12/03

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur et de la recherche scientifique

Ecole Nationale Polytechnique

Département de Génie Electrique

Mémoire de Fin d'Etude

En vue d'obtenir le diplôme d'ingénieur d'état en Automatique

Thème

Application des Algorithmes Génétiques en Identification Non-linéaire, et en commandes H∞ et par Backstepping

Proposé par :

Mr D. BOUKHETALA
Mr F. BOUDJEMA

Réalisé par :

A. BENSOUNA
H. LABDELAOUI

Promotion 2003

10, Avenue Hassan Badi, BP El Harrach Alger

Dédicace

Je dédie ce travail à mes chers parents,

à mes frères et sœurs, à ma famille,

et à mes amis

A. BENSOUNA

Je dédie ce travail à ma famille

et à tous mes amis

H. LABDELAOUI

Remerciements

Nous tenons à remercier et à exprimer notre respectueuse gratitude à nos promoteurs **Mr D. BOUKHETALA** et **Mr F. BOUDJEMA,** pour l'aide précieuse qu'ils nous ont apportée et pour l'assistance qu'ils nous ont témoignée tout au long de ce travail.

Qu'ils soient vivement remerciés **Mr M. TADJINE**, pour ses conseils et ses contributions pour l'achèvement de ce travail, ainsi que **Mr H. CHEKIREB** qui nous a fait l'honneur de juger notre travail.

Enfin, nous tenons à remercier tous ceux qui nous ont aidé, de prés ou de loin, pour la réalisation de travail.

Sommaire

Chapitre 2: Les algorithmes génétiques multiobjectif

Chapitre 3: Application des AGs en commande et en Identification des systèmes

Introduction Générale

La science survient du grand désire de l'homme de comprendre et de contrôler les phénomènes naturels qu'il rencontre dans le monde. A travers l'histoire, les êtres humains ont graduellement bâti un grand édifice de connaissance, qui leurs permet de mieux connaître et s'adapter aux environnements dans lesquels ils vivent, et de prédire, par des mesures variantes, le temps, les mouvements des planètes, les éclipses solaires et lunaires, les processus des maladies, l'évolution de la croissance économique, les étapes de développement du langage chez les enfants, et un vaste panorama d'autres phénomènes naturels, sociaux, et culturels. Par ailleurs, l'homme n'a cessé de développer des moyens de plus en plus complexes pour découvrir et contrôler divers aspects de sa vie et de ses interactions avec la nature.

L'apparition des ordinateurs électroniques a été probablement le développement le plus révolutionnaire dans l'histoire de la science et de la technologie. Les anciens scientifiques comme Alain Turing, John von Neumann, Norbert Wiener, et autres, étaient motivés surtout par des visions d'inspirer des programmes informatiques dotant d'une intelligence artificielle, avec une habilité de se répliquer, la capacité de connaître et d'interagir avec leurs environnements. Ces premières initiatives étaient très intéressées par la biologie et la psychologie par rapport à l'électronique, elles considéraient les systèmes naturels comme des métaphores d'orientation pour compléter leurs visions. Il n'est pas étonnant donc, que les premiers ordinateurs servaient non seulement à calculer les trajectoires des missiles et à déchiffrer les codes militaires, mais aussi à modéliser le cerveau, imiter l'apprentissage humain, et simuler l'évolution biologique. Ces activités dans le domaine du calcul motivées par la biologie, ont progressé et régressé pendant plusieurs années de recherche et d'étude. mais depuis le début des années 80s, elles ont tous subi une réinnovation dans la communauté de la recherche numérique. La première activité s'est développée dans le champ des réseaux

de neurones artificiels, la seconde dans l'apprentissage des machines, et la troisième dans ce qui est appelé le *Calcul Evoltionnaire*, duquel les Algorithmes Génétiques (AGs) sont l'exemple le plus proéminent.

Dans ce travail, nous appliquons les algorithmes génétiques à différents types de problèmes de commande et d'identification, en mettant en évidence les points forts qu'ils présentent et qui peuvent être exploités dans ce type de problème, afin d'améliorer les performances des régulateurs et la conduite générale des systèmes de commande.

Dans le premier chapitre, nous présentons les algorithmes génétiques comme des méthodes d'optimisation et de recherche globale, pouvant être exploitées très efficacement dans le domaine de l'automatique. Ce chapitre introduit les concepts et les principes de base qui s'avèrent nécessaires pour l'implémentation d'un AG simple, i.e. évoluant pour un seul critère d'optimisation.

Le chapitre 2 donne une base théorique nécessaire pour la manipulation des algorithmes génétiques multiobjectif. Il couvre quelques concepts de base dans l'optimisation multiobjectif, en expliquant le besoin d'utilisation du *Decision making* dans la procédure d'optimisation, et en décrivant quelques approches générales de l'articulation des préférences. Dans ce chapitre, nous présentons la nouvelle méthode du Pareto-ranking basée sur les travaux de Fonseca et Fleming.

Le dernier chapitre est consacré aux applications des AGs dans des problèmes d'automatique, entre autres la commande et l'identification. Dans le premier cas, un AG est utilisé pour optimiser les paramètres d'une loi de commande obtenue via la technique de Backstepping, qui sera par la suite, appliquée à un robot manipulateur de type Puma 560. Dans le deuxième cas, nous avons réalisé une optimisation multiobjectif via l'approche H_2 / H_∞ qui donne des régulateurs mixtes assurant le compromis H_2 / H_∞ afin de permettre à l'utilisateur de choisir parmi ses derniers selon ses préférences. Dans le dernier cas, nous utilisons un AG pour l'identification d'un système non-linéaire de type *NARMAX*.

Enfin, nous concluons notre travail par une estimation générale des différents résultats obtenus dans les applications qui sont faites, ainsi que l'évaluation des performances des AGs dans de telles applications.

Chapitre 1

Rappel sur les Algorithmes Génétiques et leurs Applications en Automatique

1.1 Introduction

A travers l'histoire, la nature a toujours été une source majeure d'inspiration et de métaphores pour le développement scientifique et technique qui, de suite, a contribué à la meilleure compréhension de la nature elle-même. Ce n'est pas difficile d'identifier les liens entre l'oreille et le microphone, l'œil et l'appareil photo, la chauve-souris et le système du sonar. le cerveau et les réseaux de neurones artificiels, et ainsi de suite. De la même façon, le processus d'évolution naturelle a inspiré un montant croissant de recherche dans les systèmes artificiels, qui est rendu possible grâce à la disponibilité grandissante de la puissance du calcul informatique moderne.

Dans ce premier chapitre, nous introduisons les principes de base du fonctionnement d'un algorithme génétique en insistant sur la capacité d'évolution des techniques artificielles dans le domaine de l'automatique. Il commence avec une description brève du calcul évolutionnaire dans son ensemble dans la section 1.2, après lequel une famille particulière de méthodes, connues sous le nom d'algorithmes génétiques, est introduite et discutée avec plus de détail. La section 1.5 décrit l'aspect théorique des algorithmes génétiques en introduisant la notion de schémas, des exemples de leurs applications antérieures en commande et en identification des systèmes sont examinés dans la section 1.6.

1.2 Le calcul évolutionnaire [Fon 1995]

Le terme Calcul Evolutionnaire (CE) est utilisé pour décrire une classe générale de méthodes de calcul qui sont inspirées des variations génétiques et du processus de la sélection naturelle. L'intérêt porté par l'utilisation de telles méthodes dans des domaines aussi divers que la biologie, la chimie, l'économie, les sciences de l'ingénierie, les sciences cognitives et les mathématiques, parmi d'autres, est double. Premièrement, le CE fournit des moyens de modélisation et de simulation de processus naturels et économiques, à travers lesquels les processus originaux peuvent être étudiés et/ou analysés. De plus, le CE a aussi prouvé son utilité dans l'orientation efficace des recherches complexes et des problèmes d'optimisation. en étendant le domaine de l'optimisation, à des régions non accessibles auparavant.

Les méthodes du calcul évolutionnaire ou Algorithmes Evolutionnaires (AEs), ont vu le jour vers la fin des années soixante. Les Stratégies de l'Evolution (SEs) (en allemand: *Evolutionstrategien*) ont été développées par Rechenberg (1973; 1994) et Schwefel (1981) en Allemagne, alors qu'aux États-Unis deux approches supplémentaires sont survenues: La Programmation Evolutionnaire de Fogel (PE) (Fogel, 1991), et les Algorithmes Génétiques de Holland (AGs) (Holland, 1975). Des méthodes similaires ont aussi été développées vraisemblablement en Pologne (Karcz-Duleba, 1994), où le terme "la sélection douce" a été utilisée pour différencier l'approche évolutionnaire des méthodes du grimpeur traditionnelles.

Les stratégies d'évolution ont principalement été vues comme des méthodes d'optimisation robustes, et toujours efficaces, pour le traitement des problèmes d'ingénierie impliquant beaucoup de variables de décision réelles. Des suppositions habituelles pré-nécessaires pour d'autres méthodes d'optimisation telles que la continuité ou la différenciabilité de la surface du coût, ont conduit à la supposition la plus générale de la forte causalité, valide pour la plupart des problèmes d'ingénierie. La Programmation Evolutionnaire a été préalablement développée dans le contexte de l'intelligence artificielle pour évoluer les tables de la transition d'état des machines (Bäck et Schwefel, 1993), mais elle a été étendue ensuite pour traiter des variables de décision réelles.

Les algorithmes génétiques ont un impact beaucoup plus fort que leurs équivalents sur les méthodes d'optimisation et de recherche globale en terme de performance, par opposition à celles locales. Les AGs ont été initialement formulés comme des algorithmes de recherche combinatoire, qui exigent la discrétisation de l'espace de recherche afin d'être appliquées à des problèmes impliquant des variables de décision réelles. Plutôt qu'être vu comme faiblesse, le besoin de la discrétisation a fait apparaître des AGs particulièrement convenables pour une

implémentation sur les calculateurs numériques, en maintenant une certaine analogie vis-à-vis de la génétique naturelle. Dans la nature, les séquences des gènes qui prennent une des quatre valeurs possibles trouvent une expression aussi bien dans le cas continu que dans le cas discret des caractéristiques somatiques, (exemples: de ce qui est, respectivement, poids du corps et son nombre d'organes). Les discussions théoriques basées sur l'hypothèse des blocks de construction et le théorème des Schémas (Holland, 1975; Goldberg, 1989) sont aussi apparues pour supporter la discrétisation.

Leurs demandes de globalité, la facilité avec laquelle ils peuvent être modifiés et appliqués à différentes classes de problèmes, et beaucoup de résultats encourageants, ont conféré une grande popularité aux AGs. Malheureusement, la théorie a été lente à accompagner leur développement pratique, conduisant à une situation de conflit entre la théorie et la pratique ou même de contradiction. Actuellement, il n'y a aucune théorie prédictive des AGs applicable à des problèmes de dimension réaliste, en dépit de quelques contributions dans cette direction (Manderick *et al.*, 1991).

1.3 Concepts et principes de base

Aux environs de l'année 1850, Gregor Mendel a développé sa théorie des *gènes*. Les gènes sont des segments de renseignements héréditaires trouvés dans les êtres vivants. Chaque gène décrit un petit aspect de l'existence de l'être duquel il fait partie, les différents états possibles des gènes sont appelés *allèles*. La collection complète de gènes dans un être décrit cet être en entier. Cette théorie est connue comme la théorie génétique ou simplement *les génétiques*.

Les gènes sont codés dans une collection de molécules très complexes qui sont appelées *ADN* ou les *chromosomes*. Chacune et chaque cellule dans le corps d'un organisme contiennent une copie de tous les différents chromosomes que l'organisme possède. Donc, chaque cellule contient une description complète de l'organisme entier.

Pendant le développement d'un organisme, l'ADN est utilisé comme un plan pour la production de nouvelles cellules dans un but spécifique. Quand un nouvel organisme est créé, son ADN est construit par combinaison des parties de l'ADN de ses parents. Les caractéristiques de ce nouvel organisme sont déterminées par l'ADN de ses parents. En 1859 Charles Darwin a publié son *Origine of Species*. Dans ce travail il explique sa théorie d'évolution, qui décrit comment l'apparition de la vie comme nous la connaissons, peut être expliqué comme un phénomène naturel.

Un aspect très important de la théorie de Darwin est qu'un organisme qui est bien adapté pour survivre, a une plus grande chance de perpétuer ses caractéristiques dans les générations futures, par rapport à un organisme qui est moins adapté. Par conséquent, les caractéristiques qui représentent un organisme convenable pour survivre vraisemblablement seront passées aux générations futures, alors que les caractéristiques qui constituent un organisme inférieur sont prédisposées à diminuer parce qu'il y a moins de chance qu'un organisme les procréera. Ce mécanisme est appelé "la survie du plus adapté" ou avec les propres mots de Darwin, la *sélection naturelle*.

Dans ce chemin, il y a un développement rapide de nouvelles existences, qui sont mieux adaptées à leurs alentours que leurs parents ou grands-parents. En outre, le mécanisme permet aux organismes de s'adapter aux changements dans l'environnement (Dawkins 1986, Dawkins 1989).

1.3.1 L'appel de l'évolution

Pourquoi utilise-t-on l'évolution comme une inspiration pour résoudre des problèmes de calcul? Pour les chercheurs dans le domaine du calcul évolutionnaire, les mécanismes de l'évolution paraissent très convenables pour la plus part des problèmes impliquant le calcul intensif, et ceci dans beaucoup de domaines. Plusieurs problèmes de calcul exigent une recherche à travers un grand nombre de possibilités pour les solutions. Un exemple est celui du problème de calcul dans l'ingénierie des protéines, dans lequel un algorithme cherche parmi un grand nombre de séquences d'aminoacides pour une protéine avec des propriétés spécifiées. Un autre exemple est celui de chercher une liste de règles ou d'équations qui peuvent prédire l'évolution d'un marché financier. De tels problèmes de recherche peuvent souvent bénéficier d'un usage effectif de la parallélisation, dans laquelle différentes possibilités sont explorées simultanément d'une façon efficace.

Plusieurs problèmes de calcul font intervenir des programmes informatiques pour être *adaptatifs*, afin de bien s'adapter continuer à un environnement évolutif. Ceci est le cas. par exemple, pour les problèmes de commande des robots, dans lesquels un robot est censé accomplir une tâche dans un environnement variable, à l'aide d'interfaces informatiques qui doivent s'adapter aux préférences des différents utilisateurs. D'autres problèmes ont besoin de programmes informatiques pour être innovateurs, et par suite construire quelque chose de vraiment nouveau et original, comme par exemple, un algorithme pour accomplir une tâche de calcul ou même une nouvelle découverte scientifique. Par ailleurs, beaucoup de problèmes de calcul exigent des solutions complexes qui sont difficiles à programmer à la main. Un

exemple frappant est celui de la création de l'intelligence artificielle. Cela faisait longtemps, les praticiens de l'IA croyaient qu'il serait possible de coder dans un programme, les règles qui engendreraient de l'intelligence; les systèmes experts étaient un des résultats de cet optimisme. De nos jours, plusieurs chercheurs de l'IA pensent que les "règles" qui régissent l'intelligence sont très complexes pour que les scientifiques puissent les coder à la main en mode "top-down". Par contre, ils pensent que le meilleur chemin vers l'intelligence est celui à travers un paradigme "bottom-up", dans lequel les êtres humains écrivent seulement de très simples règles, et les comportements complexes comme l'intelligence émergent d'une application massivement parallèle et de l'interaction de ces simples règles. Les règles de l'évolution, vues d'un haut niveau, sont remarquablement simples: les espèces évoluent par variation aléatoire (via mutation, recombinaison, et d'autres opérateurs), suivis par une sélection naturelle dans laquelle les plus adaptés tendent à survivre et à se reproduire, et donc propager leur matériel génétique aux générations futures. Ces simples règles sont considérées à l'origine, en grande partie, de l'extraordinaire variété et complexité qu'on voie dans la biosphère. [Mit 1998]

1.3.2 Les algorithmes génétiques

Les algorithmes génétiques sont des algorithmes de recherche stochastique inspirés des principes de la sélection naturelle et des génétiques naturelles. Une population de solutions candidates ou *individus*, est maintenue, et les individus rivalisent les uns avec les autres pour survivre. Une fois évalués, les individus les plus forts ont une plus grande chance à contribuer à la production de nouveaux individus (descendants) par rapport à ceux plus faibles, lesquels peuvent ne pas contribuer du tout. Les descendants sont produits à travers une recombinaison, par laquelle ils héritent des caractéristiques génétiques de chacun de leurs parents, et également à travers la mutation qui peut en conférer quelques-unes des caractéristiques certainement innovatrices. Une nouvelle population est alors générée à partir des individus crées récemment, et de certains individus appartenant à l'ancienne population. c'est la procédure de réinsertion. Dans les prochaines étapes de sélection, les descendants sont aussi amenés à rivaliser les uns avec les autres, et probablement avec leurs parents. L'amélioration dans la population survient comme une conséquence de la procédure de sélection répétée des meilleurs parents, qui sont par la suite plus rassurants à produire de bons descendants, et l'élimination conséquente des mauvais éléments. La solution finale du problème est représentée par le meilleur individu de la dernière population.

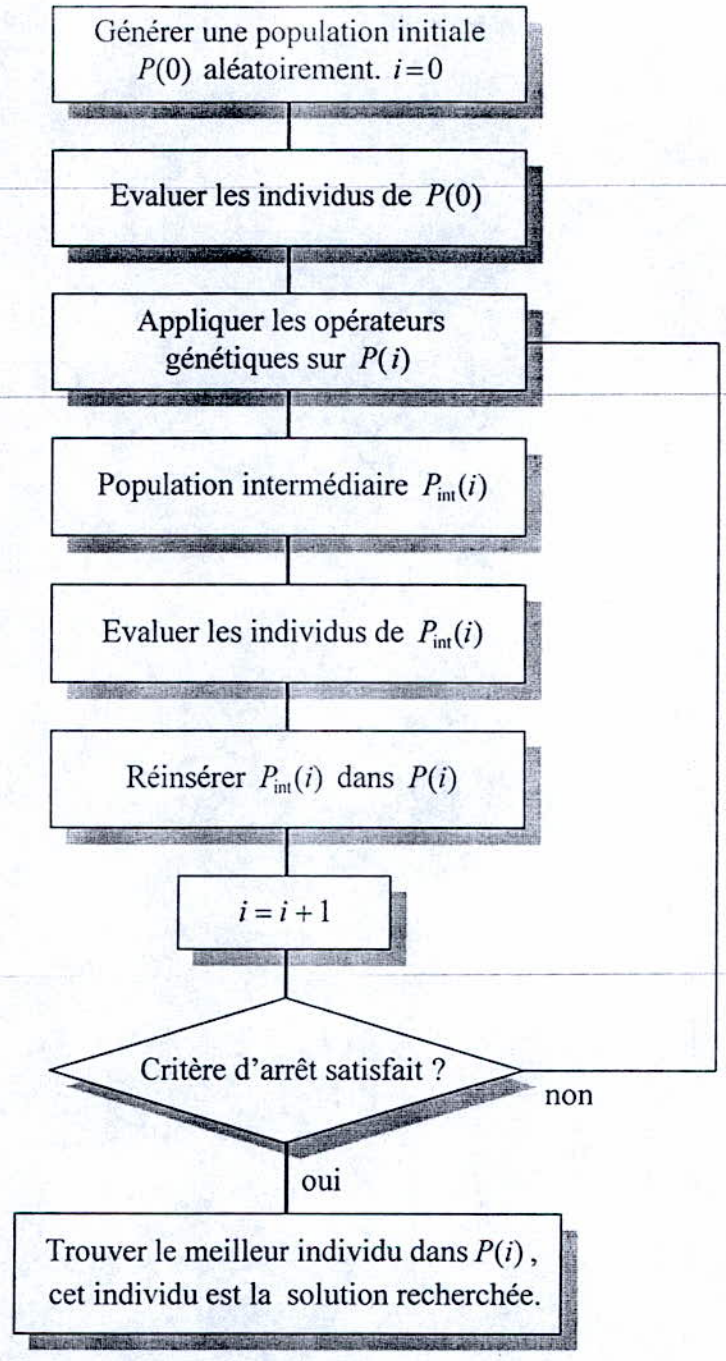


Figure 1.1: L'Algorithme Génétique de base.

