formant un vecteur de primitives.

1.3.1 Primitives locales

Ces primitives sont des objets de la forme tel que les extrémités, les croisements de traits, les boucles et les courbes. L'inconvénient de ces primitives est que leur extraction nécessite une squelettisation préalable du caractère, puisque l'épaisseur du trait ne contient pas d'information. Néanmoins ce sont des primitives très robustes vis à vis de la rotation, translation, homothétie [8]. Dans cette catégorie, il existe 4 familles de caractéristiques : intersections avec des droites, arcs concaves et occlusions, extremas et jonctions [7].

1.3.2 Primitives globales

Ces primitives sont dérivées de la distribution des pixels. Heutte et al [7] suggèrent 3 familles de caractéristiques telles que : les moments invariants, les projections, et les profils. Elles sont extraites en considérant la distribution des pixels noirs de l'objet (caractère, mot, chiffres).

Le processus d'identification de la meilleure méthode d'extraction de caractéristiques n'est pas évident. Par exemple, Trier et al [6] rapportent que les moments de Zernike s'appliquent bien sur des images à niveaux de gris et que la projection s'applique souvent sur des caractères segmentés pour résoudre leur problème de reconnaissance de l'écriture manuscrite.

Cependant, il est nécessaire d'effectuer pour chaque problème de reconnaissance une évaluation expérimentale de quelques méthodes d'extraction de primitives les plus prometteuses. Ces expériences permettront de faire un choix judicieux des primitives à extraire car souvent, l'utilisation d'une seule méthode d'extraction de caractéristiques n'est pas suffisante pour obtenir de bonne performance du système de reconnaissance.

La solution évidente est de combiner plusieurs méthodes d'extraction afin de donner une meilleure description de la forme (caractère, chiffre, mot) à classer.

1.4 Classification

La classification est l'élaboration d'une règle de décision qui transforme les attributs caractérisant les formes en appartenance à une classe (passage de l'espace de codage vers l'espace de décision) [2]. Avant qu'un modèle de décision ne soit intégré dans un système de reconnaissance de l'écriture, il faut avoir procédé auparavant à deux étapes : l'étape d'*apprentissage* et l'étape de *test*.

L'étape d'apprentissage consiste à caractériser les classes de formes de manière à bien distinguer les familles homogènes de formes. C'est une étape clé dans le système de reconnaissance. On distingue deux types d'apprentissage : apprentissage *supervisé* et apprentissage *non supervisé* [2].
Dans le cas de l'apprentissage supervisé, un échantillon représentatif de l'ensemble des formes à reconnaître est fourni au module d'apprentissage. Chaque forme est étiquetée par un opérateur appelé professeur, cette étiquette permet d'indiquer au module d'apprentissage la classe dans laquelle le professeur souhaite que la forme soit rangée. Cette phase d'apprentissage consiste à analyser les ressemblances entre les éléments d'une même classe et les dissemblances entre les éléments de classes différentes pour en déduire la meilleure partition de l'espace des représentations. Les paramètres décrivant cette partition sont stockés dans une table d'apprentissage à laquelle le module de décision se référera ensuite pour classer les formes qui lui sont présentées.
Dans le cas de l'apprentissage non supervisé, on fournit au système de reconnaissance un grand nombre de formes non étiquetées. L'étape de la classification va se charger d'identifier automatiquement les formes appartenant à une même classe. Dans le domaine de la reconnaissance de l'écriture, ce sont les méthodes basées sur un apprentissage supervisé qui sont le plus souvent utilisées ; et plus particulièrement pour les caractères manuscrits isolés car les classes naturelles sont connues et en nombre limité.

L'étape de test permet d'évaluer la performance du classifieur pour un apprentissage donné. C'est une étape importante car elle peut mettre en cause le choix des primitives ou le choix de la méthode d'apprentissage. En effet, il est difficile de trouver *a priori* les primitives pertinentes et la méthode d'apprentissage la plus adaptée au problème posé d'où l'utilité de procéder par itérations successives. Ces itérations consistent à extraire des primitives jugées utiles au problème de reconnaissance à résoudre et de tester la performance du système avec cet ensemble de primitives. Au fur et à mesure que les performances du système souhaitées ne sont pas atteintes alors il suffit de trouver à nouveau une nouvelle famille de primitives ou de combiner les primitives extraites avec de nouvelles primitives.

Ces phases d'apprentissage et de test sont réalisées préalablement à l'intégration du module de décision dans le système de reconnaissance. Dans tous les cas, on peut permettre au système de reconnaissance d'itérer les phases d'apprentissage et de test tant que l'on n'a pas atteint les performances désirées. Le calcul de cette performance est le résultat du classifieur utilisé. Pour construire un classifieur, il existe plusieurs approches. Abhijit et al [10] ont défini deux approches : l'approche structurelle et l'approche statistique. Par contre Schalkoff [11] a défini trois approches : structurelle, statistique et les réseaux de neurones. Les deux premières sont identiques à celles d'Abhijit qui a précisé que l'utilisation de l'approche statistique est adéquate lorsque l'information structurelle n'est pas disponible ou non pertinente. Dans le cas contraire c'est à dire si l'information structurelle est disponible alors l'approche structurelle est adoptée. L'approche réseau de neurones est une implémentation dérivée de l'approche statistique et de l'approche structurelle.
Dans notre cas, nous abordons les 2 approches : structurelle et statistique en considérant que les réseaux de neurones font partie des approches statistiques.

1.4.1 Approche structurelle / syntaxique

C'est une approche qui est basée sur l'extraction de primitives en prenant compte de l'information structurelle. Cette approche cherche à structurer l'information en décrivant l'organisation topologique (la structure) de la forme à partir de ses composantes les

plus élémentaires. Cette approche nécessite une mesure de la similarité entre deux représentations structurelles. On distingue plusieurs techniques telles que les structures de graphes et les structures syntaxiques. Pour plus d'informations, l'ouvrage de Miclet [12] traite cette approche en détails.

1.4.2 Approche statistique

Cette approche consiste à déterminer des caractéristiques extraites d'une forme pour les caractériser d'une manière statistique. Elle a besoin d'un nombre élevé d'exemples afin de réaliser un apprentissage correct des lois de probabilité des différentes classes. Autrement dit, cette approche bénéficie des méthodes d'apprentissage automatique qui s'appuient sur des bases théoriques connues telles que la théorie de la décision bayésienne, les méthodes de classification non supervisées et l'analyse en composantes principales. Les deux principales familles de méthodes utilisées sont les méthodes *paramétriques* et les méthodes *non paramétriques*.

Les méthodes paramétriques opèrent sous l'hypothèse que les classes étudiées suivent une distribution de probabilité d'une certaine forme connue *a priori*. La prise de décision consiste à déterminer la classe pour laquelle la forme inconnue présente la probabilité d'appartenance maximale. Elles exigent des bases d'apprentissage assez importantes pour une estimation correcte des paramètres de la distribution supposée. L'approche statistique englobe : la règle de Bayes [2, 5, 11], la distance de Mahalanobis [2], les méthodes neuronales [14, 16, 17] et les chaînes de Markov [18].

Dans le cas des méthodes non paramétriques, les lois de probabilité sont inconnues pour une des classes. Le problème revient à établir des frontières de décision entre les classes. Les techniques les plus utilisées en reconnaissance de formes sont : la méthode du plus proche voisin [2, 5, 11], la méthode de Parzen [11] et la méthode d'appariement de graphes [19]. Pour de plus amples informations, Gaillat [9] décrit un ensemble de méthodes statistiques en reconnaissance de formes.

Malgré leur nature différente, les approches statistiques et structurelles peuvent être combinées aux mêmes domaines d'application. Le choix d'une approche peut être lié à des contraintes matérielles telle que la taille de la base d'apprentissage disponible, le temps de calcul requis et la taille mémoire nécessaire. L'utilisation conjointe des deux approches peut être une solution optimale pour le problème de reconnaissance de l'écriture.

Dans le cadre de notre projet, la méthode de classification du système de reconnaissance des chiffres isolés qui est décrite au chapitre 2 repose sur la théorie des réseaux de neurones. Ces derniers manipulent des primitives globales.

1.5 Sélection des primitives

Les caractéristiques permettent de distinguer une forme appartenant à une classe par rapport aux formes des autres classes. Il est important de bien définir les caractéristiques à extraire d'un objet pour sa reconnaissance. Il existe une très grande variété de caractéristiques mesurables sur des images et trop souvent on pense que chaque caractéristique est importante pour discriminer une forme d'une autre. Kim et Kim [41] ont observé que la performance du système de reconnaissance devient mauvaise et le temps de calcul augmente à mesure que le nombre de primitives augmente dans un problème de reconnaissance de formes. Dans certains cas, il existe quelques caractéristiques qui n'aideront pas à discriminer entre les classes. Autrement dit, il existe des caractéristiques redondantes ou non pertinentes. Ces caractéristiques seront inutiles dans la classification de l'objet d'où l'utilité d'effectuer la sélection des primitives les plus pertinentes [39, 42]. Cette sélection a parfois un impact considérable dans l'efficacité des résultats de la classification [43].

L'objectif principal de la sélection des primitives est la réduction du nombre de caractéristiques utilisées tout en essayant de maintenir ou d'améliorer la performance de la classification du système de reconnaissance. Cet objectif est défini par une fonction objective à optimiser. Pour se faire, il suffit de choisir une méthode d'optimisation qui peut être adaptée au problème à résoudre. Les méthodes

d'optimisation peuvent être classées de différentes manières : Les méthodes *déterministes* et les méthodes *non déterministes.*

Les méthodes déterministes sont généralement efficaces lorsque l'évaluation de la fonction à optimiser est très rapide ou lorsque la forme de la fonction est connue *a priori.* Elles nécessitent comme étape préliminaire la localisation des extremas. Les méthodes non déterministes font appel à des tirages de nombres aléatoires. Elles permettent une exploration efficace de l'espace de recherche. Dash et Liu [44] et Yang et Honavar [45] proposent deux étapes essentielles dans la sélection des primitives : les *procédures de recherche* et les *fonctions d'évaluation.*

- ***Les procédures de recherche.*** Si l'ensemble des caractéristiques contient L primitives défini initialement alors le nombre total de possibilités des candidats générés est de 2^L. Ce nombre est très élevé et le but est de trouver une solution parmi ce nombre de probabilités. Pour résoudre ce genre de problème, la procédure de recherche regroupe 3 approches : complète, heuristiques et aléatoire. La figure 2 illustre ces approches avec l'ensemble des méthodes qui s'y rattachent.

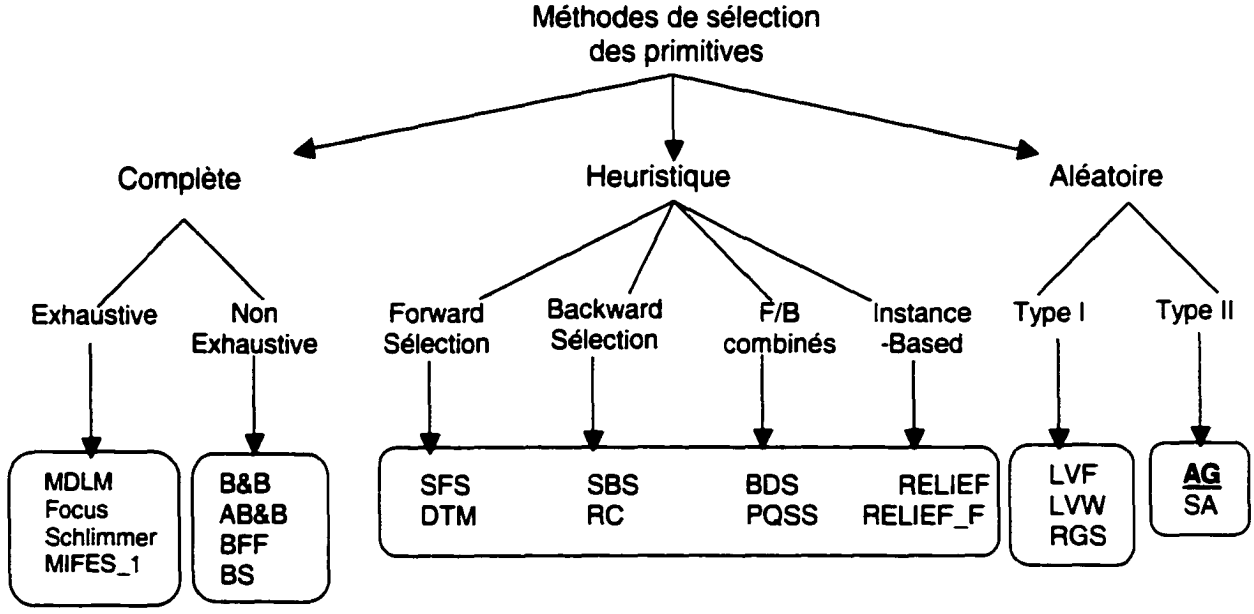


Figure 2 - Les méthodes de sélection de primitives (extraite de [44])

Complète : Ce procédé fait une recherche complète du sous-ensemble optimal de primitives selon la fonction d'évaluation utilisée. Différentes fonctions d'évaluation sont employées pour réduire l'espace de recherche sans compromettre les chances de trouver le sous-ensemble optimal. Par conséquent, bien que l'espace de recherche soit $O(2^L)$, peu de sous-ensembles sont évalués. L'optimalité du sous-ensemble de primitives, selon la fonction à optimiser, est garantie parce que le procédé peut faire des retours arrière (backtrack). Parmi les techniques utilisées : Branch and Bound (B&B), Best First Search (BFF), Beam Search (BS), Approximate Branch and Bound (AB&B), Minimum Description Length Method (MDLM), Focus, Méthode de Shlimmer et Min-Features (MIFFES_1).

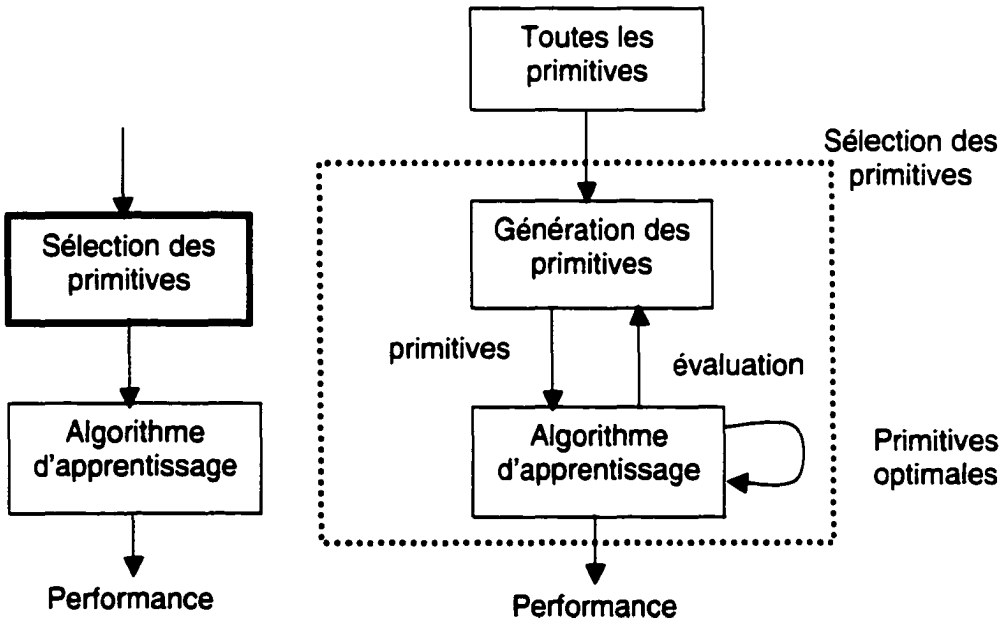
Heuristiques : À chaque itération de la procédure de recherche, toutes les caractéristiques qui restent à être sélectionnées sont considérées pour la sélection des primitives. L'espace de recherche est L^2. Ces procédés sont très simples et rapides car l'espace de recherche est quadratique en terme du nombre de primitives. Parmi les techniques utilisées : Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), bi-directional search (BDS), RELIEF, relevance in context (RC), (p,q) sequential search (PQSS).

Aléatoire : Ce procédé de recherche est plutôt nouveau dans son utilisation dans des méthodes de sélection des primitives en comparaison aux deux autres catégories. Bien que l'espace de recherche soit 2^L, ces méthodes recherchent des sous-ensembles en effectuant un maximum d'itérations. Chaque procédure de recherche aléatoire exige des valeurs de quelques paramètres pour l'appliquer au problème à résoudre. La tâche d'attribution de valeurs appropriées à ces paramètres est importante afin d'obtenir des résultats satisfaisants. Parmi les techniques utilisées, on rencontre les algorithmes génétiques (AG), le recuit simulé (Simulated Annealing (SA)), random generation plus sequential selection (RGS).

- ***Les fonctions d'évaluation.*** Elles mesurent l'ensemble des solutions candidates générées par les procédures de recherche. Ces valeurs sont comparées aux valeurs précédentes. La meilleure valeur sera gardée. Il existe plusieurs types d'évaluation :

la mesure de la distance (distance euclidienne) qui permet de mesurer les capacités de discrimination, la mesure d'information qui permet d'estimer le gain d'une caractéristique, la mesure de dépendance qui permet de quantifier la corrélation des caractéristiques, la mesure de consistance de l'ensemble des caractéristiques et la mesure du taux d'erreur lors de la classification.

Les auteurs [44, 45, 46] ont classé ces algorithmes en deux catégories : *filter* et *wrapper*. Les primitives sont sélectionnées indépendamment de l'algorithme d'apprentissage dans le cas de l'approche *filter*, tandis qu'un sous-ensemble de primitives est généré et évalué par l'algorithme d'apprentissage dans l'approche *wrapper* (voir figure 3).



(a) Approche *Filter* (b) Approche W*rapper*

Figure 3 - Deux approches pour la sélection de primitives (extraite de [45])

Yang et Honavar [45] et Hall et al [51] ont mentionné qu'il a été prouvé que le temps de calcul de l'approche *filter* est beaucoup plus rapide que l'approche *wrapper*. Kudo et Sklansky [47] ont étudié les différentes techniques d'optimisation et ont trouvé que les

AG conviennent parfaitement lorsque le nombre de primitives à sélectionner est très élevé (supérieur à 100 primitives). Ces auteurs ont rajouté que les AG ont une grande probabilité de trouver les meilleures solutions par rapport aux autres techniques d'optimisation. Ils recommandent d'appliquer les AG plusieurs fois avec différents paramètres. L'application des AG dans la sélection des primitives a été utilisée dans différents domaines de recherche tels que l'étalonnage de caméras par Ji et Zhang dans [53], la vérification de signatures par Ramesh et Murty [54], le diagnostic médical par Yang et Honovar [45], la reconnaissance de visages par Ho et Huang [55], la recherche biologique par Pei et al [52], l'agriculture par Kavzoglu et Mather [56], la reconnaissance de caractère optique par Shamik et Das [58] et la reconnaissance de l'écriture par Kim et Kim [41].

1.6 Conclusion

Ce chapitre nous a permis d'avoir un aperçu global d'un système de reconnaissance de formes à savoir : les pré-traitements, l'extraction des primitives et la classification. L'étape de la sélection des primitives a été abordée car elle permet d'améliorer la performance du système de reconnaissance.

Dans le cadre de notre projet, nous nous sommes intéressés à l'ensemble de ces étapes de reconnaissance de formes (pré-traitements, extraction et classification) car l'objectif du projet est la sélection du sous-ensemble des primitives pertinentes. Ce sous-ensemble appartient à l'ensemble des primitives extraites du système de reconnaissance des chiffres isolés. Le but de cette sélection des primitives est l'optimisation du système de reconnaissance des chiffres isolés. L'optimisation du système de reconnaissance de chiffres isolés consiste à maximiser la performance (meilleur taux de reconnaissance calculé à partir des réseaux de neurones) et à minimiser la complexité du réseau (diminuer le nombre des entrées du classifieur utilisé à savoir les réseaux de neurones). Nous aborderons en détails les réseaux de neurones au chapitre 2.

CHAPITRE 2

LES RÉSEAUX DE NEURONES

2.1 Introduction

Les méthodes de reconnaissance à base de réseaux de neurones ont été étudiées depuis plusieurs années dans le but de réaliser des performances proche de celles observées chez l'humain. Ces réseaux de neurones sont composés de plusieurs éléments (ou cellules) de calcul opérant en parallèle et arrangés à la manière des réseaux de neurones biologiques. Lippmann [14] a précisé que la force des réseaux de neurones réside dans leur capacité à générer une région de décision de forme quelconque, requise par un algorithme de classification, au prix de l'intégration de couches de cellules supplémentaires dans le réseau de type perceptron multicouche avec un apprentissage par rétropropagation du gradient de l'erreur.
Les techniques neuronales sont par nature des méthodes de classification paramétriques puisque la phase d'apprentissage consiste à estimer les paramètres qui permettront de choisir la fonction de discrimination dans un ensemble de fonctions possibles. Toutefois, ces fonctions de discrimination sont tellement complexes qu'on peut plutôt rapprocher les techniques neuronales à des méthodes de classification non paramétriques.

Un réseau se distingue en général par le type de neurone formel qu'il utilise, la règle d'apprentissage qui le décrit et l'architecture définissant les interconnections entre neurones. Le neurone formel, qui peut être considéré comme une modélisation élémentaire de neurone réel, est un automate possédant n entrées réelles $z_1,\ldots,z_n$ et dont le traitement consiste à affecter à sa sortie o le résultat d'une fonction d'activation f de la somme pondérée de ses entrées

$$o = f(\sum_{i=1}^{n} w_i z_i) = f(net_i) \qquad (2.1)$$

où les w_i sont les coefficients ou poids synaptiques associés aux entrées z_i. La fonction d'activation f est généralement de type sigmoïde. La règle d'apprentissage tente de déterminer les valeurs (optimales) des coefficients synaptiques à l'aide d'exemples permettant de minimiser une fonction de coût d'erreur définie entre la sortie effective du réseau et la sortie désirée. Enfin l'architecture du réseau définit les interconnections entre neurones et les contraintes liant les coefficients synaptiques. L'architecture peut être fixée au départ, généralement de façon heuristique ou construite progressivement pendant la phase d'apprentissage. L'architecture adoptée et la nature des connexions entre neurones définissent la règle de propagation dans le réseau encore appelée la dynamique du réseau.

Les réseaux à couches sont les modèles connexionnistes les plus couramment utilisés. Leur architecture, organisée en couches successives, comprend une couche d'entrée, une couche de sortie et une ou plusieurs couches intermédiaires appelées couches cachées car elles ne sont pas vues de l'extérieur. Chaque couche est composée d'un certain nombre de neurones. Les connexions sont établies entre les neurones appartenant à des couches successives mais les neurones d'une même couche ne peuvent pas communiquer entre eux dans le cas des réseaux à couches.

Parmi les plus simples des réseaux à couches, on cite le réseau mono couche appelé perceptron. Historiquement, le perceptron est parmi les premiers réseaux efficaces qui a été proposé et étudié en détail. Il est composé uniquement d'une couche d'entrée et d'une couche de sortie (voir figure 4). La couche d'entrée comporte autant de neurones que le vecteur de caractéristiques (primitives) a de composantes. Les connexions entre ces deux couches sont modifiables et bidirectionnelles. C'est la couche de sortie qui remplit la tâche de classification et les unités de la couche de sortie réalisent une fonction à seuil. Cette fonction d'activation réalisée par le neurone formel est un échelon qui vaut 1 si la somme pondérée de ses entrées est supérieure au seuil et 0 sinon. Chaque neurone sépare donc linéairement l'espace des entrées en deux classes. Les connexions sont modifiées par un apprentissage supervisé selon le principe de correction d'erreurs : si une unité de la couche de sortie n'est pas dans l'état désiré, toutes les liaisons que cette unité réalise avec les unités de la couche

d'entrée sont augmentées ou diminuées selon le type d'action que l'unité de la couche d'entrée réalisait sur l'unité de sortie considérée. Autrement dit, la règle d'apprentissage fixe le poids des connexions selon le principe de coût d'erreurs. A partir d'exemples présentés sur la couche d'entrée du perceptron, les poids associés aux connexions de chaque neurones de la couche de sortie sont modifiés jusqu'à obtenir la sortie désirée de ce neurone. Ainsi, *m* neurones binaires sur la couche de sortie permettent de séparer linéairement 2^m classes.

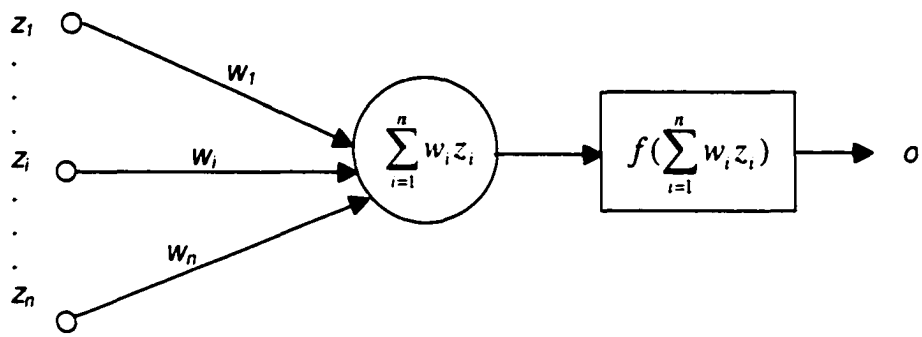


Figure 4 - Modèle d'un réseau de neurone, perceptron

Les méthodes d'apprentissage sont divisées en deux catégories : l'apprentissage *avec supervision* et l'apprentissage *sans supervision* [2]. L'apprentissage supervisé consiste à ajuster par interpolation, approximation ou optimisation les poids du réseau de neurones en comparant la sortie obtenue et la sortie désirée de façon à minimiser l'erreur. Les algorithmes d'apprentissage permettent de déterminer automatiquement certains paramètres libres du réseau (par exemple : poids) qui correspondent en fait à des paramètres permettant de définir les frontières des classes. Le réseau de neurones génère une ou plusieurs valeurs de sorties basées sur la performance de calcul des valeurs d'entrées. Les résultats de ces calculs dépendent des valeurs de poids assignées à chaque entrée obtenues après la phase d'apprentissage.

Pour faire un lien entre les réseaux de neurones et d'autres types de classifieurs, rappelons les rapports étroits qui existent entre les classifieurs bayésiens et certains modèles connexionnistes. Il a été démontré dans la littérature (voir le rapport technique

de Remm [15]) que pour un problème de classification à m classes, un perceptron multicouches avec m neurones de sorties utilisant la méthode de rétropropagation du gradient de l'erreur et ayant comme sortie désirée un neurone à 1 indiquant la classe correcte et les autres à 0, réalisera une estimation des probabilités *a posteriori*. Plus formellement, pour attribuer un vecteur Z à une classe parmi m, un classifieur bayésien estimera les probabilités *a posteriori* $P(C_i|Z)$ pour chaque classe C_i, i=1,...,m et assignera au vecteur Z la classe de probabilité maximale. Contrairement à la méthode bayésienne qui estime les probabilités *a posteriori* à partir des probabilités *a priori*, le calcul par un perceptron sera direct sans passer par les probabilités *a priori*.
On appelle probabilité *a posteriori* la probabilité conditionnelle $P(C_i|Z)$ de la classe C_i connaissant Z. En d'autres termes, quand on a une entrée Z, $P(C_i|Z)$ représente la probabilité que Z appartienne à la classe C_i.
La probabilité conditionnelle $P(Z|C_i)$ représente la probabilité d'avoir Z comme entrée quand on sait que l'on est dans la classe C_i.

Il existe plusieurs modèles de réseaux de neurones dans la littérature tels que définis par : Lippmann dans [14], Bishop dans [16], et Zurada dans [17]. Dans le cadre de notre projet, nous limiterons notre étude à l'utilisation du réseau de neurones multicouche MLP. Le choix de ce modèle a été pris en considération après une étude détaillée par le système de Oliveira et al [3] développé durant sa thèse de doctorat.

2.2 Perceptron multicouche

L'introduction de couches intermédiaires dans le réseau MLP permet de résoudre des problèmes plus complexes que la simple séparation linéaire. Lorsqu'il existe au moins une couche cachée, les états internes du réseau ne peuvent plus être donnés directement par les exemples et les sorties désirées puisque les sorties des neurones appartenant aux couches intermédiaires sont inconnues. La figure 5 représente un réseau de neurones multicouche avec comme entrée le vecteur de primitives et en sortie les classes où seront classées les formes. Pour déterminer le nombre de couches cachées dans un réseau cela dépendra du problème à résoudre (comment les classes des formes sont-elles séparées?).

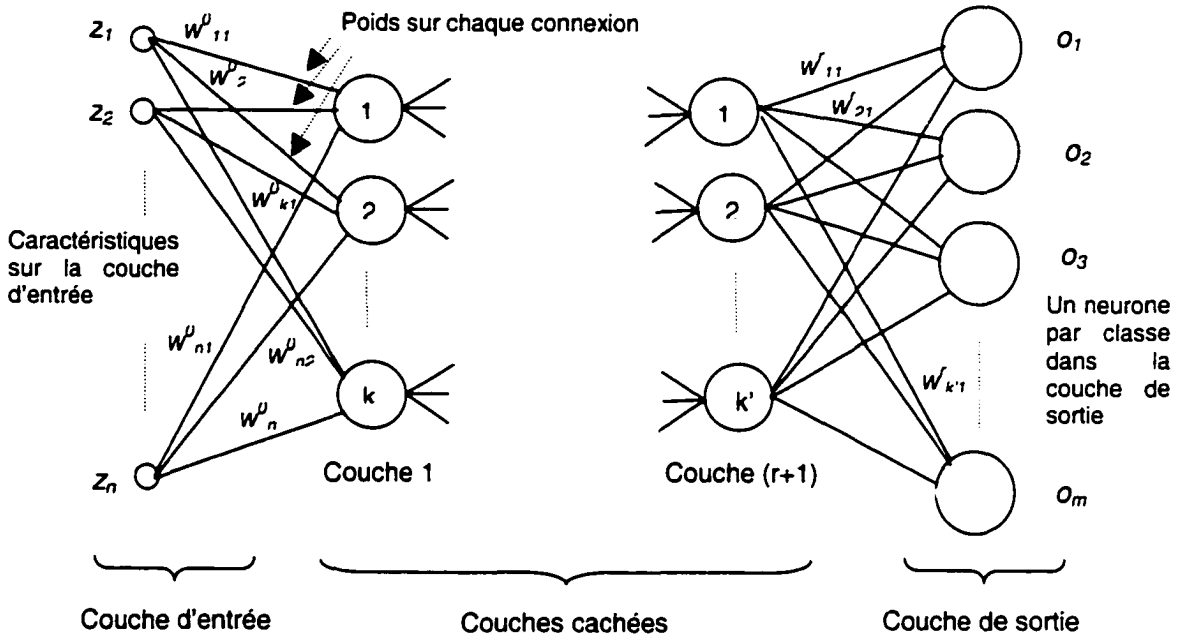


Figure 5 - Réseau de neurones multicouches (extraite de [14])

Plusieurs types de réseaux de neurones multicouches ont été développés. Nous focalisons notre étude sur le type MLP. Dans le processus de la construction de ce type de classifieur, l'apprentissage a comme objectif d'adapter les poids des connexions entre les neurones de sorte que le réseau donne en sortie la classe d'appartenance des formes qui lui sont proposées en entrée. Ce qui revient à minimiser l'erreur commise par le réseau sur l'ensemble de formes de la base d'apprentissage. Pour une forme de la classe C_i, l'état du neurone O_i de la couche de sortie du MLP doit être proche de la valeur maximale et tous les autres proches de la valeur minimale. Ce problème de minimisation de l'erreur a été résolu par des méthodes de *rétropropagation du gradient d'erreur* [14, 16, 17]. Toutefois, ces méthodes ne garantissent pas l'obtention du minimum global de la fonction à optimiser (blocage dans un minimum local). Le terme «rétropropagation» est utilisé pour décrire l'apprentissage du réseau de neurones de type MLP utilisant la descente du gradient appliquée à la fonction de la somme des erreurs quadratiques [16]. Bishop [16] et Mangalagiu [19] ont

démontré qu'un réseau de type MLP à une couche cachée peut estimer n'importe quelle fonction dans R^n avec une précision arbitraire. Ainsi, le MLP est capable d'estimer des hyper-surfaces discriminantes très complexes. Les fonctions estimées sont soit linéaires ou non linéaires. Elles sont appelées fonctions d'activation. La fonction non linéaire que nous utiliserons est la fonction *sigmoïde*.