1.3.3 Les opérateurs génétiques

La forme la plus simple d'un algorithme génétique implique trois types d'opérateurs: la sélection, le croisement, et la mutation.

La sélection Cet opérateur est un mécanisme de sélection d'individus dans la population d'après leurs *fitness* (adaptation), pour participer à la reproduction. Les individus les plus adaptés auront plus de chance d'être sélectionnés.

Le croisement Cet opérateur choisis aléatoirement un locus et échange les subséquences autour de ce locus entre deux individus, pour créer deux descendants. C'est une méthode d'exploitation des informations génétiques contenues dans un couple d'individus. Dans certaines littératures la reproduction est distinguée du croisement et est définie comme étant l'équivalent artificiel du principe de survie du plus adapté, avec cet opérateur un individu d'une population actuelle est simplement copié dans la prochaine population. Dans notre travail, nous considérons que le croisement et la reproduction ont le même sens.

La mutation Cet opérateur est l'équivalent artificiel de la mutation biologique, il change aléatoirement une portion du matériel génétique (déformation génétique) contenu dans un individu quelconque. C'est donc un outil de diversification génétique qui permet une exploitation plus large de l'espace de recherche.

1.3.4 Compromis entre l'exploration et l'exploitation

Un algorithme de recherche efficace doit manipuler deux techniques pour trouver l'optimum global: exploration de nouvelles régions inconnues dans l'espace de recherche, et l'exploitation des informations et des connaissances obtenues à partir des points visités. Cependant, ces deux techniques sont contradictoires, et un bon algorithme de recherche doit trouver un bon compromis.

Holland (1975) a montré que les AGs combinent à la fois l'exploration et l'exploitation en même temps par une façon optimale. Cela peut être théoriquement vrai, mais en pratique il y a forcément des problèmes, car Holland a fait certaines suppositions simplificatrices: population infinie, la fonction de fitness reflète parfaitement l'utilité d'une solution, et aucune interaction entre les gènes. Cependant la première supposition ne peut en aucun cas être satisfaite en pratique, et donc les AGs sont condamnés à accumuler des erreurs stochastiques. Un problème fréquemment rencontré dans la nature est celui du *mouvement génétique*, ce phénomène tout à fait naturel sera discuté plus en détail dans la section 1.3.6. La deuxième et la troisième supposition sont difficilement satisfaites dans les problèmes réels.

1.3.5 Les algorithmes génétiques et les méthodes de recherche traditionnelles

Certaines différences importantes existent entre les algorithmes génétiques et les algorithmes de recherche conventionnels tels que la méthode du grimpeur, le recuit simulé, et la méthode de recherche Tabou, la méthode de Newton et autres. On peut distinguer clairement les différences suivantes:

- Les AGs manipulent des versions codées des paramètres du problème au lieu de paramètres eux-mêmes.
- Alors que presque toutes les méthodes conventionnelles effectuent la recherche à partir d'un seul point, les AGs opèrent toujours sur une population entière de points (solutions possibles). Ceci contribue beaucoup à la robustesse des algorithmes génétiques et augmente les chances d'atteindre l'optimal global, et vice versa, et réduit le risque d'être freiné par un point local stationnaire.
- Les algorithmes génétiques standards n'utilisent pas d'information auxiliaire sur la valeur de la fonction objective comme les dérivées. Par conséquent. ils peuvent être appliqués à n'importe quel problème d'optimisation continu ou discret. La tâche la plus importante à faire est de spécifier une fonction de décodage qui soit significative.
- Les AGs utilisent des opérateurs de transition probabilistes alors que les autres méthodes conventionnelles utilisent pour une optimisation continue des opérateurs de transition déterministes. Bien qu'il existe des méthodes de recherche purement probabilistes, les AGs sont certainement bien meilleures car ils exploitent les connaissances implicites tirées des fitness des individus.

On constate que les AGs ont beaucoup de points forts, toutefois, il ne faut pas oublier les faiblesses de ces derniers. On donne ici quelques-uns des points faibles des AGs:

- Il n'existe toujours pas un cadre théorique des processus évolutionnaires. à l'opposé. par exemple, aux algorithmes statistiques.
- Il n'existe pas une garantie qu'une solution sera trouvée, même une seule qui soit mauvaise. Ceci, comme on va le voir plus loin, est un problème de choix des différents paramètres que font intervenir les AGs.
- Les AGs sont souvent lents en les comparant avec d'autres techniques conventionnelles, du fait que c'est la population entière des individus qui est examinée au lieu de solutions séparées.

1.4 Implémentation d'un algorithme génétique

La littérature sur les AGs décrit un grand nombre d'applications réussies. Cependant, ils existent aussi des cas dans lesquels les performances obtenues sont faibles. Etant donnée une application potentielle particulière, on ne peut affirmer rigoureusement si les AGs sont convenables pour cette application. Cependant, les chercheurs partagent les intuitions qu'en présence d'un des cas suivant: l'espace de recherche est large, et n'est pas parfaitement lisse et unimodal, ou bien n'est pas bien défini, ou la fonction de fitness est bruitée, et si la tâche n'exige pas de trouver l'optimum global (c'est-à-dire trouver rapidement une solution qui soit suffisamment bonne est acceptable), alors un AG aura une grande chance à être bien meilleur que les autres méthodes de recherche. Les performances d'un AG dépendront certainement des détails d'implémentation, c'est à dire du codage des solutions candidates, des opérateurs, et du choix des paramètres de l'AG et du critère d'adaptation des solutions (la fonction objective). Beaucoup de modifications faites pour le fonctionnement et les opérateurs des AGs ont permis à ces derniers d'améliorer leurs résultats, ces modifications seront discutées plus loin dans cette section.

1.4.1 Représentation et initialisation de la population

Les AGs opèrent sur un nombre potentiel de solutions, appelée population. Typiquement, une population est composée d'environs 30 à 100 individus, bien qu'une variante dite *micro-algorithme génétique* utilise de très petites populations, d'environ 10 individus, avec des restrictions sur les stratégies de reproduction et de remplacement, et ceci dans le but d'approcher le plus possible une exécution en temps réel.

Dans les AGs, les individus sont considérés à deux niveaux: le niveau *phénotypique* et le niveau *génotypique*. Le *phénotype* d'un individu est sa valeur dans le domaine dans lequel la fonction objective est définie, constituant un candidat pour le problème des variables de décision. Ceci est analogue aux caractéristiques d'un être vivant tels que la taille, le poids, le nombre d'organes, et la longueur des organes. Par ailleurs, le *génotype* d'un individu est une *représentation* de son phénotype à un niveau inférieur, analogue aux séquences génétiques contenues dans les chromosomes biologiques, qui est stockée par l'ordinateur et manipulée par l'AG. Le phénotype est donc *codé* dans le génotype, traditionnellement, sous forme de caractères de bits, mais plus généralement sous forme de n'importe quelle structure de données convenable. Les caractères ont été préférés pour simplicité et par analogie avec les

chromosomes naturels, mais d'autres structures telles que les matrices, les arbres, et éventuellement des listes de règles (Grefenstette, 1989), sont également manipulées.

La distinction entre le phénotype et sa représentation génotypique est importante et sera adoptée ici, car elle facilite l'extension de l'approche à d'autres classes de problèmes. Cependant, cette distinction n'est pas nécessairement rigide: un génotype et son phénotype correspondant peuvent être aussi les mêmes dans les cas où la relation entre eux est triviale. La représentation génotypique et la relation associée phénotype-génotype qui peut être une relation un-à-un ou plusieurs-à-un, sont des paramètres très importants pour le fonctionnement d'un AG. Les relations un-à-plusieurs et plusieurs-à-plusieurs peuvent apparaître en pratique, par exemple, comme une conséquence d'incertitude dans l'évaluation de la fonction objective.

La représentation la plus utilisée des chromosomes est celle des caractères binaires. Dans ce cas, chaque variable de décision dans l'ensemble des paramètres est codée sous forme d'une concaténation de bits (caractères prenant les valeurs 0 ou 1). La Figure 1.2 illustre cette représentation, dans ce cas l'individu est codé sur 8 bits et est formé de deux chromosomes, chaque chromosome représente une variable de décision du problème. L'utilisation du codage binaire en Gray a été préconisée comme une méthode pour éviter le biais représentationel caché dans la représentation binaire conventionnelle, du fait que la distance de Hamming entre deux valeurs adjacentes est constante. Caruana et Schaffer ont montré empiriquement que de larges distances de Hamming dans les relations représentationnelles entres les valeurs adjacentes, comme dans le cas de la représentation binaire standard, peut résulter d'un processus de recherche ayant été trompé ou incapable de trouver efficacement l'optimum global. Une autre approche, proposée par Schmitendorgf *et al.*, est l'utilisation de l'échelle logarithmique dans la conversion des valeurs binaires des chromosomes vers leurs valeurs phénotypiques réelles. Cette approche peut dans certains cas élargir l'espace de recherche avec le même nombre de bits.

D'autres stratégies alternatives de codage telles que les représentations réelles et entières, deviennent de plus en plus intéressantes, du fait qu'elles offrent certains avantages par rapport au codage binaire. En effet, l'efficacité de l'AG est augmentée puisqu'il n'y a aucun besoin de convertir les phénotypes des chromosomes avant chaque étape d'évaluation. Par ailleurs, l'espace mémoire nécessaire à l'implémentation est réduit du fait qu'on utilise la représentation en virgule flottante directement, sans oublier la liberté d'utiliser différents opérateurs génétiques.

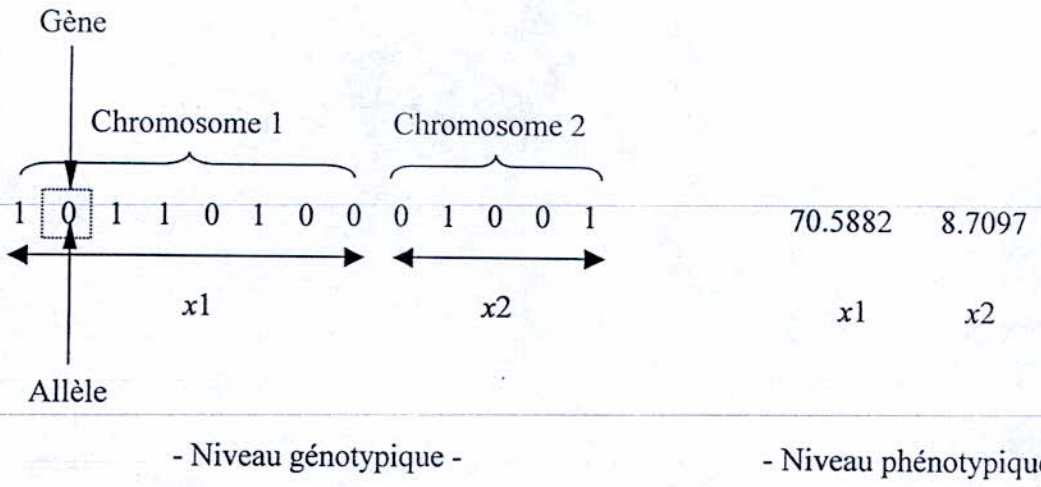


Figure 1.2: Représentation génotypique et phénotypique d'un individu dans le cas d'un codage binaire.

1.4.2 L'évaluation et le calcul de fitness

Les individus sont évalués via la fonction objective qui définint le problème. Par conséquent, ils leurs sont assignés un certain coût (en supposant un problème de minimisation) qui les distinguent. La *fitness* (ou adaptation) de chaque individu est alors dérivée de son coût, en prenant en considération tous les autres individus dans la population, un individu est d'autant plus adapté que sa fitness est grande. Les AGs sont donc d'une nature de maximisation, ils transforment le problème posé d'un problème de minimisation (coût) à un problème de maximisation (fitness).

La distinction entre coût et adaptation est très importante. En fait, le coût est quelque chose d'intrinsèque au problème et, par conséquent, peut ne pas être changé à volonté. De plus, le coût peut supposer généralement des valeurs établies suivant n'importe quel ordre. Par ailleurs, la relation coût-fitness concerne le processus de sélection, elle devrait être (largement) une relation monotone sur le sous-ensemble des réels non-négatifs.

Pour généraliser, la fonction de fitness *normalisée* d'un parent donné est définie ici comme le rapport entre le nombre d'enfants qu'il est prévu de produire à la génération suivante, et le nombre total des individus de la population. Puisqu'elle suppose des valeurs comprises entre 0 et 1, la fonction de fitness normalisée est parfois vue comme une probabilité. Cependant, une telle interprétation n'est valide que dans un contexte d'implémentation particulière de la procédure de sélection, tel que le *sampling* avec remplacement, comme on va le voir plus loin.

Un autre concept utile est celui de la fitness *relative*, c'est à dire, la fitness normalisée d'un individu par la moyenne des fitness de la population. La fitness relative fournit une indication directe de combien les individus courants sont plus bons ou plus mauvais.

Alors que le classement des individus dans la population, établi suivant leurs coûts, devrait être conservé, la relation coût-fitness peut conserver l'échelle comme elle ne le peut pas. Il existe principalement deux types de stratégies pour le calcul de la fonction de fitness:

Le scaling La fitness est calculée comme une fonction (probablement non-linéaire) des valeurs du coût. Cependant, il faut prendre soin d'assurer (ou forcer) la non-négativité de la fonction de fitness pour tous les individus, en donnant au meilleur individu dans la population un avantage contrôlé sur les autres.

Le ranking La relation coût-fitness est établie en classant la population par coût, et par suite assigner des valeurs de fitness résolues aux individus d'après leurs placements (ou rangs) dans la population plutôt que d'après leurs performances. Les renseignements de l'échelle sont donc délibérément abandonnés.

Le scaling est l'approche la plus traditionnelle. Le vecteur contenant les fitness des individus est calculé comme une fonction monotone du coût, compensé par un certain montant pour assurer la non-négativité des valeurs de la fitness, et ensuite linéairement mis en échelle. La première difficulté survient à ce stade: puisque le scaling a pour but de conserver la performance relative entre les individus, la transformation initiale ou la compensation subséquente peut affecter gravement la fitness relative finalement assignée à chaque individu. En désignant par f et F respectivement les fonctions objective et de fitness, le scaling peut être décrit comme suit:

$$F(x) = af(x) + b \tag{1.1}$$

où a est un facteur d'échelle et b est utilisé pour assurer la non-négativité des valeurs de fitness.

Avec le scaling, un individu beaucoup plus fort que tous les autres peut être assigné une très grande fitness relative et, à travers la sélection, peut encore dominer rapidement la population conduisant probablement la recherche à des solutions sous-optimales. Réciproquement, l'avantage du meilleur individu sur le reste de la population va être minimal si la plupart des individus agissent plus ou moins à égalité, la recherche dégénérera dans une évolution sans but.

Comme aucune des deux situations précédentes n'est désirable, la fitness relative calculée par scaling du meilleur individu dans la population doit être contrôlée dans un certain sens. Par exemple, le sigma-scaling, calcule le montant de compensation par la déviation moyenne et standard des valeurs du vecteur de fitness dans la population à chaque génération, afin que la fitness relative du meilleur individu est établie plus raisonnablement (Hancock, 1994).

Le ranking diminue ces mêmes difficultés en éliminant toute sensibilité au scaling dans lequel le problème est formulé. Dans ce cas, les individus leurs sont assignés des valeurs de fitness d'après leurs rangs dans la population plutôt que d'après leurs fitness elles-mêmes, et puisque le meilleur individu dans la population lui est toujours assigné la même fitness relative, les super-individus prétendus, ne peuvent jamais se reproduire excessivement. De la même façon, quand tous les individus agissent presque de la même manière, le meilleur individu est toujours préféré clairement au reste.

Le calcul de fitness à l'aide du ranking est caractérisé par le choix de la transformation rang-fitness, qui est choisie habituellement linéaire ou exponentielle. Pour une population de taille N, en classant le meilleur individu au niveau 0 et le plus mauvais au niveau $N-1$, et en désignant le rang par r et la fitness relative par $F = F(r)$, ces relations peuvent être formulées comme suit:

Linéaire

$$F(r) = s - (s-1) \cdot \frac{2r}{N-1} \tag{1.2}$$

où s, $1 < s \leq 2$, est la fitness relative désirée pour le meilleur individu, cette valeur est appelée *pression sélective*. La borne supérieure de s survient car la fonction de fitness doit être non-négative pour tous les individus, en maintenant $\sum_{i=0}^{N-1} f(i) = N$.

Exponentiel

$$F(r) = \rho^r . s \tag{1.3}$$

où $s > 1$ est la fitness relative désirée pour le meilleur individu et ρ est tel que $\sum_{i=0}^{N-1} \rho^i = N/s$. Comme il n'y a pas de borne supérieure sur s, la relation exponentielle est en quelque sorte plus flexible que la relation linéaire.

Pour $1 < s \leq 2$, la différence principale entre le ranking linaire et le ranking exponentiel est que la transformation exponentielle ne pénalise pas fortement les plus mauvais individus. par rapport à ceux plus faibles. Par conséquent, la transformation exponentielle contribue généralement à une recherche plus diverse.

1.4.3 La sélection

La sélection ou réplication, est le processus de choix des individus pour participer à la production de descendants, proportionnellement à leur fitness. Cependant, comme le nombre des répliqués doit être un nombre entier, le nombre obtenu de descendants a besoin généralement d'être plus grand ou plus petit que la valeur désirée, en introduisant ce qui est appelée *l'erreur de sélection.*

Pour éviter des erreurs systématiques de sélection, la sélection dans les AGs est exécutée habituellement d'une façon stochastique. Le nombre attendu de descendants d'un individu doit alors correspondre exactement à sa fitness, alors que les erreurs *stochastiques* de sélection devraient rester aussi petites que possible. Baker (1987) a accompli ces deux buts avec un algorithme d'échantillonnage appelé *Stochastic Universal Sampling* (SUS). SUS a été proposé comme une alternative à la sélection de *Roulette Wheel Selection* (Roulette de Casino) de Goldberg (Goldberg, 1989) (RWS, ou sampling avec remplacement) qui permet à beaucoup d'erreurs de sélection de se produire. RWS consiste en une séquence d'essais indépendants de sélection où la probabilité de sélection de chaque individu dans chaque essai (comme déterminée par le nombre relatif des fentes sur une roulette de casino, voir Figure 1.3) reste constante et égale à sa fitness relative.

La sélection avec SUS peut aussi être visualisée comme le résultat du lancement d'une roulette de casino ayant des secteurs proportionnels en largeur aux fitness des individus dans la population, mais avec plusieurs pointeurs, situés à des distances égales les uns par rapport aux autres (Figure 1.4). Une fois la roulette arrêtée, le nombre de pointeurs d'arrêt sur chaque secteur doit être soit le minimum ou le maximum du nombre désiré de descendants correspondants, donc garantir le minimum de déviations autour de la valeur désirée ou étendue. Les répliqués obtenus devraient être retirés avant que l'algorithme continue avec la recombinaison et la mutation.

Dans le cas extrême où un seul individu sera sélectionné, la sélection avec SUS est clairement équivalente à RWS. Dans le cas de plusieurs individus, l'avantage de SUS, sur RWS en termes d'extension augmente avec le nombre d'individus à générer. Il n'y a habituellement aucune raison d'utiliser RWS dans ces circonstances.

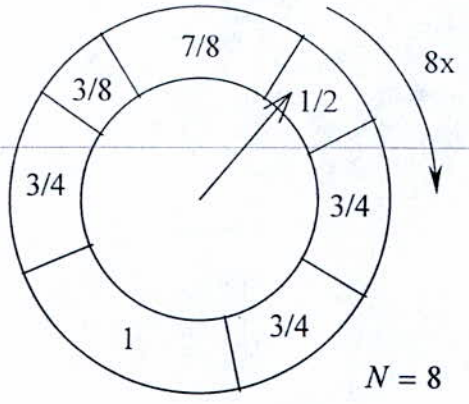


Figure 1.3: Roulette Wheel Selection.

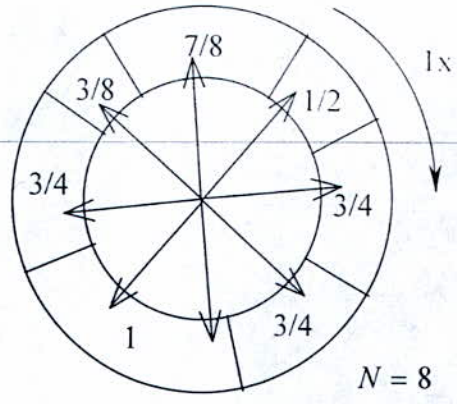


Figure 1.4: Stochastic Universal Sampling.