Par contre la difficulté majeure rencontrée lors de l'utilisation de ce type de classifieur consiste à déterminer le nombre de couches cachées, le nombre de neurones dans chacune des couches et les poids des connexions entre les différentes couches (pente de la sigmoïde, taux d'apprentissage...). De ce fait, pour une application donnée, la construction du classifieur de type MLP utilise des règles empiriques et nécessite un certain nombre d'essais afin d'obtenir des performances de généralisation intéressantes.

Pour déterminer les poids de toutes les connexions du réseau, l'utilisation des algorithmes d'apprentissage est obligatoire. L'objectif des algorithmes d'apprentissage est de minimiser l'erreur de décision effectuée par le RNA en ajustant les poids à chaque présentation d'un vecteur d'entraînement. Pour ce qui est de l'ajustement des poids à une étape donnée de la phase d'apprentissage, l'erreur à minimiser est habituellement celle produite lors de l'application d'un vecteur de l'ensemble d'apprentissage à l'entrée du réseau. Pour ce qui est de l'évaluation de la qualité d'apprentissage du réseau, l'erreur cumulée par tous les vecteurs de l'ensemble d'entraînement est évaluée. Cette erreur cumulée est calculée pour tous les cycles de la phase d'entraînement et est définie à partir de l'erreur quadratique. Cette mesure de l'erreur illustre la précision obtenue après P cycles d'apprentissage. De même l'ajout d'un momentum permettra d'accélérer la vitesse de convergence de l'algorithme d'apprentissage par la rétropropagation des erreurs. Cela consiste à ajouter à la valeur courante d'ajustement des poids une fraction de l'ajustement précédent.

En résumé, nous utiliserons pour l'apprentissage du réseau l'algorithme de rétropropagation avec minimisation du gradient d'erreur qui est défini par les étapes suivantes [14, 16] :

1. Initialiser les poids à des petites valeurs et les seuils du réseau.
2. Insérer à l'entrée du réseau une observation (exemple) de la base de données en forme de vecteur de caractéristiques, puis calculer sa valeur d'activation et sa fonction d'activation en utilisant les formules (2.3) et (2.4).
3. Évaluer le signal d'erreur des sorties du réseau en utilisant la formule (2.5).
4. Ajuster les poids en utilisant la formule (2.2).
5. Évaluer le signal d'erreur pour chaque couche cachée en utilisant la formule (2.6).
6. Ajuster les poids de la couche cachée en utilisant la formule (2.2).
7. Répéter les étapes 2 à 6 pour l'ensemble des observations de la base d'apprentissage tant que le critère d'arrêt n'a pas été atteint.

Il existe plusieurs critères d'arrêts. Ces critères peuvent être combinés entre eux. Le premier critère est basé sur l'amplitude du gradient de la fonction d'activation, puisque par définition le gradient sera à zéro au minimum. L'apprentissage du réseau du type MLP utilise la technique de recherche du gradient pour déterminer les poids du réseau. Le second critère d'arrêt est de fixer un seuil que l'erreur quadratique ne doit pas dépasser. Toutefois ceci exige une connaissance préalable de la valeur minimale de l'erreur qui n'est pas toujours disponible. Dans le domaine de la reconnaissance de formes, il suffit de s'arrêter lorsque tous les objets sont correctement classés. Le troisième critère consiste à déterminer un nombre fixe de cycles à atteindre. Finalement, la méthode du *cross-validation* [14] peut être utilisée pour surveiller l'évolution de l'apprentissage appelée la performance du système en généralisation. Toutefois l'algorithme s'arrête lorsqu'il n'y a plus d'amélioration de la performance du système de reconnaissance. Cette méthode fonctionne sur deux bases de données : la base d'apprentissage pour entraîner le réseau et la base validation pour mesurer la performance en généralisation du réseau de neurones. Durant la phase d'apprentissage, la performance du réseau sur la base d'apprentissage continue à s'améliorer mais sur la base de validation à un moment donné une chute de sa performance peut être observée. Si cette dégradation de la performance sur la base de validation existe alors on dit que le réseau est en sur-apprentissage. Si ce cas de sur-apprentissage est rencontré alors l'algorithme d'apprentissage sera arrêté à ce moment. Les trois premiers critères sont sensibles aux choix des paramètres (par

exemple : le nombre de nœuds dans la couche cachée, le seuil d'erreur, ...) et si le choix n'est pas bon alors les résultats obtenus seront mauvais ou le temps de calcul de la performance du système de reconnaissance sera plus lent (par exemple définir un grand nombre de nœuds dans la couche cachée). Cependant, la méthode du cross-validation n'a pas ce genre de problème.

En général, les formules utilisées par cet algorithme sont :

- pour l'ajustement de poids entre le nœud *j* (sortie ou cachée) et le nœud *i* (cachée ou entrée)

$$\Delta w_{ij} = \eta \delta_j o_i \tag{2.2}$$

où η est la valeur de la constante d'apprentissage. En général, $0.1 < \eta < 0.9$

et o_i est la valeur d'activation du neurone *i* tel que $o_i = f(net_i)$

et
$$net_i = \sum_j w_{ij} o_j \tag{2.3}$$

avec $f(net_i)$ est la fonction d'activation. La fonction utilisée dans le cas de notre projet est la fonction sigmoïde définie

$$f(net_i) = \frac{1}{1 + e^{-net_i}} \tag{2.4}$$

- pour le calcul du signal d'erreur du neurone *j* de la couche de sortie

$$\delta_j = (d_j - o_j) o_j (1 - o_j) \tag{2.5}$$

avec d_j est la valeur désirée du neurone *j*.

- pour le calcul du signal d'erreur du neurone *j* de la couche cachée avec *k* nœuds

$$\delta_j = o_j(1-o_j)\sum_k w_{jk}\delta_k \tag{2.6}$$

Ces formules ont été dérivées de la formule de calcul de l'erreur quadratique de l'ensemble de la base d'apprentissage définie comme suit

$$E_p = \frac{1}{2}\sum_p(\sum_k(d_{pk} - o_{pk})^2) \tag{2.7}$$

où *p* est l'indice d'un exemple de la base et *k* est l'indice du nœud de sortie.
L'objectif est de minimiser cette erreur.

- lors de l'ajout du momentum, la formule (2.2) devient alors :

$$\Delta w_{ij}(n+1) = \eta\delta_j o_i + \alpha\Delta w_{ij}(n) \tag{2.8}$$

où α est le momentum tels que $0.1<\alpha<0.8$ et $\Delta w_{ij}(n)$ représente l'ajustement à l'étape précédente.

L'inconvénient des réseaux à couches est le manque d'éléments théoriques permettant de relier d'une part le nombre de couches cachées et le nombre de neurones par couche et d'autre part le type et la complexité du problème à traiter. Il existe des heuristiques pour déterminer le nombre de neurones dans une couche cachée. La formule (2.9) nous montre une heuristique pour déterminer ce nombre

$$N_c = (\frac{N_e + N_s}{2}) + 1 \qquad (2.9)$$

où N_e est le nombre d'entrées du MLP.

N_s est le nombre de neurones en sortie.

N_c est le nombre entier de neurones cachés.

Pour déterminer le nombre maximal de neurones cachés, la formule (2.10) présente ce nombre

$$N_c \leq (2 \times N_e) + 2 \qquad (2.10)$$

Pour définir le nombre de couches cachées et de neurones par couche, le concepteur doit effectuer un grand nombre d'expériences. Par exemple, on fait varier la taille du réseau puis on effectue un apprentissage complet pour chaque taille et enfin on choisit la structure qui conduit aux meilleurs résultats.

2.3 Conclusion

Ce chapitre nous a permis d'avoir un aperçu général sur les réseaux de neurones. Notre étude s'est intéressée principalement aux réseaux de neurones de type MLP. Nous avons décrit les étapes de l'apprentissage du réseau et les différents critères d'arrêt de cet algorithme. L'objectif de notre projet est de minimiser le nombre de neurones d'entrées au réseau tout en améliorant la performance du système de reconnaissance. Ceci va permettre un gain de temps de calcul considérable si un grand nombre d'entrées est supprimé. Pour se faire, nous utilisons les éléments d'une théorie de l'optimisation basée sur les principes de la sélection naturelle qui d'après la littérature donne de meilleurs résultats dans la sélection des primitives adaptée à la reconnaissance de l'écriture manuscrite. La théorie des algorithmes génétiques est abordée dans le chapitre suivant.

CHAPITRE 3

LES ALGORITHMES GÉNÉTIQUES

3.1 Introduction

Les algorithmes génétiques (AG) représentent une famille assez riche et très intéressante d'algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique [21]. Les champs d'application sont fort diversifiés. On les retrouve aussi bien en théorie des graphes qu'en compression d'images numériques ou encore en programmation automatique et en reconnaissance de formes. Le choix des AG parmi d'autres méthodes se justifie en fonction des quatre propriétés suivantes [21] :

- Les AG utilisent un codage des paramètres et non les paramètres eux-mêmes.
- Les AG travaillent sur une population de points, au lieu d'un point unique.
- Les AG n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée ou une autre connaissance auxiliaire.
- Les AG utilisent des règles de transition probabilistes, et non déterministes.

De plus, les AG utilisent deux stratégies importantes pour trouver une solution ou un ensemble de solutions. Ces stratégies sont : l'*exploration* et l'*exploitation*. Elles permettent de trouver le maximum global (solution du problème) du fait qu'elles sont complémentaires [24]. Si l'exploration investigue l'ensemble des solutions de l'espace de recherche, la phase d'exploitation quant à elle se sert de la connaissance trouvée aux solutions précédemment visitées pour aider à trouver de meilleures solutions. La combinaison de ces deux stratégies peut être tout à fait efficace mais la difficulté est de savoir où se trouve la meilleure solution.

Les principes de base des AG ont été développés par Holland [22]. Ils ont été inspirés par le mécanisme de sélection naturelle où les meilleurs candidats sont probablement les mieux adaptés aux conditions de concurrence. L'AG utilise alors une analogie directe avec l'évolution naturelle. À travers la méthode d'évolution génétique, une

solution optimale peut être trouvée et représentée par le dernier gagnant de la technique génétique [23]. Ces algorithmes sont simples et très performants dans la recherche d'une solution optimale.

Les AG fonctionnent avec une population regroupant un ensemble d'individus appelés *chromosomes*. Chaque chromosome est constitué d'un ensemble de *gènes*. Pour chaque individu on attribue une valeur calculée par une fonction appelée fonction d'adaptation ou *fitness*. En pratique, à partir d'une population, des chromosomes sont générés d'une façon aléatoire lors de l'initialisation. Pour définir la taille de la population, Man et al [23] ont mentionné que cette taille varie d'un problème à un autre. Dans chaque cycle d'opérations génétiques, une nouvelle population appelée *génération* est créée à partir des chromosomes de la population courante. Pour cela certains chromosomes appelés '*parents*' sont sélectionnés afin d'élaborer les opérations génétiques. Les gènes de ces parents sont mixés et recombinés pour la production d'autres chromosomes appelés '*enfants*' constituant la nouvelle génération. Les étapes de l'AG sont répétées durant *t* cycles, l'arrêt de l'algorithme est fixé d'après un *critère d'arrêt*. On peut avoir plusieurs critères d'arrêt :

- Le nombre de génération fixé initialement a été atteint.
- La valeur de la fonction d'adaptation a atteint une valeur fixée *a priori*.
- L'absence d'évolution de la valeur de la fonction d'adaptation des individus d'une population à une autre.
- Les chromosomes ont atteint un certain degré d'homogénéité.

La figure 6 illustre les différentes étapes d'un AG simple [21, 23] :

```
Algorithme Génétique Simple
{
        t = 0;
        Initialisation (P(t));
        Évaluation (P(t));
        While non fin faire
        t= t + 1;
        P' = Sélection (P(t));
        Recombiner (P'(t);
        Mutation (P'(t));
        Évaluation (P'(t));
        P = Survie (P, P'(t));
}
```

Figure 6 - Les étapes d'un AG simple

On peut résumer que l'AG est fondé sur :

- Une représentation chromosomique des solutions du problème.
- Une méthode pour créer une population initiale de solutions.
- Une fonction d'évaluation (fitness) pour classer les solutions en fonction de leurs aptitudes.
- Des opérateurs génétiques qui définissent la manière dont les caractéristiques génétiques des parents sont transmises aux descendants (enfants).
- Les valeurs des paramètres utilisés par l'AG.

3.2 Codage des chromosomes

Un chromosome est un ensemble de paramètres (gènes) qui sont reliés entre eux pour former une chaîne de valeurs. Beasly et al [25] ont mentionné que traditionnellement ces valeurs (symboles) sont représentées par des bits (0 ou 1). En terme de la

génétique, l'ensemble des paramètres représentés par un chromosome particulier est désigné sous le nom de *génotype*. Le génotype contient l'information exigée pour construire une organisation qui est désignée sous le nom du *phénotype* [24].
Goldberg [21] et Holland [22] ont démontré qu'il est idéal de représenter le chromosome en une chaîne binaire. C'est pourquoi les AG utilisent généralement cette représentation. Par contre d'autres auteurs tels que Man et al [23], Janikow et Michalewicz [26], Herrera et al [27], et Davis [28] ont effectué une comparaison entre la représentation binaire et la représentation réelle. Ces auteurs ont trouvé que la représentation réelle donne de meilleurs résultats d'après leur problème à résoudre. En résumé, il n'existe pas de consensus sur la représentation idéale des individus. Soit la notation de la représentation du chromosome C suivante :

$$C = (c_1, c_2, \ldots, c_i, \ldots, c_L)$$

où L représente le nombre de variables dans le chromosome et c_i représente le $i^{ème}$ gène du chromosome C avec une représentation de type binaire ou réelle.

3.3 Initialisation de la population

La population initiale est constituée d'un ensemble d'individus (chromosomes) générés aléatoirement. Cependant rien n'empêche d'utiliser des résultats et des solutions existantes pour former la population initiale. En d'autres termes, un ensemble de solutions connues préalablement peut être injecté dans la population initiale.
Le nombre d'individus d'une population ou la taille de la population constitue un paramètre important pour l'AG qu'il faudra déterminer. La représentation de la population P est :

$$P = (C_1, C_2, \ldots, C_i, \ldots, C_{taille\_pop})$$

où C_i représente la $i^{ème}$ chromosome dans la population et *taille_pop* représente le nombre de chromosomes dans la population.

3.4 Fonction d'évaluation

La fonction d'évaluation ou d'adaptation (fitness) associe un coût à chaque chromosome. Il faut distinguer entre la fonction objective et la fonction d'adaptation.

Dans certains cas, elles peuvent être identiques mais en général la fonction d'adaptation dépend de la fonction objective, laquelle dépend de la nature du problème à résoudre. La solution optimale du problème est obtenue à partir de la fonction d'évaluation du chromosome. Dans le cas d'un problème de minimisation, la solution est associée à la plus petite valeur trouvée de la fonction d'adaptation calculée pour chaque individu de la population. Dans le cas d'une maximisation, alors la valeur la plus grande de la fonction sera prise en compte.

La fonction d'adaptation peut être soit mono critère ou multicritère. Une fonction d'adaptation mono critère signifie que la fonction dépend d'une seule fonction objective par contre la fonction d'adaptation multicritère dépend d'une combinaison de plusieurs fonctions objectives d'où la notion objective unique et objectives multiples.

- *Objectif unique*. La définition de la fonction d'adaptation (fitness) ne pose généralement pas de problème car c'est une optimisation simple à résoudre.

- *Objectifs multiples*. Les problèmes d'optimisation doivent souvent satisfaire des objectifs multiples, dont certains sont concurrents. Une méthode classique consiste à définir des fonctions objectives o_i, traduisant chaque objectif à atteindre, et de les combiner au sein de la fonction d'adaptation. On établit ainsi un compromis. Le plus simple est de se ramener à une somme pondérée des fonctions objectives. A la place d'une somme, on peut également utiliser un produit des fonctions objectives ou utiliser des expressions plus complexes. L'optimisation multicritère est un axe de recherche très actif actuellement, de par les enjeux économiques et industriels auxquels il répond. Des concepts tels que les niches écologiques ou l'optimalité de Pareto semblent prometteurs pour la résolution de ce genre de problème.

Il existe des méthodes pour la définition des fonctions à optimiser qui dépendent du problème à résoudre. Parmi ces méthodes de fonction d'adaptation, on trouve : no scaling, linear scaling, sigma truncation, power law scaling et ranking [29].

3.4.1 No scaling

C'est la méthode la plus facile pour le calcul de la valeur de la fonction d'adaptation f_i du chromosome C_i

$$f_i = o_i \tag{3.1}$$

où o_i est la fonction objective du chromosome C_i.

3.4.2 Linear scaling (échelle linéaire)

La valeur de la fonction d'adaptation f_i du chromosome C_i est en relation linéaire avec la valeur de la fonction objective o_i [23, 29]

$$f_i = a\, o_i + b \tag{3.2}$$

où a et b sont choisis en fonction des valeurs des fonctions objectives des individus [21]. Cette méthode produit une bonne solution. Cependant, cette fonction d'adaptation peut être négative. Pour éviter la valeur négative, il suffit de la remplacer par une valeur nulle.

3.4.3 Sigma truncation (Le sigma tronqué)

Cette méthode est utilisée lorsque la valeur de la fonction objective est négative. La valeur de f_i du chromosome C_i est calculée de la manière suivante

$$f_i = o_i - (\bar{o} - c\delta) \tag{3.3}$$

où c est une petite valeur entière et $\bar{o}$ est la moyenne des fonctions objectives. Cette moyenne correspond à la somme pondérée de toutes les valeurs de la fonction de chaque individu d'une population.

Autrement dit $$\bar{o} = \frac{\sum_{i=1}^{taille\_pop} f_i}{taille\_pop}$$

δ est la déviation standard de la population.
Afin d'éviter la valeur négative de f_i, chaque résultat négatif est mis à zéro.

3.4.4 Power law scaling (puissance d'échelle)

La valeur de f_i est définie par la relation suivante

$$f_i = o_i^k \tag{3.4}$$

où k est une valeur qui varie en fonction des générations.

3.4.5 Ranking (Rangement)

La valeur de f_i ne dépend pas directement de la valeur de la fonction objective mais s'associe au rang de la valeur de la fonction objective. Cette méthode permet d'éviter la convergence prématurée et d'accélérer la recherche quand la population converge vers une solution. D'autre part, elle exige des calculs supplémentaires dans le tri des valeurs objectives des chromosomes de la population.

3.5 Méthodes de sélection

Pour générer de nouveaux descendants (enfants), des parents sont sélectionnés puis des opérations génétiques leur sont appliquées. Pour sélectionner les meilleurs chromosomes dans une population, certaines techniques proposées par différents auteurs sont les suivantes : rank selection, roulette wheel selection, tournament selection, uniform selection [29].

3.5.1 Rank selection (Sélection ordonnée)

Chaque chromosome C_i d'une population P est évalué par la fonction d'adaptation f_i. Les valeurs de la fonction d'adaptation obtenues pour l'ensemble des chromosomes seront classées dans un ordre croissant ou décroissant. Les meilleurs chromosomes seront donc sélectionnés.

3.5.2 Roulette wheel selection (Roulette Biaisée)

Man et al [23] ont mentionné que la technique de la roulette biaisée est la plus utilisée dans la littérature. Elle attribue à chaque chromosome C_i une probabilité de survie p_i proportionnelle à sa valeur f_i dans la population

$$p_i = \frac{f_i}{\sum_{j=1}^{taille\_pop} f_j} \tag{3.5}$$

où $\sum_{j=1}^{taille\_pop} f_j$ représente la somme de toutes les valeurs des fonctions d'adaptation de chaque chromosome C_i de la population P.

Lors de la phase de sélection, les individus sont sélectionnés aléatoirement en respectant les probabilités p_i associées pour former la population de la nouvelle génération. Ceci s'effectue pour le calcul d'une probabilité de sélection cumulée q_i telle que

$$q_i = \sum_{j=1}^{i} p_j \tag{3.6}$$

On génère un nombre réel r aléatoirement sur l'intervalle [0,1]. Cette valeur est générée plusieurs fois en fonction de la taille de la population *taille_pop*. L'individu C_i est sélectionné lorsque $q_i\text{-}1 < r < q_i$.

3.5.3 Tournament selection (sélection par tournoi)

Cette technique utilise la méthode de la roulette biaisée pour sélectionner deux individus. On récupère celui dont la valeur de la fonction d'adaptation est la plus grande. Cette méthode choisit toujours une valeur de la fonction d'adaptation plus élevée par rapport à la technique de la roulette biaisée.

3.5.4 Uniform selection (sélection uniforme)

C'est une technique très simple qui consiste à sélectionner un individu C_i aléatoirement de la population P. La probabilité p_i pour qu'un individu soit sélectionné est définie par

$$p_i = \frac{1}{taille\_pop} \tag{3.7}$$

3.6 Opérateurs génétiques de reproduction

Durant la phase de reproduction de l'AG, des individus de la population sont sélectionnés d'après la méthode de sélection choisie et sont recombinés produisant des enfants de la prochaine génération. Cette phase utilise des mécanismes de reproduction qui sont : le *croisement* (crossover) et la *mutation*.
Les méthodes de reproduction proposées dans la littérature concernent les chromosomes dont la représentation des gènes est binaire. Dans le cas d'un problème d'optimisation où les gènes du chromosome sont des nombres réels, avant d'effectuer les opérations de reproduction, les gènes du chromosome sont convertis en bits (32 ou 16 bits), puis les opérations de reproduction sont effectuées sur cette représentation binaire. Nous illustrons ici cette stratégie dans un exemple présenté par Goldberg [21].

Supposons la fonction à optimiser suivante $f(x) = x^2$ tel que $x \in [0 , 31]$.
La représentation de la variable x est sur 5 bits. Les opérateurs de reproduction sont opérés sur des bits au lieu des réels (voir tableau I).

Tableau I

Exemple sur les opérateurs pour une représentation binaire vs réelle

Valeur réelle de x	Représentation binaire de x	Point de mutation aléatoire	Point de coupure du croisement à un point de coupure	Nouvelle valeur de x	Valeur réelle de x
13	0 1 1 0 1	2	4	0 1 1 0 0	12
24	1 1 0 0 0	1	4	1 1 0 0 1	25
8	0 1 0 0 0	4	2	1 1 0 1 1	27
19	1 0 0 1 1	3	2	1 0 0 0 0	16

La conversion de la représentation réelle en une représentation binaire est une technique utilisée pour la résolution de plusieurs types de problèmes, par exemple :

- ***Optimisation dans un domaine de recherche continue.*** Pour résoudre un problème dans un domaine de recherche continue, chaque paramètre (gène) c_i du chromosome C défini dans l'intervalle $[a_i , b_i]$ est converti en code binaire tels que l'intervalle $[a_i , b_i]$ soit transformé en un ensemble $\{0,\ldots,2^{L_i}\}$ où L_i est le nombre de bits. Le chromosome ainsi converti aura une grande dimension répondant à la grande précision numérique requise pour la résolution du problème. Herrera et al [27] indiquent que la performance de cette représentation binaire n'est pas très conviviale à l'utilisation. Durant l'exécution de l'AG, plusieurs opérations inutiles seront effectuées entraînant une exploration trop étendue de l'espace de recherche. Cette exploration va balayer des solutions sans aucune signification impliquant que l'algorithme ne convergera pas. Pour palier ce genre de problème, la représentation réelle sera donc utilisée au lieu de la représentation binaire.

- ***La redondance.*** Dans le cas de la codification d'un paramètre dont le cardinal n'est pas une puissance de deux, alors il existe certains codes qui n'appartiennent pas au domaine du paramètre. Ces codes sont appelés les *codes redondants*. Soit $x_i \in S_i$ où S_i est un ensemble discret fini avec $| S_i | = 10$. L'utilisation d'une représentation binaire contient au minimum 4 bits pour codifier un élément de l'ensemble S_i. Si les

codes 0000 jusqu'à 1001 sont sélectionnés pour représenter les éléments de S_i, que représentent alors les codes de 1010 à 1111? Lorsque les opérateurs de reproduction seront effectués, certains codes qui n'appartiennent pas à l'ensemble S_i seront générés. Pour palier à ce problème, Herrera et al [27] proposent deux mécanismes : *fixed remapping* et *random remapping*. Ils ont observé que la représentation réelle est la plus adaptée.

A partir de ces problèmes sur la codification des chromosomes dont la représentation est binaire, il est donc préférable de représenter les gènes du chromosome en réel d'après la définition des variables du problème à résoudre.

Pour le bon fonctionnement de l'AG, plusieurs opérateurs ont été proposés dans la littérature et appliqués dans différentes applications utilisant soit un codage binaire ou un codage réel [21, 23, 24, 26, 27, 28, 29, 30].

3.6.1 Techniques de croisement

Cette technique nécessite deux parents afin d'effectuer l'échange des gènes entre eux. Cet échange permet de former deux descendants possédant des caractéristiques issues des deux parents. Chaque individu se voit attribuer une même probabilité p_{cross} de participer à un croisement.
L'opérateur de croisement favorise l'exploration de l'espace de recherche. Il assure le brassage du matériel génétique et l'accumulation des mutations favorables. En d'autres termes, cet opérateur permet de créer de nouvelles combinaisons ayant des caractéristiques communes avec leurs parents. Souvent les meilleures caractéristiques sont transmises aux descendants. Cette transmission est appelée *héritage*. Seuls les individus les mieux adaptés vivent suffisamment longtemps pour se reproduire. Ceci va conduire à des progénitures encore mieux adaptées.
Il existe différentes techniques de croisement. Chacune des techniques s'applique sur des chromosomes dont la représentation est soit binaire ou réelle. Nous citerons quelques techniques. Une notation des chromosomes qui est adoptée dans les prochaines sections est la suivante :

le chromosome parent 1 : $C^1 = (c^1_1 , c^1_2 , \ldots , c^1_i , \ldots , c^1_L)$
le chromosome parent 2 : $C^2 = (c^2_1 , c^2_2 , \ldots , c^2_i , \ldots , c^2_L)$.
où : $c^k_i \in [a_i , b_i]$ ou bien $c_i \in \{0 , 1\}$ avec $k = 1,2$.

3.6.1.1 Un point de coupure (croisement simple)

On choisit aléatoirement un point de croisement pour chaque couple d'individus sélectionné. Notons que le croisement s'effectue directement au niveau des gènes représentés soit en binaires ou en réels. Un chromosome ne peut pas être coupé au milieu d'un gène. La figure 7 illustre ce croisement d'un seul point de coupure dans le cadre d'une représentation binaire ou réelle des gènes des chromosomes.

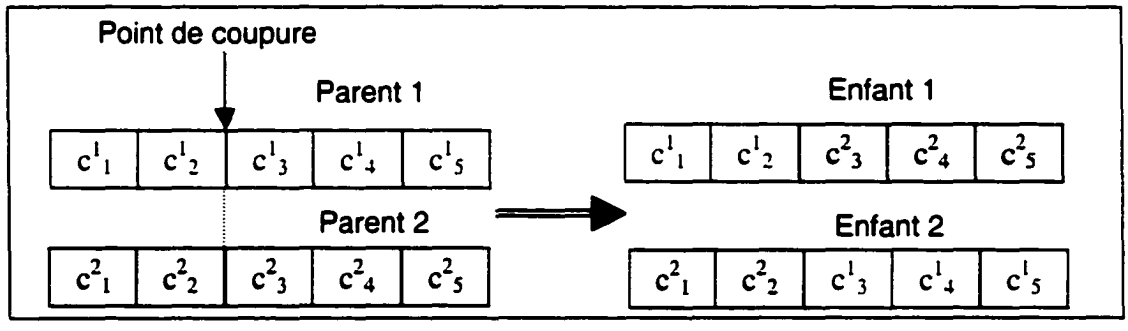


Figure 7 - Croisement avec un seul point de coupure

3.6.1.2 Croisement multiple (multipoint)

Plusieurs auteurs se sont penchés sur l'utilisation de plusieurs points de coupure concernant l'opérateur de croisement. Le nombre de points de coupure généré est en moyenne $L/2$ où L est la taille du chromosome. L'individu est représenté sous la forme d'un anneau et l'échange de gènes entre les deux parents s'effectue de la façon suivante (voir figure 8) :

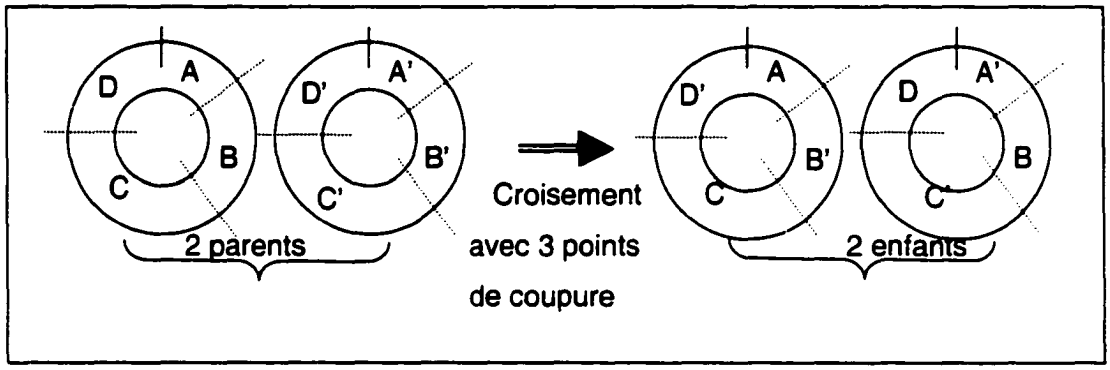


Figure 8 - Croisement avec 3 points de coupure

Cette technique s'applique autant pour une codification binaire que réelle des chromosomes. C'est une technique très utilisée dans différentes applications du fait que les résultats obtenus sont satisfaisants [38].