Une approche nettement différente est connue sous le nom de *sélection par tournoi* (Hancock, 1994). Dans sa forme la plus simple, des paires d'individus sont tirées au hasard de la population, et les individus dans chaque paire sont comparés l'un avec l'autre. Le meilleur des deux individus dans chaque paire est déclaré vainqueur de ce tournoi et est sélectionné.

La sélection par tournoi est particulièrement commode quand il est plus facile de comparer les individus que de calculer leurs performances absolues, comme c'est souvent le cas dans la programmation génétique. Cependant, comme la sélection de RWS, la sélection avec des tournois aléatoires peut mener à de grandes erreurs de sélection. Lors de la sélection de plusieurs individus, il devrait être possible de classer la population étant donné un nombre quelconque de paires de comparaison (Uppuluri, 1989), ou tournois, et d'utiliser SUS pour sélectionner les individus par rapport à ce classement. Le processus de sélection est d'autant plus précis que le nombre d'individus comparés (dimension du tournoi) est plus grand.

La sélection par tournoi est semblable au ranking puisqu'elle ne compte pas sur les informations apportées par le scaling. En faisant varier la dimension et le nombre des tournois, l'adaptation relative effective du meilleur individu peut être contrôlée, bien que d'une façon moins flexible qu'avec le ranking. Par ailleurs, en posant la probabilité qu'un individu emporte un tournoi, sachant qu'il est sorti gagnant de ce tournoi, inférieur à 1 permet un contrôle plus doux (diminué) de la fitness relative attendue du meilleur individu dans la population. La prise en compte d'une probabilité dans les tournois selon la fitness relative des deux adversaires, résulte d'une baisse de différentiel de la fitness quand la population converge.

1.4.4 La recombinaison et la mutation

La sélection seule, répétée dans la même population ne produirait rien de nouveau par rapport à la population initiale. Pour qu'une amélioration puisse apparaître, quelques nouveautés doivent être introduites dans la population entre les étapes de sélection. Les opérateurs génétiques modifient les parents sélectionnés en manipulant leurs génotypes, ils peuvent être divisés en deux catégories principales:

La recombinaison provoque l'échange d'informations génétiques entre les individus, dans des paires ou de larges groupes. Les caractéristiques de deux parents ou plus peuvent donc être héritées par le même descendant.

La mutation provoque des changements au niveau des génotypes des individuels suivant certaines règles probabilistes. Habituellement, seulement une petite partie du chromosome est changée par mutation, causant l'héritage de la plupart des caractéristiques des parents. Une mutation "déterministe" peut aussi être implémentée, communément connue sous le nom de *réparation.*

Les opérateurs de mutation sont généralement les plus simples des deux catégories, mais certainement pas les moins importants. L'évolution à travers la mutation et la sélection sans recombinaison n'est pas seulement possible mais peut être aussi très effective. Les stratégies d'évolution, par exemple, ont été originairement conçues comme des algorithmes de mutation-sélection. La recombinaison, de l'autre côté, contribue différemment à la recherche. En apportant dans un seul individu les caractéristiques génétiques de deux parents ou plus, il est possible d'accélérer considérablement le processus de recherche, surtout dans le cas où les variables de décision sont vaguement associées. Pour cette raison, la recombinaison des individus sélectionnés est souvent accomplie avec une grande probabilité, typiquement entre 0.6 et 0.9.

L'implémentation de la recombinaison et de la mutation dépend nécessairement de la représentation génotypique prise en compte. En outre, les performances des mêmes opérateurs dépendent généralement de la classe du problème considéré. Un grand travail de recherche a été effectué pour identifier les meilleurs opérateurs génétiques pour plusieurs classes de problèmes. Le développement de l'opérateur de recombinaison de Bord pour le problème du voyageur de commerce (Whitley *et al.*, 1991) en est un bon exemple.

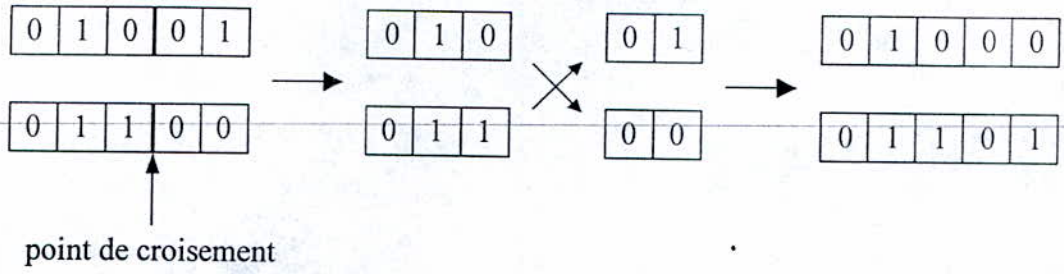


Figure 1.5: Croisement en un seul point.

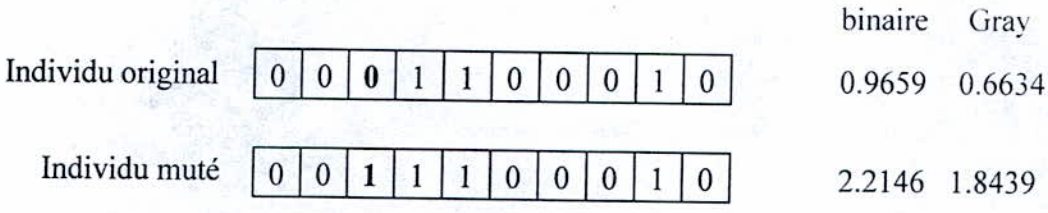


Figure 1.6: Mutation binaire.

Un opérateur de recombinaison typique pour un codage binaire ou avec d'autres caractères des chromosomes est le croisement en un seul point, par lequel deux individus échangent une portion (droite ou gauche) de leurs chromosomes pour créer un descendant. comme illustré dans la Figure 1.5. Le point de croisement est sélectionné au hasard. D'autres opérateurs de recombinaison communément utilisés avec des caractères binaires sont:

Croisement en deux points Deux points de croisement sont sélectionnés au lieu d'un seul (Booker, 1987).

Croisement uniforme chaque bit est échangé indépendamment, avec une probabilité donnée, (Syswerda, 1989).

Shuffle crossover Les chromosomes sont mélangés avant que le croisement en un seul point n'est appliqué, et par suite réorganisés (Caruana *et al.*, 1989).

Reduced-surrogate crossover Les bits non-identiques dans les chromosomes sont premièrement identifiés, et un des types du croisement précités plus haut est appliqué à la chaîne de caractères ainsi obtenue (Booker, 1987). Cela à pour effet de garantir la production de nouveaux individus différents de leurs parents.

Dans le cas d'un codage réel, l'opérateur de recombinaison est appliqué différemment. En effet, les descendants ne sont pas crées par échange de matériel génétique proprement dit,

mais plutôt via des opérations mathématiques appliquées sur les valeurs réelles des chromosomes. On distingue dans ce cas les différents opérateurs suivants:

Recombinaison intermédiaire Dans ce cas les valeurs des descendants sont choisies autour et entre les valeurs des phénotypes des parents. Les descendants sont produits à travers la règle suivante:

$$\text{Descendants} = \text{Parent1} + \alpha(\text{Parent2} - \text{Parent1}) \tag{1.4}$$

La variable α est un paramètre d'échelle choisi aléatoirement dans un certain intervalle, typiquement [-0.25, 1.25] et Parent1 et Parent2 sont les chromosomes des parents. Chaque variable dans la descendance est le résultat de la combinaison des variables suivant l'expression précédente avec un nouvel Alpha, choisit pour chaque paire de gènes des parents. En termes géométriques, la recombinaison intermédiaire est capable de produire de nouvelles variables dans un hypercube légérement plus large que celui définit par les parents, comme il est montré dans la Figure 1.7.a.

Recombinaison en ligne La recombinaison en ligne est similaire à la recombinaison intermédiaire, à l'exception du choix unique de la valeur de Alpha utilisée pour la recombinaison. La Figure 1.7.b montre comment la recombinaison en ligne peut générer n'importe quel point sur la ligne définie par les parents pour une recombinaison à deux variables.

Le rôle de la mutation est souvent vu comme une garantie pour que la probabilité de recherche d'un caractère quelconque ne soit jamais nulle d'une part, et d'autre part comme action d'un organe de recherche de bonnes matières génétiques qui peuvent être perdues à travers l'action de la sélection et la recombinaison.

L'effet de la mutation sur un caractère binaire est illustré dans la Figure 1.6 dans le cas d'un chromosome de 10 bits représentant un décodage réel dans l'intervalle [0 10] en utilisant le codage standard et le codage en Gray. Ici, la mutation binaire provoque l'inversion de la valeur du troisième bit. Etant donné l'application de l'opérateur de mutation uniformément sur une population entière, il est possible qu'un caractère binaire donné soit muté en plus d'un point.

Avec les représentations non-binaires, la mutation est accomplie soit en perturbant les valeurs des gènes ou bien à travers une sélection aléatoire de nouvelles valeurs dans la gamme permise. Wright, Janikow et Michalewicz ont démontré comment les AGs codés-réel peuvent prendre l'avantage des taux de mutation élevés par rapport aux AGs codés-binaire, en

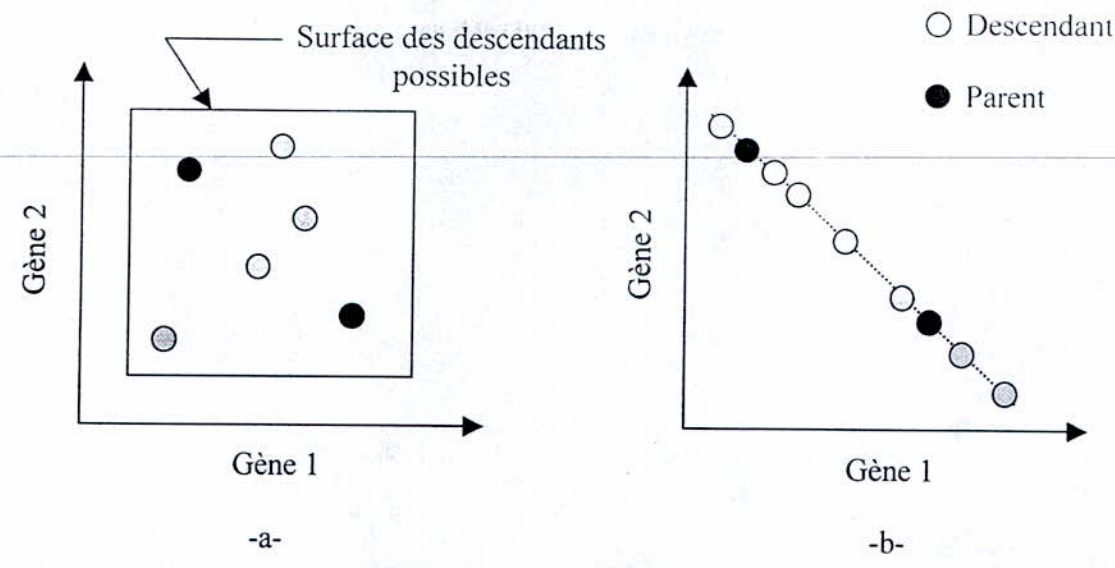


Figure 1.7: Recombinaison intermédiaire (a) et en ligne (b).

augmentant le niveau de l'exploration possible de l'espace de recherche sans affecter de façon défavorable les caractéristiques de la convergence. En effet, Tate et Smith pensent que pour les codages plus complexes que le codage binaire, des taux de mutation élevés peuvent être à la fois désirables et nécessaires et ont montré comment, pour un problème d'optimisation combinatoire complexe, des taux de mutation élevés avec un codage non-binaire ont apporté considérablement de meilleures solutions que l'approche normale.

Plusieurs variations sur l'opérateur de mutation ont été proposées. Par exemple, biaiser la mutation vers les individus qui ont les plus faibles valeurs de fitness, dans le but d'augmenter l'exploration dans la recherche sans perdre les informations des individus les plus adaptés, ou bien paramétrer la mutation, d'une façon à ce que le taux de mutation diminue avec la convergence de la population. Mühlenbein a introduit un opérateur de mutation pour l'AG codé-réel qui utilise un terme non-linéaire pour la distribution de la gamme de mutation appliquée aux valeurs du gène.

1.4.5 La réinsertion

Dès qu'une nouvelle population est générée par sélection et recombinaison d'individus de la population précédente, la fitness des individus dans la nouvelle population peut être déterminée. Si moins d'individus sont produits par recombinaison par rapport à la taille de la population originale, alors la différence fractionnelle entre la nouvelle et l'ancienne taille de la population est appelée *generation gap*.

L'insertion des descendants dans la population peut être effectuée par plusieurs manières. Dans le cas le plus simple, par exemple, la population entière des parents est remplacée par leurs descendants inconditionnellement. En dehors de la supposition d'une population de taille constante, cette stratégie de remplacement montre des similarités dans beaucoup d'espèces naturelles où la reproduction prend place de façon saisonnière, et les parents meurent avant que leurs enfants ne soient nés. Ceci est connu comme le remplacement *générationnel*.

Dans la nature, il y a aussi un intérêt dans les AGs où les descendants ont une chance de rivaliser avec au moins quelques-uns de leurs parents. Dans le cas de l'AG *steady-state* ou l'AG *incrémental*, un nombre minime d'enfants (typiquement un ou deux) est produit à travers recombinaison et mutation. Ces descendants sont alors évalués, et probablement réinsérés dans la population, en remplaçant :

- Des membres aléatoirement choisis de la population des parents.
- Les membres les plus anciens de la population des parents.
- Leurs propres parents.
- Les membres les moins adaptés de la population des parents.

La réinsertion actuelle peut se faire :

- Inconditionnellement.
- Seulement si les enfants sont plus adaptés que les individus qu'ils vont les remplacer.
- D'une façon probabiliste, suivant l'adaptation des descendants par rapport à celles des individus qu'ils vont les remplacer.

Si quelques individus ne montrent pas une habilité à se reproduire plus loin, la réinsertion aura finalement le même effet que la sélection. La pression sélective totale ou le biais sur le meilleur individu, imposée sur la population n'est pas déterminée seulement par la stratégie du calcul de fitness, mais elle est également influencée par l'instant et la façon d'application de la procédure de réinsertion. En particulier, remplacer toujours les individus les moins adaptés dans la population augmente fortement l'effectif, la différence des fitness entre les individus les plus forts et les plus faibles dans la population. Cela est dû, en plus d'être moins appropriés pour la sélection, au fait que les individus les plus faibles tendent à mourir plus rapidement, donc auront moins de chance pour participer aux essais de sélection que ceux

plus forts. Les stratégies de réinsertion qui garantissent la conservation du meilleur individu sont dites *élitistes*.

L'utilisation de différentes combinaisons de sélection et de stratégies de réinsertion est largement rapportée à la littérature des AGs. Par exemple, les AGs steady-state sont supposés fournir de meilleurs résultats que ceux générationnels, prétendument parce qu'ils exploitent les bons individus, récemment produits, d'une façon plus rapide par rapport aux autres cas. Comme il a été signalé par Harvey (1992), cela se traduit par une augmentation de la pression sélective, même si des remplacements aléatoires sont utilisés. Le remplacement basé sur la fitness fournirait même de plus grande pression sélective.

La pression sélective supplémentaire fournie par la reproduction steady-state en prenant en compte sa contrepartie générationnelle peut être la simple raison pour laquelle l'AG steady-state paraît donner de meilleures performances dans certains cas. Une pression sélective suffisante est spécialement importante si de grandes erreurs de sélection peuvent se produire. Le travail de Syswerda (1991) ne révèle aucune différence fondamentale entre les AGs générationnels et les AGs steady-state avec remplacement aléatoire inconditionnel. L'augmentation de la pression sélective dans un AG générationnel, d'une façon explicite à travers le calcul de fitness peut produire des résultats semblables implicites à travers la stratégie de remplacement.

1.4.6 Le mouvement génétique et le sharing

À long terme, une population finie qui subit une sélection stochastique tendra à évoluer vers une petite région de l'espace de recherche, même si d'autres régions de hautes adaptations similaires existent (par exemple, il peut y avoir plus qu'un optimum local). Ce phénomène certainement indésirable, connu sous le non du *mouvement génétique*, est dû à la composition d'erreurs de sélection sur les générations, et peut être observé dans les deux domaines d'évolution naturel et artificiel.

Un souci pour l'estimation de la précision est, par conséquent, justifié. En effet, plus la population est petite, plus les erreurs de sélection et le mouvement génétique sont vulnérables, le plus important c'est d'assurer un mécanisme d'estimation précis. C'est pourquoi une préférence heuristique pour l'AG générationnel avec sélection SUS était exprimée antérieurement. Malheureusement, bien qu'elle donne une précision optimale, la sélection SUS ne peut par elle-même éliminer le mouvement génétique. Aussi long que les populations doivent rester finies, les erreurs de sélection sont inévitables. [Fon 1995]

Dans ces circonstances, c'est l'accumulation des erreurs de sélection qui doivent être contrôlée afin de prévenir un mouvement génétique. Des taux de mutation plus grands peuvent accomplir ceci dans une certaine mesure en introduisant systématiquement une diversité dans la population, mais comme discuté plus haut, les taux de mutation ne peuvent pas être rendus arbitrairement grands. De la même façon, introduire aléatoirement un petit pourcentage d'immigrés dans la population à chaque génération (Grefenstette, 1992) rend l'AG plus apte à retrouver les informations perdues à travers la sélection et, donc, du mouvement génétique. Une alternative supplémentaire consiste à pénaliser la production d'individus semblable les uns aux autres, par exemple, en forçant les individus à remplacer les parents leurs plus semblables, cette technique est appelée crowding (Goldberg, 1989).

Une technique plus compliquée, connue sous le non du *sharing* (Goldberg et Richardson, 1987), modélise une compétition individuelle pour des ressources finies dans un environnement fermé. Les individus semblables sont affaiblis par l'obligation de partager les mêmes ressources entre eux, alors que d'autres individus différents retiennent leurs fitness originales. Comme une conséquence de l'application du sharing, le processus de sélection reçoit une réaction négative décourageant la reproduction d'individus toujours abondants, et entraînant la population à se regrouper autour de différents optima locaux dans l'espace de recherche. De tels groupes représentent des régions favorables dans l'espace de recherche, appelés *niches*. Le sharing a pour effet de modifier les fitness des individus de la population avant l'étape de sélection, les nouvelles valeurs de fitness F_i seront calculées de la façon suivante:

$$F_i = \frac{F_i}{m_i};\ m_i = \sum_{j=1}^{N} S(d(x_i, x_j)) \tag{1.5}$$

avec

$$S(d) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{share}}\right)^{\alpha} & \text{si } d < \sigma_{share} \\ 0 \quad \text{si } d > \sigma_{share} \end{cases} \tag{1.6}$$

où σ_{share} est le paramètre qui représente l'indice de ressemblance entre les individus, il permet de délimiter le voisinage d'un point dans l'espace de recherche, α est un paramètre qui détermine l'efficacité du sharing, ainsi pour $\alpha < 1$ on pénalise les groupes très agglomérés.

La difficulté principale avec le sharing concerne la décision à prendre sur la valeur de σ_{share}, pour lequel les individus commencent à diminuer réciproquement leurs fitness. Une distance adéquate pour les mesures de distance peut être définie dans l'espace génotypique ou bien dans l'espace phénotypique, néanmoins la valeur du seuil approprié de σ_{share} était estimée seulement sur la base de suppositions concernant l'emplacement et le nombre préalable d'optima locaux dans l'espace de recherche, qui sont généralement inconnus (Deb et Goldberg, 1989). [Fon 1995]

1.4.7 La viabilité et la restriction du mating

Si la population est divisée en groupes pour peupler différentes niches, la capacité de recombinaison pour produire des descendants mieux adaptés à partir des individus dans de tels groupes peut diminuer. En effet, ceci mène souvent à la création de descendants à l'extérieur des régions favorables (Deb et Goldberg, 1989). De tels individus non adaptés sont généralement dits *lethals*.