3.6.1.3 Croisement uniforme

Cette technique est complètement différente des deux techniques précédentes. Un masque de croisement est généré aléatoirement pour chaque couple d'individus ou pour chaque génération. Les valeurs de ce masque sont binaires. Sa taille est identique à celle du chromosome. Son fonctionnement est illustré par la figure 9 :

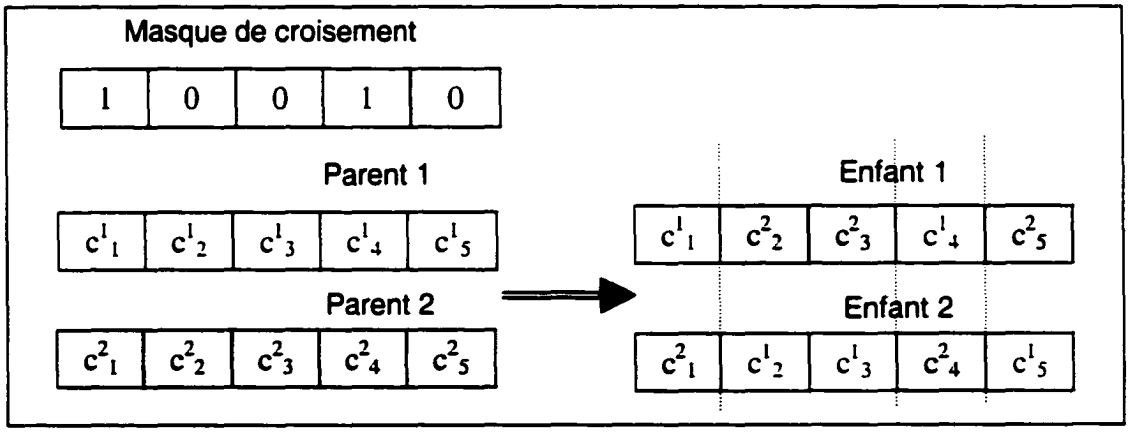


Figure 9 - Croisement uniforme

Le fonctionnement du croisement uniforme est le suivant : si la valeur du bit du masque est égale à 1 alors la valeur du gène du parent1 est copiée chez l'enfant1 et si la valeur du bit du masque est égale à 0 alors la valeur du gène du parent2 est transmise à l'enfant1. Les valeurs des gènes de l'enfant2 sont les suivantes : les valeurs des gènes du parents1 lorsque la valeur du bit du masque est égale à 0 et les valeurs des gènes du parents2 lorsque la valeur du bit du masque est égale à 1.
Malgré que cette technique soit différente des deux autres au niveau conceptuel, on remarque qu'il existe une ressemblance entre ces techniques. Le croisement uniforme peut être un cas général des deux techniques.

Ces trois techniques de croisement s'appliquent dans les deux cas de figure : la représentation binaire et la représentation réelle. Les techniques étudiées par Herrera et al [27] s'appliquent sur une représentation réelle mais dans certains cas peuvent être appliquées sur une représentation binaire. La figure 10 représente un ensemble d'opérateurs décrivant l'espace de recherche balayé (exploration vs exploitation). Nous étudierons ci dessous quelques opérateurs que nous jugions intéressants dans le cadre de notre projet.

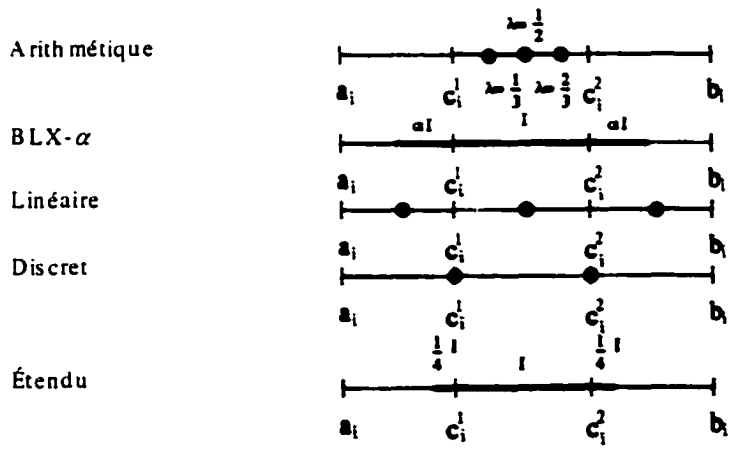


Figure 10 - Représentation des opérateurs de croisement (extraite de [27])

3.6.1.4 Croisement arithmétique

Cette technique a été développée par Michalewicz [31]. Lorsque cette opération est opérée sur les parents C^1 et C^2, deux enfants (progénitures) sont générés $H^k = (h^k_1 , k^k_2 , \dots , h^k_i , \dots , h^k_L)$ avec k=1,2 tels que

$$h^1_i = \lambda_i c^1_i + (1-\lambda_i)c^2_i \qquad \text{et} \qquad h^2_i=\lambda_i c^2_i + (1-\lambda_i)c^1_i \qquad (3.8)$$

Dans le cas d'un croisement arithmétique uniforme, la valeur de λ_i est une constante choisie par l'utilisateur, par contre si la valeur de λ_i est générée aléatoirement alors nous sommes dans le cas d'un croisement arithmétique non uniforme. La figure 10 montre un domaine d'exploration borné du fait que les valeurs des gènes des parents c^k_i appartiennent à un intervalle $[a_i , b_i]$ et que la valeur de λ appartient au domaine $[0, 1]$.

3.6.1.5 Croisement BLX-α

Eshelman et al [32] ont développé cet opérateur de croisement. Soit l'enfant généré $H=(h_1 , k_2 , \dots , h_i , \dots , h_L)$ où h_i est choisi aléatoirement dans l'intervalle $[c_{min} - I\alpha , c_{max} + I\alpha]$ tels que

$$c_{max} = max(c^1_i , c^2_i), c_{min} = min(c^1_i , c^2_i) \text{ et } I = c_{max} - c_{min} \qquad (3.9)$$

Il est difficile de fixer la valeur du paramètre α. Eshelman et al [32] suggèrent que la valeur de α soit égale à 0.5 afin d'avoir un rapport équilibré entre la divergence (exploration) et la convergence (exploitation).

Nous remarquons que le domaine des valeurs des gènes du chromosome généré n'appartient plus au domaine $[a_i , b_i]$ des parents. Au fur et à mesure, ce domaine va s'élargir ou se rétrécir. L'explication est la suivante :

sachant que $c_{min} \in [a_i , b_i]$ et $c_{max} \in [a_i , b_i]$ alors $I \in [a_i , b_i]$.

Par contre l'intervalle dont les valeurs obtenues du produit $I.\alpha$ est défini en fonction de la valeur de α. Pour rester dans le domaine de recherche $[a_i , b_i]$, alors certaines conditions doivent être respectées pour déterminer un paramètre α ayant de bonnes propriétés. La figure 10 montre un domaine borné dans le cas du croisement BLX-α pour lequel les auteurs ont supposé que $c^1_i \leq c^2_i$ et $f(C^1) \geq f(C^2)$ [27].
En résumé, ce type de croisement élargit le domaine de recherche, d'où une exploration très vaste, ceci rend difficile la convergence vers des solutions envisageables.

3.6.1.6 Croisement linéaire

Le croisement linéaire génère trois progénitures $H^k = (h^k_1 , k^k_2 , \dots , h^k_i , \dots , h^k_L)$ avec k=1,2,3 tels que

$$h^1_i = 1/2\,(c^1_i + c^2_i) \qquad h^2_i = 3/2\,c^1_i - 1/2\,c^2_i \quad h^3_i = -1/2\,c^1_i + 3/2\,c^2_i \tag{3.10}$$

Lors de la sélection, la nouvelle génération sera formée des deux meilleurs de ces enfants. Nous remarquons que le domaine d'exploration est borné. Cet opérateur a été développé par Wright [33].

3.6.1.7 Croisement discret

Les valeurs des gènes h_i des enfants générés H sont égales aux valeurs des gènes c^1_i du parent C^1 ou aux valeurs des gènes c^2_i du parent C^2. Le choix du parent est effectué d'une façon aléatoire ayant une distribution uniforme. L'intervalle d'exploration est borné.

3.6.1.8 Croisement étendu

Le gène h_i du chromosome H généré par cet opérateur est défini comme suit

$$h_i = c^1{}_i + \alpha\,(c^2{}_i - c^1{}_i) \qquad (3.11)$$

tels que la valeur de α est choisie aléatoirement ayant une distribution uniforme de l'intervalle *[-d, 1+d]*. Dans ce type de croisement, Muhlenbein [34] a déterminé la valeur de *d* à 0. Dans le cas contraire ce type de croisement est appelé le *croisement intermédiaire étendu*. Le bon choix de la valeur de *d* est 0.25 et la valeur de α varie pour chaque gène. L'opérateur de Muhlenbein ne s'applique que sur des gènes de type réel. Autrement dit la formule (3.11) s'écrit

$$h_i = c^1{}_i + \alpha_i\,(c^2{}_i - c^1{}_i) \qquad (3.12)$$

Nous remarquons que les valeurs des gènes des chromosomes générés n'appartiennent plus à l'intervalle $[a_i\ ,\ b_i]$. Ce qui fait que le domaine d'exploration est assez vaste (même remarque que le croisement BLX-α).

3.6.1.9 Quelle technique choisir?

Les avis des auteurs divergent en ce qui concerne le choix d'une technique de croisement lorsque la représentation du chromosome est binaire. Certains préfèrent le croisement multipoint (multiple) et d'autres sont portés sur le croisement uniforme. Plusieurs articles ont comparé un ensemble de techniques de croisement. Les conclusions sont assez diversifiées dû à la nature des différents problèmes étudiés. Par exemple, Eshelman et al [35] ont observé que le croisement multipoint (multiple) et uniforme donnent de meilleurs résultats. Il s'avère que le croisement avec 8 points de coupure est la meilleure technique pour obtenir une solution adéquate en fonction du problème à optimiser. De Jong et Spears [36] ont précisé que le croisement à 2 points de coupure ne fonctionne pas à la perfection lorsque la taille de la population est grande. Ils ont ajouté que les performances du système à optimiser sont robustes lors de l'utilisation du croisement uniforme sur une population de petite taille.

Pour les cas où la représentation des chromosomes est réelle, Herrera et al [27] traitent un ensemble d'opérateurs pour la résolution des problèmes d'optimisation standards. Ils ont obtenu des résultats satisfaisants en utilisant l'opérateur de croisement BLX-α.

A partir de l'ensemble des techniques de croisement citées précédemment, nous constatons que peu importe la technique utilisée, il faut respecter le concept de base du croisement. Ce concept est de permuter les valeurs des gènes de deux parents pour former des progénitures et d'explorer le domaine de recherche dans l'espace des solutions. Pour faire un bon choix d'une technique de croisement, il suffit d'effectuer plusieurs expériences avec différentes techniques. Pour raffiner cette recherche, les techniques de sélection standards (par exemple la roulette biaisée) et l'opérateur de mutation vont nous permettre de converger plus rapidement vers des solutions.

Certaines remarques sur le croisement sont illustrées dans les points suivants :

- Le croisement est la clef de la puissance des AG. Il est directement lié à l'aptitude qu'a une population d'individus d'explorer son espace de recherche et de combiner entre eux les meilleurs résultats. Grâce au croisement, les AG se concentrent sur les parties les plus prometteuses de l'espace des solutions du fait que cet opérateur de croisement combine des chaînes contenant des solutions partielles.
- Le croisement n'est habituellement pas appliqué à toutes les paires d'individus choisis aléatoirement lors de la reproduction. La probabilité du croisement appliquée est comprise entre 0.6 et 1.0 [38]. Dans le cas où le croisement ne s'appliquerait pas, alors les enfants sont semblables aux parents.

3.6.2 Techniques de mutation

Cet opérateur est appliqué sur chaque chromosome issu de l'opération de croisement ou appartenant à une population. L'action de l'opérateur de mutation consiste à changer ou à permuter des valeurs des gènes du chromosome C avec une probabilité p_{mut}. Cette probabilité de mutation est assez faible en pratique. On peut observer à un moment du cycle de l'AG que des individus générés n'évoluent plus. Pour les faire évoluer, l'opération de mutation peut modifier les valeurs des gènes pour constituer des

individus non similaires. Cet opérateur permet une recherche de la solution au problème à optimiser dans un domaine très restreint. L'utilité de cet opérateur est donc l'exploitation de l'espace de recherche des solutions. Ces mutations ne créent généralement pas de meilleures solutions au problème mais elles évitent l'établissement de populations uniformes incapables d'évoluer. Ceci permet à l'AG de converger vers des solutions globales. A partir d'une exploration de l'espace de recherche, la mutation permet de passer de l'exploration vers l'exploitation et de trouver un ensemble de solutions. Cependant, plusieurs techniques de mutation ont été développées dans la littérature. Certaines d'entre elles s'appliquent sur des gènes dont la représentation est binaire et d'autres sur des gènes de type réel. Pour déterminer le nombre de positions dont les gènes doivent subir un changement, il suffit de connaître la taille du chromosome L et la probabilité de mutation p_{mut}. Ce nombre est défini par le produit de $L \times p_{mut}$.

Posons la notation suivante du chromosome qui doit subir la mutation

$C = (c_1 , c_2 , \dots , c_i , \dots , c_L)$ tels que $c_i \in [a_i , b_i]$ ou $c_i \in \{0,1\}$. Le résultat obtenu par l'opérateur de mutation est un chromosome C' tels que $C' = (c'_1 , c'_2 , \dots , c'_i , \dots , c'_L)$.

3.6.2.1 Mutation aléatoire

Dans le cas binaire, si la valeur du gène à muter est égale à 1 alors elle est inversée à 0 et si la valeur du gène est égale à 0 alors elle est inversée à 1. La figure 11 illustre l'effet de la mutation sur une chaîne binaire d'un chromosome. La position du gène qui doit subir la mutation est déterminée aléatoirement.

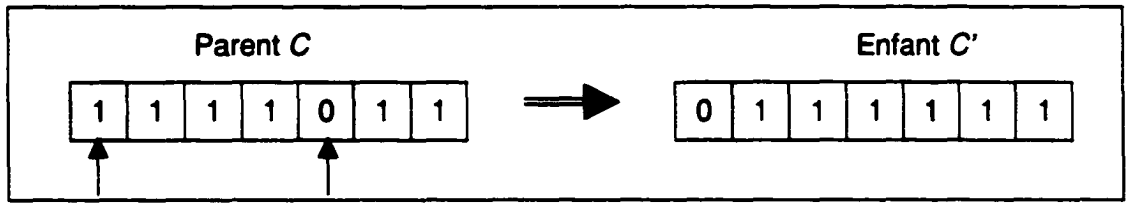


Figure 11 - Mutation aléatoire binaire

Dans le cas réel, le gène c_i est modifié par le gène c'_i. Cette valeur du gène c'_i est choisie aléatoirement de l'intervalle $[a_i , b_i]$.

3.6.2.2 Mutation non uniforme

Cette technique est appliquée en fonction de la génération t courante et le nombre maximum de générations *gen_max*. Le gène c'_i est défini comme suit

$$c'_i = \begin{cases} c_i + \Delta(t, b_i - c_i) & \text{si } \tau = 0 \\ c_i - \Delta(t, c_i - a_i) & \text{si } \tau = 1 \end{cases} \qquad (3.13)$$

où τ est généré aléatoirement tels que $\tau \in \{0 , 1\}$ et

$$\Delta(t, y) = y(1 - r^{(1 - \frac{t}{gen\_max})^b}) \qquad (3.14)$$

où r est un nombre aléatoire de l'intervalle [0,1]
et b est un paramètre choisi par l'utilisateur. Ce paramètre permet de déterminer le degré de dépendance sur le nombre d'itérations.
Cette fonction (3.14) retourne une valeur appartenant à l'intervalle $[0 , y]$ et permet de définir les intervalles de types exploration et exploitation. Cette technique s'applique sur des gènes dont la représentation est réelle.

Pour visualiser le comportement de la fonction (3.13) (soit la notation *f(t)*), nous avons effectué quelques simulations en faisant varier le nombre maximum de générations de 0 à *gen_max* et la valeur du paramètre *b* sur la plage des valeurs comprise entre 0 et 100. Les résultats obtenus sont illustrés par les graphiques 1 et 2.
Nous avons choisi deux valeurs du nombre maximum de génération : 10 000 et 1000. Le graphique 1 représente le cas où le nombre de génération est égale à 10 000 et le graphique 2 représente le second cas ou le nombre de génération est égale à 1000. Le choix de ces deux valeurs permet d'analyser l'influence du nombre maximal de générations sur la forme de la fonction (3.13).

Posons $r = 0.0037$, $c_i = 0.0897$, $a_i = 0$ et $b_i = 1$. L'équation (3.13) s'écrit

si $\tau = 0$ alors $$c'_i = 1 + (c_i - 1) r^{(1 - \frac{t}{Nbre\_gen})^b} \quad (3.15)$$

→ si la valeur de t est très petite (tend vers 0)

alors $r^{(1 - \frac{t}{Nbre\_gen})^b}$ tend vers r.

et l'équation (3.15) tend vers $1 + r(c_i - 1)$.

→ si la valeur de t est très grande (tend vers *Nbre_gen*)

alors $r^{(1 - \frac{t}{Nbre\_gen})^b}$ tend vers 1.

et l'équation (3.15) tend vers c_i.

si $\tau = 1$ alors $$c'_i = c_i r^{(1 - \frac{t}{Nbre\_gen})^b} \quad (3.16)$$

→ si la valeur de t est très petite (tend vers 0)

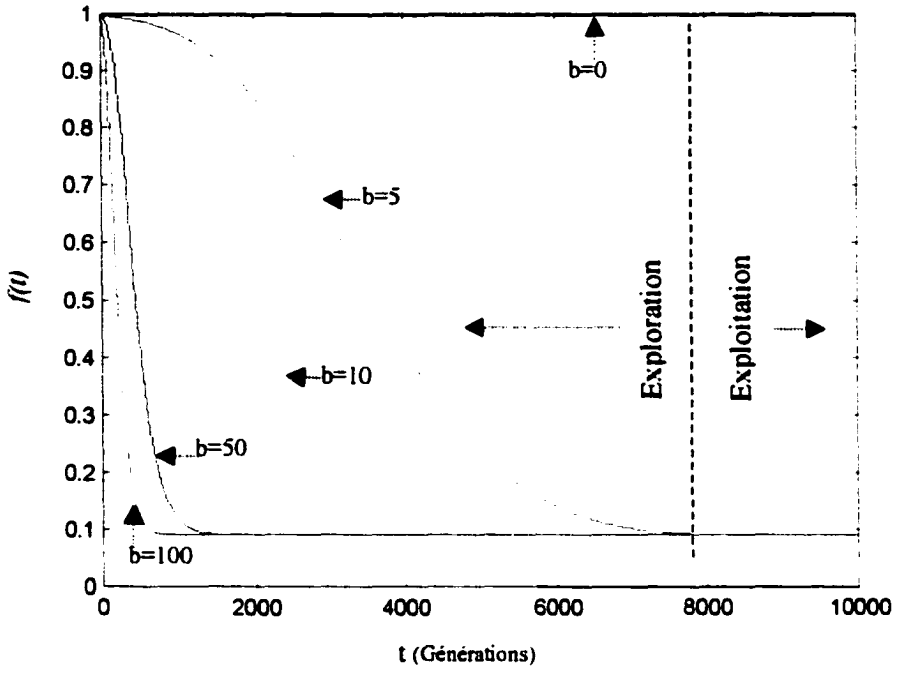
alors $r^{(1 - \frac{t}{Nbre\_gen})^b}$ tend vers r.

et l'équation (3.16) tend vers $r\, c_i$.

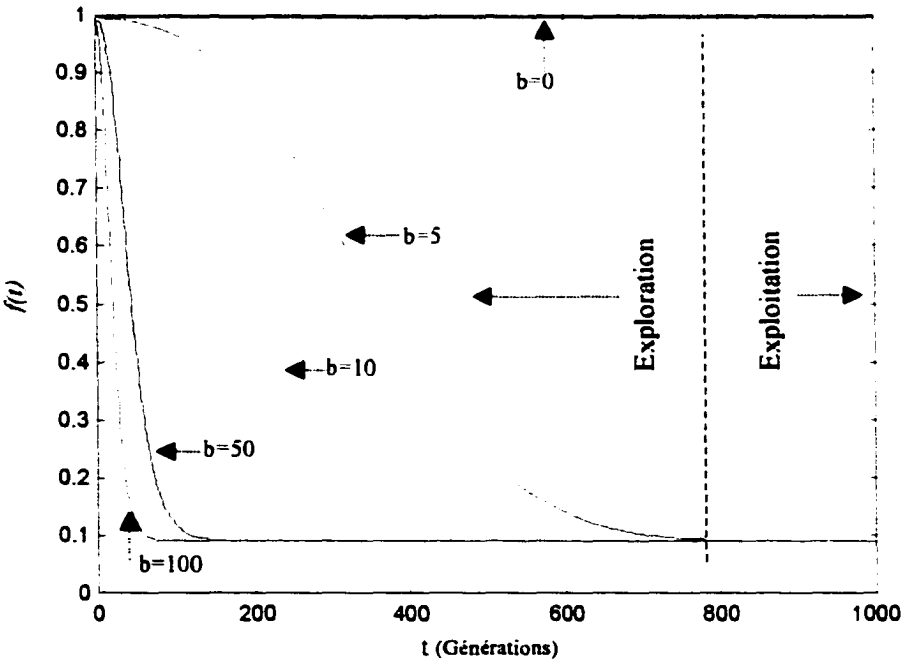
→ si la valeur de t est très grande (tend vers *Nbre_gen*)

alors $r^{(1 - \frac{t}{Nbre\_gen})^b}$ tend vers 1.

et l'équation (3.16) tend vers c_i.



Graphique 1 - Opérateur de mutation non uniforme avec *gen_max*=10 000



Graphique 2 - Opérateur de mutation non uniforme avec *gen_max*=1000

Lorsque la valeur du paramètre *b* est égale à 0, nous observons dans les deux graphiques 1 et 2 que les valeurs des gènes des enfants sont toutes égales aux valeurs des gènes des parents et la fonction *f(t)* n'évolue pas. Ce cas de figure est bien entendu à éviter. Par contre lorsque la valeur de *b* augmente, le domaine d'exploration diminue au fur et à mesure et le domaine d'exploitation s'agrandit. Par exemple lorsque la valeur de *b* = 5, nous observons que jusqu'à la génération 8000 (graphique 1) et la génération 800 (graphique 2), l'AG ne fait qu'explorer l'espace des solutions. Pour le reste des générations, la phase d'exploitation est entamée pour trouver les solutions globales. Dans le cas ou le nombre de générations est de 10000, le temps de calcul est plus grand que celui de 1000 générations et la solution obtenue est identique dans les deux cas. Nous constatons que le choix du nombre de génération n'influe pas sur les résultats. Pour le choix de la valeur du paramètre *b*, il est préférable qu'elle soit égale à 5 car l'espace de recherche est très bien exploré et le passage vers l'exploitation donne une solution assez rapide. Par contre lorsque la valeur de *b* est assez grande (supérieure à 10), l'espace de recherche n'est pas exploré de façon satisfaisante contrairement à l'exploitation.

3.7 Réinsertion

Il existe plusieurs stratégies qui peuvent être adoptées dans la génération d'une nouvelle population. Dans un cycle de génération, un individu de la population courante est remplacé par un nouvel enfant. Il est fort probable que le meilleur individu de la génération courante ne sera pas reproduit dans la nouvelle génération. Alors il est intéressant d'introduire la stratégie de l'*élitisme* [37]. Elle consiste à copier le meilleur ou un sous-ensemble des meilleurs chromosomes de la génération courante à la prochaine génération. Cette stratégie d'élitisme permet d'augmenter la dominance des meilleurs individus dans une population ce qui permet une amélioration des solutions à obtenir.

3.8 Le choix des paramètres d'un algorithme génétique

Pour lancer l'AG, il faut définir certains paramètres tels que : la taille de la population, les probabilités de mutation et de croisement et le nombre de générations. Il est difficile de les fixer ou de trouver les meilleurs avant l'exécution de l'algorithme [37,38]. C'est un problème de réglage qui doit être optimisé pour chaque type de problème traité. Cela constitue une part importante du travail de l'expérimentateur. Dans la littérature, la définition de ces paramètres diffère d'une application à une autre. Les paramètres que nous abordons sont : la taille de la population, le nombre de générations et les probabilités de reproduction.

3.8.1 Taille de la population

L'AG nécessite la détermination du nombre d'individus qui constituent la population *P*. Le problème qui se pose est comment fixer la taille de cette population. Une population trop petite évolue probablement vers un optimum local peu intéressant. Une population trop grande met plus de temps pour converger vers des solutions envisageables. La taille de la population doit être choisie de façon à réaliser un bon compromis entre le temps de calcul et la qualité du résultat. Eiben et al [38] mentionnent que plusieurs chercheurs se sont penchés sur ce problème qui consiste à déterminer la taille optimale de la population pour atteindre la meilleure solution. Certains d'entre eux l'ont fixé entre 20 et 100 de manière empirique. D'autres essayent de l'ajuster en respectant le taux d'erreur de sélection ou de la faire varier d'une population à une autre. Par contre, Bautista et al [39] ont fixé le nombre d'individus à 50 et ont mentionné qu'il peut exister une relation entre la taille de la population et le nombre de gènes. Eiben et al recommandent que la taille d'une population moyenne soit comprise entre 20 et 30.

3.8.2 Nombre de générations

C'est un chiffre que l'expérimentateur doit fixer. Il est préférable qu'il soit assez grand afin de mieux visualiser la convergence de la solution. Certains auteurs utilisent un

nombre de 1000 générations et d'autres 10 000. L'essentiel est de trouver des solutions en un nombre réduit de générations.

3.8.3 Probabilité des opérateurs génétiques

Le choix de la probabilité de mutation p_{mut} et la probabilité du croisement p_{cross} est un problème d'optimisation non linéaire complexe à résoudre [23]. En outre, pour les fixer, ces probabilités de mutation et de croisement dépendent de la nature de la fonction objective du problème à résoudre. Man et al [23] ont cité quelques exemples tels que :

- Pour une population de taille grande = 100
  $p_{mut} = 0.001$ $p_{cross} = 0.6$
- Pour une population de taille petite = 30
  $p_{mut} = 0.01$ $p_{cross} = 0.9$

Mitchell [40] a, quant à elle, suggéré de fixer p_{mut} à 0.001 et p_{cross} à 0.6. Einben et al [38] rapportent une formule proposée par plusieurs auteurs pour déterminer la probabilité de mutation. Cette probabilité dépend de la taille du chromosome et est exprimée par la l'équation suivante

$$p_{mut} = 1/L \tag{3.17}$$

où L est la taille du chromosome en gènes.

Lorsque la technique de mutation est non uniforme, l'équation (3.17) ne peut être appliquée. Car cette technique nécessite une mutation sur plusieurs valeurs des gènes. Ainsi, il préférable de choisir une probabilité qui approxime $10/L$ au lieu de la probabilité $1/L$. La probabilité $1/L$ signifie la mutation maximale d'un gène dans un chromosome de taille L.

Concernant la probabilité de croisement, les recherches n'ont pas pu déterminer la valeur optimale due au fait que la probabilité de croisement s'opère sur une paire de chromosomes contrairement à la probabilité de mutation qui s'opère sur les gènes de

chaque chromosome. Les probabilités de croisement les plus couramment utilisées appartiennent à l'intervalle [0.6,0.95]. Il est préférable de ne pas utiliser un taux trop faible. Eiben et al [38] mentionnent que les valeurs inférieures à 0.6 sont rarement utilisées.

3.9 Algorithme génétique simple versus algorithme génétique itératif

Les algorithmes génétiques itératifs (AGI) décrits par Man et al [23] sont utilisés conjointement avec l'AG décrit ci-haut appelé AG simple (AGS). L'objectif des AGI consiste à trouver une solution optimale tout en appliquant les même opérateurs de reproduction. Théoriquement, l'utilisation des AGI permet une convergence plus rapide car au fur et à mesure l'espace de recherche diminue à chaque itération. Oliveira et Benahmed [57] décrivent en détail la démarche suivie par l'AGI pour la sélection de primitives.
Les étapes de l'AGI sont décrites ci-dessous :

- ***Itération 1 :*** Soit N_2 le nombre total de primitives considérées par le réseau de neurones (au départ $N_2 = L = 132$ primitives). L'AGS est appliqué et donne comme solution x_1 avec $f_1(x_1) \approx 0$. La fonction f_1 représente un des objectifs de la fonction d'adaptation (fitness). La fonction f_2 représente le nombre de primitives à optimiser (voir figure 12a).

- ***Itération 2 :*** Supposons que $f_2(x_1)=N_3$. Le domaine de recherche est réduit et la complexité de la topologie du réseau de neurones est également réduite de N_2 à N_3-1. L'AGS est appliqué encore une fois jusqu'à ce qu'une autre solution avec $f_1(x_2) \approx 0$ soit obtenue (voir figure 12b).

- ***Itération 3 :*** Répéter l'étape 2 de sorte qu'à chaque itération, le domaine de recherche sera réduit en passant par une topologie de complexité plus petite. Éventuellement la solution optimale x_{opt} avec $f_2(x_{opt})=N_1$ sera obtenue (voir figure 12c).

- ***Itération 4 :*** Une autre itération avec le nombre de primitives borné par N_1-*1* est effectuée. Évidemment aucune solution ne peut être trouvée par l'AG simple. Le processus de recherche est terminé en fixant un nombre maximum de générations pour l'AGS. Si aucune solution n'est trouvée lorsque le nombre maximal d'itérations est atteint alors la solution obtenue à l'étape 3 sera considérée comme la solution optimale pour ce problème. Ainsi le nombre de primitives à minimiser est atteint. (voir figure 12d).

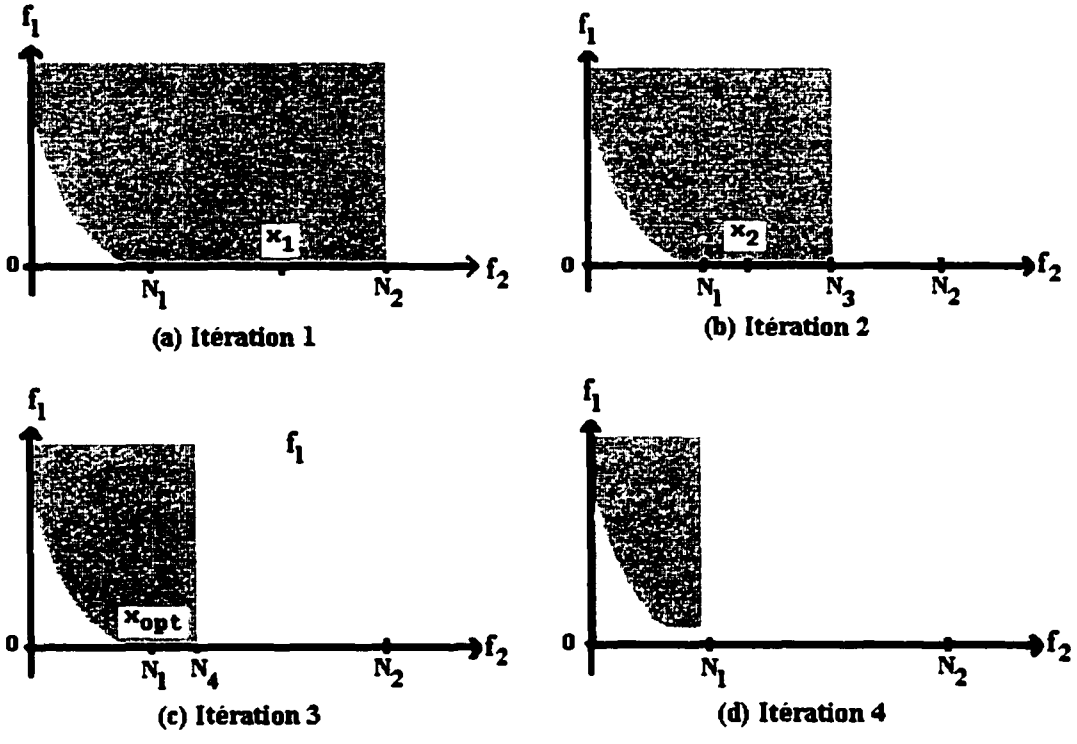


Figure 12 - Approche itérative

La même méthodologie et les mêmes paramètres définis pour l'AGS sont utilisés par l'AGI. Il existe une légère différence entre les AGS et les AGI. Cette différence se situe en deux points : le mécanisme de recherche et l'initialisation de la population.
Le mécanisme de recherche est illustré à la figure 12. La meilleure solution trouvée dans l'itération précédente est utilisée pour initialiser la nouvelle population de l'itération courante.

L'initialisation de la population a été modifiée dans cette approche afin de permettre une recherche plus localisée à chaque itération. Nous avons calculé la distance de Hamming *(d)* entre la solution injectée et les chromosomes produits par la population initiale car celle-ci permet une exploration en profondeur dans le domaine de recherche restreint et est définie par

$$d(C_1,C_2)<\psi \tag{3.18}$$

où C_1 :est le chromosome qui représente la meilleure solution trouvée à l'itération précédente de l'algorithme (dans le cas de l'itération 1, C_1 est représenté par des valeurs des gènes égaux tous à 1, dans le cas de la sélection des primitives, toutes les primitives sont alors considérées),

C_2 : est le chromosome généré lors de l'initialisation de la population.

ψ : est un seuil qui définit la distance maximale entre C_1 et C_2. Ce seuil est déterminé par l'expérimentateur et d'après le problème à résoudre.

L'AGI permet donc de balayer un sous espace de recherche d'une manière très efficace.

3.10 Conclusion

Ce chapitre nous a permis d'avoir une vue générale sur les concepts des AGS. Nous pouvons conclure que les AG sont des algorithmes simples de conception et peuvent résoudre des problèmes assez complexes. La résolution de ces problèmes est obtenue grâce aux opérateurs de reproduction. Ces AG sont des procédures assez robustes pour résoudre un problème d'optimisation pour la sélection des primitives. Néanmoins elles présentent certaines limites et des difficultés. Ces difficultés reposent sur le choix des bons paramètres tels que : la taille de la population, le nombre de génération, les probabilités de croisement et de mutation et les méthodes des opérateurs de reproduction. Ces paramètres dépendent du problème à résoudre et d'une codification appropriée au problème à solutionner.

Nous avons observé que certains résultats obtenus sont mauvais lorsque la convergence tend vers un optimum local. L'origine de ce type de phénomène est causée par une mauvaise initialisation du domaine de recherche ou par un mauvais choix de paramètres. Lors de l'initialisation du domaine de recherche, il est difficile d'atteindre la solution. Pour déterminer des bons paramètres et un bon choix des méthodes de reproduction, il est nécessaire d'effectuer plusieurs expériences. Le principal impact négatif d'un mauvais choix des paramètres, autrement dit lorsque les paramètres sont fixés à des valeurs non extrêmes, est un temps d'exécution trop long de l'algorithme. Un autre aspect important est le choix du critère d'arrêt. Le problème qui se pose est comment savoir que la solution optimale du problème a été trouvée. Il est difficile de répondre à cette question mais nous pouvons envisager quelques réponses comme : déterminer un grand nombre de génération supérieur à 10000 jusqu'à que la valeur de la fonction d'adaptation n'évolue plus durant quelques centaines de générations.

Dans le cadre de notre projet, nous avons effectué plusieurs expériences pour déterminer ces paramètres tels que la taille de la population, le nombre de génération et les probabilités de mutation et de croisement. Ces expériences sont décrites au chapitre 5.

CHAPITRE 4

L'OPTIMISATION D'UN SYSTÈME DE RECONNAISSANCE DE CHIFFRES ISOLÉS

4.1 Introduction

L'objectif de notre étude est la sélection des primitives en utilisant la technique d'optimisation que sont les AG. Cette sélection permet la détermination des primitives appropriées aux chiffres isolés et l'élimination des primitives redondantes ou non significatives. Tout en effectuant cette sélection, il est important de maintenir ou d'améliorer la performance du système de reconnaissance. Le système que nous étudions est l'optimisation du nombre des primitives pour la reconnaissance des chiffres isolés basée sur les travaux effectués par Oliveira dans le cadre de sa thèse de doctorat actuellement en cours. Les résultats qui seront obtenus dans notre projet vont servir à l'optimisation d'une partie de son système qui est constitué de 3 sous-systèmes. Notre étude est considérée comme une phase exploratoire de la thèse de doctorat de Oliveira. La méthode d'optimisation utilisée et les résultats obtenus vont orienter une partie des travaux de Oliveira concernant la sélection des primitives.

Nous expliquons dans ce chapitre la méthodologie adoptée pour la sélection des primitives par les algorithmes génétiques. En premier lieu, nous donnons un aperçu sur la base de données sur laquelle nos expériences ont été effectuées. Par la suite nous décrivons les différentes étapes de reconnaissance de formes en expliquant les pré-traitements effectués sur la base de données, les primitives extraites et l'utilisation du réseau de neurones de type MLP.

4.2 Base de données

Dans tout système de reconnaissance de formes, il est nécessaire d'avoir une base de données afin d'effectuer tous les traitements. La base de données que nous avons utilisée dans notre projet est la base *NIST SD19*. Cette base contient des chiffres manuscrits, isolés et stockés sous forme d'images binaires. Le nombre de classes est égale à 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). La figure 13 donne un petit aperçu de cette base :

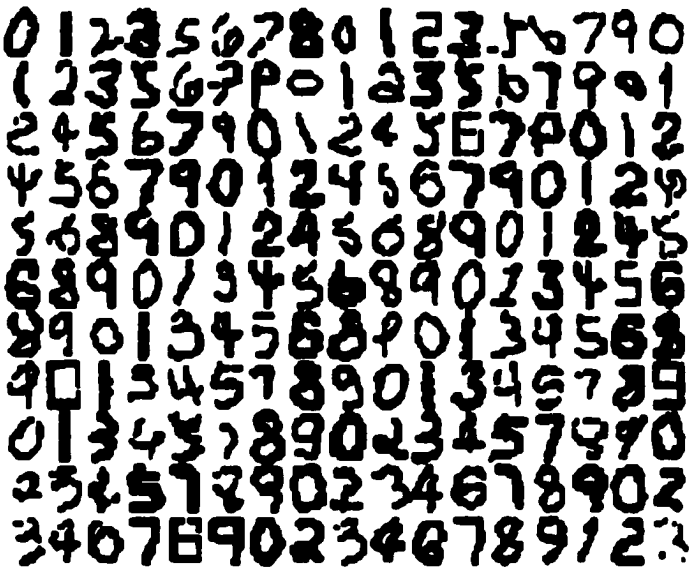


Figure 13 - Aperçu sur la base de données

Cette base de données est partagée en 3 sections : hsf_{0123}, hsf_7 et hsf_4. Chacune des sections est constituée des répertoires (0,1,2,3,7,4) qui contiennent des images de chiffres isolés. Ces sections correspondent respectivement à l'apprentissage, à la validation et au test. Le tableau (II.a) montre le nombre de chiffres par classe et par section. La base d'apprentissage à son tour est subdivisée en deux bases : une base contenant 195 000 chiffres isolés et le reste c'est à dire les 28 123 chiffres sur une autre base. Une appellation pour chaque base de données est utilisée et est décrite dans le tableau (II.b).

Tableau II

Répartition des bases de données

Classe	hsf _{0123}	hsf_7	hsf_4
0	22971	5893	5560
1	24771	6567	6655
2	22131	5967	5888
3	23172	6036	5819
4	21549	5873	5722
5	19545	5684	5539
6	22128	5900	5858
7	23208	6254	6097
8	22029	5889	5695
9	21619	6026	5813
Total	**223123**	**60089**	**58646**
	Base d'apprentissage	Base de validation	Base de test

(a)

Base A	Base B	Base C	Base D
195000	28123	60089	58646

(b)

4.3 Pré-traitements

Les pré-traitements effectués sur cette base de données ont pour objectif le nettoyage des images. Autrement dit le bruit a été supprimé par la technique de filtrage simple [60]. Cette technique utilise un masque de dimension 3x3. Ce masque balaye toute l'image binaire (pixel par pixel) permettant de détecter le bruit et finalement l'éléminer.

4.4 Extraction des primitives

Oliveira a effectué plusieurs simulations pour déterminer les primitives qui semblent être appropriées au système de reconnaissance de chiffres isolés. Les primitives extraites sont : le zonage, la concavité, le contour et la surface.

4.4.1 Zonage

Le zonage consiste à partager l'image en région (ou zones). L'image de départ est une image binaire de taille *NxM*. Cette image est partagée en 6 zones (3x2). Pour chaque zone, un ensemble de primitives est calculé à partir de la concavité (13 primitives), du contour (8 primitives) et du calcul de la surface pour chacune des régions (1 primitive). La figure 14 représente une image partagée en zones.

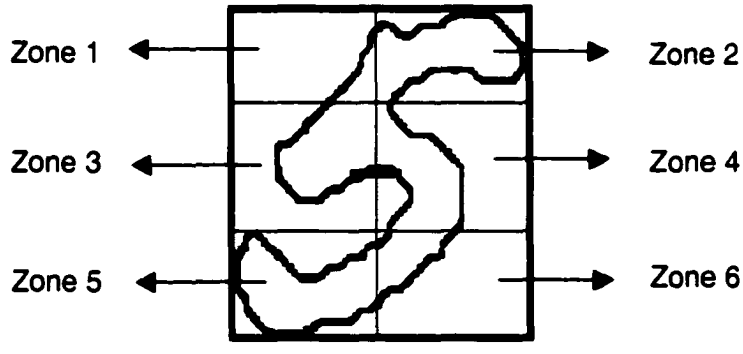


Figure 14 - Division d'une image en zones

Les images n'ont pas la même taille (dimensions) ce qui fait que lors du partage de l'image en zones, les dimensions des zones ne sont pas nécessairement identiques.

4.4.2 Concavité

Les primitives extraites à partir d'une analyse des parties concaves et/ou convexes du chiffre permettent de mettre en évidence les propriétés topologiques et géométriques de la forme. Dans chaque zone d'une image, on utilise le codage de freeman à 4

voisins qui va balayer l'image pixel par pixel. La figure 15 illustre le codage de Freeman à 4 voisins avec les directions dans le cas d'une concavité fermée rencontrée.

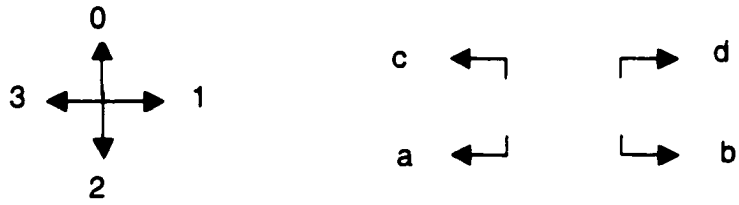


Figure 15 - Codage de Freeman à 4 directions

Chaque composante du vecteur de primitives correspond au nombre de points blancs d'une zone de l'image présentant l'un des types de concavités.

La configuration utilisée est f^i_j où i est la primitive dont le nombre de bits est à 1 et j est la direction parmi les 4 possibilités du code de freeman utilisée. Seules les concavités sans discontinuités, qui sont les plus fréquentes en pratique, sont prises en considération. Certaines configurations sont ignorées car il est peu fréquent ou rare d'atteindre les configurations f^0 et f^1. Dans le cas de la configuration f^0, le nombre de bit à 1 est égal à zéro cela signifie l'existence d'aucun bits aux alentours. De la même façon, la configuration f^1 ne nécessite pas d'être prise. Lorsqu'une concavité fermée se présente (f^4) alors 4 cas de figures se présentent pour pouvoir sortir de cette concavité à savoir les directions : nord ouest, nord est, sud ouest et sud est. La figure 16 illustre le cas d'une concavité fermée.

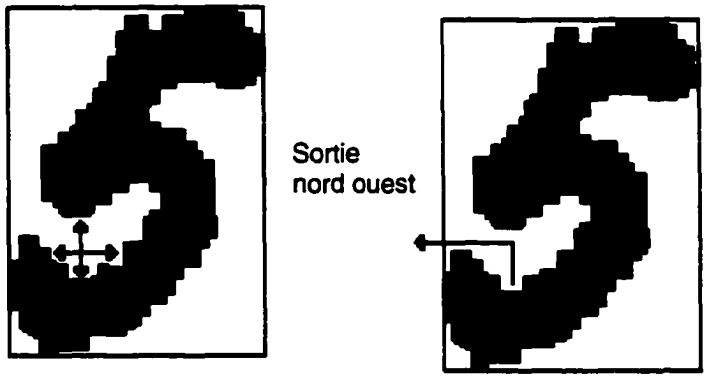


Figure 16 - Concavité fermée

Le nombre total de primitives de type concavité est de 13. Ces primitives sont :

f^2_{01}	f^2_{12}	f^2_{23}	f^2_{03}	
0011	1001	1100	0110	
f^3_0	f^3_1	f^3_2	f^3_3	
0111	1011	1101	1110	
f^4	f^4_a	f^4_b	f^4_c	f^4_d
1111	1111	1111	1111	1111

4.4.3 Contour

L'extraction des contours d'un caractère est aussi une opération qui permet de réduire la quantité d'information à traiter. Les contours d'un caractère apportent autant d'informations que le caractère (chiffre) lui-même. Les contours d'une forme binaire peuvent être obtenus très facilement à partir du code de Freeman à 8 voisins et le nombre de primitives extraites est ainsi de 8. La recherche du contour est appliquée sur chaque zone d'une image.

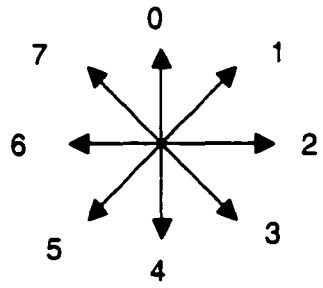


Figure 17 - Codage de Freeman à 8 directions

Chaque caractéristique représente le cumul du nombre de points (pixels) noirs dans la direction donnée par le codage de freeman. La figure 18 représente un exemple de contour d'un chiffre numérique isolé.

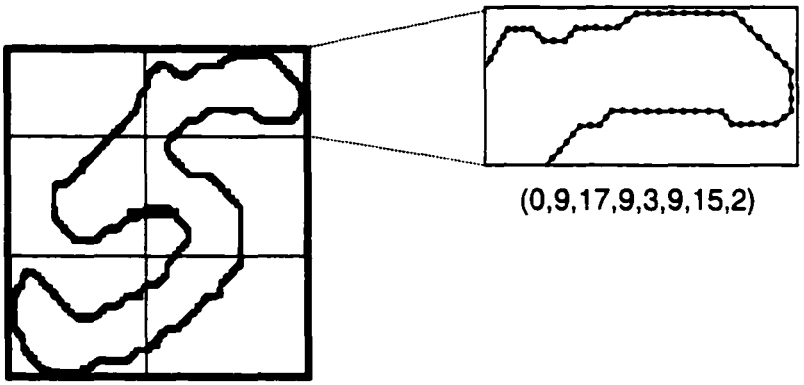


Figure 18 - Détection du contour d'un chiffre

4.4.4 Surface

La surface consiste à calculer le nombre de pixels noirs dans chaque zone de l'image binaire (représentant le chiffre). Par exemple dans la figure 18, la surface de la zone 2 qui est égale à 64 correspond au nombre total de pixels noirs dans cette zone.

4.4.5 Constitution du vecteur de caractéristiques

On obtient un vecteur de caractéristiques qui regroupe 132 primitives réparties de la manière suivante :

13 primitives de type concavité par région.
08 primitives de type contour par région.
01 primitive de type surface par région.
06 régions.

Le format du vecteur de primitives est illustré dans la figure 19 :

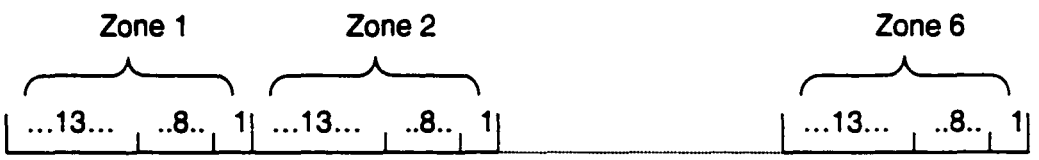


Figure 19 - Présentation du vecteur de primitives

Chaque primitive du vecteur est normalisée dans un intervalle [0,1] d'après le type de primitive (concavité, contour et surface). La valeur de la primitive après sa normalisation appliquée pour chaque type de primitive est la suivante

$$z'^{i}_{j} = \frac{z^{i}_{j}}{\sum_{j} z^{i}_{j}} \qquad (4.1)$$

où z^{i}_{j} représente la $j^{ème}$ primitive pour un type donnée dans la $i^{ème}$ zone.
Par exemple, dans la zone 2 (voir figure 18), les primitives de type contour obtenues sont (0,9,17,9,3,9,15,2). Après avoir appliqué la normalisation (équation 4.1), nous obtenons les nouvelles valeurs normalisées des primitives de type contour qui sont (0/64, 9/64, 17/64, 9/64, 3/64, 9/64, 15/64, 2/64).

Avant d'obtenir cette famille de primitives (concavité, contour, surface et zonage), plusieurs expériences ont été effectuées avec d'autres types de primitives. Par conséquent, le choix a été bien exploré sauf qu'il faudra faire une sélection afin de garder les primitives les plus utiles et de supprimer toutes celles qui sont redondantes et inutiles en appliquant les AG. Ceci est l'objectif de ce mémoire.

4.5 Sélection des primitives

Notre objectif est la sélection des primitives pertinentes dans le cadre de notre problème de reconnaissance de chiffres isolés tout en essayant de maintenir ou d'améliorer la performance du ce système de reconnaissance. Ces primitives sont représentées dans le vecteur qui regroupe *L* primitives. Pour résoudre ce problème de sélection des primitives, on utilise les AG. La fonction d'adaptation définie est multicritère : minimiser le nombre de primitives et minimiser l'erreur en généralisation du système de reconnaissance de chiffres. Il faudra donc trouver *M* primitives tels que $M < L$. Les étapes de construction d'un AG utilisé pour la sélection des primitives sont :

- ***Le codage des chromosomes.*** Le codage des chromosomes est assez simple dans le cas de la sélection des primitives car le codage est binaire. Cela signifie que le chromosome sera sous la forme d'une chaîne binaire. La présence du bit 1 signifie que la primitive est sélectionnée et la présence du bit 0 signifie que la primitive ne l'est pas. La taille du chromosome est identique à la taille du vecteur de primitives, soit $L = 132$.

- ***La définition de la fonction d'adaptation.*** Notre fonction d'adaptation (fitness) est une fonction multicritère. Nous utilisons deux fonctions objectives à atteindre. Nous avons choisi une représentation de la fonction la plus commode qui est une somme pondérée des fonctions objectives. Le premier objectif consiste à minimiser le nombre de primitives. Le second objectif est la minimisation du taux d'erreur lors de la classification. Ainsi, la fonction d'adaptation est une combinaison des deux fonctions objectives qu'il faudra minimiser. Cette fonction d'adaptation est décrite par la formule (4.2). Chaque chromosome généré sera évalué par cette fonction.

$$fitness = Minimiser(\alpha f_1 + \beta f_2) \qquad (4.2)$$

où f_1 : est le taux d'erreur obtenu lors de la classification en utilisant le réseau de neurones. Son intervalle d'appartenance est [0,1].
f_2 : est le rapport entre le nombre de primitives sélectionnées du chromosome et le nombre total de primitives (L=132). Son intervalle d'appartenance est [0,1].
α et β : sont deux paramètres en fonction de la normalisation et de la pondération pour chaque objectif.

Le problème qui se pose est comment déterminer les valeurs de ces deux paramètres α et β. Nous les avons fixés arbitrairement à 1000 la valeur de α et 1 la valeur de β. La technique utilisée pour trouver ces valeurs est la suivante

$$\alpha = c_1.w_1 \text{ et } \beta = c_2.w_2 \qquad (4.3)$$

où c_i : est le paramètre de normalisation de l'objectif f_i.

w_i : est le paramètre de pondération de l'objectif f_i, tel que $\sum \omega_i = 1$

Après remplacement, l'équation (4.2) devient

$f = Minimiser(c_1.w_1.f_1 + c_2.w_2.f_2)$

Puisque $w_1 + w_2 = 1$, nous obtenons

$$f_2 = -\frac{w_1 c_1}{w_2 c_2} f_1 + \frac{f}{w_2 c_2} \tag{4.4}$$

La pente de cette fonction f_2 est $-\frac{w_1.c_1}{w_2.c_2}$. Le fait que la fonction d'adaptation est à minimiser et que l'objectif est la sélection des primitives alors la valeur de cette pente doit être très petite. Étant donné que le plus important pour le problème de la sélection des primitives est de diminuer le nombre des primitives alors la plus grande importance est donnée à la fonction objective f_2 au lieu de f_1. D'où le choix des valeurs de α et de β est respectivement 1000 et 1. A partir de ces paramètres, les pondérations de chacun des objectifs sont calculées: $w_1 = 0.3$ et $w_2 = 0.7$. On peut conclure que l'importance de la sélection des primitives est environ deux fois plus grande par rapport à la performance du taux d'erreur.

Une fois ces paramètres définis, la question que nous posons est quelle sera la valeur de la primitive non sélectionnée à l'entrée du réseau. Faut-il lancer l'apprentissage pour chaque chromosome généré afin de déterminer la matrice poids?
Pour palier ces problèmes, la technique de l'analyse de sensibilité répond à ces questions [48, 49, 50].

4.5.1 Analyse de sensibilité

Cette technique a été développée par Moody et al [48, 49, 50]. L'analyse de sensibilité SBP calcule la mesure de sensibilité S_i pour évaluer le changement d'erreur d'apprentissage causée par la suppression de l'entrée z_i du réseau de neurones. La sensibilité de l'entrée i du réseau de neurones est définie par

$$S_i = \frac{1}{N}\sum_j S_{ij} \tag{4.5}$$

où S_{ij} est la sensibilité calculée pour l'exemple z_j de la base de données telle que

$$S_{ij} = SE(\overline{z_i}, w_\lambda) - SE(z_{ij}, w_\lambda) \tag{4.6}$$

où $\overline{z_i} = \frac{1}{N}\sum_{j=1}^{N} z_{ij}$

w : représente la matrice poids généré lors de l'apprentissag

λ : représente une étiquette de l'architecture du réseau de neurones.

S_i : mesure l'effet sur l'erreur quadratique (*SE*) lors de l'apprentissage.

La $i^{ème}$ entrée de l'exemple z_i est remplacée par la moyenne $\overline{z_i}$ calculée pour l'ensemble des exemples de la base d'apprentissage. Le remplacement de la variable par la valeur moyenne supprime son influence sur la sortie du réseau. Lors du calcul de S_i, aucun apprentissage n'est effectué pour évaluer $SE(\overline{z_i}, \omega_\lambda)$. De même il n'est pas suffisant de mettre la valeur z_{ij} à zéro quel que soit j, car les valeurs des connexions des couches cachées sont évaluées lors de l'apprentissage et il est préférable que les valeurs des entrées d'un réseau de neurones ne soient pas nulles mais proches de 0. Cette technique de sensibilité permettra d'éviter la valeur nulle en la remplaçant par la moyenne calculée de la primitive correspondante à cette entrée. L'utilité de cette technique permet de ne pas relancer l'apprentissage du réseau de neurones pour chaque chromosome généré d'où un gain de temps de calcul considérable.

4.6 Pondération des primitives

La pondération consiste à donner un poids à chaque primitive. L'objectif de la pondération est de classer les primitives par ordre d'importance lors de la reconnaissance [52, 59].

Ceci dit, la différence entre la sélection et la pondération des primitives lors des étapes suivies d'un AG est la codification du chromosome et la définition de la fonction d'adaptation. Le choix des paramètres tels que : la taille de la population, le nombre de génération et les méthodes des opérateurs de reproduction, reste un problème. Dans le cadre de notre projet, nous avons effectué plusieurs expériences tout en gardant les même valeurs de certains paramètres (voir chapitre 5).

- ***Codage des chromosomes.*** Les valeurs des gènes du chromosome $C=(c_1,c_2,\ldots,c_i,\ldots,c_L)$ appartiennent à un domaine de valeurs réelles tels que $c_i \in [a_i,b_i]$. C'est une codification réelle où c_i représente le poids du $i^{ème}$ gène de la $i^{ème}$ primitive. Dans notre cas, nous posons $a_i = 0$ et $b_i = 1$.

- ***Définition de la fonction d'adaptation.*** C'est une fonction mono objective. Elle consiste à l'optimisation de la performance du système. Il suffit de maximiser le taux de reconnaissance calculé par le MLP. Les entrées du réseau sont définies comme suit : pour chaque chromosome généré $C = (c_1 , c_2 , \ldots , c_i , \ldots , c_L)$ par l'AG, nous effectuons l'opération de multiplication du chromosome avec le vecteur de primitives. La fonction d'adaptation est définie par l'équation suivante

$$fitness_{pond.} = Maximiser(f_3) \qquad (4.7)$$

où f_3 représente le taux de reconnaissance calculé pour un chromosome donné.

4.7 Classification

Le classifieur utilisé dans ce travail est un réseau de neurones de type MLP à une couche cachée. En général, le nombre de nœuds sur la couche cachée est déterminé à partir de l'heuristique utilisant la formule (2.9) ou par tâtonnement. Plusieurs expériences ont été effectuées pour déterminer le nombre de nœuds sur la couche cachée. Il a été observé que lorsque le nombre de nœuds sur la couche cachée est compris entre 80 et 120, le MLP donne les même performances en taux de reconnaissance des chiffres isolés. Avec 60 nœuds cachés, nous avons observé le

meilleur taux de reconnaissance. L'apprentissage de ce réseau a été effectué par l'algorithme de rétropropagation utilisant la technique de cross-validation pour déterminer le critère d'arrêt.

Les paramètres utilisés sont :

Tableau III

Paramètres de l'algorithme d'apprentissage

- Nombre de couche cachée	= 1
- Nombre de nœuds dans la couche cachée	= 60
- Constante d'apprentissage	= 0.9
- Momentum	= 0.93
- Signal d'erreur	= 0.001
- Nombre maximal d'itérations (cycles)	= 170
- Nombre d'entrées	= 132
- Nombre de sorties	= 10
- Type de la fonction d'activation	= Sigmoïde définie par (2.4)

L'apprentissage du réseau est effectué sur la base A utilisant ces paramètres décrits au tableau III. Une baisse du taux de reconnaissance évaluée sur la base de données B ou le nombre de cycles vont servir de critère d'arrêt de l'apprentissage du réseau. Lorsque un sur-apprentissage est rencontré alors le réseau d'apprentissage s'arrête sinon l'algorithme termine ces traitements jusqu'à atteindre le nombre de cycles maximal. Les bases C et D vont servir à la fin de l'apprentissage pour tester la performance du système de reconnaissance. En utilisant toutes les primitives à l'entrée du réseau de neurones, sans aucune sélection effectuée, nous avons obtenu les résultats suivants :

Tableau IV

Résultats avec toutes les primitives

	Taux de reconnaissance (%) (132 primitives)
Base A	99.65
Base B	99.62
Base C	99.15
Base D	97.47

4.8 Méthodologie

L'objectif de notre projet est d'effectuer une sélection des primitives pertinentes et la pondération des primitives. Pour se faire, nous utilisons la méthode d'optimisation qui est les AG. Pour déterminer les bons paramètres de ces AG, nous avons effectué plusieurs expériences (voir chapitre 5). La méthodologie adoptée pour la sélection et la pondération des primitives est identique. La figure 20 illustre les étapes suivies pour réaliser ce projet.

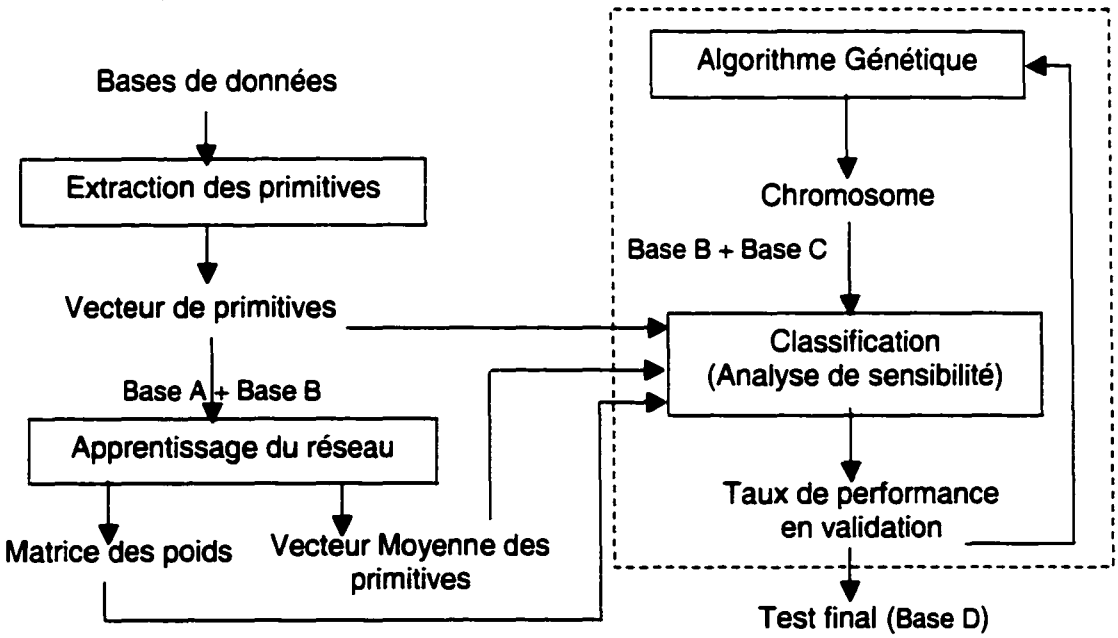


Figure 20 - Aperçu général du système mis en place

La première étape consiste à nettoyer les images de la base de données (Bases : A, B, C et D) puis faire l'extraction des primitives. Les primitives extraites de chaque image sont stockées dans un vecteur de primitives (voir la section 4.4.5). La taille de ce vecteur est de L primitives telle que L = 132 primitives. La seconde étape consiste à constituer la matrice des poids du réseau qui est calculé lors de l'apprentissage. Cet apprentissage est effectué sur la base A et est validé sur la base B (voir la section 4.7). Pour optimiser les entrées du réseau de neurones, les AG sont adoptés. Pour chaque chromosome généré par l'AG, l'évaluation de la fonction d'adaptation est effectuée sur la base B. La base C servira de validation et de critère d'arrêt pour cette optimisation. Dans le cadre de la sélection de primitives, l'analyse de sensibilité est utilisée (voir section 4.5.1). Cette analyse permet de ne pas relancer l'apprentissage du réseau pour chaque chromosome généré ce qui permet de diminuer le temps de calcul. Dans le cas de la pondération des primitives, l'analyse de sensibilité ne peut être utilisée. La dernière étape consiste à utiliser les bases de données C et D pour tester la performance du système de reconnaissance après l'apprentissage du réseau. Dans le cas de la sélection et de la pondération des primitives par l'AG, la base D servira uniquement comme une base de test.

4.9 Conclusion

Nous avons dans ce chapitre décrit la démarche suivie et adoptée pour réaliser notre projet. Le travail est opéré sur la base NIST_SD19. Sur cette base, les différentes étapes de la reconnaissance de formes ont été suivies à savoir les pré-traitements et l'extraction des primitives. L'étape de classification est utilisée dans l'apprentissage du réseau et dans l'optimisation du système de reconnaissance des chiffres isolés. Lors de l'apprentissage du réseau, les poids du réseau de neurones ont été déterminés en présentant les 132 primitives à l'entrée du réseau. L'objectif de cette étude est de minimiser ce nombre d'entrées du réseau tout en améliorant ou en maintenant la performance du système de reconnaissance. La méthode des AG est utilisée pour cette minimisation. L'utilisation classique de l'approche *wrapper* décrite à la section 1.5 consiste lors de la sélection des entrées d'un réseau de supprimer les primitives redondantes ou inutiles, par la suite le réseau d'apprentissage est relancé. Afin d'éviter

de relancer cet apprentissage, la technique de sensibilité développée par Moody et al [48, 49] a été adoptée dans ce travail. Ainsi cette technique de sensibilité permet de minimiser le temps de calcul.