Comme dans la nature où l'accouplement a tendance à se produire seulement entre des individus raisonnablement semblables (ceux de la même espèce), un schéma de restriction pour le mating (accouplement) peut être ajouté à l'AG dans le but de réduire la formation de lethals, comme proposé par Deb et Goldberg (1989). Encore une fois, une mesure de distance entre les individus doit être définie, le paramètre qui régit la reproduction noté σ_{mate}, est utilisé pour restreindre la procédure de croisement. Le deuxième individu choisi pour participer à un croisement doit être situé à une distance inférieure ou égale à σ_{mate}, sinon, si aucun individu dans la population ne satisfait pas cette condition alors un individu quelconque est choisi aléatoirement. Bien que le mating aléatoire améliore le fonctionnement de l'AG, la restriction du mating, basée sur une mesure de distance ne résolue pas entièrement le problème de sélection du partenaire. En particulier, le choix du degré de ressemblance entre individus pour pouvoir se croiser est difficile à faire exactement que dans le cas du sharing (en pratique, le paramètre du sharing et du mating sont généralement assignés la même valeur).

Le succès de la recombinaison ne dépend pas seulement de l'état génétique des parents qui participent au croisement, mais aussi, et peut-être le plus, sur l'interaction entre le codage des chromosomes, l'opérateur de recombinaison, et la structure de la perspective du coût. Manderick *et al.* (1991) ont montré que, pour certains types de perspectives du coût, la

performance des différents opérateurs de recombinaison peut être apparentée aux propriétés statistiques de la surface du coût elle-même. La question qui se pose sur la façon par laquelle la recombinaison et/ou le codage des chromosomes peuvent être évolués concurremment avec la recherche est une question naturelle, pour laquelle il n'existe aujourd'hui aucunes réponses bien définies.

1.4.8 Le taux de mutation et la pression sélective [Fon 1995]

Le choix "optimal" des taux de mutation et de croisement était probablement l'un des premiers soucis apparus par l'utilisation des AGs dans les problèmes d'optimisation. Des études initiales, d'une nature expérimentale, sont faites dans le but de trouver un bon appui pour le choix de ces paramètres pour plusieurs fonctions de test. La plupart de ces études sont concentrées sur les chromosomes binaires où le taux de mutation est clairement défini comme la probabilité que chaque bit soit indépendamment inversé par mutation. Néanmoins, quelques résultats s'appliquent également aux chromosomes plus généraux.

Les larges taux de mutation proposés par quelques auteurs ont élevé une controverse sur le rôle effectif de la mutation dans la recherche génétique, qui est considérée comme secondaire par d'autres. De nos jours, la mutation est mieux comprise, et reconnue être un opérateur de recherche important et actif. Le choix du taux de mutation a été montré dépendant de plusieurs autres aspects de l'AG, à savoir:

La pression sélective Dans l'absence de recombinaison, la probabilité que des individus ne subissent pas une mutation devrait être tel qu'au moins une copie exacte du meilleur individu dans chaque génération survivra après la mort de ses parents. Ce choix, suggéré par le travail de Schuster, maximise l'exploration de l'espace de recherche par la population en garantissant l'exploitation du meilleur individu. En fait, des taux de mutation élevés entraînent rapidement une dégénération de la recherche vers une évolution aléatoire, car la sélection ne serait plus capable de se rétablir des erreurs génétiques introduites par la mutation. Ce changement brusque dans le comportement qualitatif de l'AG avec l'augmentation du taux de mutation est appelé *catastrophe de l'erreur*. Des taux de mutation inférieurs au *seuil de l'erreur* correspondant préservent le meilleur individu, mais aussi diminuent la diversité de la recherche.

La longueur du chromosome Les considérations précitées soulèvent une question importante dans le cas des AGs élitistes, par exemple. Puisque le meilleur individu n'est jamais perdu dans de telles situations, il paraîtrait que les descendants pourraient être crées

arbitrairement différents de leurs parents. Ceci n'est clairement pas le cas, comme ce serait équivalent à une recherche aléatoire pure. Donc, quel est le degré d'influence de la mutation sur la structure du chromosome?

Les taux de mutation pour les chromosomes composés de caractères sont généralement posés comme l'inverse de la longueur du chromosome, et ceci a été prouvé théoriquement comme optimal (avec certaines restrictions) (Bäck, 1993), et jugé expérimentalement comme donnant de bonnes performances sur une grande gamme de problèmes.

La distance de l'optimum Comme il est noté par Bäck (1993), plus l'individu est loin de l'optimum, plus le taux de mutation doit être élevé dans le but de minimiser la distance moyenne du descendant correspondant à cet optimum. Cela a la forte implication que les taux de mutation doivent, en principe, ne pas rester constants durant une exécution d'un AG. Bien qu'il ait été constaté qu'une variation du taux de mutation avec le temps peut accélérer l'optimisation, une telle pratique ne paraît pas être bien établie. Comme Bäck l'a indiqué, le taux de mutation standard $1/l$, où l est la longueur du chromosome, est assez suffisant pour ce qui pourrait être accompli avec des taux de mutations optimaux et variant dans le temps.

Dans la pratique, les taux de mutation déterminés à partir des directives précitées sont généralement acceptables.

1.4.9 Implémentation parallèle et populations structurées

Bien qu'ils aient tendance à être exigeants du point de vue calcul, les algorithmes génétiques sont aussi très répondants à l'implémentation parallèle. En effet, l'étape de l'évaluation dans un AG générationnel peut être accomplie d'une façon parallèle. Supposant que l'AG lui-même est lancé sur un seul processeur, les descendants produits à chaque génération peuvent être simplement mis en dehors pour l'évaluation sur d'autres processeurs, comme illustré dans la Figure 1.8. Cette approche est particulièrement appropriée quand le montant de calcul exigé pour chaque évaluation est élevé comparé avec celui de l'AG lui-même et avec la quantité de données qui ont besoin d'être communiquées entre les processeurs par une seule évaluation. Les AGs croissants peuvent être également parallélisés dans ce sens, mais le nombre de descendants crées à un instant donné devrait être égale au nombre de processeurs disponibles.

Jusqu'ici, il a été supposé implicitement que tous les individus prennent le même montant de temps de calcul pour être évalués. Quand ce n'est pas le cas, une implémentation synchrone (générationnelle ou incrémentale) peut ne pas utiliser les processeurs disponibles

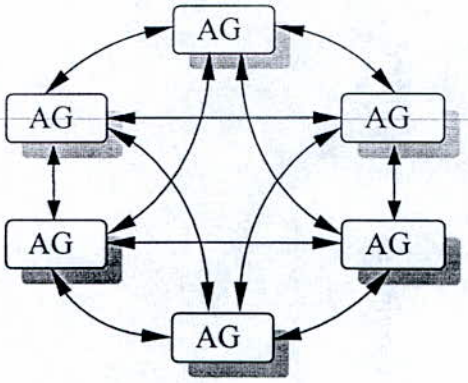


Figure 1.8: Le modèle en île.

simultanément. Cependant, les AGs peuvent être aussi implémentés d'une façon asynchrone: après l'évaluation de la population entière, un nouvel individu est sélectionné pour l'évaluation chaque fois qu'un processeur termine l'évaluation d'un autre antérieur, lequel est alors réinséré dans la population. En tant qu'effet latéral, cette approche est préférable en réalité pour les individus les moins difficiles à évaluer, puisque ceux-ci tendent à participer souvent à la sélection. Cela peut ou non être désirable.

Les approches précédentes sont centrées autour du "fermier de l'AG", et ne paraissent pas comme des modèles très exacts de l'évolution naturelle. Dans de tels AGs, les individus sont autorisés à se croiser, mais aussi forcés à rivaliser, avec chaque autre individu dans la population. Dans la nature, cependant, l'isolement géographique implique que les individus dans un continent, par exemple, n'ont aucune chance pour interagir réciproquement avec les Individus dans un autre continent, probablement avec quelques exceptions. Les populations sont alors *structurées*, et l'évolution est conditionnée par cette structure.

Dans le modèle *en îles*, plusieurs AGs sont lancés indépendamment, par exemple, sur différents processeurs. De plus, ces AGs échangent périodiquement de petites fractions de leurs populations, en implémentant ce qui appelé *la migration* (voir Figure 1.8). Dans le modèle *de diffusion* ou le modèle *cellulaire*, la population est distribuée sur une grille, et chaque individu réagit réciproquement avec son voisin immédiat (voir Figure 1.9) afin d'éviter la complexité et le fort coût des communications.

Le paradigme d'isolement géographique rend même la parallèlisation plus attirante. à raison qu'il diminue les communications entres processeurs, toutefois, il n'exige pas une implémentation sur une architecture parallèle. Par ailleurs, les algorithmes génétiques basés

sur les populations structurées exposent de mieux des propriétés de recherche globale que leurs équivalents, qui par eux même justifient leur implémentation séquentielle.

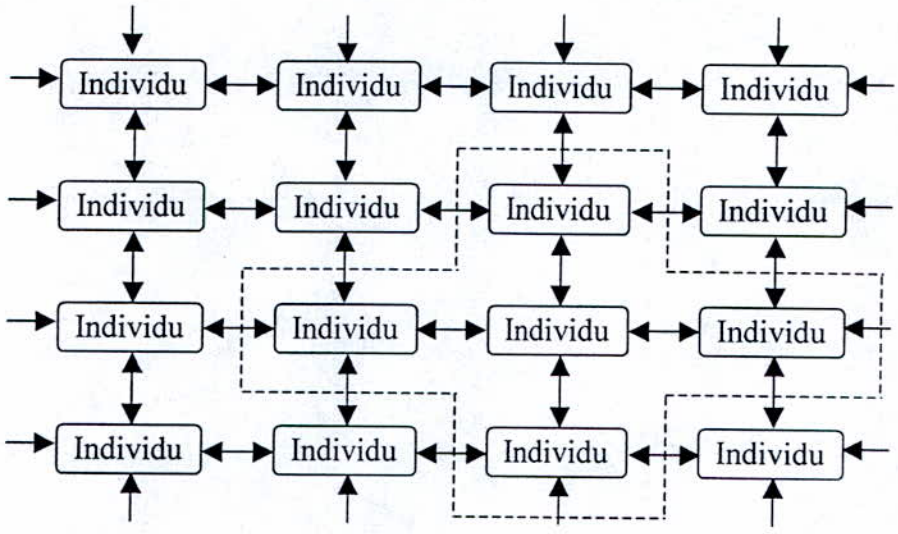


Figure 1.9: Le modèle de diffusion.

1.5 Schemata [Mit 1998]

La théorie traditionnelle des AGs suppose, en général, que ces derniers fonctionnent en découvrant, en favorisant, et en recombinant de bons "blocks de construction" des solutions à un très haut niveau de parallélisation. L'idée ici est que les bonnes solutions tendent à être construites à partir de bons blocks de construction c à d : combinaison de valeurs de bits qui confèrent une fitness élevée aux caractères dans lesquels ils se trouvent.

Dans le but d'analyser les mécanismes internes des AGs, et de formaliser la notion informelle des "blocks de construction", John Holland a introduit le concept de *schémas* (ou schemata). Un schéma est un ensemble de caractères de bits qui définit un sous-ensemble de caractères dans une population, lesquels sont semblables entres eux à certaines positions. Un schéma est présenté comme un caractère qui a la même longueur que tous les autres caractères dans la population (les individus), et utilise le même alphabet, à qui un symbole général est ajouté, d'habitude un astérisque. Un schéma définit les caractères dans la population qui sont égaux au schéma dans toutes ses positions fixes, c'est à dire, ceux qui ne contiennent pas le symbole général. Par exemple, supposons que les caractères sont de longueur 8, et l'alphabet des caractères est {0,1}. Alors l'alphabet du schéma est {0,1,*} et tous les schematas sont

aussi de longueur 8. Le schéma $H = 000*01*0$ décrit le sous-ensemble {00000100, 00000110, 00010100, 00010110}, et les caractères appartenant à ce sous-ensemble sont dits *instances* de H. Le schéma 00****** décrit le sous-ensemble dont les caractères commencent avec deux zéros, et le schéma ******** décrit la population entière. Par ailleurs, l'ordre d'un schéma H est définit comme étant le nombre de positions fixes présentes dans le schéma, et on appelle *longueur* d'un schéma la distance entre le premier et le dernier caractère fixé en position. Par exemple, le schéma $H = *11**0**$ est d'ordre 3 et de longueur 4.

Comment les AGs traitent-ils les schémas? Chaque caractère de bits de longueur l est une instance de 2^l schémas différents. Par exemple, le caractère 11 est une instance de ** (tous les caractères possibles de longueur 2), *1, 1*, et 11 (le schéma qui contient un seul caractère, 11). Donc, une population donnée de n caractères contient des instances entre 2^l et $n.2^l$ schémas différents. Si tous les caractères sont identiques, alors il y a des instances exactement de 2^l schémas différents, sinon, le nombre est inférieur ou égal à $n.2^l$. Cela veut dire, qu'à une génération donnée, quand l'AG évalue explicitement les fitness des n caractères dans la population, il est implicitement en train d'estimer la fitness moyenne d'un nombre beaucoup plus grand de schémas, où la fitness moyenne d'un schéma est définie comme étant la valeur moyenne des fitness de toutes les instances possibles du schéma. Par exemple, dans une population de n caractères générés aléatoirement, en moyenne, la moitié des caractères seront des instances de 1***...* et l'autre moitié sera des instances de 0***...* . Les évaluations de $n/2$ caractères qui sont approximativement des instances de 1***...* donnent une estimation de la fitness moyenne de ce schéma (ceci est une estimation car les instances évaluées dans une population de taille typique ne sont qu'une partie de toutes les instances possibles). Puisque les schémas ne sont pas explicitement représentés ou évalués par l'AG, les estimations des fitness moyennes des schémas ne sont pas calculées ni stockées explicitement par l'AG. Cependant, le comportement des AGs, en termes d'augmentation ou de diminution d'instances des schémas donnés dans la population, peut être décrit comme si les valeurs moyennes des fitness des schémas étaient calculées et stockées.

On peut calculer les dynamiques approximatives de cette augmentation et diminution d'instances des schémas comme suit. Soit H un schéma avec au moins une instance présente dans la population à l'instant t. Soit $m(H,t)$ le nombre d'instances de H à l'instant t, et soit $\hat{f}(H,t)$ la fitness moyenne de H à l'instant t (c'est à dire la valeur moyenne des fitness des instances de H dans la population à l'instant t). On veut calculer $E(m(H,t+1))$, qui représente le nombre attendu d'instances de H à l'instant $t+1$. Le nombre attendu des

descendants d'un caractère x est égal à $f(x)/\bar{f}(t)$, où $f(x)$ est la fitness de x et $\bar{f}(t)$ la fitness moyenne de la population à l'instant t. Par ailleurs, supposons que x est dans la population à l'instant t, et que $x \in H$ signifie que x est une instance de H, et ignorons (dans ce cas) les effets du croisement et de la mutation, on a

$$E(m(H,t+1)) = \sum_{x \in H} f(x)/\bar{f}(t)$$
$$= (\hat{f}(H,t)/\bar{f}(t))\, m(H,t) \qquad (1.7)$$

Par définition, $\hat{f}(H,t) = (\sum_{x \in H} f(x))/m(H,t)$ pour tout x appartenant à la population à l'instant t. Donc, malgré que l'AG ne calcule pas explicitement $\hat{f}(H,t)$, les augmentations et les diminutions des instances d'un schéma dans la population dépendent de cette quantité.

Le croisement et la mutation peuvent tous les deux, détruire et créer des instances de H. Pour notre cas, incluons seulement les effets destructifs du croisement et de la mutation c à d : ceux qui diminuent le nombre d'instances de H. En incluant ces effets, nous modifions le coté droit de l'équation 1.7 pour donner une borne inférieure de $E(m(H,t+1))$. Soit p_c la probabilité de croisement, et supposons qu'une instance du schéma H est choisie pour être un parent. Le schéma H est dit survivant après croisement (en un seul point) si un de ses descendants est une instance du schéma H. On peut donner une borne inférieure sur la probabilité $S_c(H)$ que H survivra après croisement:

$$S_c(H) \geq 1 - p_c\left(\frac{d(H)}{l-1}\right) \qquad (1.8)$$

où $d(H)$ est la longueur définie de H et l est la longueur des caractères de bits dans l'espace de recherche. Autrement dit, les croisements se situant à l'intérieure de la longueur définie de H peuvent détruire H (c'est à dire peuvent produire des descendants qui ne sont plus des instances de H), donc on multiplie la fraction du caractère que H occupe par la probabilité de croisement, pour obtenir une borne supérieure de la probabilité qu'il se détruit. (La valeur est une borne supérieure car certains croisements entre les positions définies d'un schéma ne le détruiront pas, par exemple, si deux caractères identiques se croisent entre eux). Soustraire cette valeur de 1 donne une borne inférieure sur la probabilité de survie $S_c(H)$. En résumé, la probabilité de survie après croisement est plus grande pour les schémas courts.

Les effets de perturbation de la mutation peuvent être quantifiés comme suit: soit p_m la probabilité de mutation. Donc $S_m(H)$, la probabilité qu'un schéma H survivra après mutation d'une instance de H, est égale à $(1-p_m)^{o(H)}$, où $o(H)$ est l'ordre de H (c'est à dire le nombre de bits définis dans H). Autrement dit, pour chaque bit, la probabilité que le bit ne sera pas muté est $1-p_m$, donc la probabilité qu'aucun bit défini du schéma H ne soit muté est cette quantité élevée à la puissance $o(H)$. En d'autres termes, la probabilité de survie après mutation est plus grande pour des schémas d'ordres petits.

Ces effets de perturbation peuvent être utilisés pour ajuster l'équation 1.7:

$$E(m(H,t+1)) \geq \frac{\hat{f}(H,t)}{\bar{f}(t)} m(H,t)\left(1-p_c\frac{d(H)}{l-1}\right)\left[(1-p_m)^{o(H)}\right] \qquad (1.9)$$

Ceci est connu sous le non du théorème des schémas (Holland 1975; Goldberg 1989). Il décrit la croissance d'un schéma d'une génération à l'autre. Le théorème des schémas est souvent interprété comme si on sous-entendait que les schémas courts et d'ordres petits dont la fitness moyenne reste au-dessus de la moyenne, recevront une augmentation exponentielle du nombre de spécimens (instances évaluées) à travers le temps, puisque le nombre de spécimens de ces schémas qui ne sont pas perturbés et restent au-dessus de la moyenne en fitness augmente par un facteur de $\hat{f}(H,t)/\bar{f}(t)$ à chaque génération.

Le théorème des schémas décrit dans l'équation 1.9 est une borne inférieure, car il traite seulement les effets destructifs des opérations de croisement et de mutation. Cependant, le croisement est reconnu être la majeure source de la puissance des AGs, avec l'habilité de recombinaison des instances de bons schémas pour former des instances de schémas d'ordres élevés à qualité égale ou meilleure. Dans ce qui précède, il est supposé que le processus par lequel les AGs fonctionnent est connu comme l'hypothèse des blocks de construction (Goldberg, 1989).

Le théorème des schémas donné plus haut s'adresse non seulement aux schémas, mais aussi à n'importe quels sous-ensembles de caractères dans l'espace de recherche. La raison sur le choix des schémas est motivée (en particulier, les schémas courts et de fitness moyenne élevée) par, une bonne description des types de blocks de construction qui sont combinés effectivement par le croisement en un seul point. Les chercheurs des AGs ont défini d'autres opérateurs de croisement qui traitent différents types de blocks de construction, et ont analysé les schémas généralisés que manipule effectivement un opérateur de croisement.

1.6 Applications des AGs en commande et en identification des systèmes [Fle 2001]

Les algorithmes génétiques ont été appliqués avec succès à une grande partie d'applications off-line. Dans le domaine de la commande des systèmes, ces applications incluent la synthèse des régulateurs, l'identification des modèles, l'analyse de la stabilité robuste, et le diagnostic des défaillances. Dans certains cas, les AGs sont utilisés comme les seuls moyens de conception et de synthèse. Dans d'autres cas, ils ont été combinés avec d'autres méthodes existantes. Dans certains cas, ils ont été combinés avec d'autres techniques intelligentes ou métaheuristiques. Un système intelligent peut prendre des décisions appropriées, autonomes, et généralement incorpore un processus d'apprentissage (bien qu'aucune définition ferme d'un tel système n'existe). Toute technique qui découvre de nouvelles solutions, basée sur l'expérience acquise à partir des solutions antérieures, peut être classée comme une méthode métaheuristique.