Enfin, ce chapitre nous a permis de décrire la méthodologique adoptée dans le cadre de notre projet. Les expériences effectuées et les résultats obtenus seront décrits au chapitre 5.

CHAPITRE 5

LES EXPÉRIENCES ET LES RÉSULTATS

5.1 Introduction

Nous avons effectué plusieurs expériences dans le cadre de ce projet. Le choix de bons paramètres est une étape essentielle dans les AG. D'après la littérature, il n'existe pas un standard pour déterminer ces paramètres. La détermination de ces derniers diffère d'un problème à un autre. Il faudra simuler plusieurs expériences avec différentes valeurs des paramètres. Sachant que le temps de calcul d'une expérience est prohibitif, il faudra songer à trouver une fonction d'adaptation qui s'apparente à notre problème de sélection de primitives. Nous avons observé que la sélection des primitives se rapproche (ou s'apparente) à celui de la fonction *One-Max* [22]. Cette dernière concerne l'optimisation de la fonction standard qui calcule le nombre maximal ou minimal des valeurs de bits égales à 1 dans un chromosome. Le temps de calcul négligeable par rapport à celui de la sélection des primitives rend cette fonction avantageuse même en utilisant l'analyse de sensibilité. De plus, c'est une fonction qui ne nécessite pas la manipulation de la base de données. Nous avons donc effectué un ensemble d'expériences avec cette fonction *One-Max* pour déterminer les bons paramètres que nous utiliserons par la suite pour résoudre notre problème de sélection de primitives.

Dans ce chapitre, nous abordons dans la première partie le choix des paramètres en utilisant la fonction *One-Max*. La seconde partie est consacrée à la sélection des primitives en appliquant les AGS et les AGI. La dernière partie traite des expériences effectuées dans le cadre de la pondération des primitives. Cette dernière est appliquée dans deux cas de figure. Le premier cas est de réaliser une pondération sur toutes les primitives du système de reconnaissance à savoir les 132 primitives et le second cas consiste à appliquer la pondération après la phase de sélection des primitives.

Pour réaliser ce projet, nous avons utilisé la librairie GALib développé par le MIT [29] à laquelle nous avons rajouté certains opérateurs de reproduction qui n'existaient pas. Les codes source des programmes sont écrits en C et C++.

5.2 Choix des paramètres

Pour déterminer les bons paramètres des AG dont certains sont prédéfinis, nous avons effectué plusieurs expériences. Nous avons travaillé sur la fonction standard d'optimisation *One-Max* qui consiste à minimiser le nombre de bits 1.
Les paramètres pris par défaut sont :

Tableau V

Choix de paramètres de l'AG pour la fonction *One-Max*

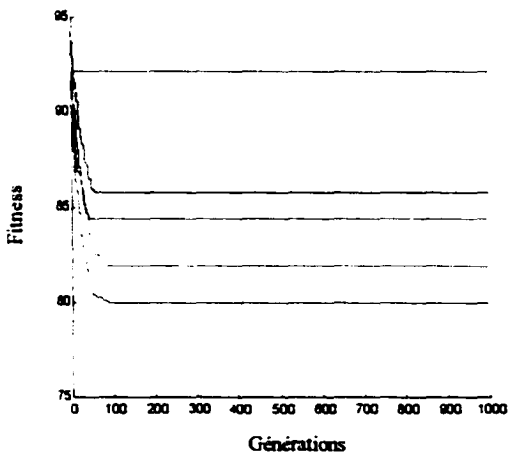
- Codage du chromosome	: Binaire.
- Technique de sélection	: Roulette Wheel.
- Élitisme	: Vrai.
- Taille de la population	: 32.
- Nombre de générations	: 1000.
- Fonction d'adaptation	: Minimiser le nombre de bits à 1 dans le chromosome de taille *L*.
- Taille du chromosome	: $L = 132$.
- Initialisation de la population	: Injecter le chromosome ayant des bits tous à 1.
- Nombre de réplications	: 10.
- Méthode de mutation	: Aléatoire.

Pour déterminer le choix de la méthode de croisement, nous avons utilisé les techniques les plus communes à savoir : le croisement à un point de coupure, le croisement à deux points de coupure et le croisement uniforme. Pour chacun de ces cas, nous avons varié les probabilités de croisement et de mutation. Les deux ensembles de probabilité de croisement et mutation que nous avons utilisé sont respectivement {1, 0.8, 0.6, 0.4, 0.2, 0} et {0, 0.1/L, 1/L, 10/L, 1}. La combinaison entre

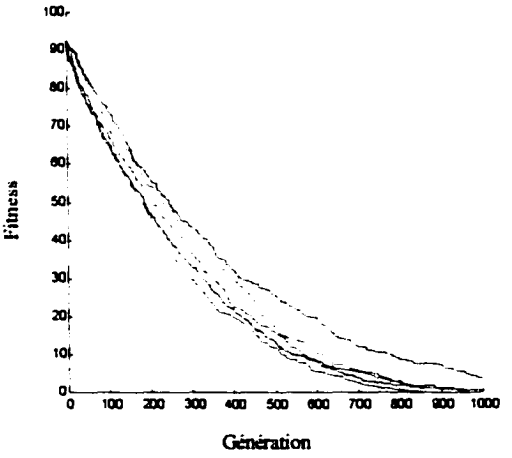
ces probabilités engendre plusieurs cas d'expériences (30 cas). Une dizaine de réplications est effectuée pour chacun de ces cas. Chacun des trois graphiques 3, 4 et 5 regroupent un ensemble d'expériences intitulées : a, b, c, d et e. Chacune des courbes de ces expériences correspondent à un choix des probabilités de croisement et de mutation et chacune d'elles représente les tendances moyennes évaluées sur 10 réplications.

1/ Croisement à un point de coupure

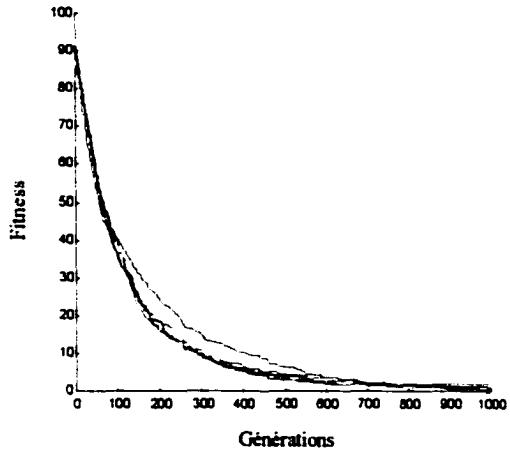
Pour déterminer les meilleures valeurs de probabilité de croisement et de mutation, nous avons effectué les expériences présentées au graphique 3. Ce dernier représente toutes les courbes avec les différentes probabilités de croisement et de mutation. Nous observons que le cas du graphique (3.c) présente une meilleure vitesse de convergence quand la probabilité de mutation p_{mut} est égale à $1/L$ et la probabilité de croisement p_{cross} égale à 0.8. La courbe correspondante converge plus rapidement contrairement aux autres valeurs de probabilité et la solution optimale est atteinte lors de la 800ème génération.



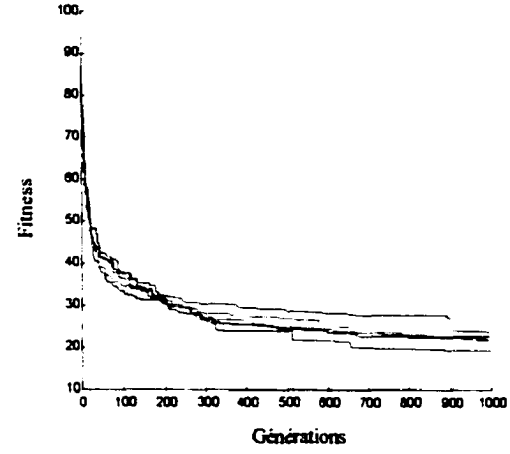
(a) $p_{mut} = 0.0$



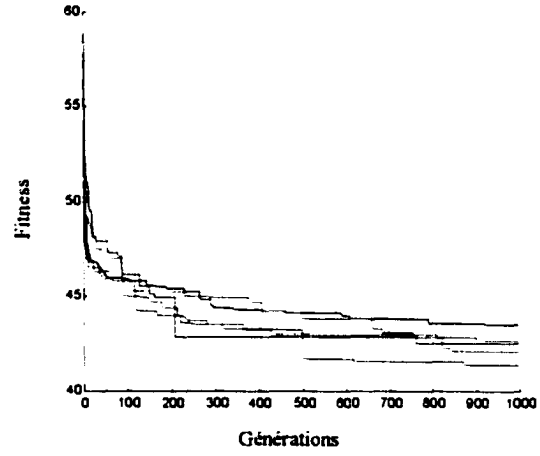
(b) $p_{mut} = 0.1/L$



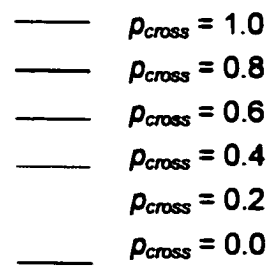
(c) $p_{mut} = 1/L$



(d) $p_{mut} = 10/L$



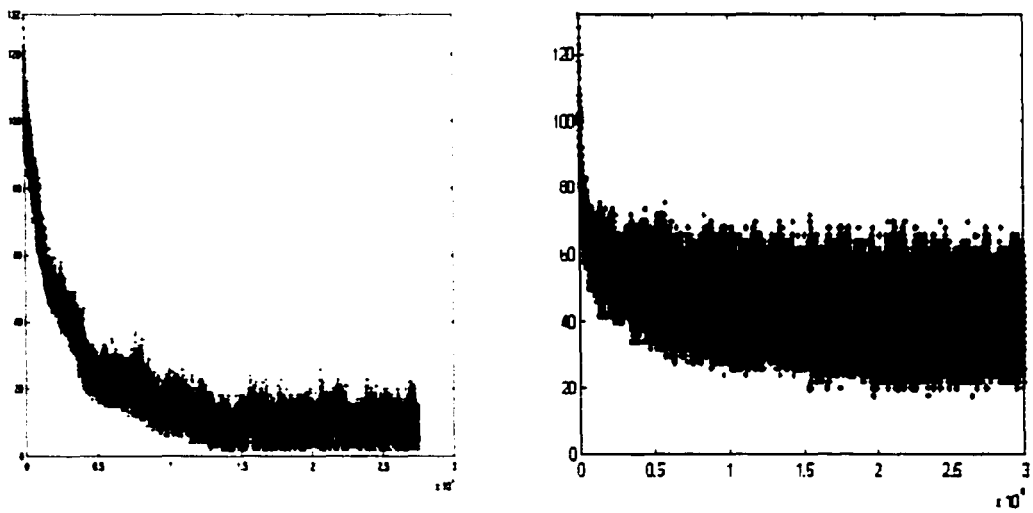
(e) $p_{mut} = 1.0$



Graphique 3 - Courbes représentant l'évaluation moyenne de la fonction *One-Max* dans le cas d'un croisement à un point de coupure

Une constatation peut être faite concernant les graphiques (3.a), (3.d) et (3,e) est que ces courbes n'atteignent pas la solution optimale. Par contre celle ci est atteinte dans le cas des graphiques (3.b) et (3.c) sauf que la convergence est plus lente quand la probabilité de mutation p_{mut} est égale à 0.1/L.

La solution optimale est obtenue lorsqu'une valeur nulle de la fonction d'adaptation associée à un chromosome est atteinte, celle-ci est observée à la figure (21.a) correspondant à la distribution des chromosomes générés par l'AGS. Le nombre maximal généré est de 30 000 chromosomes et l'évaluation de la fonction d'adaptation est effectuée pour chaque chromosome. La figure (21.a) montre un aperçu de la distribution des chromosomes générés par l'AGS dans le cas où p_{cross}=0.8 et p_{mut}=1/*L*. Nous observons également que l'espace de recherche de solutions est bien balayé (couvert) et que la solution optimale est atteinte. A la figure (21.b) nous observons un balayage partiel du domaine de recherche dans le cas où p_{cross}=0 et p_{mut}=10/*L*. La solution optimale obtenue ne correspond pas à la valeur nulle contrairement à ce qui est observé à la figure (21.a). L'axe des abcisses représente les individus générés durant toutes les générations et l'axe des coordonnées représente l'évaluation de la fonction d'adaptation calculé pour chaque individu généré.



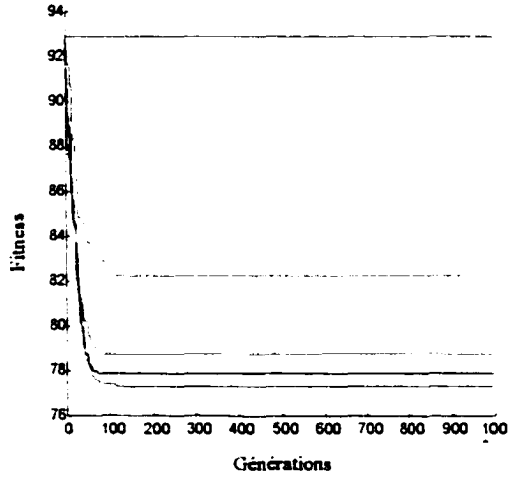
(a) p_{mut} = 1/*L* et p_{cross} = 0.8 (b) p_{mut} = 10/*L* et p_{cross} = 0

Figure 21 - Distribution des individus dans le cas du croisement à un point de coupure

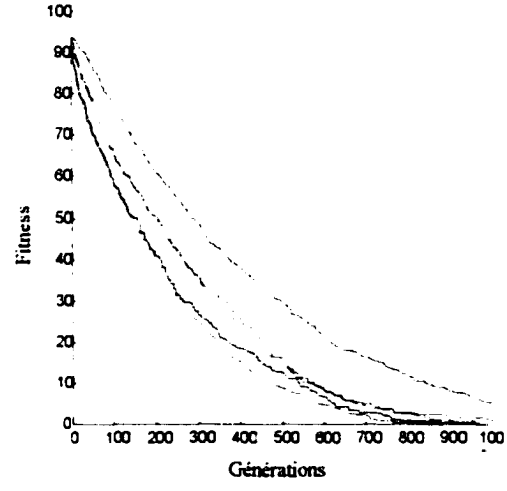
En résumé, les meilleurs paramètres des probabilités utilisant la méthode de croisement à un point de coupure sont p_{cross} = 0.8 et p_{mut} = 1/*L*, ce qui est conforme au choix effectué par Eiben et al [38].

2/ Croisement à deux points de coupure

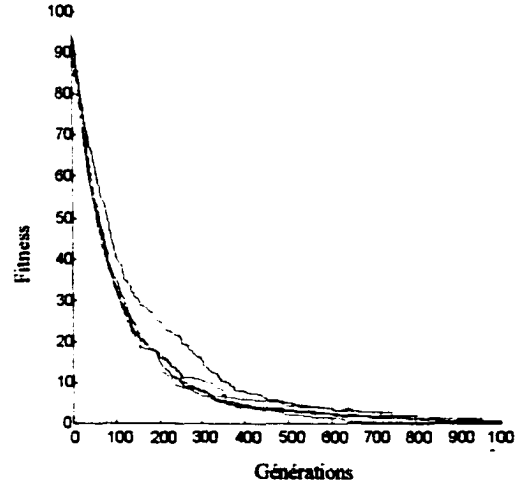
Pour ce qui est de l'approche de croisement à 2 points de coupure, nous observons des résultats presque identiques aux résultats obtenus pour le cas précédent. Néanmoins, une légère différence peut être constaté au niveau de la vitesse de convergence. En effet, la solution optimale est atteinte à la 900ème génération avec les probabilités $p_{mut} = 1/L$ et $p_{cross} = 0.6$. Le graphique 4 représente l'ensemble des courbes de l'évaluation moyenne de 10 réplications.



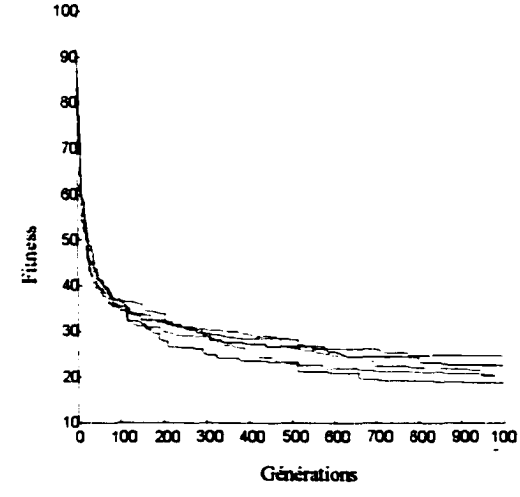
(a) $p_{mut} = 0.0$



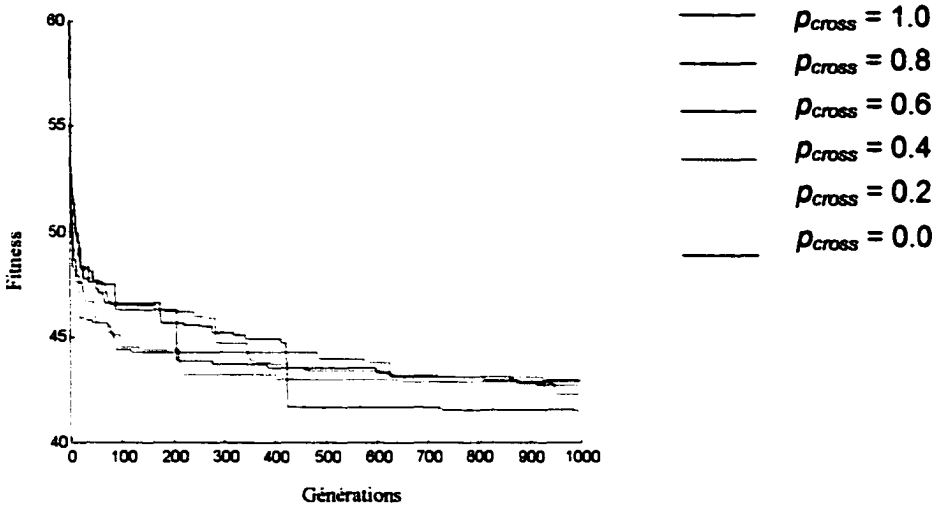
(b) $p_{mut} = 0.1/L$



(c) $p_{mut} = 1/L$



(d) $p_{mut} = 10/L$



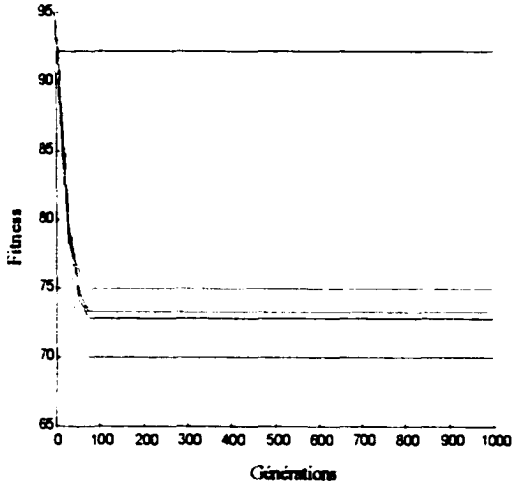
(e) $p_{mut} = 1.0$

Graphique 4 - Courbes représentant l'évaluation moyenne de la fonction *One-Max* dans le cas d'un croisement à deux points de coupures

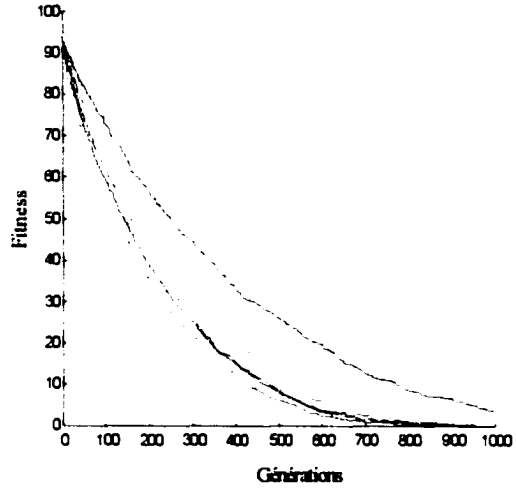
En résumé, les meilleurs paramètres des probabilités utilisant la méthode de croisement à deux points de coupure sont $p_{cross} = 0.6$ et $p_{mut} = 1/L$.

3/ Croisement uniforme

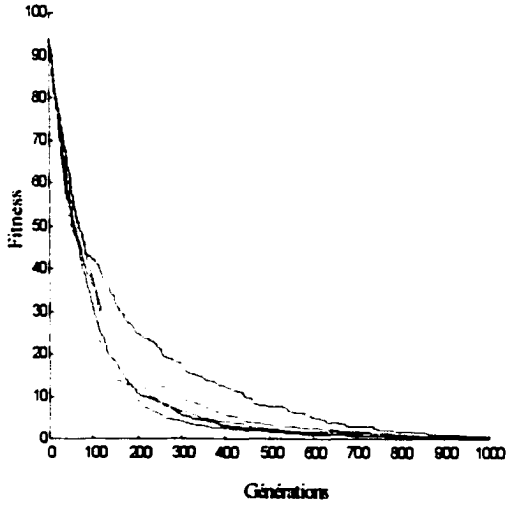
De même, pour ce qui est de l'approche de croisement uniforme, nous observons des résultats identiques aux résultats obtenus dans les deux cas du croisement à un point de coupure et à deux points de coupure. Toutefois, nous observons une légère différence au niveau du choix de la probabilité de croisement qui est dans ce cas égale à 1. La solution optimale est atteinte à la 900ème génération avec les probabilités de mutation $p_{mut} = 1/L$ et de croisement $p_{cross} = 1$ (voir graphique (5.c)). Le graphique 5 illustre l'ensemble des courbes de l'évaluation moyenne de 10 réplications.



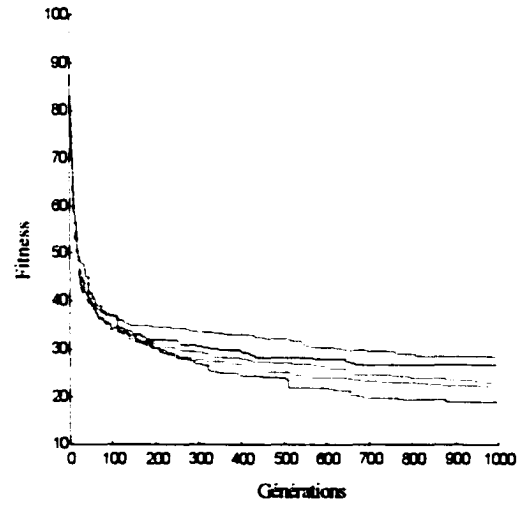
(a) $p_{mut} = 0.0$



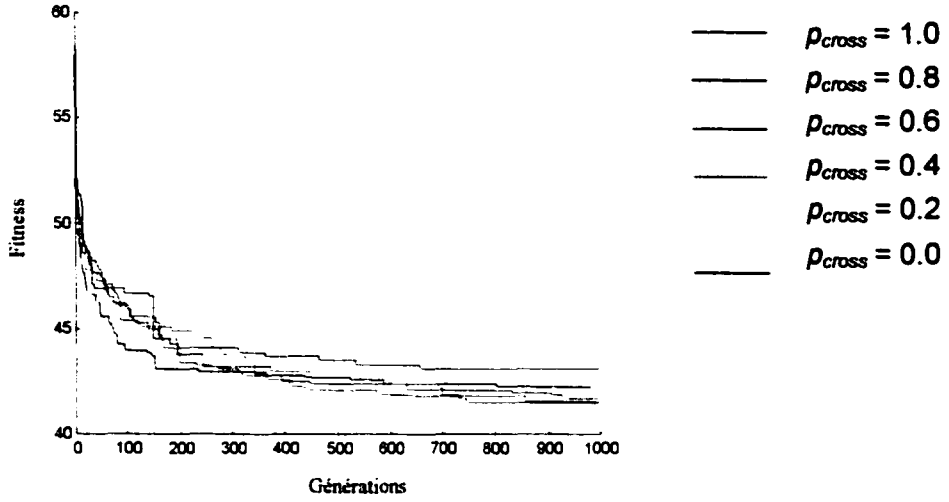
(b) $p_{mut} = 0.1/L$



(c) $p_{mut} = 1/L$



(d) $p_{mut} = 10/L$



(e) p_{mut} = 1.0

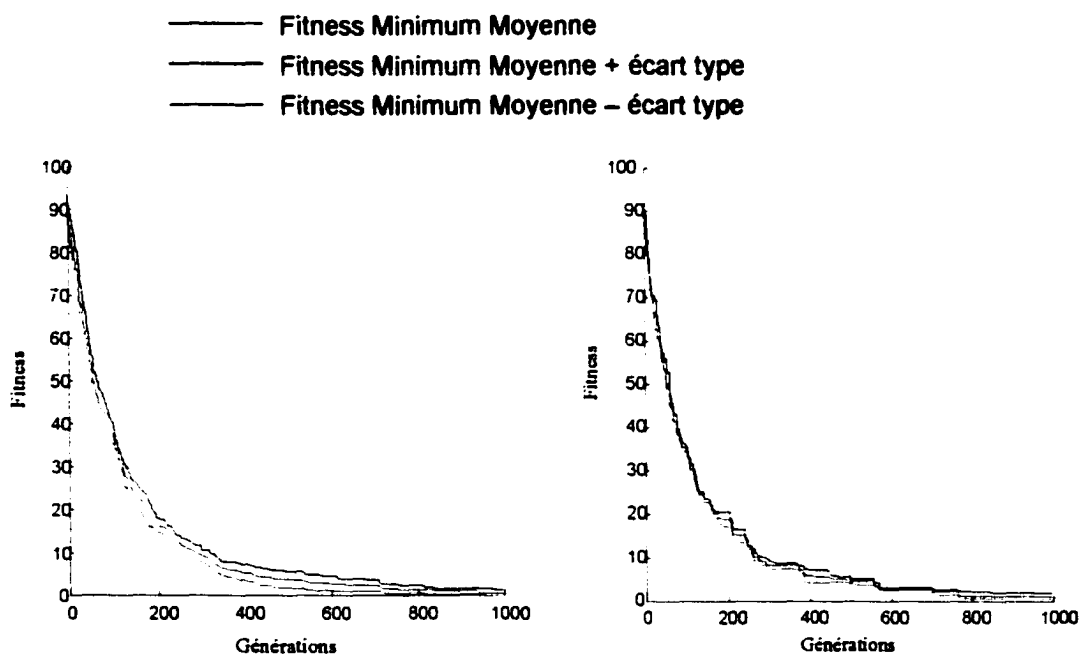
Graphique 5 - Courbes représentant l'évaluation moyenne de la fonction *One-Max* dans le cas d'un croisement uniforme

En résumé, les meilleurs paramètres de probabilités utilisant la méthode de croisement uniforme sont p_{cross} = 1 et p_{mut} = 1/*L*.

Enfin, en conclusion du choix des probabilités de croisement, de mutation et des méthodes appropriées de reproduction, nous constatons que la technique de croisement à un point de coupure semble la plus adaptée au problème *One-Max*. Les probabilités de croisement et de mutation sont égales respectivement à 0.8 et 1/*L*, du fait que la vitesse de convergence pour obtenir la solution optimale est la plus rapide.

Et toujours dans le même contexte du problème *One-Max*, nous avons effectué des expériences pour observer l'influence de la taille de la population. Pour se faire, nous avons utilisé les paramètres prédéfinis initialement (voir tableau V) et les résultats des expériences précédentes à savoir : le croisement à un point de coupure avec les probabilités p_{cross} = 0.8 et p_{mut} = 1/*L*. Nos résultats étant basés sur une population de seulement 32 individus, nous avons effectué une dizaine de réplications supplémentaires avec une population de plus grande taille, c'est-à-dire une population

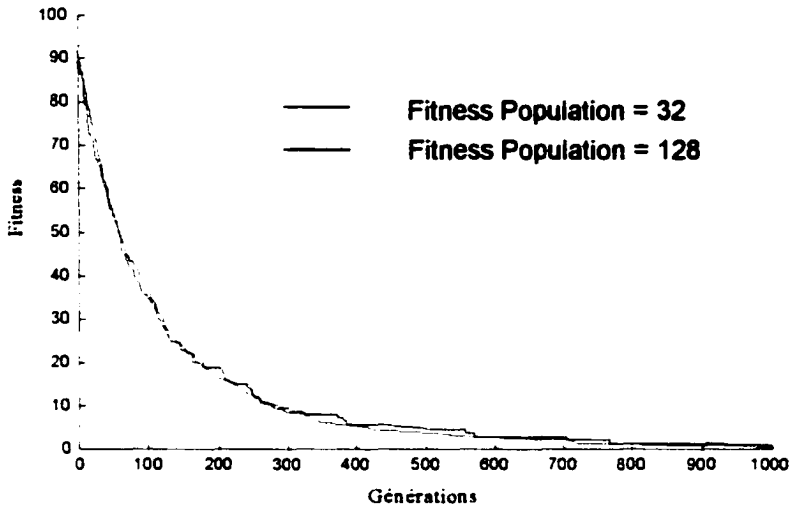
de 128 individus. Le graphique 6 représente l'évaluation moyenne de 10 réplications de la fonction *One-Max* sur une population élargie. Dans le graphique 6, nous observons qu'il n'existe pas de différence notable entre les résultats obtenus précédemment et ceux obtenus avec les 128 individus. La différence se trouve alors au niveau de l'écart type représenté par les graphiques (6.a) et (6.b). Ces deux derniers illustrent respectivement une population de 32 individus et de 128 individus. Le calcul de l'écart type est effectué de la manière suivante : pour chaque génération, les chromosomes sont évalués par la fonction d'adaptation *One-Max* et la moyenne de cette fonction est calculée pour l'ensemble des chromosomes de cette génération et enfin à partir de cette moyenne, nous calculons l'écart type pour visualiser la dispersion des individus dans une génération. Alors, nous pouvons constater que lorsque la taille de la population est grande, la dispersion des individus de celle-ci est petite.



(a) Population = 32 (b) Population = 128

Graphique 6- Courbes représentant l'évaluation moyenne de la fonction *One-Max* dans le cas de deux populations de taille différente

Concernant la vitesse de convergence des deux populations, nous avons observé qu'elle est presque identique car la solution optimale est atteinte à la 800ème génération (voir graphique 7).



Graphique 7- Comparaison de la convergence de l'évaluation moyenne de la fonction *One-Max* de deux populations de taille différente

Donc, il est préférable d'utiliser une population de taille égale à 32 individus au lieu d'une population de 128 individus car le temps de calcul pour une population de grande taille est plus lent qu'une population de petite taille.

En conclusion, cette partie du choix des paramètres nous a permis de fixer certaines valeurs des paramètres. Ces derniers concernent le choix de la méthode de croisement, les valeurs des probabilités des opérateurs de reproduction et l'influence de la taille de la population. Ces paramètres en combinaison avec les paramètres prédéfinis initialement (voir tableau V) seront utilisés pour effectuer la sélection des primitives dans le cadre de notre projet.