1.6.1 Régulateurs optimaux

Les algorithmes génétiques, et d'autres algorithmes évolutionnaires tel que la programmation génétique, ont été intensivement appliqués à la synthèse off-line des régulateurs. Les AGs ont été utilisés pour obtenir les paramètres ou la structure du régulateur ou parfois les deux. Ils ont été aussi combinés avec les régulateurs flous et neuronaux pour former une stratégie de commande intelligente.

Dans le début des années 1990s, les AGs ont été en premier temps investigués comme des moyens alternatifs pour l'ajustement des régulateurs PID. Oliveira *et al.* (1991) ont utilisé un AG standard pour déterminer des estimations initiales pour les valeurs des paramètres du PID. Ils ont appliqué leur méthodologie à une variété de classes des systèmes linéaires invariants dans le temps (LTI), comprenant des classes de systèmes à phase minimale, phase non-minimale, et systèmes instables. Wang et Kwok (1992) ont appliqué un AG qui utilise des 'micro-opérateurs' d'inversion et de présélection à l'ajustement du régulateur PID. Ils ont insisté sur l'avantage de flexibilité en prenant en compte la fonction du coût et, en particulier, ils ont fait allusion au concept d'optimalité au sens de Pareto (fournir la capacité d'adresser de multiples fonctions objectives simultanément, telles que la réponse transitoire et le rejet de perturbation).

Récemment, Vlachos *et al.* (1999) ont appliqué un AG à l'ajustement d'un régulateur PI décentralisé pour des processus multivariables. La performance du régulateur était définie en termes des bornes dans le temps sur les réponses en boucle fermée pour les deux problèmes de poursuite et d'interactions dans la boucle. Cette approche a offert une bonne visualisation de la performance des solutions potentielles.

Une autre approche pour l'utilisation des AGs dans la synthèse des régulateurs consiste à appliquer la méthodologie indirectement. Dans un tel plan, l'AG manipule les paramètres d'entrée d'un processus de synthèse établi, qui, à son tour produit le régulateur final. La méthode quadratique linéaire gaussienne (LQG) et le régulateur H-infini ont été tous les deux utilisés de cette manière. Dans la méthode LQG, un AG peut être utilisé pour rechercher les meilleures valeurs des facteurs pondérés de la matrice, puisque la sélection manuelle des éléments de la matrice n'est pas simple. Une approche de recherche similaire peut être utilisée en conjonction avec une procédure de synthèse H-infini. Dans ce cas, un AG peut chercher des fonctions de poids pour le système dans le but d'assurer de bonnes performances en boucle fermée, tandis qu'un régulateur robuste est garanti comme un résultat de la synthèse H-infini. Ces approches de synthèse *indirectes* ou hybrides (LQG et H-infini) ont été étendues à une synthèse dont le critère est multiobjectif (qui traite plusieurs objectifs simultanément), accomplie via une incorporation d'un algorithme génétique multiobjectif (MOGA).

Les AGs ont été également appliqués avec succès *directement* dans le domaine de la commande par H-infini. Dans cette approche, le régulateur actuel est synthétisé via un AG. En admettant la difficulté dans l'implémentation des régulateurs H-infini sans contraintes, dans laquelle l'ordre du régulateur est beaucoup plus élevé que celui du système. Chen et Cheng (1998) ont proposé une structure qui spécifie un régulateur H-infini. Un AG a été utilisé pour chercher de bonnes solutions se situant dans un domaine admissible des paramètres du régulateur (obtenu via le critère de stabilité de Routh-Hurwitz).

La programmation génétique a été utilisée pour la synthèse automatique des valeurs des paramètres et la topologie des régulateurs (Koza *et al.*, 2000). Un modèle bouclé en îles de 66 algorithmes de GP (Genetic Programming), chacun avec 1000 individus, ont été implémentés. Cette approche implique un calcul très intensif par rapport à la plupart des AEs. Le système a dupliqué les parents existants (pour les régulateurs PI et PID) et a redécouvert les anciens (une technique de réglage faisant intervenir la deuxième dérivée de l'erreur entre le signal de référence et le signal de sortie).

Les algorithmes génétiques multiobjectifs ont été utilisés dans le contexte d'optimisation des paramètres des régulateurs. Par exemple, MOGA a été utilisé pour sélectionner la

structure du régulateur et les paramètres convenables pour une régulation multivariable d'une turbine à gaz (Chipperfield et Fleming, 1996).

1.6.2 Application à la commande neuro/floue

Les limitations des régulateurs conventionnels pour des applications à des systèmes dynamiques compliqués, ont motivé la recherche dans ce qui est appelé les systèmes de commande intelligents. Les deux techniques les plus populaires sont la commande floue, dans laquelle une connaissance d'expertise peut être incorporée dans la conception du réglage, et la commande par réseaux de neurones qui est plus convenable pour les systèmes mal modélisés et pour les systèmes non-linéaires.

Les algorithmes génétiques ont été utilisés pour optimiser plusieurs aspects des régulateurs intelligents. Dans la commande floue, un AG peut être utilisé pour générer la table des règles floues, et d'ajuster les paramètres associés de la fonction d'appartenance. Dans la commande par réseaux de neurones, un AG peut fonctionner comme un choix alternatif pour l'apprentissage des valeurs des poids. Les AGs ont aussi été considérés capables d'optimiser la topologie d'un réseau de neurones.

Ichikawa et Sawa (1992) ont utilisé un réseau de neurones (RN) comme un remplacement direct des régulateurs conventionnels. Les poids ont été obtenus en utilisant une approche basée sur un AG. Chaque individu dans la population représente une distribution des poids pour le réseau. La structure et la fonction d'activation sont choisies a priori. Cette approche offre l'avantage de ne pas exiger des modèles d'apprentissage et que la fonction objective peut ne pas être mathématiquement appropriée.

La recherche dans le domaine d'application des AGs à la commande floue est divisée en deux grandes catégories: réglage des fonctions d'appartenance, et l'établissement de la base des règles, ainsi que leurs ajustement. Les praticiens de l'approche précédente ont tous tendance à admettre que la forme des règles sera reconnue vraisemblablement a priori, et que la plupart des incertitudes se manifestent dans le développement des fonctions d'appartenance associées. L'utilisation d'une base de règle statique réduit aussi le niveau nécessaire de la complexité du calcul, qui peut être une raison supplémentaire pour la popularité de cette approche.

Linkens et Nyongesa (1995) ont examiné l'application des deux approches on-line et off-line des AGs à la commande floue. Ils considèrent le processus complet de la commande

floue, incluant l'établissement des règles de commande et l'optimisation des fonctions d'appartenance.

1.6.3 Identification

L'application des AGs à l'identification des systèmes de types boîte noire et boîte grise a reçu un intérêt considérable depuis la publication de Kristinsson et Dumont en 1992. Ils ont appliqué les AGs à l'identification des systèmes continus et des systèmes discrets. La technique employée peut être utilisée aussi bien dans des applications on-line qu'off-line. L'AG a été utilisé pour identifier directement les pôles et les zéros, ou bien pour obtenir des valeurs physiques des paramètres. La fonction coût utilisée était l'erreur entre les sorties estimées et celles mesurées sur une fenêtre de données, qui est constituée de la paire entrée-sortie courante et les 30 échantillons antérieurs. Pour chaque point d'échantillonnage, la population a évoluée pour trois générations supplémentaires. Kristinsson et Dumont ont apporté des résultats comparables ou même meilleurs aux techniques célèbres, mais ils ont noté toutefois la grande dépense de calcul résultante.

Un des problèmes centraux dans l'identification des systèmes est le choix des termes d'entrée, de sortie, et du délai qui contribuent au modèle. Les AGs fournissent une méthode simple pour chercher l'espace de la structure pour les termes qui contribuent le plus considérablement à la sortie du processus.

La sélection des termes non linéaires pour les modèles de type NARMAX (Nonlinear AutoRegressive Moving Average eXogenous) (Leontaritis et Facturations, 1985) a été appliquée en utilisant un AG (Fonseca *et al.*, 1993). Le problème peut être vu comme une sélection de sous-ensembles où chaque individu dans la population représente un terme spécifique qui peut être utilisé dans le modèle. Dix termes sont habituellement considérés comme un nombre acceptable pour capturer le comportement du système. Dans un autre exemple, Luh et Wu (1999) ont utilisé des AGs basés sur la migration pour identifier des modèles NARX (Nonlinear AutoRegressive eXogenous).

Dans les cas où le modèle identifié est linéaire vis-à-vis des paramètres, les techniques des moindres carrées standards peuvent être utilisées pour obtenir de bonnes évaluations des paramètres du modèle. Dans d'autres circonstances, comme pour les modèles rationnels non-linéaires, d'autres techniques peuvent fournir des résultats meilleurs. Les méthodologies basées sur les AEs ont été exploitées comme des solutions potentielles.

Choi *et al.* (2000) ont utilisé un algorithme basé sur les *stratégies d'évolution* pour identifier les paramètres des modèles de frottement statique (Karnopp) et dynamique (LuGre). Les structures du modèle ont été prédéfinies (basé sur des résultats existants dans la littérature). Les résultats ont été utilisés dans un système de compensation de frottements, qui utilise un régulateur par mode de glissement.

Facturations et Mao (1998) ont appliqué les AGs à l'identification des modèles rationnels non-linéaires. Les informations associées avec la structure et les paramètres ont été codées dans chaque individu pour faciliter l'optimisation simultanée des deux éléments. Les modèles rationnels ne sont pas linéaires vis-à-vis des paramètres et, par conséquent, les paramètres ne peuvent pas être estimés par les méthodes standards correctement. Le modèle peut être manipulé pour s'assurer qu'il est linéaire vis-à-vis des paramètres, mais cela introduit de sévères problèmes de bruit.

1.6.4 Diagnostic des défaillances

Patton *et al.* (1997) ont formulé la *FDI* (fault detection and isolation) basée sur le modèle comme un problème d'optimisation multi-critères dans lequel la tâche était de maximiser l'effet des défaillances sur le résidu, et par ailleurs de minimiser l'effet des incertitudes. Ils ont formulé une fonction de coût globale qui utilise la méthode d'inégalités et ont optimisé le critère en utilisant un AG. L'approche était appliquée à la découverte des défaillances de la sonde dans un système de contrôle de vol. Des observateurs robustes pour la détection des défaillances ont été aussi synthétisés en utilisant une approche basée sur le concept de Pareto (Kowalczuk *et al.*, 1999). Dans ces deux techniques, l'utilisation d'un AG a permis l'inclusion directe de plusieurs critères de performance (incluant les renseignements tirés du domaine fréquentiel inutilisés précédemment).

Les AGs ont été également appliqués aux méthodes FDI qui ne sont pas basées sur le concept des résidus. Marcu *et al.* (1997) ont formulé le diagnostic basé sur le modèle des défaillances du processus comme une caractéristique d'un problème de sélection et de classification. Un algorithme évolutionnaire multiobjectifs a été utilisé pour une reconnaissance off-line de régions correspondant à des défaillances *connues* et pour des conditions de défaillances libres (utilisant les formes des composants), et pour l'identification on-line des coefficients du processus.

Painton et Campbell (1995) ont développé une technique pour améliorer la fiabilité totale du système en considérant des choix au niveau des composants. Pour chaque composant

possible, une distribution du taux d'échec et un coût ont été définis. Un AG a été utilisé pour chercher la configuration du choix du composant pour le système le plus fiable, en termes de MTBF (temps moyen avant l'échec), avec une limitation pré-définie du coût. La distribution du MTBF a été obtenue pour chaque membre de la population en conduisant 200 essais (en utilisant un échantillonnage Latin d'hypercubes). L'AG a été constaté d'un intérêt préférable à l'alternative de base: une recherche exhaustive.

1.6.5 Analyse des systèmes

Les algorithmes génétiques ont été utilisés dans le contexte de conception des systèmes de commande robuste efficace (Marrison et Stengel, 1997). Dans l'analyse stochastique de la robustesse, la probabilité qu'un système en boucle fermée donne des performances inacceptables, avec la présence possible de variations des paramètres, est évaluée. Un AG peut être utilisé pour manipuler une population de points dans l'espace de synthèse (chacun correspond au vecteur de synthèse d'un compensateur). Chaque vecteur de synthèse peut être évalué en utilisant la méthode Monte Carlo Evaluation (MCE). Marrison et Stengel ont utilisé des outils statistiques pour comparer deux synthèses et éviter le calcul par les MCEs. La comparaison des synthèses a montré une différence statistique significative entre les deux alternatives. Ce résultat était convenable pour l'utilisation de l'algorithme de sélection par tournoi de l'AG.

L'analyse de la stabilité robuste des systèmes discrets peut être faite au moyen d'une recherche évolutionnaire pour les pôles du système qui se trouvent en dehors du cercle d'unité (Fadali *et al.*, 1999). Cette méthode teste des conditions suffisantes pour l'instabilité dans les systèmes LTI, discrets, les systèmes incertains avec des polynômes de structures non linéaires (dépendances entre les différents paramètres).

1.6.6 Applications on-line

Les avantages d'un AG pour les applications on-line en commande des systèmes sont les mêmes que ceux discutés pour les applications off-line. Cependant, il faut prendre en considération certaines mesures, il est important qu'un signal de commande approprié soit généré à chaque instant d'échantillonnage. S'il n'y a pas de contraintes, les actions du 'meilleur' individu sur l'AG peut infliger de sévères conséquences sur le processus. Cela est inacceptable dans la plupart des applications, en particulier, dans le cas des systèmes de sécurité ou les systèmes accomplissant des missions critiques.

Dans une application en temps réel, l'intervalle de temps pour lequel une procédure d'optimisation peut être exécutée entre deux points de décision est limité. Etant donnée la puissance de calcul actuelle, il n'est pas évident qu'un AG s'exécutera avec convergence pendant la période d'échantillonnage d'une application de commande typique. Donc, seulement un certain nombre de générations peut être évolué. Pour les systèmes avec des périodes d'échantillonnage assez grandes, un niveau de convergence acceptable peut être établit.

Il existe trois approches générales pour l'utilisation des AGs en commande on-line (Linkens et Nyongesa, 1995):

- Utiliser le modèle du processus.
- Utiliser le processus directement.
- Permettre un *réglage restreint* d'un régulateur existant.

1.7 Conclusion

Les algorithmes génétiques possèdent des caractéristiques d'évolution qui leurs permettent d'être très utilisés à une grande classe des problèmes d'optimisation. Ces mêmes caractéristiques font distinguer les AGs des autres mécanismes de recherche conventionnels, en ce qui concerne la capacité de trouver les solutions optimales d'un problème donné, tout en ne requérrant que peu d'informations sur ces dernières et en manifestant de très grande habilité d'éviter le piège des optima locaux, sans oublier la possibilité d'une implémentation parallèle. Cependant, il devrait être signalé qu'il n'y a aucune base théorique solide des AGs, et qu'il n'y a aucune garantie qu'une solution soit trouvée.

Parmi les divers champs d'applications des AGs, l'optimisation multiobjectif serait peut être la plus prometteuse des alternatives dans les AGs. En effet, la plupart des applications pratiques dans le domaine de la commande et d'identification des systèmes font appel à des problèmes d'optimisation de plusieurs critères (performances, robustesse, énergie, etc). Le prochain chapitre introduit les concepts théoriques nécessaires de l'optimisation multiobjectif pour une nouvelle formulation du problème d'optimisation avec les AGs.

Chapitre 2

Les Algorithmes Génétiques Multiobjectif

2.1 Introduction

Les algorithmes génétiques sont reconnus pour être convenus aux problèmes d'optimisation impliquant plusieurs fonctions objectives, i.e., la minimisation de plusieurs critères. L'AG devrait rechercher plusieurs solutions pour un problème multiobjectif et non pas une seule, en prenant l'avantage de toutes les ressemblances disponibles dans la famille des solutions possibles de ce problème.

La capacité de traiter des problèmes complexes, qui ont des propriétés particulières (discontinuité, multi-modalité, espaces faisables disjoints, des solutions objectives bruitées, etc) renforce l'effectivité des AGs ainsi que d'autres AEs dans la recherche multiobjectif.

Dans ce chapitre, nous présentons quelques concepts de base de l'optimisation multiobjectif en se basant sur le principe d'optimalité au sens de Pareto, afin de pouvoir aborder le cas de l'AG multiobjectif (MOGA, Multiobjective Genetic Algorithms) basé sur l'approche du Pareto-ranking proposée par Fonseca et Fleming.

2.2 L'optimisation multiobjectif [Fon 1995]

Les problèmes pratiques sont souvent caractérisés par des mesures de performances non-commensurables (non partagées) qui sont généralement en concurrence. Le problème d'optimisation multiobjectif peut être définit comme un problème de minimisation simultanée des n composantes f_k, $k = 1,....,n$, d'un vecteur de fonctions probablement non-linéaires f d'une variable de décision générale x dans un univers U, où

$$f(x) = (f_1(x), \ldots, f_n(x)).$$

Généralement, le problème n'accepte pas une solution unique, parfaite, mais plutôt un ensemble de solutions alternatives, dites non-dominées, connu comme l'ensemble de Pareto-optimal (Ben-Tal, 1980). En supposant un problème de minimisation, la dominance est définie comme suit

Définition 2.1 (La dominance au sens de Pareto)

Un vecteur $u=(u_1,\ldots,u_n)$ domine $v=(v_1,\ldots,v_n)$ ssi u est partiellement inférieur a v, i.e.,

$$\forall i \in \{1,\ldots,n\},\ u_i \le v_i \ \wedge\ \exists i \in \{1,\ldots,n\} : u_i < v_i.$$

Définition 2.2 (L'optimalité au sens de Pareto)

Une solution $x_u \in U$ est dite optimale au sens de Pareto ssi il n'y a aucun $x_v \in U$ pour lequel $v=f(x_v)=(v_1,\ldots,v_n)$ domine $u=f(x_u)=(u_1,\ldots,u_n)$.

Les solutions optimales au sens de Pareto sont aussi dites efficaces, non-dominées, et non-inférieures. Les vecteurs objectifs correspondants sont simplement dits non-dominés. L'ensemble des vecteurs non-dominés (ou ensemble non-dominé) définit la surface de compromis du problème considéré.

L'optimalité au sens de Pareto peut être illustrée graphiquement dans la Figure 2.1 en considérant l'ensemble de toutes les valeurs objectives faisables, i.e., l'ensemble de tous les points dans l'espace objectif correspondant à au moins une image de la variable de décision. L'ensemble non-dominé de ce problème est composé des points qui s'allongent sur toute la frontière en trait solide de la région grise. La Figure 2.2 illustre la notion de concavité et convexité de l'ensemble non-dominé. En général, les surfaces de compromis peuvent être ni convexes ni concaves. La convexité de la surface de compromis dépend sur la transformation d'échelle des fonctions objectives. Une transformation d'échelle non-linéaire peut convertir une surface non-convexe en une surface convexe, et vice-versa, comme il est illustré dans la Figure 2.3. La surface sombre correspondant aux objectifs (f_1, f_2) est l'originale, elle est non-convexe. Les autres surfaces mois sombres correspondent à $(f_1^{\alpha}, f_2^{\alpha})$ pour $\alpha = 5, \alpha = 9$. la dernière devient clairement convexe, néanmoins, ces formulations représentent le même

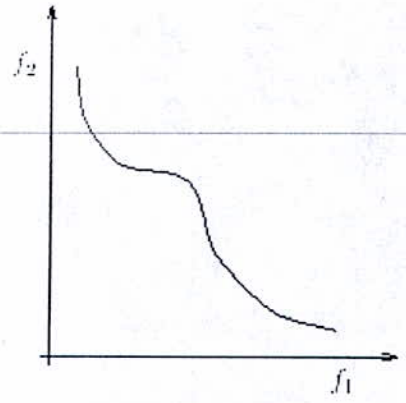


Figure 2.1: Interprétation graphique de l'ensemble non-dominé.

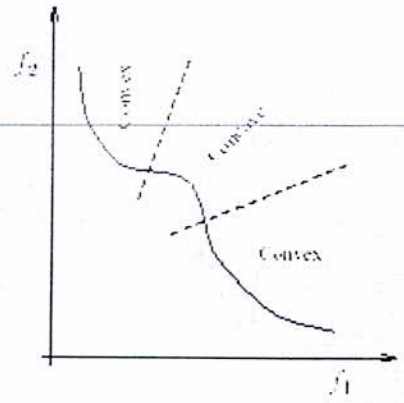


Figure 2.2: Régions concave et convexe de la surface de compromis.

problème de minimisation au sens de Pareto, et admettent exactement le même ensemble de solutions dans l'espace des variables de décision.

2.2.1 L'optimisation multiobjectif et le Decision Making

La notion d'optimalité au sens de Pareto n'est que la première étape dans la résolution d'un problème d'optimisation multiobjectif. Le choix d'une solution convenable à partir de toutes les alternatives non-inférieures ne dépend pas seulement du problème considéré, mais aussi sur les préférences subjectives d'un agent de décision, le Decision Maker (DM). Donc, la solution finale du problème est le résultat d'un processus *d'optimisation* et un autre de *décision.*

D'après la façon par laquelle les processus d'optimisation et de décision sont combinés dans la recherche d'un compromis de solutions, trois grandes classes de méthodes pour l'optimisation multiobjectif peuvent être identifiées (Hwang et Masud. 1979):

Articulation des préférences a priori Le DM exprime les préférences en combinant les différents objectifs dans une seule fonction de coût, le problème d'optimisation multi-objectifs devient alors un simple objectif ou mono-objectif.

Articulation des préférences a posteriori L'optimisateur présente l'ensemble des solutions non-dominées au DM, sans aucune préférence, le compromis est choisi à partir de cet ensemble.

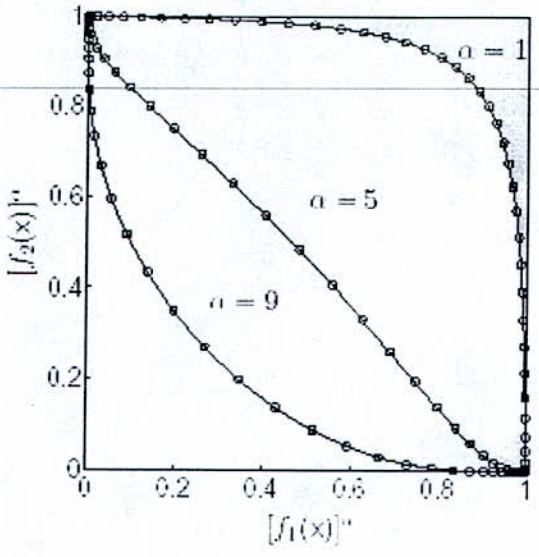


Figure 2.3: L'influence de la transformation d'échelle de chaque fonction objective sur la surface de compromis.

Articulation progressive des préférences L'optimisation et la décision sont faites d'une façon alternative. A chaque étape d'optimisation, l'optimisateur présente un ensemble de solutions non-dominées, des informations partielles sont fournies par le DM à l'optimisateur pour continuer la recherche et ainsi de suite, ces informations servent à guider l'optimisateur pour aboutir au bon compromis.

Les méthodes pour l'articulation des préférences a priori ont les avantages de ne requérir aucune autre interaction avec le DM. Pour cette raison, l'effort de calcul nécessaire peut être concentré sur la production de la solution finale. Cependant, si la solution trouvée ne peut pas être acceptée comme un bon compromis, de nouveaux essais de l'optimisateur sur des fonctions de coût modifiées doivent probablement être réalisés, jusqu'à ce qu'une solution convenable soit trouvée. Ces méthodes ont également le désavantage de requérir de nouveaux essais de l'optimisateur chaque fois que les préférences du DM changent.

Les méthodes pour l'articulation des préférences a posteriori diminuent ces difficultés mais nécessitent un temps de calcul plus grand. Du fait qu'une connaissance a priori est supposée, la surface de compromis entière doit être sélectionnée avec précision. Ces méthodes sont particulièrement convenables pour les problèmes impliquant des objectifs statiques, mais dans lesquels les préférences du DM changent relativement souvent. Dans de tels cas, les changements de préférences ne demandent pas un calcul supplémentaire.

L'articulation progressive des préférences établit un compromis entre ces deux classes de méthodes, mais avec une interaction plus intense entre le DM et l'optimisateur.

2.2.2 Articulation des préférences

L'articulation des préférences définit une fonction de coût qui distingue entre les solutions. Bien qu'une fonction de coût peut être difficile à formaliser, les approches basées sur ce qui suit sont largement utilisées.

2.2.2.1 Coefficients de poids

Les coefficients de poids sont des valeurs réelles qui expriment l'importance relative des objectifs et contrôle leurs contributions dans la mesure de la valeur de coût globale. L'approche de la somme pondérée est l'exemple classique d'une méthode basée sur la pondération des objectifs (Hwang et Masud, 1979). La fonction de coût est formulée comme une somme pondérée des objectifs:

$$F_{ws}(x) = \sum_{i=1}^{n} w_i f_i(x)$$

où n est le nombre de fonctions objectives, et les poids w_i sont des constantes positives. Il est évident que pour des poids positifs donnés, l'optimum global de $F_{ws}(x)$ est toujours une solution non-dominée du problème multiobjectif original. Cependant le contraire n'est pas vrai, par exemple, des solutions non-dominées situées dans des régions concaves de la surface de compromis ne peuvent pas être obtenues par cette méthode, car leur coût est sous-optimal (Fleming et Pashkevich, 1985).

2.2.2.2 Priorités

Les priorités sont des valeurs entières qui déterminent l'ordre par lequel les objectifs sont optimisés, d'après leur importance. La méthode lexicographique (Ben-Tal, 1980), par exemple, commence par assigner différentes priorités à chaque objectif. Considérons n objectifs $f_1, \ldots, f_n$, chaque objectif f_i, $i=1,\ldots,n$, est lui est assigné une priorité i, où des valeurs grandes de i correspondent à des priorités élevées. La fonction de coût résultante est telle que

$$F_{lex}(x) < F_{lex}(y) \Leftrightarrow$$

$$\exists i \in \{1,.....,n\} : \forall j \in \{i,.....,n\},\ f_j(x) \le f_j(y) \ \wedge \ f_i(x) < f_i(y).$$

En pratique, l'objectif correspondant à la plus grande priorité, f_n, est minimisé en premier temps, la solution trouvée sera notée x_n. Ensuite, f_{n-1} est minimisée en considérant $f_n(x) \le f_n(x_n)$, et une nouvelle solution sera trouvée (x_{n-1}). La troisième étape consiste à minimiser f_{n-2} en considérant $f_k(x) \le f_k(x_k)$, pour $k = n-1, n$, et ainsi de suite. La solution finale x_1 est prise comme la solution du problème. Cette solution est aussi une solution optimale au sens de Pareto.

2.2.2.3 Buts

Les valeurs des buts indiquent les niveaux de performance désirés dans chaque dimension des objectifs. Une approche particulière qui interprète ceci est dite l'approche du minimax, qui consiste à minimiser la fonction de coût suivante

$$F_{mM}(x) = \max_{i \in \{1,...,n\}} \left\{ \frac{f_i(x) - g_i}{w_i} \right\}$$

où les constantes g_i représentent les buts à atteindre, et w_i sont des poids positifs qui indiquent la direction de recherche désirée dans l'espace objectif. Cette approche est capable de produire des solutions dans les régions concaves, toutefois, il n'est pas garanti que les solutions trouvées seront toujours des solutions strictement non-dominées.

2.2.3 L'optimisation avec des contraintes

La solution d'un problème pratique peut être soumise à des contraintes à travers un nombre de restrictions imposées sur la variable de décision. D'habitude, les contraintes sont imposées à partir de l'une des deux catégories suivantes:

Contraintes sur le domaine Elles expriment le domaine de définition de la fonction objective. Dans la commande des systèmes, la stabilité du système en boucle fermée peut être donnée comme un exemple pour les contraintes sur le domaine, du fait que la plupart des mesures de performances ne sont pas définies pour les systèmes instables.

Contraintes de préférences Elles imposent des restrictions supplémentaires sur la solution du problème d'après des connaissances acquises à partir d'un plus haut niveau. Une marge de stabilité donnée, par exemple, exprime une préférence (subjective) du concepteur.

Généralement, les contraintes sont exprimées en termes d'inégalités de type

$$f(x) \leq g$$

où f est une fonction à valeurs réelles de la variable x, et g une constante. L'inégalité peut être également stricte. Les contraintes d'égalité de type

$$f(x) = g$$

peuvent être formulées comme des cas particuliers des contraintes d'inégalité, par exemple

$$g-\delta_1 \leq f(x) \leq g+\delta_2$$

avec δ_1 et δ_2 sont des constantes positives, arbitraires de petites valeurs.

En général, le problème d'optimisation avec contraintes est celui de minimiser une fonction multiobjectif $(f_1, \ldots\ldots, f_n)$ d'une variable de décision x dans un univers U, en présence d'un nombre $n-k$ de conditions sur x, pouvant être éventuellement exprimées sous forme d 'un vecteur d'inégalité de type

$$(f_{k+1}(x), \ldots\ldots, f_n(x)) \leq (g_{k+1}, \ldots\ldots, g_n)$$

Il est implicitement supposé qu'il existe au moins un point dans U qui satisfait toutes les contraintes, bien qu'en pratique ceci n'est toujours pas garanti. Si toutes les contraintes ne peuvent pas être simultanément satisfaites, le nombre de contraintes violées est alors pris en compte afin de donner une solution acceptable au problème considéré.

2.3 Multiobjective Genetic Algorithms (MOGA) [Fon 1995]

Le problème d'implémentation des AGs pour les problèmes d'optimisation multiobjectif réside en fait dans le calcul de fitness pour les individus de la population, à partir d'un espace objectif multidimensionnel. Pour un problème à objectif unique comme il est déjà vu dans le chapitre précédent, une simple transformation par scaling ou ranking suffis pour calculer les valeurs de fitness, ce qui n'est pas le cas pour les problèmes impliquant de multiples objectifs.

Pour cette raison, Goldberg (1989) a proposé une méthode basée sur l'approche de Pareto, l'idée était d'affecter le rang 1 à tous les individus non-dominés puis les extraire de la population, et ensuite chercher un nouvel ensemble non-dominé et lui assigner le rang 2, et ainsi de suite. Quand tous les individus de la population leurs sont assignés un rang, une transformation comme le ranking ou le scaling est alors effectuée pour estimer l'adaptation de chaque individu.

Fonseca et Fleming (1993) ont proposé une méthode légèrement différente, basée aussi sur l'approche de Pareto, avec l'introduction d'un aspect de décision assez avancé qui incorpore des opérateurs relationnels puissants, qui permettent de mieux guider la recherche selon des buts et des préférences.

Le rang attribué à un individu quelconque est le nombre des individus qu'ils le dominent. Les individus non-dominés sont assignés le même rang qui est minimal, alors que les individus dominés sont pénalisés selon la concentration de la population dans la surface du compromis. Cette approche est simple à formuler et à analyser mathématiquement, elle est présentée dans la section 2.3.2.2.

2.3.1 L'optimisation évolutionnaire multiobjectif

L'optimisation évolutionnaire multiobjectif peut être généralisée (Fonseca et Fleming, 1993) et peut être vue comme une interaction entre le DM et l'algorithme de recherche évolutionnaire (voir Figure 2.6). L'algorithme de recherche évolutionnaire génère l'ensemble des solutions candidates ou individus selon leurs rangs qui sont calculés par le DM, qui peut aller d'une somme pondérée à un générateur de décision intelligent, le bloc EA représente tout optimisateur évolutionnaire. Dans les prochaines sections on abordera l'implémentation du Pareto-ranking qui est une méthode de calcul de rang dans le cas multiobjectif, ainsi que le cadre théorique élaboré par Fonseca et Fleming (1993) concernant la combinaison du principe

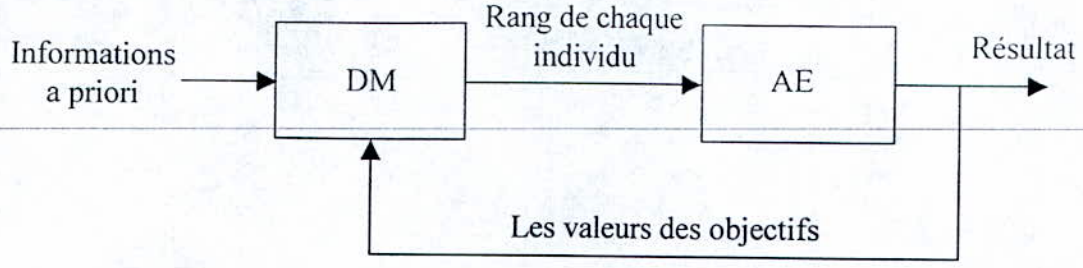


Figure 2.4: Structure générale d'un AE multiobjectif.

de dominance avec les préférences pour produire une stratégie d'affectation de coût ou rang unique.

2.3.2 Decision Making multiobjectif basée sur des buts et des priorités

La spécification des buts et des priorités peut être adaptée pour une grande variété de formulations de problèmes multiobjectif. Les buts et les priorités peuvent dans certains cas être tirés à partir de la description du problème. Une extension de la stratégie de décision proposée par Fonseca et Fleming (1993), est formulée dans la prochaine section en terme des opérateurs relationnels, qui incorporent toute information sur les buts et les priorités donnés. Le ranking de la population entière est basé sur de tels opérateurs.

2.3.2.1 L'opérateur de comparaison

Considérons un vecteur de fonctions f de dimension n de la même variable de décision x et deux vecteurs objectifs $u = f(x_u)$ et $v = f(x_v)$, où x_u et x_v sont des valeurs particulières de x. On considère aussi le *vecteur de préférences*

$$g = [g_1, \ldots\ldots, g_p]$$
$$g = [(g_{1,1}, \ldots\ldots, g_{1,n1}), \ldots\ldots, (g_{p,1}, \ldots\ldots, g_{p,np})]$$

où p est un nombre entier, $n_i \in \{0, \ldots\ldots, n\}$ pour $i = 1, \ldots\ldots, p$ et

$$\sum_{i=1}^{p} n_i = n$$

De la même façon, u peut être écrit comme suit

$$u = [u_1, \ldots\ldots, u_p]$$
$$u = [(u_{1,1}, \ldots, u_{1,n1}), \ldots, (u_{p,1}, \ldots\ldots, u_{p,np})]$$

et de même pour v et f.

Les sous-vecteurs g_i du vecteur de préférence g, où $i = 1, \ldots, p$ associe les priorités i et les buts g_{i,j_i}, où $j_i = 1, \ldots, n_i$, aux fonctions objectives correspondantes f_{i,J_i} composants de f_i. Cela suppose une permutation possible des composantes de f, sans perdre la généralité. Des valeurs grandes de i inférieures ou égales à p, indiquent des priorités élevées.

Généralement, chaque sous-vecteur g_i sera tel qu'un nombre $k_i \in \{0, \ldots\ldots, n_i\}$ de ses composantes atteignent leurs buts tandis que les autres non. Par ailleurs, u est tel que, pour $i = 1, \ldots\ldots, p$, on peut écrire

$$\exists\, k_i \in \{0, \ldots\ldots, n_i\} \mid \forall l \in \{1, \ldots\ldots, k_i\}, \forall\, m \in \{k_i + 1, \ldots\ldots, n_i\},$$
$$(u_{i,l} \leq g_{i,l}) \wedge (u_{i,m} > g_{i,m}).$$

Pour une raison de simplification, les premières k_i composantes des vecteurs u_i, v_i et g_i seront représentés respectivement par $u_i^{\cup}, v_i^{\cup}$ et $g_i^{\cup}$. Les dernières $n_i - k_i$ composantes des mêmes vecteurs seront dénotées par $u_i^{\cap}, v_i^{\cap}$ et $g_i^{\cap}$, respectivement.

Définition 2.1 (préférabilité) Le vecteur $u = [u_1, \ldots\ldots, u_p]$ est préférable à $v = [v_1, \ldots\ldots, v_p]$ d'après le vecteur de préférences $g = [g_1, \ldots\ldots, g_p]$ ($u \underset{g}{\prec} v$) ssi

$$p = 1 \Rightarrow (u_p^{\cap} \mathrm{p} < v_p^{\cap}) \vee \{(u_p^{\cap} = v_p^{\cap}) \wedge [(v_p^{\cup} \not\leq g_p^{\cup}) \vee (u_p^{\cup} \mathrm{p} < v_p^{\cup})]\}$$

et

$$p > 1 \Rightarrow (u_p^{\cap} \mathrm{p} < v_p^{\cap}) \vee \left\{(u_p^{\cap} = v_p^{\cap}) \wedge [(v_p^{\cup} \not\leq g_p^{\cup}) \vee (u_{1,\ldots,p-1} \underset{g_{1,\ldots,p-1}}{\prec} v_{1,\ldots,p-1})]\right\}$$

En d'autres termes, les vecteurs u et v sont premièrement comparés d'après de leurs composantes avec les priorités les plus élevées, sans la prise en considération de celles qui ont atteint leurs buts, $u_p^{\cup}$. Dans le cas où les deux vecteurs atteignent leurs buts avec cette priorité, ou s'ils violent quelques uns de ces buts ou tous, mais exactement de la même

manière, alors le prochain niveau de priorité $(p-1)$ est considéré. Le processus continue jusqu'à ce que la priorité 1 soit satisfaite, dans laquelle le résultat est donné en comparant les composantes de priorité 1.

Définition 2.2 (équivalence) Le vecteur $u=[u_1,.....u_p]$ est équivalent à $v=[v_1,.....,v_p]$ d'après le vecteur de préférence $g=[g_1,.....,g_p]$ $(u \underset{g}{\equiv} v)$ ssi

$$(u^{\cap}=v^{\cap}) \wedge (u_1^{\cup}=v_1^{\cup}) \wedge (v_{2,...,p}^{\cup} \leq g_{2,...,p}^{\cup}).$$

Le concept de préférabilité peut être rapporté à celui d'infériorité comme suit

Lemme 2.1 Pour n'importe quels vecteurs objectifs u et v, si $u \, p < v$, alors u est soit préférable ou équivalent à v, étant donné un vecteur de préférence $g=[g_1,.....,g_p]$ quelconque.

Lemme 2.2 (transitivité) La relation de préférabilité est transitive, i.e., étant donnés trois vecteurs objectifs u, v et w, et un vecteur de préférence $g=[g_1,.....,g_p]$.

$$u \underset{g}{\prec} v \underset{g}{\prec} w \underset{g}{\prec} \Rightarrow u \underset{g}{\prec} w$$

2.3.2.2 Le ranking de la population

A l'opposé du cas d'un seul objectif, le ranking d'une population dans le cas multiobjectif n'est pas unique. Dans le cas présent, il est désiré que tous les individus préférés leurs soient assignés le même rang, et que ces individus seront placés à des niveaux de rangs élevés par rapport à ceux qui sont plus préférables.

Considérant l'individu x_u à la génération t avec le vecteur objectif u, et soit $r_u^{(t)}$ le nombre d'individus dans la population actuelle qui sont préférables que lui. La position courante de x_u est donnée par :

$$rang(x_u,t)=r_u^{(t)}$$

ce qui assure que tous les individus préférables dans la population courante leurs sont attribués le rang zéro.

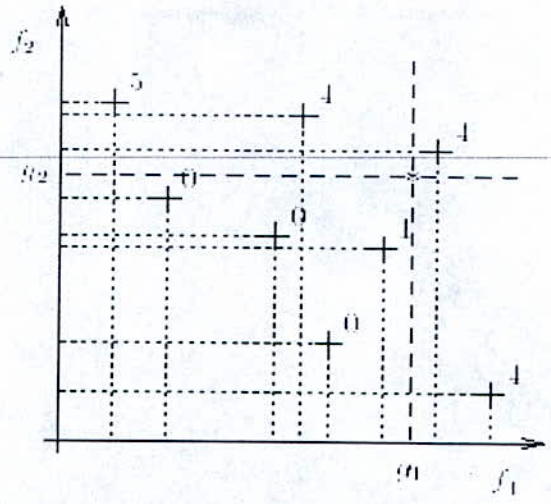


Figure 2.5: f_1 et f_2 ont la même priorité.