5.3 Sélection des primitives

Notre objectif est la sélection des primitives en utilisant une technique d'optimisation connue sous le nom d'AG. Cette méthode doit permettre de réduire le nombre de primitives L en M primitives telle que $M<=L$. La méthodologie adoptée est décrite à la section 4.8. Nous avons effectué deux sortes d'expériences pour la sélection de primitives : la sélection des primitives par les AGS et la sélection des primitives par les AGI. Dans ces deux cas, les paramètres définis ci-dessous sont utilisés :

Tableau VI

Paramètres de l'AGS pour la sélection des primitives

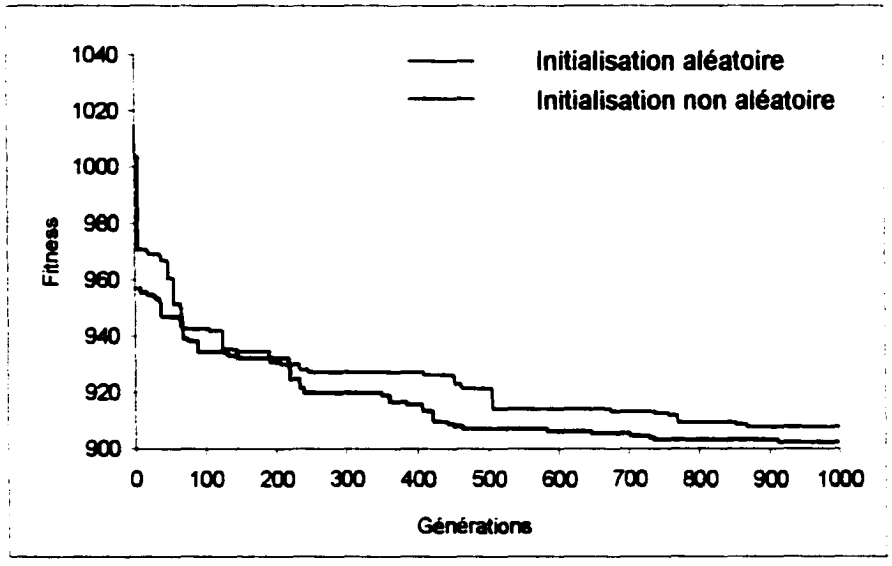
- Codage	: Binaire
- Sélection	: Roulette Wheel
- Élitisme	: Vrai
- Taille population	: 32
- Nombre générations	: 1000
- Fonction d'adaptation	: *fitness* =*Minimiser*($\alpha f_1 + \beta f_2$) (voir formule (4.2)).
- Taille du chromosome	: L = 132.
- Nombre de réplications	: 10.
- Méthode de mutation	: Aléatoire.
- Probabilité de mutation	: $p_{mut} = 1/L$.
- Méthode de croisement	: Un point de coupure.
- Probabilité de croisement	: $p_{cross} = 0.8$.
- Critère d'arrêt	: Nombre de générations maximale et/ou la performance maximale sur la base de validation.

5.3.1 Algorithme génétique simple (AGS)

Avant de se lancer dans plusieurs réplications, nous avons voulu connaître l'impact de l'initialisation de la population *P* sur le processus de sélection de primitives. Lors de cette initialisation, nous suggérons deux cas possibles :

- *initialisation aléatoire* qui signifie que les valeurs des gènes c_i des individus *C* de la population *P* sont générées aléatoirement. Dans le codage binaire, des bits de valeur égale à 0 et à 1 sont produits pour constituer les gènes c_i des chromosomes *C*.
- *initialisation non aléatoire* qui consiste à injecter dans la population initiale *P* une ou plusieurs bonnes solutions. Cette injection permet à l'AG de converger plus rapidement vers des solutions optimales.

Étant donné que le choix de l'initialisation de la population *P* est un paramètre important, nous avons effectué une expérience pour chacun des cas en utilisant les paramètres du tableau VI. La solution que nous avons injecté représente le chromosome dont la valeur des gènes est égale à 1.



Graphique 8 - Comparaison entre l'initialisation aléatoire et l'initialisation non aléatoire

Le graphique 8 illustre les résultats obtenus pour une seule réplication où nous observons que la vitesse de convergence lors de l'initialisation non aléatoire est plus rapide et que la solution obtenue est meilleure.

Pour toutes les expériences réalisées ultérieurement, nous avons injecté la meilleure solution *a priori* dans la première population de l'AGS. Cette solution est un chromosome *C* dont les valeurs des gènes c_i sont égales à 1 (toutes les primitives sont alors sélectionnées initialement).L'optimisation du système de reconnaissance par les AGS s'effectue sur la base de données B. A chaque 10 générations de l'AGS, nous calculons le taux d'erreur sur la base C (base de validation) qui servira comme critère d'arrêt de l'AGS. L'évaluation de la performance en généralisation est calculée sur la base D à la fin du processus de sélection. Pour plus de détails, la méthodologie adoptée est décrite à la section 4.8.

Le tableau suivant illustre les résultats obtenus des 10 réplications :

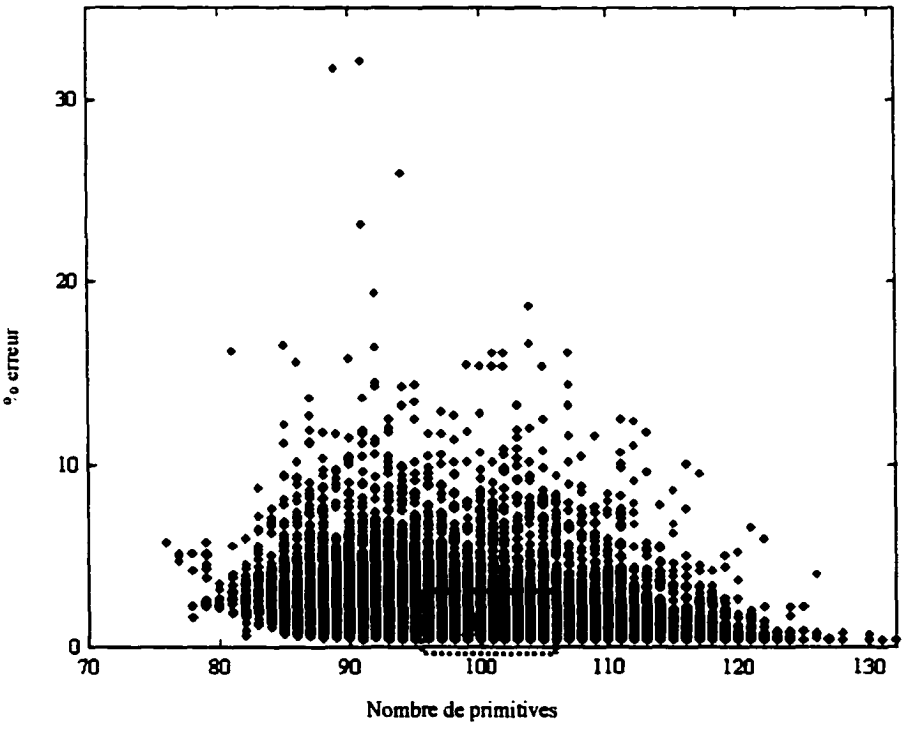
Tableau VII

Résultats de l'AGS

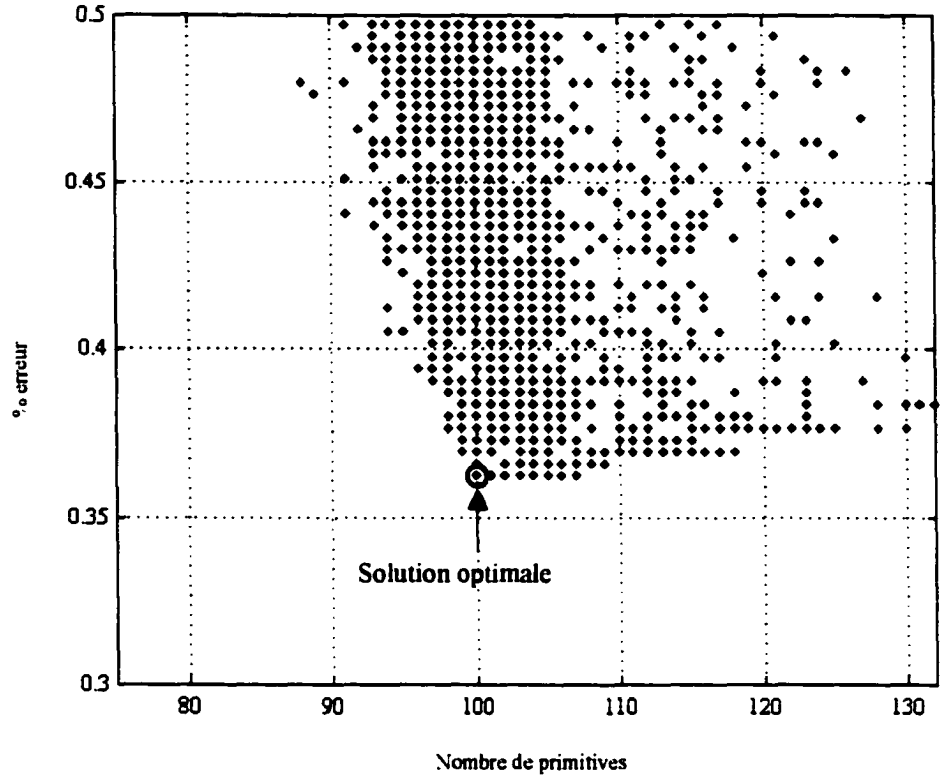
		Taux de reconnaissance (%)	
# Réplication	**# Primitives**	**Base B**	**Base C**
Initialement	132	99.62	99.15
1	100	99.63	99.02
2	100	99.63	99.02
3	90	99.63	98.96
4	100	99.63	99.02
5	96	99.64	99.04
6	100	99.63	99.02
7	100	99.63	99.02
8	94	99.63	99.01
9	100	99.64	99.05
10	98	99.63	99.07
Moyenne	97.8	99.63	99.02
Ecart-type	11.96	1.78E-5	8.23E-4

Nous remarquons que 6 réplications parmi les 10 ont un nombre de primitives sélectionnées de 100. En analysant les positions de chacune de ces 6 réplications,

nous avons constaté que 5 solutions sont identiques. Étant donné que la moyenne calculée est de sélectionner 98 primitives, il est impossible de connaître les positions des primitives supprimées. Il est donc préférable de sélectionner les 100 primitives. De plus, nous observons que la dispersion calculée sur le nombre de primitives est importante ce qui signifie que l'espace de recherche a bien été exploré. Cette exploration est illustrée par les figures (22.a) et (22.b) où la distribution des individus générés par l'AGS est observée concernant la réplication 9. Pour faciliter une analyse visuelle de l'espace des solutions, la figure (22.b) montre un agrandissement du cadre A où la solution optimale est atteinte. La valeur de l'écart type des taux de reconnaissance calculés sur la base B étant très faible, cependant rien n'empêche que l'espace de recherche a bien été balayé (voir figure 22).



(a)

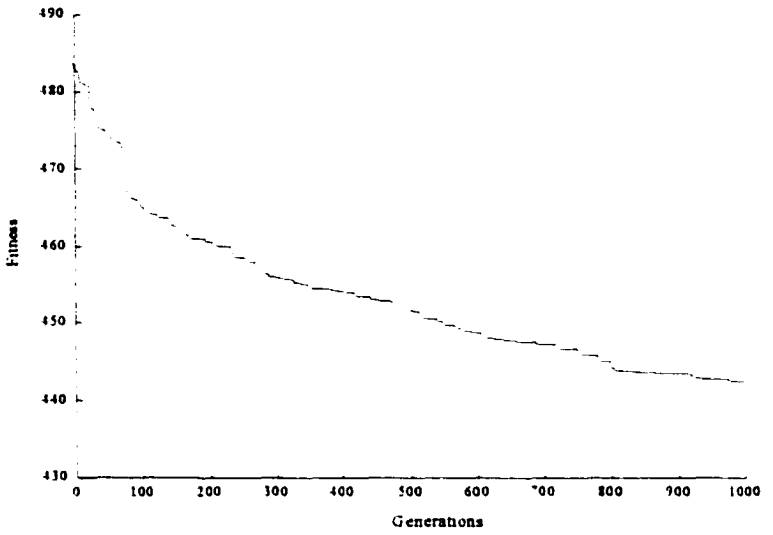


(b)

Figure 22 - Distribution des individus dans l'espace de recherche

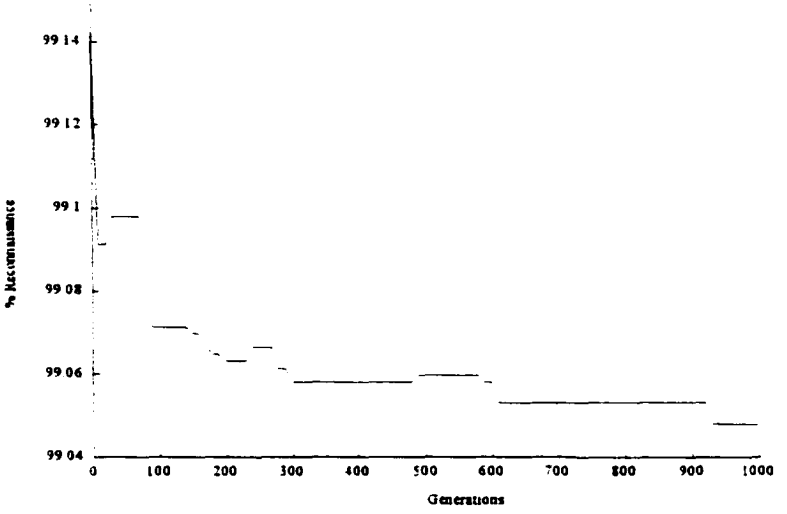
Sachant que l'optimisation du système de reconnaissance est calculée sur la base B, la validation sur la base C et la généralisation sur la base D (voir section 4.8), nous avons représenté l'évolution de la fonction d'adaptation sur la base B par le graphique 9 et l'évolution de la performance de ce système par le graphique 10.

Nous observons que l'évolution de la fonction d'adaptation diminue au fur et à mesure que les générations progressent pour arriver à une forme de plateau où la fonction d'adaptation n'évolue plus. Dans ce graphique 9, nous n'observons pas de plateau au cours des 1000 premières générations mais après 1500 générations on voit en apparaître un. Nous pouvons augmenter le nombre de générations pour visualiser ce plateau pour l'ensemble des expériences mais le temps de calcul est déjà assez lent pour 1000 générations.



Graphique 9 - Évolution de la fonction d'adaptation de la base B

Le graphique 10 représente l'évolution de la performance du système de reconnaissance de la réplication 9 sur la base de validation C. Cette évolution est calculée à chaque 10 générations lors du déroulement de l'AGS pour la sélection des primitives du système de reconnaissance des chiffres isolés. Nous observons que le taux de reconnaissance diminue légèrement de 0.1% passant de 0.38% à 0.37%.



Graphique 10 - Évolution de la performance du système sur la base de validation C

Ainsi, le système mis en place a réduit le nombre de primitives, les faisant passer de 132 primitives à, en moyenne, 98 primitives et la performance du système de reconnaissance a légèrement été améliorée (voir tableau VII). Par exemple, les positions des primitives non pertinentes et redondantes à éliminer du réseau observée à la réplication 9 sont :

0, 7, 10, 17, 20, 40, 41, 42, 47, 53, 64, 69, 73, 79, 80, 82, 84, 90, 91, 97, 112, 113, 114, 115, 122, 123, 124, 125, 126, 128, 129, 130.

L'interprétation de ces primitives supprimées sera détaillée au chapitre 6.

Les performances en taux de reconnaissance sur les bases de données sont assez satisfaisantes. En généralisation, le taux de reconnaissance obtenu sur la base de test D est égal à 97.28%, soit une légère dégradation de 0.19% en taux de reconnaissance est observée (voir tableau VIII).

Nous avons donc relancé l'apprentissage du réseau avec la meilleure solution, à savoir les 100 primitives obtenues lors de la réplication 9. La taille de la couche d'entrée du réseau de neurones est diminué de 32 primitives passant de 132 neurones à 100. L'entraînement du réseau se fait sur la base A et la validation sur la base B tel que décrit par la méthodologie présentée à la section 4.8. Cet apprentissage a mis moins de temps pour générer la matrice poids en utilisant les 100 primitives sélectionnées par rapport aux 132 primitives originales (voir tableau IX). Les bases de données C et D vont servir de bases de test pour l'évaluation finale du réseau. Les résultats obtenus sont illustrés dans le tableau suivant :

Tableau VIII

Résultats de la sélection des primitives par l'AGS

	Taux de reconnaissance (%)		
	(1)	**(2)**	**(3)**
Base A	99.65	99.65	99.64
Base B	99.62	99.64	99.61
Base C	99.15	99.05	99.16
Base D	97.47	97.28	97.54

(1) : Taux de reconnaissance avec 132 primitives, aucune sélection n'est effectuée.

(2) : Taux de reconnaissance en sélectionnant 100 primitives. Les primitives supprimées sont remplacées par la valeur moyenne d'après l'analyse de sensibilité vue à la section 4.5.1.

(3) : Taux de reconnaissance avec seulement 100 primitives obtenu après l'apprentissage du réseau de neurones.

Les résultats observés dans le tableau VIII sont presque équivalents. Nous pouvons constater que l'analyse de faisabilité a été d'une grande utilité. Le fait de ne pas relancer l'apprentissage du réseau nous a permis de réduire considérablement le temps de calcul requis pour l'optimisation du réseau. A titre indicatif, le tableau IX fournit une estimation du temps de calcul sur les différents ordinateurs utilisés. Le tableau X décrit sommairement les caractéristiques de ces ordinateurs :

Tableau IX

Estimation du temps de calcul

	Nombre d'heures de calcul		
	(1)	**(2)**	**(3)**
Apprentissage avec 132 primitives.	120	-	4h sur un ordinateur
AG simple.	360	120	8h sur 8 ordinateurs en cluster
Apprentissage avec 100 primitives.	-	120	3,5h sur un ordinateur

Tableau X

Caractéristiques des ordinateurs

	(1)	**(2)**	**(3)**
Type de machine	Sun Ultra1	Sun Blade 100	PC
CPU	Ultra Sparc IIe 143 MHz	Ultra Sparc 500 MHz	1000 MHz
Mémoire	128	256	256

Nous constatons que la configuration de l'ordinateur (2) est 3 fois plus rapide que (1) et la machine (3) est 45 fois plus rapide que (1). Le dernier ordinateur a été acquis dans le cadre du projet des AG parallèles. Initialement, toutes nos expériences ont été effectuées au sein du laboratoire LIVIA où les ordinateurs de type (1) sont disponibles pour ce type d'expérience. Le laboratoire LCA a acquis des ordinateurs de type (2) et le reste des expériences s'y est déroulé.

En conclusion de cette section, le nombre de primitives sélectionnées par les AGS a diminué de 25% passant de 132 primitives à 100 ($M = 100 < L = 132$). Cette sélection des caractéristiques pertinentes a permis d'augmenter légèrement la performance en taux de reconnaissance et de diminuer le temps de calcul requis pour l'apprentissage du réseau. Ainsi, cette optimisation a réduit d'une part la complexité du réseau de neurones et d'autre part le temps nécessaire pour l'extraction des primitives.

5.3.2 Algorithme génétique itératif (AGI)

Toujours dans le cadre de la sélection des primitives, nous avons appliqué les AGI pour la sélection des primitives dans la reconnaissance des chiffres isolés. Leur principe est identique à celui des AGS. Pour se faire, nous avons mis en place un AGI afin de comparer sa vitesse de convergence à celle des AGS. Cependant la méthodologie présentée à la section 4.8 n'a pas été appliquée sur les bases de données B et C mais plutôt sur les bases C et D car au début de ce projet la fonction d'adaptation était calculée sur la base de données C. Les paramètres prédéfinis précédemment pour la sélection des primitives par les AGS (voir tableau VI) sont identiques à ceux appliqués par les AGI sauf que ces derniers nécessitent l'ajustement de paramètres supplémentaires tels que :

Tableau XI

Paramètres de l'AGI pour la sélection de primitives

- Distance de Hamming	: $\psi = 5$.
- Nombre d'itérations	: 10.
- Nombre de générations	: 100 par itération.
- Nombre de réplications	: 1.

La distance de Hamming permet de balayer un sous-espace dans le domaine de recherche des solutions. Lors de l'initialisation de la population *P*, la distance calculée entre chaque chromosome *C* doit être inférieure d'un seuil ψ que l'expérimentateur doit fixer. Dans le cadre de notre projet, ce seuil est fixé à 5. A chaque itération, un sous espace de recherche A est balayé. La taille de cet espace est réduite au fur et à mesure que l'algorithme progresse. A titre d'exemple, la figure 23 illustre l'itération 1 de l'AGI pour laquelle la solution obtenue est de 125 primitives.

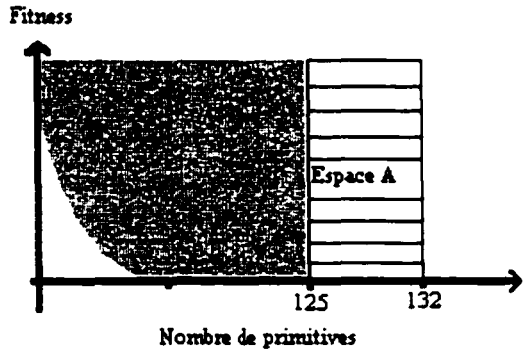
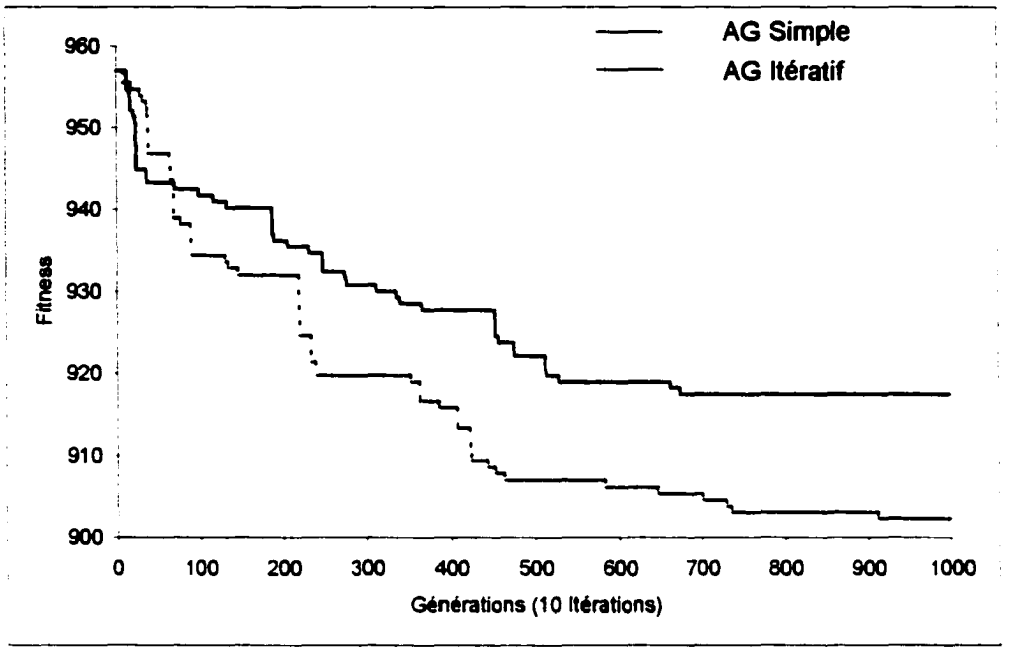


Figure 23 - Balayage du sous-espace à l'itération 1 de l'AGI

Les résultats obtenus par les AGI ne sont pas satisfaisants si nous les comparons à ceux de l'AGS (voir graphique 11).



Graphique 11 - Comparaison de l'évolution de la fonction d'adaptation entre AGS et AGI

Nous observons que la vitesse de convergence des AGI est plus lente que celle des AGS. Effectivement, la solution optimale pour la sélection des primitives par les AGI est obtenue au bout de la 7ème itération. Cette solution représente la sélection de 104 primitives au lieu de 132 primitives initiales. Dans le cas de la sélection des primitives par les AGS, nous avons obtenu 95 primitives. Cette dernière diffère de la solution trouvée à la section 5.3.1 car l'optimisation a été effectuée sur une autre base que celle utilisée par la méthodologie décrite à la section 4.8. La différence entre ces deux solutions pour les AGS se situe au niveau du choix des primitives à supprimer. Il existe quelques primitives communes à supprimer. Le tableau XII présente les résultats obtenus par les deux approches :

Tableau XII

Résultats de la sélection des primitives par les deux approches

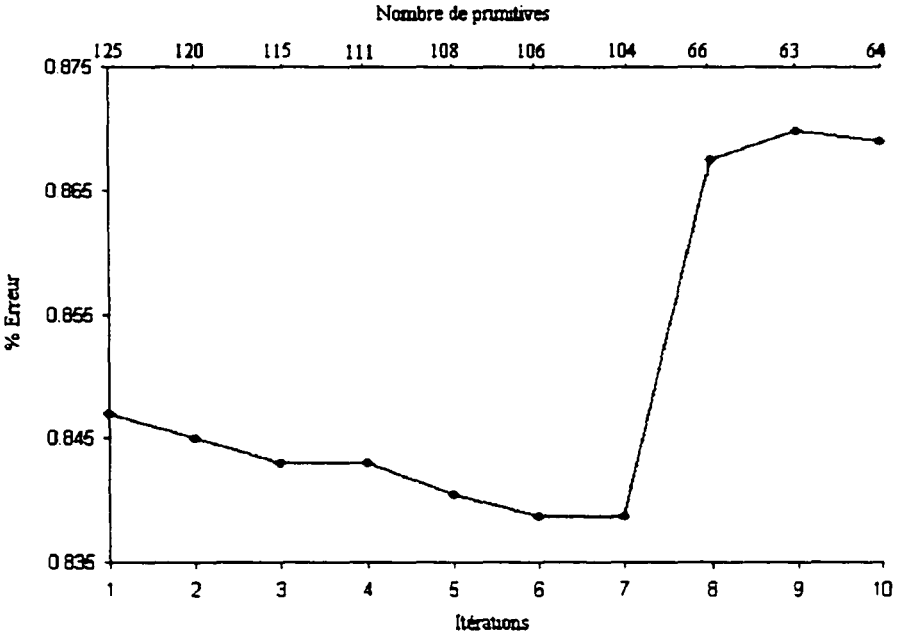
	Primitives originales		AG simple		AG itératif	
Base	**Primitives**	**% Erreur**	**Primitives**	**% Erreur**	**Primitives**	**% Erreur**
Base A	132	0.35	95	0.34	104	0.34
Base B	132	0.38	-	-	-	-
Base C	132	0.87	95	0.83	104	0.84
Base D	132	2.48	95	2.60	104	2.58

Nous avons illustré par le graphique 12 l'évolution de la performance de la fonction d'adaptation calculée par les AGI sur les bases de données C et D. Le graphique 12 met en évidence un sur-apprentissage à partir de la $7^{ème}$ itération. Au delà de la $7^{ème}$ itération, nous observons une chute du taux d'erreur considérable (environ 35%) sur la base D (voir graphique (12.b)). Ce taux d'erreur correspond à la sélection de 64 primitives à la $10^{ème}$ itération. Le fait de rencontrer ce sur-apprentissage nous amène à conclure que la solution optimale a été obtenue à la $7^{ème}$ itération. En effet, pour éviter ce problème du sur-apprentissage, la méthodologie décrite à la section 4.8 a été adoptée à notre système de reconnaissance des chiffres isolés tels que la base B est utilisée pour l'optimisation du réseau, la base C pour la validation et la base D pour la généralisation.

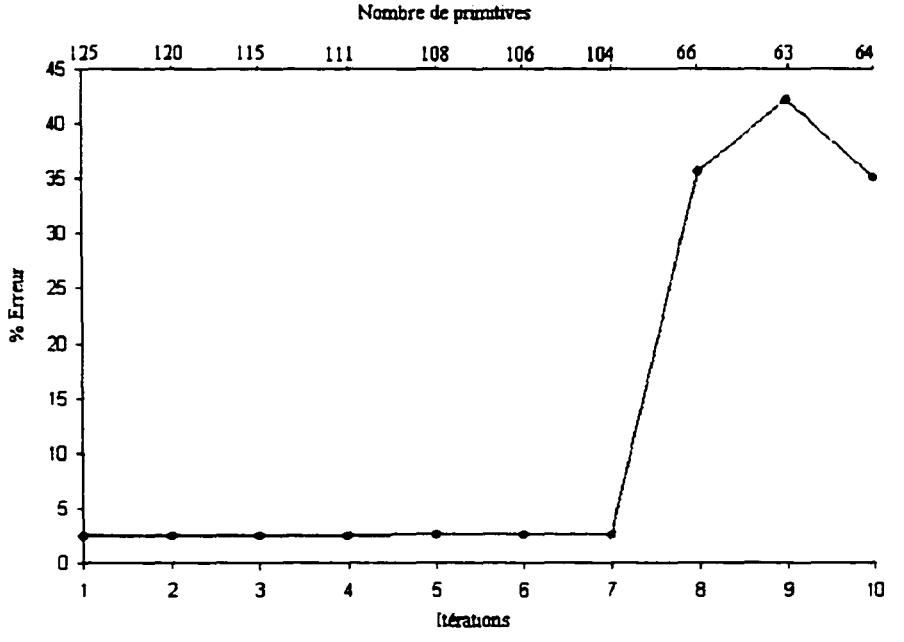
La solution optimale obtenue par les AGI correspond donc à 104 primitives sélectionnées dont les positions des primitives supprimées sont :

10, 11, 18, 19, 20, 35, 38, 39, 41, 42, 73, 79, 80, 85, 86, 90, 91, 93, 94, 97, 113, 117, 124, 125, 126, 128, 129, 130.

L'interprétation de ces primitives supprimées sera détaillée au chapitre 6.



(a) Base C



(b) Base D

Graphique 12 - Évolution de la performance dans le cas de l'AGI

5.4 Pondération des primitives

Le second objectif de notre étude est de pondérer et de classer les primitives d'après leur importance. Cette dernière consiste à déterminer les primitives les plus discriminantes en leur attribuant un poids. Nous avons utilisé les AGS pour résoudre le problème de reconnaissance des chiffres isolés en adaptant la méthodologie décrite à la section 4.8. L'optimisation du système de reconnaissance s'effectue sur la base de données B. A chaque 10 générations, nous calculons la performance du système sur la base C qui sert de validation. L'évaluation de la performance en généralisation est calculée sur la base D à la fin du processus de pondération. Pour toutes les expériences réalisées dans le cadre de la pondération des primitives, nous avons injecté la meilleure solution *a priori* dans la première population de l'AGS. Cette solution est un chromosome C dont les valeurs de gènes c_i sont toutes égales à 1. Certains paramètres sont fixés *a priori* et d'autres doivent être choisis expérimentalement. Les paramètres génétiques prédéfinis sont présentés dans le tableau XIII :

Tableau XIII

Paramètres de l'AGS pour la pondération des primitives

- Codage	: Réel sur l'intervalle [0,1]
- Sélection	: Roulette Wheel
- Élitisme	: Vrai
- Taille population	: 32
- Nombre de générations	: 1000
- Fonction d'adaptation	: $fitness_{pond.} = Maximiser(f_3)$ (voir formule (4.7))
- Taille du chromosome	: $L = 132$
- Probabilité de mutation	: $p_{mut} = 10/L$
- Probabilité de croisement	: $p_{cross} = 0.8$
- Critère d'arrêt	: Nombre de générations maximale et/ou la performance maximale sur la base de validation.