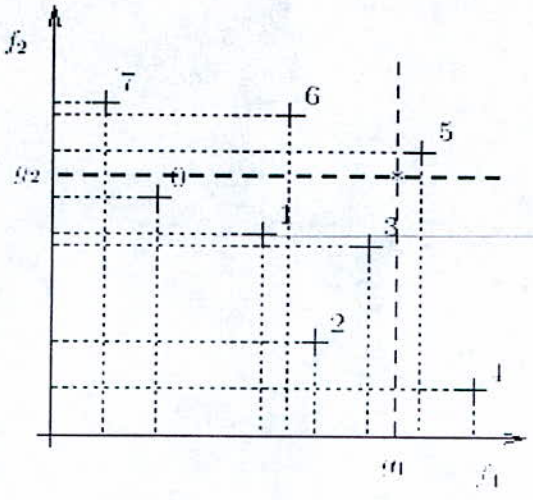


Figure 2.6: f_2 est prioritaire sur f_1.

Dans le cas d'une grande population qui a une distribution uniforme, le rang normalisé $r^{(i)} / N$ constitue une estimation de l'espace de recherche préférable pour chaque individu.

Dans le cas général d'une population qui a une distribution non uniforme, l'estimation biaisé est obtenue qui, néanmoins, préserve l'ordre strict des relations entre les individus comme il est désiré.

Lemme 2.3 Si un vecteur objectif $u = f(x_u)$ associé à l'individu x_u est préférable qu'un autre vecteur objectif $v = f(x_v)$ associé à l'individu x_v dans la même population, alors $rang(x_u,t) < rang(x_v,t)$. Par équivalence, si $rang(x_u,t) \geq rang(x_v,t)$, alors u n'est pas préférable à v.

La Figure 2.5 illustre le ranking d'une population pour deux vecteurs de préférences g_1, g_2. Dans le premier cas, les deux objectifs ont la même priorité. Notons que les individus qui ont atteint leurs buts sont préférables à tous les autres individus, et par suite, ils leur sont assignés un rang minimal. Dans le deuxième cas (Figure 2.6), l'objectif f_2 est prioritaire sur f_1, les individus qui n'ont pas atteint le but g_2 sont les plus mauvais, indépendamment de leur performance théorique suivant l'objectif f_1. Une fois le but g_2 est atteint f_1 est utilisée pour le ranking. Les individus qui ont satisfait les deux buts g_1, g_2 sont des solutions satisfaisantes, alors que ceux qui ont atteint uniquement le but g_2 sont faisables mais pas satisfaisants.

2.3.3 Le calcul de fitness

La fitness est interprétée ici comme le nombre de descendants qu'un individu donné est prévu de produire à travers la sélection. Cette interprétation est fondamentalement différente du coût ou de l'utilité, qui reflètent le résultat donné par le processus du Decision Making, et qui sont indépendants du processus de recherche. Le processus de sélection déterminent quels individus influencent actuellement la production de la génération suivante, et, par conséquent, doit être considéré comme partie de la stratégie de recherche.

Le calcul de fitness à partir d'un espace multiobjectif ou multicritères se base sur le Pareto-ranking, donné comme suit

1. Trier la population, en appliquant un algorithme de tri basé sur le Pareto-ranking.
2. Calculer la fitness à partir du meilleur individu ($rang = 0$) jusqu'au plus mauvais ($rang = \max r^{(i)} < N$) d'après une certaine fonction, généralement linéaire ou exponentielle, mais peut être également de type quelconque.
3. Assigner une même valeur de fitness pour tous les individus qui ont le même rang. Cette valeur est la moyenne des fitness de ces individus.

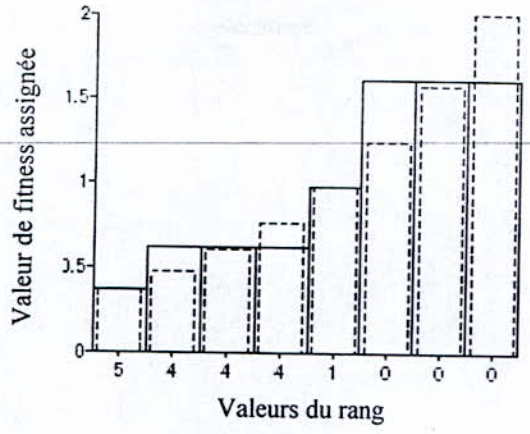


Figure 2.7: Calcul de fitness par la méthode de Pareto-ranking.

La Figure 2.7 illustre la procédure du calcul de fitness dans le cas d'une population à huit individus. Dans cet exemple, trois individus seulement représentent des solutions non-dominées, et ils leurs sont, par conséquent, assignés le rang 0. Après avoir attribué des rangs aux huit individus de la population, d'après l'approche du Pareto-ranking, décrite plus haut, ces individus leurs sont alors assignés des valeurs de fitness par ranking non-linéaire (dans ce cas). Par suite, une valeur de fitness unique pour chaque groupe d'individus de même rang est alors calculée comme étant la moyenne des fitness.

2.4 Conclusion

Dans ce chapitre, une nouvelle méthode d'implémentation des AGs pour les problèmes d'optimisation à plusieurs objectifs est introduite, et est présentée comme étant un simple problème de calcul de fitness. Cette méthode est basée sur l'approche proposée par Fonseca et Fleming, qui consiste à attribuer des rangs aux individus en utilisant un opérateur de comparaison particulier, ce dernier permet de faire la distinction entre les adaptations des individus. La combinaison entre les préférences et le Pareto-ranking permet d'estimer le coût (rang) afin de pouvoir calculer la fitness des individus. Chaque individu lui sera attribué une seule valeur de fitness qui reflète l'adaptation de ce dernier par rapport à tous les autres individus de la population, en prenant en compte toutes les fonctions objectives qui définissent le problème multiobjectif dinné.

Chapitre 3

Application des AGs en commande et en identification des systèmes

3.1 Introduction

Ce chapitre est consacré à l'application des AGs dans des problèmes de commande et d'identification des systèmes. Dans la première section, nous présentons une application qui utilise un AG pour l'optimisation des paramètres d'une loi de commande obtenue via la technique de Backstepping. Dans ce cas, les paramètres recherchés seront codés sur chaque individu de la population. La deuxième section est consacrée à un problème d'optimisation multiobjectif, concernant la commande mixte H_2 / H_∞. Dans cette application, un AG multiobjectif a été implémenté pour chercher l'ensemble des solutions représentant des régulateurs satisfaisant les performances H_2 / H_∞ afin de donner le choix selon le compromis désiré. Dans la dernière section, un AG utilisant des opérateurs de croisement et de mutation modifiés est introduit dans le but de résoudre un problème de sélection de termes dans le cadre de l'identification non-linéaire des systèmes.

3.2 Optimisation des paramètres d'une loi de commande obtenue par la technique de Backstepping

L'établissement d'une loi de commande via la technique de Backstepping implique généralement la supposition de valeurs arbitraires pour les paramètres de cette loi de commande. Ces valeurs sont généralement choisies par tâtonnement, ce qui ne permet pas dans certains cas d'assurer un bon compromis entre les différentes performances du système régulé ainsi que de bonnes caractéristiques du signal de commande (énergie, variance, etc). Dans ce qui suit nous utilisons un AG pour déterminer les valeurs optimales des paramètres de la loi commande pour un robot manipulateur Puma 560.

3.2.1 Synthèse de la commande via Backstepping pour le robot manipulateur Puma 560 [Gha 1998]

Dans cette section, nous présentons le robot manipulateur Puma 560 ainsi que son modèle dynamique, et à base de ce modèle nous élaborons une synthèse d'une loi de commande via la technique de Backstepping. Cette technique est une procédure récursive qui combine entre le choix de la fonction de Lyapunov et la synthèse de la loi de commande, en transformant le problème de synthèse de commande pour le système global à une synthèse de séquences de commande pour des systèmes réduits. En exploitant la flexibilité des systèmes réduits, le Backstepping peut répondre aux problèmes de régulation, poursuite et de robustesse avec des conditions moins restrictives.

3.2.1.1 Modèle dynamique du robot Puma 560

Le robot manipulateur Puma 560 réalise trois mouvements rotationnels, le premier dans le plan vertical, le second et le troisiéme suivant deux axes horizontaux. Son modèle dynamique peut être représenté par un système d'équations différentielles non linéaires du second ordre donné sous la forme matricielle suivante:

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + G(q) = u - mJ^T(q)\left[J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} + g\right] \qquad (3.1)$$

où $q=[q1\ q2\ q3]^T$ représente les positions angulaires des trois articulations principales du robot Puma 560, les trois autres articulations sont maintenues à la position zéro. Dans ce

modèle, m représente la charge utile et $g=[0\ 0\ 9.81]^T$ est le vecteur de gravité. Les matrices M, B, G et J sont données dans l'annexe.

3.2.1.2 Synthèse de la commande

Considérons le i^{eme} sous système du système global, la procédure récursive se déroule comme suit: à chaque étape, on augmente l'ordre du système considéré et on génère une commande virtuelle afin de stabiliser ce système augmenté, jusqu'à ce que la commande réelle apparaisse à la p^{eme} étape. Dans notre cas, chaque articulation du robot Puma 560 représente un sous système d'ordre 2. Le sous système i du système global est représenté par les équations d'état suivantes:

$$\begin{cases} \dot{z}_{i,1} = z_{i,2} \\ \dot{z}_{i,2} = \dfrac{1}{I_i} u_i + \phi_i \end{cases} \qquad i=1,2,3 \tag{3.2}$$

où les éléments du vecteur $z_i = [z_{i,1}\ z_{i,2}]$ représentent respectivement la position et la vitesse angulaires de l'articulation i, u_i est l'entrée de commande du sous système soit le couple développé par les actionneurs au niveau de cette articulation. Les coefficients I_i et les quantités ϕ_i sont donnés dans l'annexe.

En tenant en compte que la vitesse de rotation et l'accélération de chaque articulation sont bornées (cela est du aux actionneurs utilisés), les quantités ϕ_i sont telles que:

$$\left|\phi_i\right| \le \sum_{k=0}^{p} \sum_{l=1}^{3} \zeta_{i,l}^{k} \left|y_l\right|^k, \ y_l = z_{i,1} \tag{3.3}$$

où les quantités $\zeta_{i,l}^{k}$ sont des termes non linéaires qui sont fonctions de la variable $z_{i,1}$ qui est la position angulaire de la i^{eme} articulation ainsi que $v_{\max}$ et $a_{\max}$ qui représentent respectivement les valeurs maximales de la vitesse et de l'accélération de cette i^{eme} articulation.

Etape 1

On définit l'erreur de poursuite pour le i^{eme} sous système comme suit:

(3.4)

$$e_{i,1} = z_{i,1} - y_{i,ref}$$

donc

$$\dot{e}_{i,1} = \dot{z}_{i,1} - \dot{y}_{i,ref} \tag{3.5}$$

en utilisant (3.2) on obtient:

$$\dot{e}_{i,1} = z_{i,2} - \dot{y}_{i,ref} \tag{3.6}$$

Considérons la fonction de Lyapunov initiale suivante:

$$V_0 = \sum_{i=1}^{3} \left\{ \sum_{k=1}^{p} e_{i,1}^{2k} + \Gamma_i^{-1} (\beta_i - \beta_i^*)^2 \right\} \tag{3.7}$$

où β_i est le gain d'adaptation variant dans le temps injecté pour contourner l'effet des interconnections, sa valeur désirée est β_i^*, et Γ_i^{-1} est une constante positive. En dérivons V_0 par rapport au temps, on obtient :

$$\dot{V}_0 = \sum_{i=1}^{3} \left\{ \sum_{k=1}^{p} 2k e_{i,1}^{2k} \dot{e}_{i,1} + 2\Gamma_i^{-1} (\beta_i - \beta_i^*) \right\} \tag{3.8}$$

en utilisant (3.6), $\dot{V}_0$ devient:

$$\dot{V}_0 = \sum_{i=1}^{3} \left\{ \sum_{k=1}^{p} 2k e_{i,1}^{2k-1} (z_{i,2} - \dot{y}_{i,ref}) + 2\Gamma_i^{-1} \dot{\beta}_i (\beta_i - \beta_i^*) \right\} \tag{3.9}$$

Considérons $z_{i,2}$ comme étant une commande virtuelle:

$$z_{i,2} = \alpha_{i,1}(e_{i,1}, \beta_i) \tag{3.10}$$

en choisissons

$$\alpha_{i,1}(e_{i,1}, \beta_i) = -k_{i,1} e_{i,1} - \beta_i e_{i,1} - \gamma_i e_{i,1}^{2p-1} + \dot{y}_{i,ref}$$

et β_i tel que

$$\dot{\beta}_i = \Gamma_i . \left(\sum_{k=1}^{p} k e_{i,1}^{2k} - \sigma_i \beta_i \right) \tag{3.11}$$

Nous aurons

$$\dot{V}_0 = \sum_{i=1}^{3}\left\{\sum_{k=1}^{p} -2k k_{i,1} e_{i,1}^{2k} - 2k\gamma_i \, e_{i,1}^{(2p+2k-2)} - 2\sigma_i \, \beta_i \, (\beta_i - \beta_i^*)\right\} \tag{3.12}$$

La condition de stabilité $\dot{V}_0 < 0$ est assurée en choisissons des valeurs positives appropriées pour $k_{i,1}$, β_i et σ_i .

Etape 2

Considérons maintenant la fonction de Lyapunov augmentée suivante:

$$V_1 = V_0 + \sum_{i=1}^{3} e_{i,2}^2 \tag{3.13}$$

où

$$e_{i,2} = z_{i,2} - \alpha_{i,1} \tag{3.14}$$

La dérivée de $\dot{V}_1$ est donnée par:

$$\dot{V}_1 = \dot{V}_0 + \sum_{i=1}^{3} 2e_{i,2}\,\dot{e}_{i,2} \tag{3.15}$$

$$= \dot{V}_0 + \sum_{i=1}^{3} 2e_{i,2}[(k_{i,1} + \beta_i + \gamma_i (2p-1) e_{i,1}^{2p-2})(z_{i,2} - \dot{y}_{i,ref}) + e_{i,1}\Gamma_i \, (\sum e_{i,1}^{2k} - \sigma_i \, \beta_i) + \phi_i + \frac{1}{I_i} u_i - \ddot{y}_{i,ref}]$$

En tenant compte de (3.3), on peut assurer la stabilité du système global ($\dot{V}_1 < 0$) en prenons:

$$u_i = -I_i \, [k_{i,2} e_{i,2} + (k_{i,1} + \beta_i + \gamma_i \, (2p-1) e_{i,1}^{2p-2})(z_{i,2} - \dot{y}_{i,ref}) + e_{i,1}\Gamma_i \, (\sum_{k=1}^{p} e_{i,1}^{2k} - \sigma_i \, \beta_i) - \ddot{y}_{i,ref}] \tag{3.16}$$

Cette loi de commande comprend 4 paramètres, soient 12 paramètres pour le système global. Le choix manuel de ces paramètres n'est pas évident dans le cas où on veut obtenir un bon compromis entre l'erreur de poursuite et l'énergie de la commande. Nous sommes donc dans une situation où on veut augmenter les performances du réglage mais au pris d'un faible coût à dépenser.

3.2.2 Configuration de l'algorithme génétique

Le problème à résoudre est de trouver des valeurs optimales pour les paramètres de la loi de commande établie plus haut, c'est à dire $k_{i,1}$, $k_{i,2}$, γ_i et σ_i pour $i =$ 1,2,3, dans le but d'obtenir un bon compromis entre erreur de poursuite et énergie de la commande. La fonction objective

qui définit ce problème peut être formulée sous forme d'un critère de minimisation qui combine les deux objectifs de minimisation de l'erreur et de l'énergie, soit:

$$J=\sum_{i=1}^{3}\sum_{j=0}^{n} a_i e_{i,1}^2(j)+b_i u_i^2(j) \qquad a_i, b_i \in [0,1] \qquad (3.17)$$

L'AG est lancé sur une population de taille 50, chaque individu est codé en binaire sur $16\times12=192$ bits, en considérant l'intervalle de variation [0, 500] pour les paramètres $k_{i,1}$ et $k_{i,2}$ et l'intervalle [0, 30] pour les deux autres paramètres. La fonction objective est calculée via le critère précédent en prenant $a_i=1$ et $b_i=10^{-10}$ pour $i=1,2,3$. Le calcul de fitness se fait par ranking non-linéaire avec une pression sélective égale à 2. Les opérateurs génétiques utilisés sont la sélection SUS, le croisement en un seul point, et la mutation binaire, avec $p_c=0.8$ et $p_m=0.01$ et un generation gap égal à 0.9. Enfin, La réinsertion utilisée est basée sur la fitness (réinsertion élitiste) et le nombre de génération est de 30.

3.2.3 Résultats et discussions

La trajectoire de référence choisie pour chaque articulation du robot Puma 560 est de type cycloïde. Les articulations se déplacent respectivement des positions { -50°, 135°, -135°} aux positions {135°, -50°, 50°} en un temps de mouvement égal à 1.5 sec. Les vitesses de départ et d'arrivée des articulations sont nulles.

Les variations de la valeur minimale et moyenne de la fonction objective sont données dans la Figure 3.1, nous remarquons que l'AG a convergé progressivement vers une solution optimale représentée par le meilleur individu de la population à la dernière génération. Cet individu donne les valeurs des paramètres recherchés:

$k_{1,1}=491.90$ $k_{2,1}=487.22$ $k_{3,1}=490.60$

$k_{1,2}=499.36$ $k_{2,2}=485.50$ $k_{3,2}=499.18$

$\gamma_1=245.86$ $\gamma_2=52.47$ $\gamma_3=216.37$

$\sigma_1=7.99$ $\sigma_2=6.82$ $\sigma_3=1.23$

avec

$$J=3.235\times10^{-4}$$

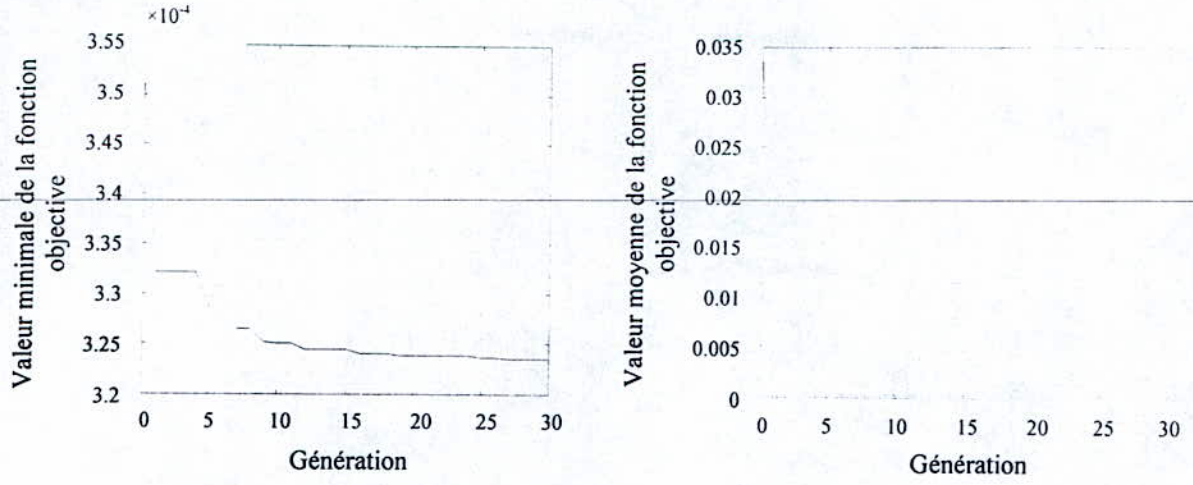


Figure 3.1: Valeurs minimale et moyenne de la fonction objective.

Les réponses du système global, en lui appliquant une commande correspondant aux paramètres précédents ainsi que les variations des valeurs des couples développés aux niveaux des trois articulations du robot sont données dans la Figure 3.2. Nous constatons que les articulations du robot ont suivi leurs consignes avec une grande précision tout en minimisant le plus possible les énergies des commandes appliquées, cependant, les résultats obtenus sont spécifiques aux valeurs de a_i et b_i prises dans notre cas. Enfin, pour valider ces résultats nous faisons un test de robustesse de la commande en considérant une chute soudaine de la masse à $t = 0.75\ s$. Les résultats de simulations sont donnés dans la Figure 3.3. Ces résultats affirment la robustesse de la commande puisque la perturbation a été rejetée rapidement aussi bien pour les trois sous-systèmes considérés.