Le choix des méthodes de reproduction (mutation et croisement) n'est pas une tâche facile dans le cas de la pondération des primitives. En effet, nous ne pouvons pas faire des simulations sur des fonctions standards comme la fonction *One-Max* utilisée pour la sélection des primitives car *a priori* il n'existe pas de fonction qui s'apparente à la fonction de notre problème de reconnaissance. Il faudra donc utiliser la fonction d'adaptation du problème à résoudre définie par la formule (4.7). Toutefois, nous nous sommes limités à effectuer une seule réplication pour chacune des méthodes de reproduction utilisées à cause des temps de calcul qui sont prohibitifs. Les meilleurs résultats obtenus avec les différentes méthodes utilisées (voir tableau XIV) vont nous permettre de relancer 10 réplications avec les méthodes de croisement et de mutation appropriées. Dans la littérature, Herrera et al [27] ont mentionné que les deux opérateurs de croisement BLX-α et arithmétique, ainsi que l'opérateur de mutation non uniforme donnent de meilleurs résultats par rapports aux autres opérateurs. Durant l'exécution de l'algorithme, nous avons observé que les valeurs des gènes c_i des individus en utilisant l'opérateur BLX-α (présenté à la section 3.6.1.5) devienent de plus en plus grandes, d'où l'intervalle de recherche s'agrandit au fur et à mesure des nouvelles populations. Dès lors, il est nécessaire d'effectuer une normalisation au niveau de chaque valeur du gène c_i de l'enfant C généré par l'opérateur de croisement. Cette normalisation consiste à transformer l'intervalle $[c_{min} - I\alpha, c_{max} + I\alpha]$ en un intervalle $[a_i, b_i]$ à l'aide d'une normalisation linéaire simple. L'opérateur de normalisation est défini par

$$\begin{aligned} [c_{min} - I\alpha\,,\, c_{max} + I\alpha] \quad &\Rightarrow \quad [a_i\,,\, b_i] \\ x \quad &\rightarrow \quad y = \omega x + \varepsilon \end{aligned} \tag{5.1}$$

après transformation linéaire de cette formule (5.1), nous obtenons la formule suivante

$$y = \left(\frac{a_i - b_i}{(c_{min} - I\alpha) - (c_{max} + I\alpha)}\right)x + \left(\frac{(c_{min} - I\alpha)b_i - (c_{max} + I\alpha)a_i}{(c_{min} - I\alpha) - (c_{max} + I\alpha)}\right) \tag{5.2}$$

Dans notre cas $a_i = 0$ et $b_i = 1$ alors,

$$y = \left(\frac{1}{(c_{max} + l\alpha) - (c_{min} - l\alpha)}\right)(x - (c_{min} - l\alpha)) \qquad (5.3)$$

La valeur y définie par la formule (5.3) est calculée pour chaque gène c_i du chromosome C. La valeur du paramètre α est fixée à 0.5 telle que proposée par Herrera et al [27]. Concernant l'opérateur de croisement arithmétique défini par la fonction (3.8) (présenté à la section 3.6.1.4), nous observons que l'espace de recherche s'accroît progressivement en fonction des valeurs de λ et celles des gènes c_i des parents. Ceci dit, une normalisation est donc nécessaire du fait que les bornes de l'espace de recherche s'élargissent; la démarche à suivre est identique à la normalisation précédente. Dans le cas du croisement arithmétique uniforme, nous avons effectué des simulations avec les différentes valeurs de λ à savoir : 0.25, 0.50 et 0.75. Contrairement au croisement arithmétique non uniforme, la valeur de λ_i sera générée aléatoirement pour chaque valeur du gène c_i.

L'opérateur de mutation que nous avons utilisé est l'opérateur non uniforme décrit à la section 3.6.2.2 avec une valeur de b = 5. La valeur de la probabilité de mutation est égale à $10/L$ au lieu de $1/L$ lors de la sélection des primitives. Le choix de cette probabilité élevée permet à plusieurs gènes c_i du chromosome C de muter et permet également une bonne exploitation de l'espace de recherche. Rappelons qu'il est nécessaire de bien explorer le domaine des solutions pour ensuite passer à la phase d'exploitation. Les différentes expériences effectuées sont décrites au tableau XIV et les résultats obtenus de chacune de ces expériences sont représentés au tableau XV.

Tableau XIV

Liste des expériences

# Expérience	Croisement	Mutation
1	BLX-α avec $\alpha = 0.5$	Non uniforme avec $b = 5$
2	---	Non uniforme avec $b = 5$
3	Arithmétique avec $\lambda = 0.25$	---
4	Arithmétique avec $\lambda = 0.5$	---
5	Arithmétique avec $\lambda = 0.75$	---
6	Arithmétique avec λ_i aléatoire	---
7	Arithmétique avec λ_i aléatoire	Non uniforme avec $b = 5$

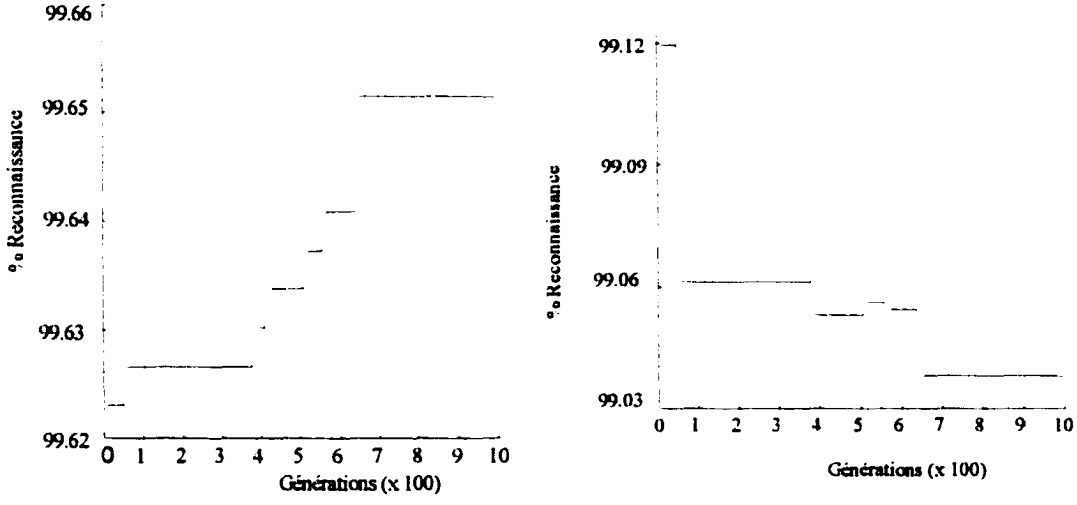
Tableau XV

Résultats des expériences

	Taux de reconnaissance (%)			
# Expérience	**Base A**	**Base B**	**Base C**	**Base D**
Tous les poids sont égaux à 1	99.65	99.62	99.15	97.47
1	99.65	99.62	99.14	97.45
2	99.58	99.65	99.04	97.28
3	99.63	99.62	99.14	97.43
4	99.64	99.62	99.16	97.46
5	99.63	99.62	99.13	97.43
6	99.64	99.63	99.16	97.46
7	99.64	99.63	99.04	97.28

Expérience 1 : La combinaison des deux opérateurs BLX-α et la mutation non uniforme n'améliorent pas la performance du système. Les résultats obtenus sont un ensemble de poids unitaires (tout égaux à 1) pour chacune des primitives.

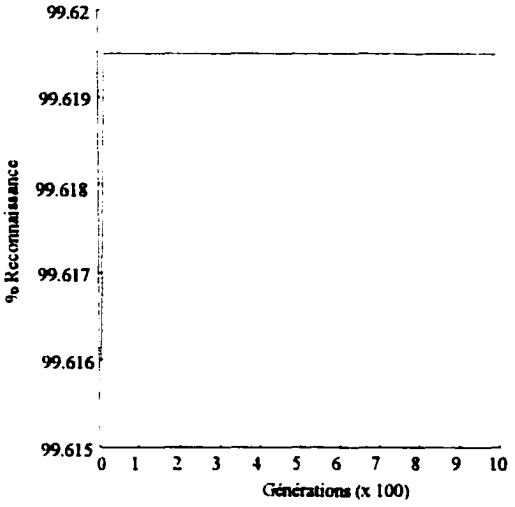
Expérience 2 : En appliquant seulement l'opérateur de mutation sans l'utilisation d'un opérateur de croisement, nous remarquons une amélioration de la performance du système sur la base B. Par contre lorsqu'une évaluation est faite sur la base C lors de la validation et sur la base D, des légères baisses de 0.11% et de 0.13% des taux de reconnaissance sont observées. Cette expérience a montré que les phases d'exploitation et d'exploration sont à égale importance. Le graphique 13 illustre les courbes obtenues lors de l'évaluation de la performance sur la base B et sur la base C.



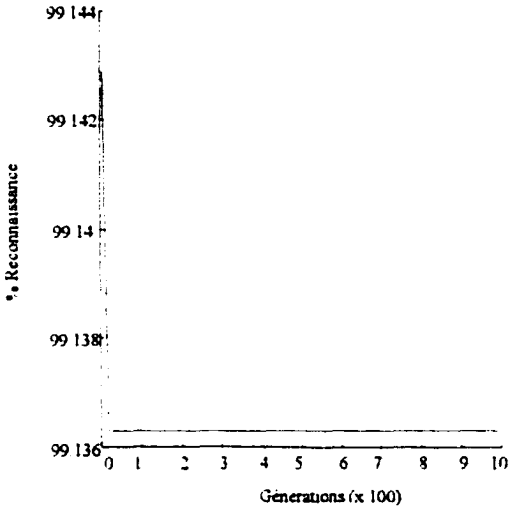
(a) Base B (b) Base C

Graphique 13 - Performance : mutation non uniforme

Expérience 3, 4 et 5 : Le système de reconnaissance à optimiser en utilisant l'opérateur de croisement arithmétique uniforme avec des valeurs λ : 0.25, 0.50 et 0.75 n'est pas efficace. Au delà des 100 premières générations, la performance en taux de reconnaissance calculée sur chacune des bases B et C ne s'améliore pas telle que nous observons par les graphiques 14, 15 et 16. Nous avons remarqué que les valeurs des gènes des chromosomes n'évoluent plus.

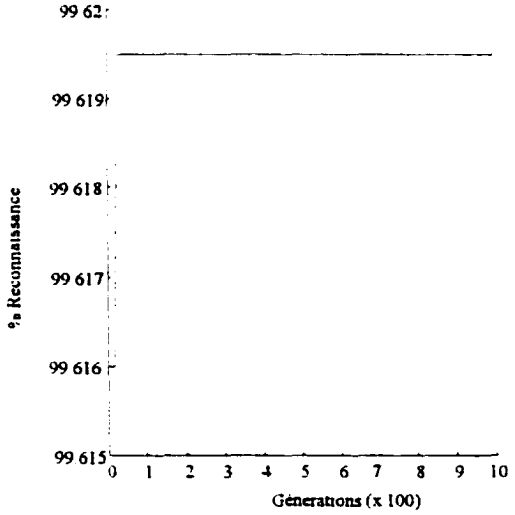


(a) Base B

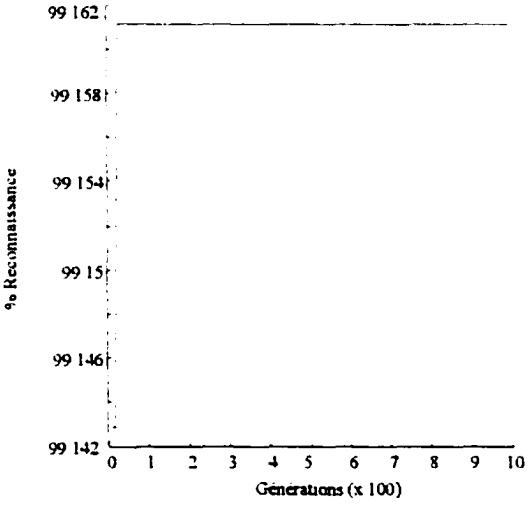


(b) Base C

Graphique 14 - Performance : croisement arithmétique $\lambda = 0.25$

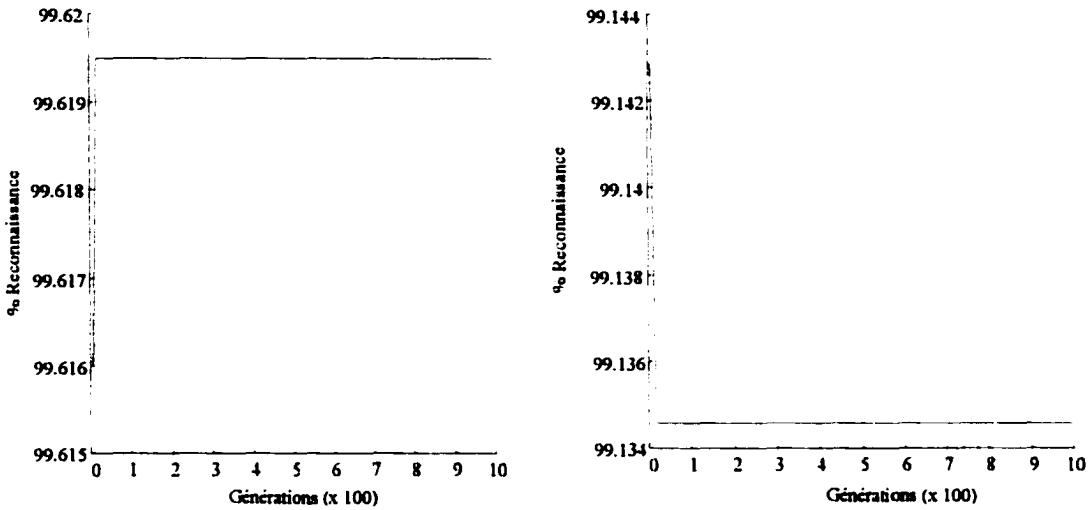


(a) Base B



(b) Base C

Graphique 15 - Performance : croisement arithmétique $\lambda = 0.50$

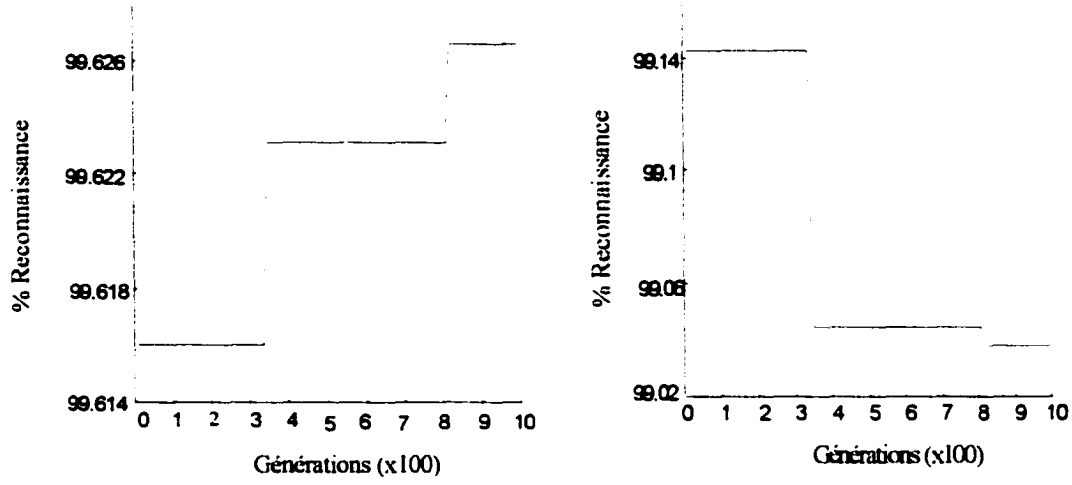


(a) Base B (b) Base C

Graphique 16 - Performance : croisement arithmétique $\lambda = 0.75$

Expérience 6 : L'utilisation exclusive du croisement arithmétique non uniforme donne une performance du système satisfaisante pour toutes les bases (voir tableau XV). La solution a été atteinte dès les premières générations.

Expérience 7 : La combinaison des opérateurs de croisement arithmétique non uniforme et l'opérateur de mutation non uniforme donne de bons résultats. Nous observons une évolution progressive de la performance du système de reconnaissance au fur et à mesure des générations contrairement aux expériences précédentes. Lors de la validation, nous avons constaté une baisse de la performance de 0.11% des taux de reconnaissance sur la base C. Le graphique 17 illustre les courbes de performance de la fonction d'adaptation évaluée sur les bases B et C.



(a) Base B (b) Base C

Graphique 17 - Performance : croisement arithmétique non uniforme et mutation non uniforme

En résumé, l'ensemble de ces expériences nous a permis de conclure et de confirmer qu'en général l'opérateur de croisement sert à explorer l'espace de recherche et l'opérateur de mutation permet de diversifier les valeurs des gènes des individus de telle sorte que l'AGS converge vers des solutions globales. L'utilisation conjointe de ces deux opérateurs permet d'obtenir de meilleures solutions. Cependant, il est préférable de choisir entre l'expérience 1 et 7 pour effectuer une simulation de 10 réplications avec les même paramètres. Nous avons donc choisi l'opérateur de croisement non uniforme et l'opérateur de mutation non uniforme avec la valeur de b = 5 (voir section 3.6.2.2) pour effectuer les 10 réplications. Lors de l'initialisation de la population, nous avons injecté un ensemble de bonnes solutions telles que : le chromosome C dont les valeurs des gènes c_i sont égales à 1 et les chromosomes obtenus comme solutions de chacune des réplications effectuées dans le cadre de la sélection des primitives par les AGS (voir section 5.3.1). L'objectif d'insérer de bonnes solutions dans la population initiale est de guider la recherche vers des solutions qui se rapprochent de celles obtenues lors de la sélection des primitives par les AGS. Les résultats de ces réplications sont présentés au tableau XVI où nous avons calculé les taux de reconnaissance sur toutes les bases de données.

Tableau XVI

Résultats de la pondération des primitives

	Taux de reconnaissance en %			
# Réplication	**Base A**	**Base B**	**Base C**	**Base D**
Tous les poids sont égaux à 1	99.65	99.62	99.15	97.47
1	99.60	99.64	99.08	97.38
2	99.59	99.64	99.08	97.29
3	99.61	99.66	99.10	97.39
4	99.59	99.63	99.12	97.34
5	99.53	99.64	99.04	97.18
6	99.56	99.63	99.05	97.19
7	99.55	99.66	99.04	97.25
8	99.60	99.66	99.08	97.30
9	99.60	99.64	99.07	97.31
10	99.60	99.65	99.07	97.37
Moyenne	99.60	99.64	99.01	97.30
Ecart-type	7.12E-04	1.39E-04	6.46E-04	5.58E-03

Nous avons observé que les solutions obtenues dans chacune des réplications sont différentes les unes des autres et les valeurs des gènes de ces solutions se rapprochent de la valeur 1. En effet, dans aucun cas, la valeur du gène c_i d'une solution C d'une réplication est identique à une autre valeur du gène c'_i d'une solution C' d'une autre réplication. La performance des taux de reconnaissance calculée sur la base B a bien été améliorée particulièrement pour les réplications 3, 7 et 8. En validation et en généralisation, nous observons une légère chute du taux de reconnaissance sur les bases C et D. Cette chute n'est pas très importante soit 0.14% sur la base C et 0.17% sur la base D. Les solutions obtenues seront décrites au chapitre 6.

En conclusion à cette section de la pondération des primitives, nous n'avons pas obtenu de bons résultats. Par conséquent, nous avons pensé effectuer une sélection des primitives par les AGS, puis de relancer l'apprentissage du réseau avec la solution obtenue (diminuer la complexité du réseau de neurones) et par la suite effectuer la pondération des primitives. Cette étude fait l'objet de la prochaine section.

5.5 Sélection puis pondération des primitives

Cette partie est inspirée des résultats obtenus lors de la pondération des primitives où nous avons observé que la pondération des primitives n'est pas une tâche facile. Nous avons voulu effectuer une autre approche sur la pondération des primitives une fois que la sélection soit réalisée. Les étapes de cette approche se détaillent de la manière suivante :

1/ Sélection des primitives par les AGS qui consiste à utiliser la méthodologie décrite à la section 4.8 et à récupérer la solution optimale obtenue à la section 5.3.1 à savoir les 100 primitives sélectionnées.

2/ Apprentissage du réseau de neurones avec les 100 meilleures primitives sélectionnées à l'étape 1 en utilisant la technique de cross-validation et en appliquant la méthodologie décrite à la section 4.8. La matrice poids ainsi constituée servira à l'étape 3.

3/ Pondération des primitives par les AGS. Ici, le réseau de neurones obtenus à l'étape 2 fait l'objet d'une dernière phase d'optimisation.

Les paramètres de l'AGS utilisés à l'étape 1 sont similaires aux paramètres de la sélection des primitives par les AGS (voir tableau VI) et les paramètres de l'AGS utilisés à l'étape 3 sont similaires aux paramètres retenus pour la pondération des primitives par les AGS (voir tableau XIII). Les paramètres de l'étape 2 sont identiques à ceux décrits à la section 4.7 (voir tableau III). Nous avons effectué 5 réplications de cette expérience pour cette approche. Le tableau XVII rappelle les résultats obtenus avec les 100 primitives dont les poids sont tout égaux à un (étape 2) et le tableau XVIII récapitule les résultats des réplications effectuées à l'étape 3.

Tableau XVII

Résultats des 100 primitives avant pondération

Taux de reconnaissance en %			
Base A	**Base B**	**Base C**	**Base D**
99.64	99.61	99.16	97.54

Tableau XVIII

Résultats de la pondération des primitives après la sélection

	Taux de reconnaissance en %			
# Réplication	**Base A**	**Base B**	**Base C**	**Base D**
1	99.55	99.65	99.00	97.20
2	99.57	99.65	99.11	97.42
3	99.59	99.65	99.01	97.33
4	99.55	99.63	99.04	97.17
5	99.58	99.64	99.12	97.35
Moyenne	99.57	99.65	99.04	97.33
Ecart-type	3.2E-04	8E-05	3.13E-03	1.11E-02

Nous avons obtenu des solutions différentes les unes des autres pour chacune des réplications. Il est difficile d'ordonner l'importance des primitives d'après les valeurs des gènes obtenues car ces valeur se rapprochent de la valeur 1. La performance du système de reconnaissance des chiffres isolés montre une légère augmentation en taux de reconnaissance de 0.04% sur la base B. Par contre en validation une chute de 0.12% des taux de reconnaissance a été observée sur la base C. Les solutions obtenues pour chaque réplications seront détaillées et interprétées au chapitre 6.

5.6 Conclusion

L'objectif principal de notre travail porte sur la sélection des primitives pertinentes du système de reconnaissance des chiffres isolés. Cet objectif a été atteint car les résultats obtenus par l'approche de la sélection des primitives par les AGS sont satisfaisants. Nous avons pu diminuer de 25% le nombre de primitives. En effet, le nombre de primitives a chuté de 32 primitives, passant de 132 à 100 primitives pertinentes. Le second objectif qui concerne la pondération des primitives ne donne pas de bons résultats. La majorité des valeurs des gènes obtenues pour chacune des primitives se rapprochent de la valeur initiale de 1. Ainsi, la pondération des primitives par les AGS effectuée soit par l'approche simple de la pondération (voir section 5.4) ou par l'approche sélection suivie par la pondération (voir section 5.5), n'est pas adaptée pour l'optimisation de notre système de reconnaissance de chiffres isolés. Une explication envisageable du non fonctionnement de la pondération des primitives est que le réseau effectue déjà cette pondération lors de la phase d'apprentissage. Ces poids sont associés à chaque interconnexion des nœuds du réseau de neurones de type MLP (voir figure 5) ce qui fait que la pondération est déjà effectuée par le réseau, rendant la méthode de la pondération des primitives par les AGS inopérante. Cette observation explique que les solutions obtenues montrent que les valeurs des gènes correspondant aux primitives avoisinent la valeur 1.

Les solutions obtenues par la sélection des primitives par les AGS et les AGI, la pondération des primitives par les AGS et la sélection suivie par la pondération des primitives par les AGS sont illustrées au chapitre 6.

CHAPITRE 6

DISCUSSION ET INTERPRÉTATION DES RÉSULTATS

6.1 Introduction

Ce chapitre est consacré à l'interprétation des résultats obtenus par les approches citées au chapitre 5. Dans un premier temps, nous abordons la sélection des primitives où les solutions seront commentées en détails pour les deux cas de sélection à savoir : les AGS et les AGI, et dans un deuxième temps, nous traitons de manière sommaire l'analyse de la pondération des primitives et la sélection postérieure à la pondération.

6.2 Sélection des primitives

Nous avons obtenu des solutions au problème d'optimisation pour la sélection des primitives par les AGS et les AGI. Nous analysons en détail un exemple de solution obtenu par chacun des algorithmes d'optimisation.

6.2.1 Algorithme génétique simple

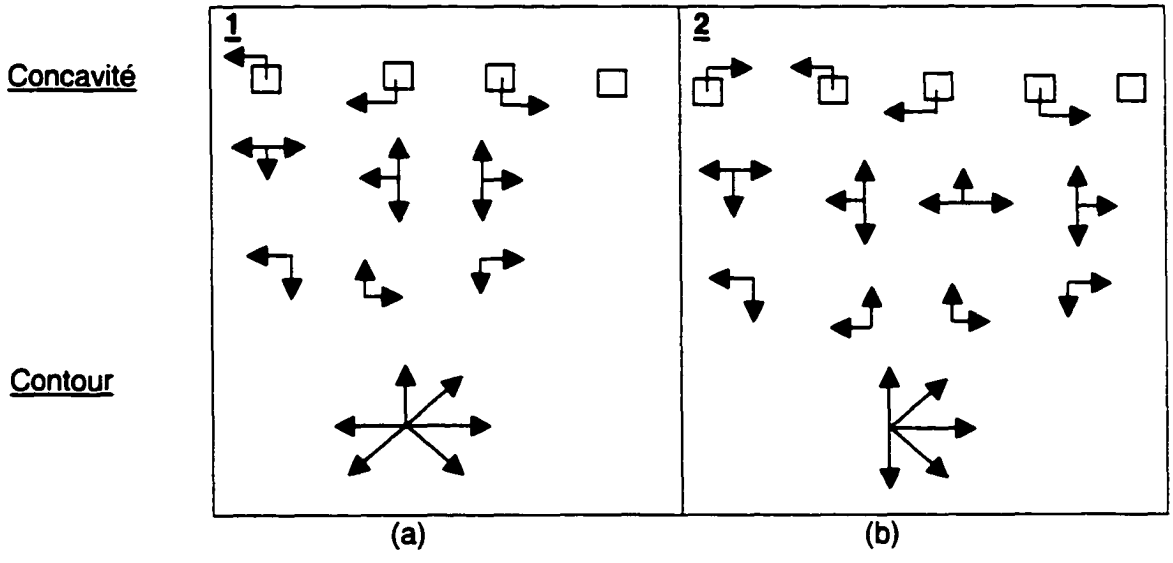
Nous traitons la meilleure solution obtenue à savoir la sélection des 100 primitives de la réplication 9 où nous représentons les primitives supprimées au tableau XIX. Les lignes de ce tableau représentent les 6 primitives de type zone et les colonnes représentent les 13 primitives de concavité, les 8 primitives du contour et 1 primitive de la surface. Nous procédons à l'analyse des primitives dans chaque zone dont chacune d'elle est constituée en total de 21 primitives (voir la constitution du vecteur de primitives à la section 4.4.5). Les positions (d'après la constitution du vecteur de primitives, chaque position est associée à une primitive) des primitives supprimées de la réplication 9 sont :
0, 7, 10, 17, 20, 40, 41, 42, 47, 53, 64, 69, 73, 79, 80, 82, 84, 90, 91, 97, 112, 113, 114, 115, 122, 123, 124, 125, 126, 128, 129, 130.

Tableau XIX

Présentation de la solution de la sélection par les AGS

	Concavité													Contour								
	f^4_d	f^4_c	f^4_b	f^4_a	f^4_0	f^3_0	f^3_1	f^3_2	f^3_3	f^2_{01}	f^2_{12}	f^2_{23}	f^2_{03}	0	1	2	3	4	5	6	7	Surf
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
1	x							x			x							x			x	
2																			x	x	x	
3				x						x											x	
4				x				x						x	x		x		x			
5			x	x						x												
6			x	x	x	x							x	x	x	x	x		x	x	x	
x = primitive supprimée																						

Pour effectuer une analyse visuelle des primitives pertinentes qui ont été sélectionnées (réplication 9), nous les avons représenté par la figure 24. A partir de cette figure et la figure 25, nous pouvons observer l'importance des primitives discriminantes sélectionnées.



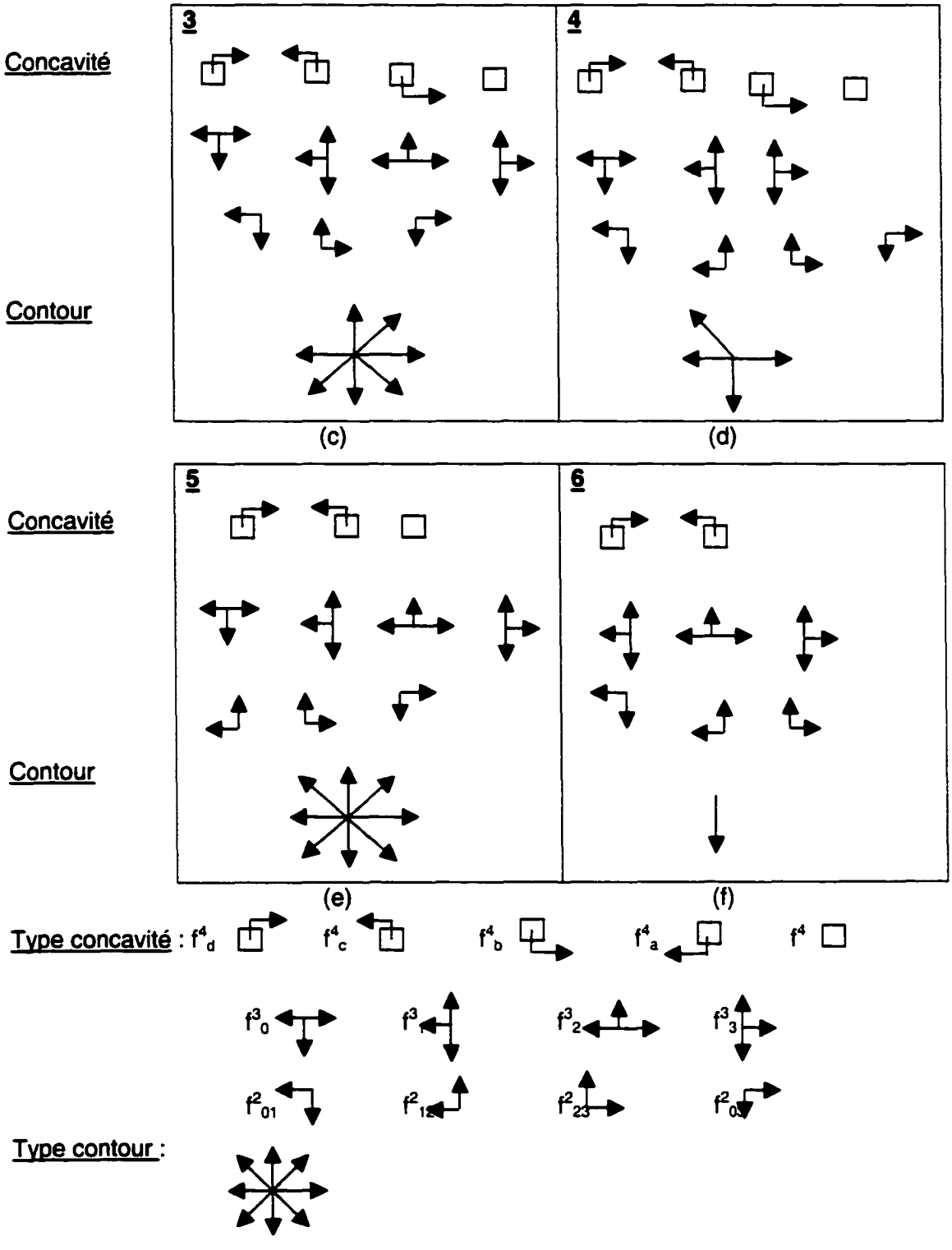


Figure 24 – Primitives pertinentes pour la reconnaissance de chiffres isolés

Zone 1 : Les primitives supprimées dans cette zone sont : f^4_d, f^3_2, f^2_{12}, la $4^{ème}$ et la $7^{ème}$ primitives du contour. Dans chacune des images illustrant des chiffres de la figure 25, nous pouvons confirmer que les primitives f^4_d , f^3_2 f^2_{12} peuvent être supprimées car il est rare de rencontrer ces types de primitives dans cette zone.

Zone 2 : Les primitives supprimées dans cette zone sont : la $5^{ème}$, $6^{ème}$ et la $7^{ème}$ primitives du contour. Effectivement ces primitives sont inutiles. Il n'est pas nécessaire de les calculer dans la zone (voir figure 25) car les pixels noirs se dirigent souvent vers le bas ou vers la droite. Nous observons également que les primitives de type concavité n'ont pas été supprimées (voir figure (24.b)).

Zone 3 : Les primitives supprimées dans cette zone sont : f^4_a, f^2_{01} et la $7^{ème}$ primitive du contour. Il n'est pas fréquent de rencontrer dans cette zone les deux primitives de concavité qui ont été supprimées. Les primitives du contour sont souvent calculées vers la droite donc elles sont importantes dans le cadre de notre système de reconnaissance de chiffres isolés (voir figure (24.c)).

Zone 4 : Les primitives supprimées dans cette zone sont : f^4_a, f^3_2 et les primitives du contour 0, 1, 3, 5. Il difficile de visualiser les primitives supprimées dans cette zone. Certaines primitives sont importantes dans cette zone telles que : les primitives 2, 4, 6 et 7 de type contour ainsi que la majorité des primitives de type concavité (exemple le chiffre 6 représenté par la figure 25).

Zone 5 : Les primitives supprimées concernent uniquement des primitives de concavité telles que : f^4_b, f^4_a et f^2_{01}. Par contre, aucune des primitives de types contour n'a été supprimée dans cette zone (voir figure (24.e) et 25).

Zone 6 : Cette zone peut ne pas contenir beaucoup de pixels ce qui fait que plusieurs primitives de contour et de concavité peuvent être éliminées parmi lesquelles : f^4_b, f^4_a, f^4_0, f^3_0, f^2_{03} de type concavité et 0, 1, 2, 3, 5, 6, 7 de type contour.

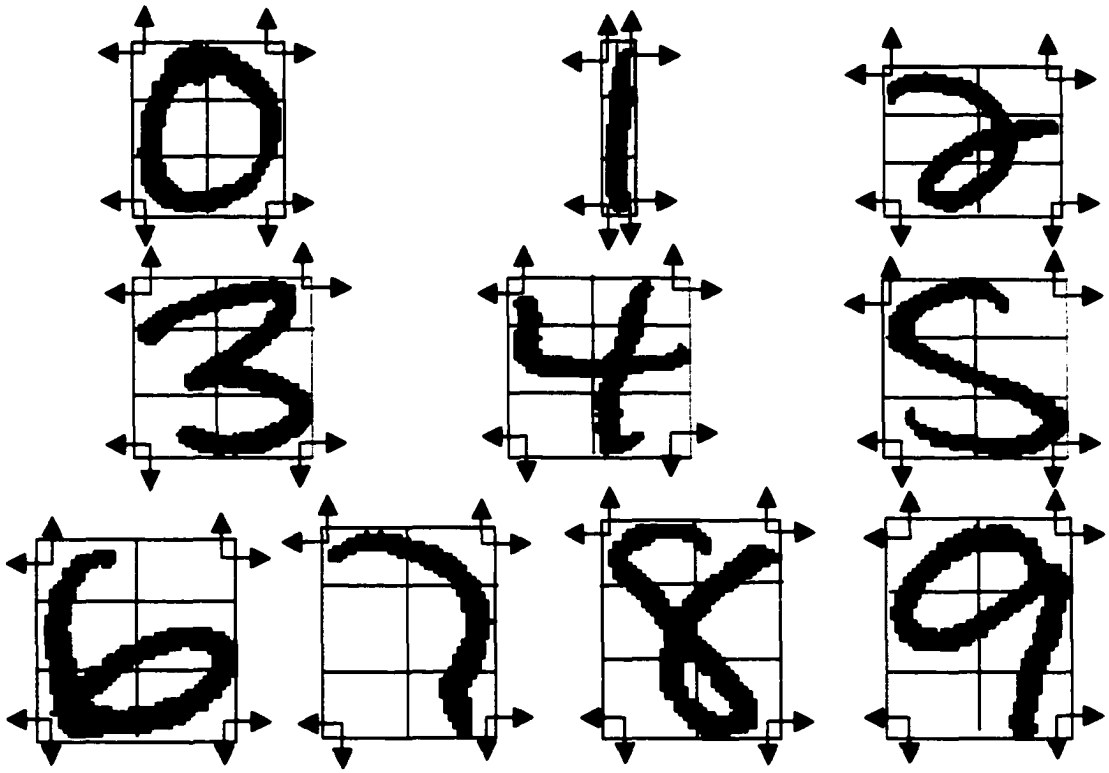


Figure 25 - Échantillon des chiffres isolés

En résumé, nous remarquons que la primitive de la surface n'a été supprimée dans aucune zone (voir figure 25), par conséquent c'est une primitive importante à extraire pour le système de reconnaissance des chiffres isolés. En plus, nous pouvons conclure qu'au niveau des extrémités des zones 1, 2, 5 et 6 qu'il est difficile d'extraire des primitives de concavité et/ou de contour. Effectivement dans chaque zone, il est rare de rencontrer des pixels noirs dans les directions nord ouest de la zone 1, nord est de la zone 2, sud ouest de la zone 5 et sud est de la zone 6 (voir figure 25). Plusieurs primitives de type contour sont également supprimées dans la zone 6.

Le tableau XX représente la liste des solutions obtenues pour chacune des réplications décrites à la section 5.3.1 (tableau VII). Nous avons effectué une intersection entre ces solutions afin de visualiser les primitives supprimées. Ces dernières sont communes à toutes les solutions (tableau XXI).

Tableau XX

Représentation des solutions des expériences de la sélection des primitives par AGS

# Réplication	La position des primitives supprimées
1	0,9,10,20,33,40,41,42,47,63,69,73,77,78,80,81,82,83,84,85,86,91,97, 114,115,122,124,125,126,128,129,130
2	0,9,10,20,33,40,41,42,47,63,69,73,77,78,80,81,82,83,84,85,86,91,97, 114,115,122,124,125,126,128,129,130
3	0,5,6,14,18,20,22,33,36,38,40,41,42,47,53,57,60,69,73,78,79,80,81,82, 83,84,85,86,90,91,93,94,112,113,115,118,122,125,126,128,129,130
4	0,9,10,20,33,40,41,42,47,63,69,73,77,78,80,81,82,83,84,85,86,91,97, 114,115,122,124,125,126,128,129,130
5	0,6,10,13,16,33,40,41,42,44,47,50,53,61,63,64,73,79,80,81,82,85,86, 90,91,93,94,97,112,113,123,124,125,126,128,129
6	0,9,10,20,33,40,41,42,47,63,69,73,77,78,80,81,82,83,84,85,86,91,97, 114,115,122,124,125,126,128,129,130
7	0,9,10,20,33,40,41,42,47,63,69,73,77,78,80,81,82,83,84,85,86,91,97, 114,115,122,124,125,126,128,129,130
8	0,5,7,10,19,20,22,33,40,41,42,47,50,53,57,59,61,63,69,78,79,80,81,82, 83,85,86,91,97,112,113,122,124,125,126,128,129,130
9	0,7,10,17,20,40,41,42,47,53,63,69,73,79,80,82,84,90,91,97,112,113, 114,115,122,123,124,125,126,128,129,130
10	0,5,6,7,10,18,23,33,36,40,41,42,45,47,53,54,79,80,82,84,85,86,91,94, 97,112,113,122,123,125,126,128,129,130

Une première constatation est que 5 réplications parmi 10 ont obtenu la même solution. Ces réplications sont :1, 2, 4, 6 et 7. Les positions des primitives communes de toutes ces solutions sont : 0, 40, 41, 42, 47, 80, 82, 91, 125, 126, 128 et 129. Ces positions correspondent respectivement aux primitives suivantes :

Tableau XXI

Primitives communes des solutions de la sélection des primitives par AGS

# Position	0	40	41	42	47	80	82	91	125	126	128	129
# Zone	1	2	2	2	3	4	4	5	6	6	6	6
Type	K	C	C	C	K	C	C	K	C	C	C	C
Primitive	f^4_d	5	6	7	f^4_a	1	3	f^4_a	2	3	5	6

C : contour K : concavité

Effectivement, si nous analysons chacune des primitives du tableau XXI en essayant de visualiser ces primitives sur les chiffres illustrés à la figure 25 alors nous constatons que ces primitives sont rarement utilisées par le système de reconnaissance des chiffres isolés. Aussi nous avons constaté l'importance des primitives de type contour de la zone 5 et des primitives de type concavité de la zone 2. Effectivement ces primitives n'ont pas été supprimées par l'ensemble des réplications.

6.2.2 Algorithme génétique itératif

L'approche des AGI pour la sélection des primitives n'applique pas la méthodologie décrite à la section 4.8. Nous avons obtenu une solution assez satisfaisante en optimisant le réseau sur la base C au lieu de la base B et en calculant la performance du système de reconnaissance en validation sur la base D au lieu de la base C. La solution obtenue est illustrée dans le tableau XXII où les primitives supprimées sont représentées. Nous n'allons pas illustrer les primitives pertinentes sélectionnées par les AGI telle que nous l'avions effectué à la section précédente car l'analyse est identique.

Tableau XXII

Présentation de la solution de la sélection par AGI

	f^4_d	f^4_c	f^4_b	f^4_a	f^4_0	f^3_0	f^3_1	f^3_2	f^3_3	f^2_{01}	f^2_{12}	f^2_{23}	f^2_{03}	0	1	2	3	4	5	6	7	Surf.
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
1											x	x							x	x	x	
2														x			x	x		x	x	
3																						
4								x						x	x					x	x	
5			x	x		x	x			x												
6				x				x							x	x	x		x	x	x	

Nous observons dans la zone 3 qu'aucune primitive n'est supprimée et dans la zone 5 que les primitives de contour sont importantes. La démarche pour analyser chacune des primitives supprimées est identique à celles de la sélection des primitives (voir section 6.2.1). Certaines primitives supprimées lors de la sélection par les AGS et la sélection par les AGI sont communes.

En résumé, il est préférable de refaire la sélection des primitives par les AGI en adoptant la méthodologie décrite à la section 4.8 afin de pouvoir comparer entre les deux approches de sélection.

6.3 Pondération des primitives

Il est très difficile d'analyser les solutions obtenues par les deux approches de la pondération à savoir : utiliser uniquement la pondération des 132 primitives originales (section 5.4) ou effectuer la sélection puis la pondération (section 5.5). Nous avons observé que les valeurs des gènes des solutions se rapprochent de la valeur 1. Il est rare ou quasiment impossible de trouver une valeur qui tend vers 0. A titre d'exemple, voici pour chacune des approches, une solution typique :

L'approche pondération des primitives : la solution présentée est obtenue par la réplication 4 du tableau XVI :

0.9185, 0.7093, 0.8841, 0.7583, 0.9414, 0.9175, 0.8800, 0.7619, 0.8385, 0.9146, 0.9376, 0.7694, 0.9580, 0.8399, 0.8797, 0.9038, 0.5636, 0.4410, 0.6688, 0.9072, 0.4899, 0.8414, 0.9427, 0.8374, 0.9597, 0.7400, 0.8839, 0.9480, 0.9327, 0.7436, 0.9360, 0.9033, 0.9426, 0.8789, 0.9000, 0.9155, 0.9487, 0.9159, 0.9012, 0.6192, 0.7477, 0.6905, 0.8232, 0.9408, 0.6990, 0.9313, 0.8371, 0.7551, 0.7828, 0.8801, 0.9335, 0.8620, 0.9243, 0.8690, 0.8932, 0.8558, 0.9332, 0.8682, 0.6253, 0.9110, 0.8719, 0.9056, 0.3938, 0.3075, 0.8686, 0.9523, 0.9099, 0.7157, 0.8816, 0.8808, 0.9032, 0.9129, 0.6658, 0.7286, 0.9050, 0.9666, 0.9379, 0.4603, 0.2336, 0.8112, 0.1455, 0.9216, 0.7620, 0.6922, 0.9071, 0.7820, 0.8525, 0.9521, 0.8346, 0.8560, 0.4172, 0.6612, 0.9189, 0.9337, 0.9134, 0.9110, 0.9312, 0.1744, 0.9319, 0.9421, 0.9321, 0.8811, 0.8674, 0.8786, 0.9318, 0.9387, 0.9234, 0.9521, 0.8894, 0.9577, 0.8195, 0.8262, 0.8127, 0.8757, 0.8886, 0.8967, 0.9101, 0.8334, 0.9334, 0.9216, 0.9720, 0.7002, 0.6523, 0.9076, 0.8180, 0.8638, 0.6636, 0.8161, 0.9452, 0.6655, 0.6122, 0.9687.

Les performances en taux de reconnaissance du système sont légèrement en chute sur la base C en validation et sur la base D en généralisation.

L'approche sélection puis pondération des primitives : la solution présentée est obtenue par la réplication 2 du tableau XVIII :

0.9033, 0.4674, 0.7903, 0.9516, 0.9417, 0.4147, 0.6346, 0.9609, 0.9172, 0.8973, 0.8215, 0.7188, 0.9438, 0.7806, 0.9237, 0.9344, 0.9407, 0.6764, 0.9010, 0.9212, 0.9694, 0.8985, 0.9548, 0.9361, 0.9106, 0.9374, 0.9379, 0.9145, 0.9234, 0.5252, 0.8530, 0.9219, 0.6664, 0.8479, 0.7631, 0.9300, 0.9493, 0.7678, 0.8568, 0.9320, 0.7587, 0.9607, 0.8910, 0.8875, 0.5369, 0.9282, 0.9783, 0.8748, 0.9173, 0.5547, 0.9441, 0.9183, 0.8991, 0.4654, 0.9494, 0.7758, 0.9401, 0.9657, 0.9222, 0.9415, 0.9238, 0.9398, 0.9449, 0.9477, 0.7901, 0.7722, 0.7546, 0.8766, 0.4871, 0.9512, 0.9112, 0.8405, 0.9174, 0.9202, 0.9256, 0.9827, 0.9228, 0.9360, 0.9703, 0.9439, 0.8927, 0.7436, 0.8604, 0.8885, 0.7266, 0.9172, 0.7999, 0.8628, 0.9256, 0.9542, 0.9537, 0.2418, 0.9132, 0.9000, 0.9290, 0.9304, 0.9159, 0.9385, 0.8358, 0.9583.

En observant ces valeurs, nous pouvons conclure que la pondération des primitives en général ne peut être appliquée dans notre système de reconnaissance des chiffres isolés. Il n'est donc pas nécessaire de représenter les solutions des autres réplications des tableaux XVI et XVIII.

6.4 Conclusion

L'interprétation des résultats n'est pas une tâche facile. Bien que les résultats soient difficiles à interpréter, nous avons établi des constatations d'après nos observations car la majorité des primitives sont corrélées entre elles. En effet, il existe un lien assez fort entre les primitives de type concavité et celles de type contour. Nous avons constaté que la sélection des primitives du système de reconnaissance de chiffres a donné de bons résultats contrairement à la pondération des primitives. En effet, lors de la phase de la sélection, le nombre de primitives a diminué de 25% par rapport aux primitives initiales, passant de 132 à 100 primitives sélectionnées. La suppression des primitives telle que nous l'avons analysée dans ce chapitre (voir section 6.2) concerne une bonne partie des primitives qui se trouvent dans les extrémités des zones 1, 2, 5 et 6. Une observation intéressante s'ajoute est que les primitives de type contour de la zone 5 et toutes les primitives de type surface sont essentielles à la reconnaissance de chiffres. Contrairement aux bons résultats de cette sélection, les résultats obtenus lors de la phase pondération des primitives ne sont pas satisfaisants. Nous avons observé que la valeur des gènes des solutions se rapproche du poids initiale de valeur 1. Une explication peut être avancée pour le non fonctionnement de cette phase est que lors de la phase d'apprentissage du réseau de neurones, la pondération des entrées est effectuée à chaque connexion synaptique de la couche cachée. Cette phase d'apprentissage génère alors la matrice de poids qui correspondent à l'importance de chacune des primitives correspondantes. Il n'est donc pas nécessaire d'effectuer une pondération des primitives par les AG lors de l'utilisation des réseaux de neurones, mais par contre si d'autres types de classifieur sont utilisés alors nous pourrions voir l'utilité de la pondération des primitives. A titre d'exemples, les auteurs des articles [41, 43] ont obtenu des résultats satisfaisants en utilisant le classifieur du plus proche voisin pour la pondération des primitives.

CONCLUSION

Les travaux présentés dans ce mémoire abordent les différentes étapes nécessaire à la construction d'un système de reconnaissance de chiffres manuscrits isolés. Pour chacune de ces étapes à savoir : les pré-traitements, l'extraction des primitives et la classification, nous avons tenté de proposer une méthode d'optimisation pour la sélection des primitives pertinentes du système de reconnaissance des chiffres. De nombreuses études ont été proposées dans la littérature [47]. Notre orientation s'est focalisée sur les AG.

Ainsi, dès l'étape d'extraction des primitives, la sélection des primitives pertinentes et non redondantes du système est effectuée. Cette sélection consiste à réduire les entrées du classifieur (réseau de neurones de type MLP) tout en améliorant ou en maintenant le taux de reconnaissance de la classification. Notre contribution principale est l'étude de la mise enoeuvre des AG pour sélectionner les primitives discriminantes. Nous avons effectué des tests sur la base de chiffres isolés NIST-SD19 une fois que nous avions déterminé les bon paramètres de l'AG tels que : la taille de la population, le nombre de génération, les probabilités de mutation et de croisement et les méthodes de reproduction. Ces paramètres de l'AG sont définis par une étude théorique permettant un gain de temps de calcul considérable. Cette étude consiste à optimiser la fonction *One-Max* qui s'apparente au problème de la sélection des primitives.

La fonction d'adaptation utilisée dans le cadre de notre étude est multi-critère. Le premier critère consiste à minimiser la dimension du vecteur de primitives (entrée du réseau) et le second est l'amélioration de la performance du système de reconnaissance des chiffres. Il est donc difficile de déterminer l'importance de chacun de ces deux critères. Dans notre cas d'études, nous avons déterminé que l'importance de la sélection des primitives est deux fois plus grande par rapport à la performance du système (voir section 4.5 où $\alpha = 1000$ et $\beta = 1$).

Les résultats obtenus de cette sélection sont satisfaisants. Nous avons réduit de 25% le nombre de primitives initiales passant de 132 à 100 primitives sélectionnées. Cette réduction signifie que le nombre d'entrée du réseau de neurones a diminué tout en maintenant (augmentation moyenne de 0.02%) les taux de reconnaissance calculés sur la base B (voir tableau VIII). Ainsi, l'optimisation du réseau a permis de réduire le temps

de calcul requis pour l'apprentissage du réseau et de reduire également la complexité du réseau de neurones.
Pour pallier au problème du choix des poids accordés à chaque fonction objective, il est recommandé de passer à l'optimisation multi-critère basée sur la technique des Fronts de Pareto. Cette approche a l'avantage de s'affranchir du problème de la définition des poids de la fonction d'adaptation, c'est-à-dire la détermination arbitraire de l'importance de chaque critère afin de trouver un ensemble de solutions optimales à l'optimisation du système de reconnaissance [61].

Le second aspect de notre projet est la pondération des primitives qui consiste à déterminer les primitives les plus discriminantes en leur attribuant un poids. Les expériences effectuées n'ont pas donné de bonnes solutions. Nous avons observé que les valeurs des poids accordés aux primitives se rapprochent de la valeur initiale (valeur égale à 1). Nous concluons donc que la pondération des primitives n'est pas adaptée pour l'optimisation de notre système de reconnaissance de chiffres isolés car le réseau effectue déjà cette pondération des primitives lors de la phase d'apprentissage. Cependant, il a été rapporté dans la littérature que cette approche peut fonctionner avec d'autres types de classificateurs tel que la méthode du plus proche voisin [41, 43].

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Tappert C.C, Suen C.Y., & Wakahara T. (1990). The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8), 787-808.

[2] Richard O. Duda, Peter E. Hart , & David G. Stork. (2001). *Pattern Classification.* (Second edition). New York: Wiley-Interscience.

[3] Oliveira L.S., Sabourin R., Bortolozzi F., & Suen C.Y. (2001). A Modular system to recognize numerical amounts on Brazilian bank cheques. *6th International Conference on Document Analysis and Recognition*, 389-394. Seattle, USA.

[4] Oliveira L.S., Lethelier E, Bortolozzi F, & Sabourin R. (2000). A new segmentation approach for handwritten digits. *15th International conference on pattern recognition*, 2, 323-326. Barcelona, Spain.

[5] Belaid A., & Belaid Y. (1992). *Reconnaissance de formes: Méthodes et Applications.* Paris : InterEditions.

[6] Oivid Due Trier, Anil K. Jain, & Torfinn Taxt. (1996). Feature extraction methods for character recognition – a survey. *Pattern recognition*, 29(4), 641-662.

[7] Heutte L., Paquet T., Moreau J.V., Lecourtier Y, & Olivier C. (1998). A structural/statistical feature based vector for handwritten character recognition. *Pattern recognition letters*, 19, 629-641.

[8] Guyon I., Poujaud I., Personnaz L., Dreyfus G., Denker J., & Le Cun Y. (1989). Comparing different neural networks architectures for classifying handwritten digits2. *Int. J. Conf. on Neural Networks,* 2, 127-132. Washington, USA.

[9] Plamondon Réjean, & Srihari Sargur. (2000). On-Line and Off-Line Handwriting Recognition : A comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 63-84.

[10] Abhijit S. Pandya, & Robert B. Macy (1996). *Pattern Recognition with Neural Networks in C++.* Floride: A CRC book published in cooperation with IEEE press.

[11] Robert Schalkoff (1992). *Pattern Recognition: Statistical, structural and Neural Approaches.* USA: John Wiley and Sons, Inc.

[12] Miclet L. (1984). *Méthodes structurelles pour la reconnaissance des formes.* Paris : Eyrolles.

[13] Gaillat G. (1983). *Méthodes statistiques de reconnaissance de formes.* Centre d'édition et de documentation de l'École Nationale de Techniques Avancées.

[14] Lippmann R.P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine.*

[15] Remm Jean-François *Probabilités et réseaux de neurones.* CRIN-CNRS et INRIA-Lorraine Rapport Technique RR n°2909.

[16] Bishop Christopher M. (1995). *Neural networks for pattern recognition.* Birmingham: Clarendon press Oxford.

[17] Zurada J.M. (1992). *Introduction to artificial neural systems.* Minnesota: West publishing company.

[18] El Yacoubi Abdenaim (1996). *Modélisation Markovienne de l'écriture manuscrite : Application à la reconnaissance des adresses postales.* Thèse de doctorat de l'Université de Rennes 1.

[19] Mangalagiu Diana (1999). *Accélération de l'algorithme des K plus proches voisins par réorganisation.* France : Thèse de doctorat de l'École de polytechnique.

[20] Guoqiang Peter Zhang (2000). Neural Networks for classification : a survey. (2000). *IEEE Transactions on systems, man, and cybernetics – Part C : applications and reviews*, (30)4, 451-462.

[21] Goldberg David E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley.

[22] Holland J.H. (1975). *Adaptation in natural and artificial systems.* MIT Press.

[23] Man K.F., Tang K.S., & Kwong S. (2000). *Genetic algorithms. Concepts and designs.* Springer.

[24] Beasly D., Bull D.R., & Martin R.R. (1993). *An overview of genetic algorithms : Part1.* Research topics. University computing, UCISA, (15)2, 58-69.

[25] Beasly D., Bull D.R., & Martin R.R. (1993). *An overview of genetic algorithms : Part2.* Research topics. University computing, UCISA.

[26] Janikow C.Z., & Michalewicz Z. (1991). An experimental comparaison of binary and floating point representations in genetic algorithms. *Proceeding of the fourth international conference on Genetic Algorithms*, Morgan Kaufmann, 31-36.

[27] Herrera F., Lozano M., & Verdegay J.L. (1998). Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis. *Artificial Intelligence Review.* 12(4). 265-319.

[28] Davis L. (1991). *Handbook of genetic algorithms.* Vau Nostrand Reinhold.

[29] Matthew Wall (1996). *GALIB (2.4.4) : A C++ library of genetic algorithms components.* Massachusetts Institute of Technology MIT press. Availaible: http://lancet.mit.edu/ga/

[30] Srinivas M., & Lalit M. Patnaik (1994). Genetic Algorithm : *A Survey. Computer.* 27(6), 17–26.

[31] Michalewicz Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* New York: Springer Verlag.

[32] Eshelman L.J., & Schaffer J.D. (1993). Real-Coded Genetic Algorithms and Interval Schemata. *Foundation of Genetic Algorithms* 2. San Mateo: L. Darrel Whitley (Morgan Kaufmann Publishers), 187-202.

[33] Wright A. (1991). Genetic Algorithms for Real Parameter Optimization. *Foundation of Genetic Algorithms 1.* San Mateo: G.J.E Rawlin (Morgan Kaufmann), 205-218.

[34] Muhlenbein H., & Schlierkamp Voosen D. (1993) Predective Models for the Breeder Genetic Algorithm I. *Continuous Parameter Optimization. Evolutionary Computation 1*, 25-49.

[35] Eshelman L.J., Caruana R., & Schaffer J.D. (1989). Biases in the Crossover Landscape. *Proc. 3rd Int'l Conference on Genetic Algorithms.* J.David Shaffer (Morgan Kaufmann Publishing).

[36] De Jong K.A., & Spears W.M. (1992). A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms. *Proceedings of the Foundations of Genetic Algorithms Workshop.* 5, 1-26. Indiana.

[37] Man K.F, & Tang K.S. (1997). Genetic Algorithms for Control and Signal Processing. *Industrial Electronics, Control and Instrumentation IECON. 23rd International Conference on*, 4, 1541-1555.

[38] Eiben A.E., Hinterding R., & Michalewicz Z. (1999). Parameter Control in Evolutionary Algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2), 124-141.

[39] Maria J. Martin-Bautista, & Maria-Amparo Vila (1999). A survey of Genetic Feature Selection in Mining Issues. *Evolutionary Computation, CEC, Proceedings of the 1999 Congress on*, 2, 1314-1321

[40] Mitchell M. (1996). *An Introduction to Genetic Algorithms.* MIT Press.

[41] Gyeonghwan Kim, & Sekwang Kim. (2000). Feature Selection Using Genetic Algorithms for Handwritten character Recognition. *7th International Workshop on Frontiers in Handwriting Recognition*, 103-112. Amsterdam.

[42] Niadapam Chaikla, & Yulu Qi. (1999). Genetic Algorithms in Feature Selection. *Systems, Man and Cybernetics, IEEE SMC Conference Proceedings*, 5, 538-540.

[43] Raymer M.L., Punch W.F., Godman E.D., Kuhn L.A., & Jain A.K. (2000). Dimensionality Reduction Using Genetic Algorithms. *Evolutionary Computation, IEEE Transaction*, 4(2), 164-171.

[44] Dash M., & Liu H. (1997). Feature Selection for Classification. *Intelligent Data Analysis 1*. 131-156.

[45] Yang Jihoon, & Honavar Vasant (1997). Feature Subset Selection Using A Genetic Algorithm. *Genetic Programming.* A data mining perspective. New York: Kluwer Academic.

[46] Andreas Moser (1999). *Distributed Genetic Algorithms for Feature Selection.* Thèse de doctorat de l'université de Kaiserslautern en Allemagne.

[47] Kudo Mineichi, & Sklansky Jack (2000). Comparaison of Algorithms that Select Features for Pattern Classifiers. *Pattern Recognition*, 33, 25-41.

[48] John Moody, & Joachim Utans (1991). Principled architecture selection for neural networks : Application to corporate bond rating prediction. *Appears in J.E. Moody, S.J. Hanson and R.P. Lippmann, editors, Advances in neural information processing systems 4.* San Mateo, CA : Morgan Kaufmann Publishers.

[49] Joachim Utans, & John Moody (1991). *Selecting neural network architectures via prediction risk: Application to corporate bond rating prediction.* 1st International conference on artificiel intelligence applications on Wall Street. Los Alamitos, CA : IEEE Computer Society Press.

[50] Joachim Utans, John Moody, Steve Rehfuss, & Hava Siegelmann (1995). Input Variable selection for neural networks : application to predicting the U.S. business cycle. *Computational Intelligence for Financial Engineering, Proceedings of the IEEE/IAFE*, 118–122.

[51] Mark A. Hall, & Lloyd A. Smith (2000). Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford University, CA. Morgan Kaufmann Publishers.

[52] Punch W.F, Goodman E.D., Min Pei, Hovland P., & Enbody R. (1993). Further Research on Feature Selection and Classification Using Genetic Algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 557-564.

[53] Ji Q., & Zhang Y. (2001). Camera Calibration with Genetic Algorithms. *IEEE Transaction on Systems, Man and Cybernetics, Part A: systems and Humans*, 31(2), 120-130.

[54] Ramesh V.E., & Murty N. (1999). Off-Lin signature Verification Using Genetically Optimized Weighted Features. *Pattern Recognition*. 32(2), 217-233.

[55] Ho S.Y., & Huang H.L. (2001). Facial Modeling from an Uncalibrated Face Image Using a Coarse-to-Fine Genetic Algorithm. *Pattern Recognition*, 34(5), 1015-1031.

[56] Kavzoglu Taskin, & Mather Paul M. (2000). The Use of Feature Selection Techniques in the Context of Artificial Neural Networks. *Proceedings of the 26th Annual Conference of the Remote Sensing Society.*

[57] Oliveira L.S., Benahmed N., Sabourin R., Bortolozzi F., & Suen C.Y. (2001). Feature Subset Selection Using Genetic Algorithms for Handwritten Digit Recognition. *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*, 362 –369.

[58] Shamik Sural, & Das P.K. (2001). A Genetic Algorithm for Feature Selection in a Neuro-Fuzzy OCR System. *6th International Conference on Document Analysis and Recognition.* Seattle USA, 987-991.

[59] Faten Hussein, Nawwaf Kharma, & Rabah Ward (2001). Genetic Algorithms for Feature Selection and Weighting, Review and Study. *6th International Conference on Document Analysis and Recognition.* Seattle USA, 1240-1244.

[60] Nicholas W. Strathy (1993). *A Method for Segmentation of Touching Handwritten Numerals.* Mémoire de maîtrise au Computer Science Department, Concordia University.

[61] Eckart Zitzler (1999). *Evolutionary Algorithms for Multiobjective Optimization : Methods and Application.* Thèse de doctorat de l'institut de technologie fédéral de Zurich.