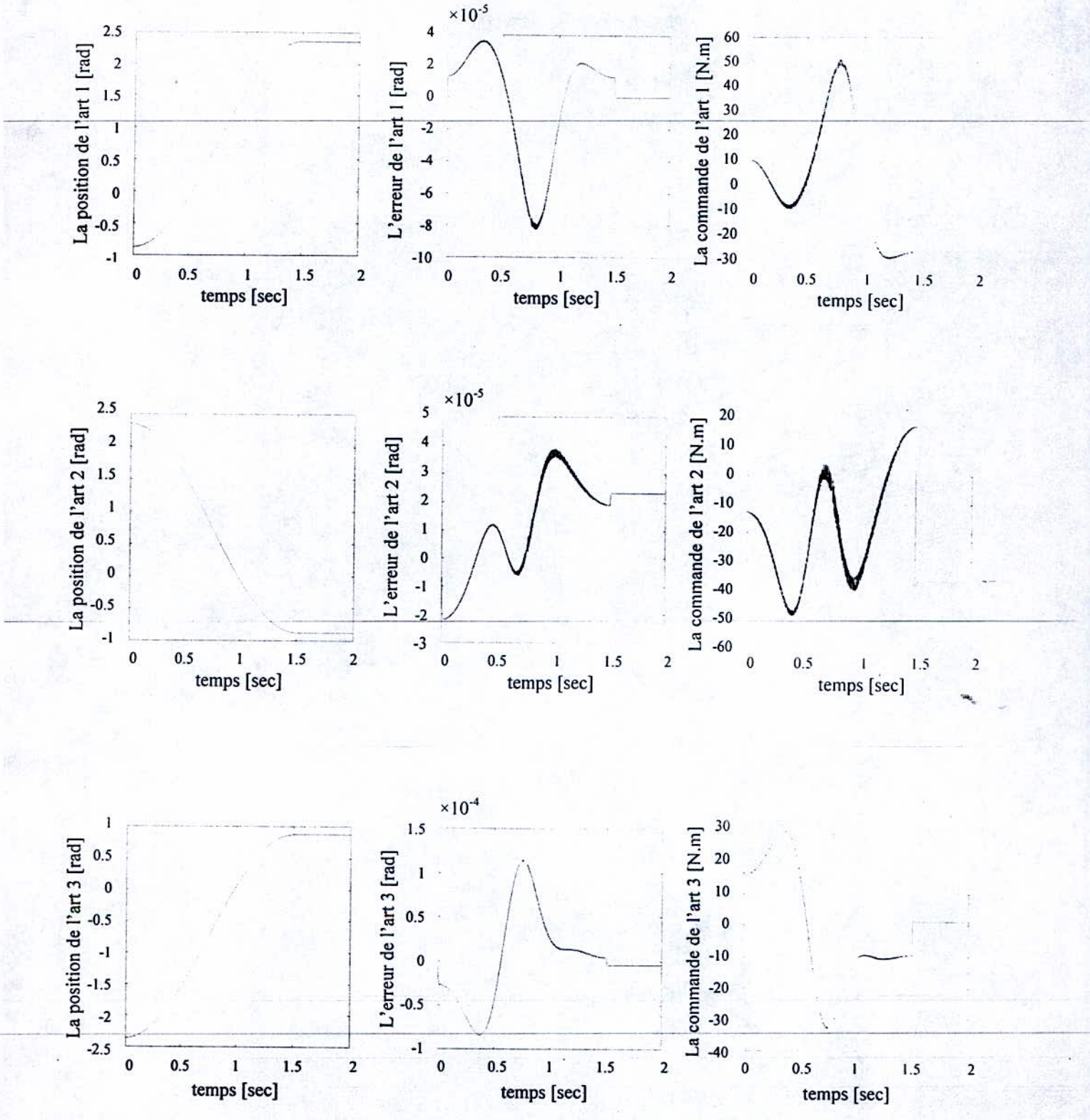


Figure 3.2: Allures des positions, des erreurs et des commandes des trois articulations du robot.

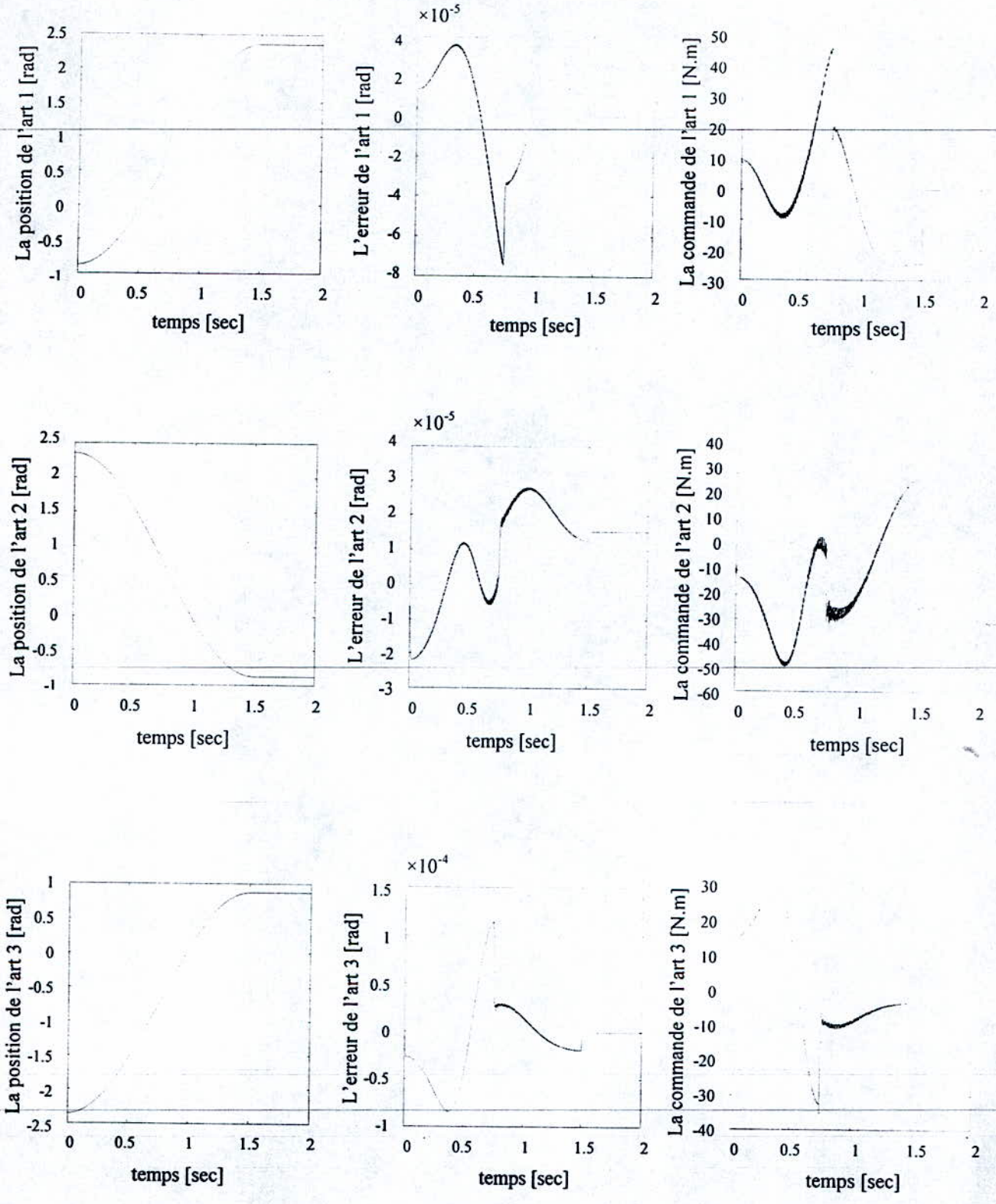


Figure 3.3: Allures des positions, des erreurs et des commandes des trois articulations du robot avec une chute de la masse à t = 0.75 s.

3.3 Approche génétique multiobjectif pour la synthèse des régulateurs H_2 / H_∞ par retour d'état

L'obtention des régulateurs qui assurent une grande robustesse avec une énergie minimale est primordiale dans le domaine industriel, où les objectifs sont dans la plupart des cas en concurrence (énergie, robustesse, temps de répons, etc.).

L'approche H_∞ donne des régulateurs très robustes, mais elle ne tient pas en compte de l'énergie du signal de commande généré par le régulateur, les régulateurs synthétisés par H_2 sont optimaux, mais pas largement robustes, une dégradation de robustesse est probable.

Le mixage H_2 / H_∞, envisage d'obtenir des régulateurs optimaux au sens H_2 avec une contrainte sur les performances H_∞, généralement l'optimisation commence par la minimisation du critère H_2 avec un certain niveau de robustesse demandé, en vérifiant si le critère H_∞ est satisfait, s'il n'est pas vérifié on dégrade le nivaux de performance H_∞ préféré et on refait le même travail, dès fois le problème est non convexe, il est donc très difficile d'atteindre les performances désirées.

Pour cette raison nous avons utilisé une approche génétique multiobjectif pour résoudre ce problème en exploitant toute la puissance de l'algorithme génétique d'un point de vue exploitation et exploration, dans le but d'aboutir à l'ensemble total des régulateurs optimaux.

3.3.1 Commande H_2 / H_∞ pour le réglage par retour d'état

Soit le modèle suivant :

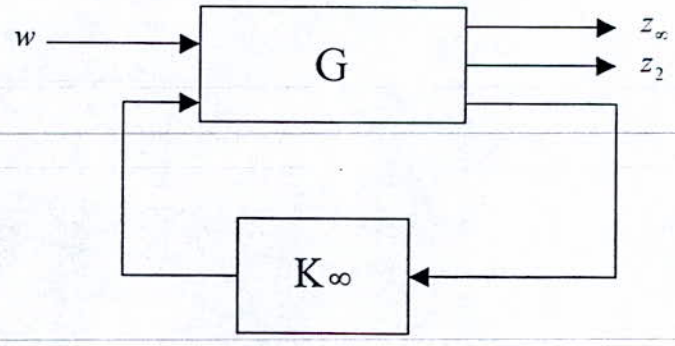


Figure 3.4: Système augmenté par R.E H_2 / H_∞.

avec

$$\begin{aligned} \frac{dx}{dt} &= Ax + B_1 w + B_2 u \\ z_\infty &= C_1 x + D_{11} w + D_{12} u \\ z_2 &= C_2 x + D_{21} w + D_{22} u \end{aligned} \tag{3.18}$$

où w_2 , w_∞ sont des signaux exogènes, z_2 , z_∞ sont des quantificateurs de performances et x est le vecteur d'état qui engendre la loi de commande $u = K_\infty x$.

Le problème H_2 / H_∞ est posé comme suit :

Etant donnée γ un certain nivaux de performance H_∞, trouver un retour d'état H_∞ stabilisant et vérifiant

$$\begin{aligned} &\text{Min } \| T_{w2-z2} \|_2 \\ &K_\infty \textit{ admissible} \end{aligned} \tag{3.19}$$

$$\text{sous : } \| T_{w\infty-z\infty} \|_\infty \leq \gamma \tag{3.20}$$

D'après les relations (3.19) et (3.20) nous pouvons constater que l'objectif est d'obtenir des performance H_2 avec une contrainte sur les performance H_∞, cette dernière perturbe l'optimisation, et la solution est dès fois sous-optimale.

3.3.2 Le compromis entre performances nominales et robustesse

L'utilité du mixage entre H_2 et H_∞ est de séparer les performances nominales et la robustesse, en décomposant le problème du compromis en deux sous-problèmes, performances nominales traitées par H_2, et robustesse traitée par H_∞.

La surface de Pareto des performances H_2 / H_∞ (voir Figure 3.5) donne une vision claire sur le compromis qui existe entre les deux performances. [Who 1997]

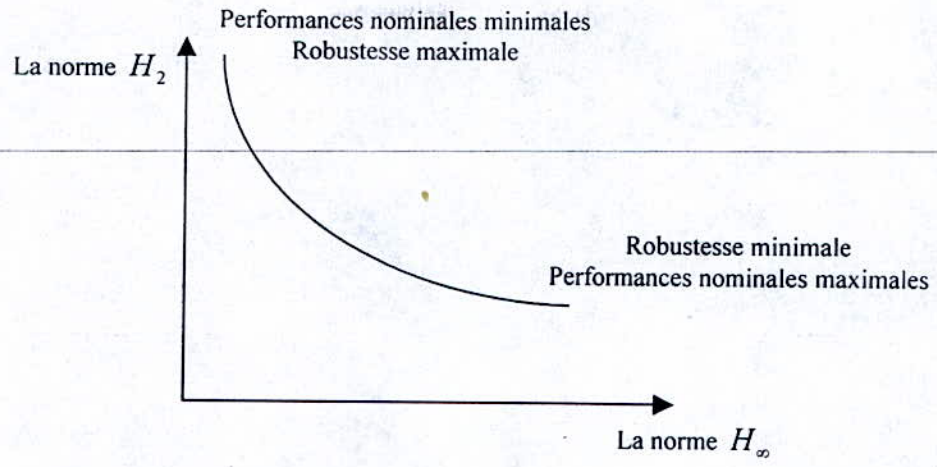


Figure 3.5: Compromis entre performances et robustesse.

3.3.3 Calcul du retour d'état H_∞

Il existe un retour d'état admissible satisfaisant

$$\|T_{w-z\infty}\| \leq \gamma$$

Si et seulement si l'Hamiltonien

$$H_\infty = \begin{bmatrix} A & \gamma^{-2} B_2 B_2^T - B_1 B_1^T \\ C_1^T C_1 & -A^T \end{bmatrix} \tag{3.21}$$

n'a aucune valeur propre purement imaginaire, et X_∞ symétrique définie positive (X_∞ est la solution de l'équation du Riccati:

$$A^T X_\infty + X_\infty A + X_\infty (\gamma^{-2} B_2 B_2^T - B_1 B_1^T) X_\infty + C_1^T C_1 = 0 \tag{3.22}$$

Le régulateur est donné par: $$K_\infty = B_2^T X_\infty \tag{3.23}$$

Il est claire que le régulateur par retour d'état H_∞ dépend indirectement de γ.

3.3.4 Approche génétique multiobjectif

En vue d'obtenir des régulateurs qui donnent le meilleur compromis entre les performances H_2 et performances H_∞, le problème est posé comme suit:

Problématique: Etant donné le modèle standard (voir Figure 3.4) pour le retour d'état H_∞ .

Le problème de H_∞ standard / H_2 avec l'approche génétique multiobjectif, est de trouver l'ensemble des régulateurs tels que:

.

$$\begin{aligned} Min\ J_2 &= \left\| T_{W-Z2} \right\|_2 \\ Min\ J_\infty &= \left\| T_{W-Z_\infty} \right\|_\infty \end{aligned} \tag{3.24}$$

Sous :

$$\begin{aligned} J_2 &\le \nu \\ J_\infty &\le \gamma \quad (H_\infty \text{ Standard}) \end{aligned} \tag{3.25}$$

L'ensemble des compensateurs optimaux ont des performances qui balancent entres performances nominales et robustesse.

3.3.5 Objectifs de la synthèse

Stabilité : la stabilité du système augmenté est primordiale pour une optimisation efficace, cet objectif peut être considéré à la fois comme but à atteindre et également une contrainte.

Les performances H_2 : Elles sont caractérisées par la minimisation de $\left\| T_{w-z2} \right\|_2$. En termes déterministes la norme H_2 d'un système linéaire stable et est égale à l'intégrale du carré de la réponse impulsionnelle, c'est à dire l'énergie totale du signal obtenu, en réponse à une impulsion de Dirac. D'un point de vue stochastique, c'est la puissance du signal de sortie, en réponse à un bruit blanc Gaussien normalisé. [Maz 1999]

L'idée de base de la commande H_2, vient de la transformation du problème de la commande LQG (qui est typiquement temporelle), en un problème dans le domaine fréquentiel, donc l'approche H_2 traite bien le problème d'énergie de la commande, et le suivi en régime permanent.

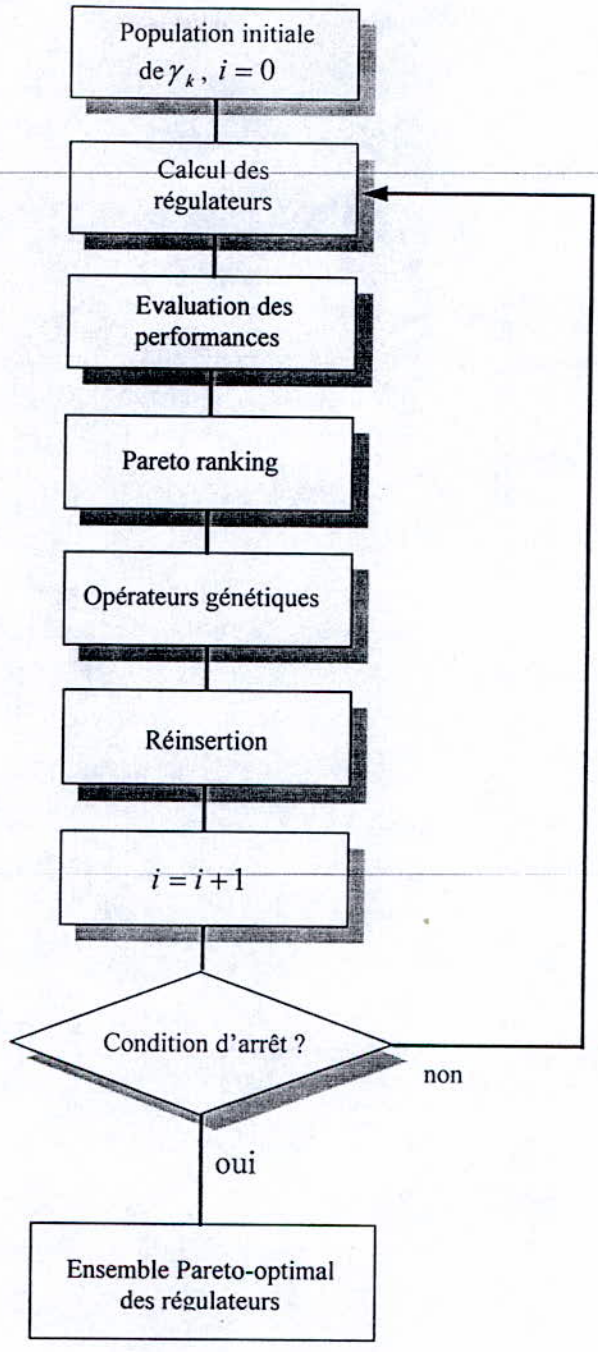


Figure 3.6: Implémentation d'un MOGA pour trouver les régulateurs Pareto-optimaux.

Les performances H_∞ : Elles sont caractérisées par la minimisation de $\|T_{w-z\infty}\|_\infty$, afin d'obtenir une meilleure robustesse vis-à-vis des incertitudes et autres performances comme le rejet de perturbations et le suivi.

Placement de pôles : Le placement de pôles influe sur la forme des réponses du système en boucle fermée vis-à-vis des bruits, des réponses bien amorties sont préférables.

3.3.6 Déroulement de l'algorithme

Premièrement, une population γ_i a été initialisée aléatoirement. Ensuite, chaque retour d'état est calculé selon les équations (3.22) et (3.23) ce qui permet l'évaluation des performances du système bouclé, tout régulateur ne vérifiant pas la stabilité et la condition (3.25) est écarté, une sélection selon l'ordre de dominance donné par le Pareto-ranking a été faite pour déterminer les régulateurs non-dominés. Enfin des opérateurs génétiques comme le sharing, le croisement et la mutation sont appliqués pour donner à la fois diversité et convergence, et ceci pour plusieurs générations (voir Figure 3.6).

3.3.7 Exemple d'application

Dans cette section nous appliquons l'approche génétique multiobjectif sur un problème posé dans **[MATLAB LMI Toolbox User Guide]**.

Le système est un satellite composé de deux corps rigides (corps principal et module d'instrumentation) lié par un lien flexible. Ce dernier est modélisé comme une force avec un moment k et le facteur visqueux f.

Modèle mathématique

Les équations dynamiques du satellite sont données par :

$$\begin{cases} J_1\ddot{\theta} + f(\dot{\theta}_1 - \dot{\theta}_2) + k(\theta_1 - \theta_2) = T + w \\ J_2\ddot{\theta}_2 + f(\dot{\theta}_2 - \dot{\theta}_1) + k(\theta_2 - \theta_1) = 0 \end{cases} \tag{3.26}$$

où : θ_1, θ_2 sont les angles d'orientation du corps principal et du module d'instrumentation respectivement.

T : le moment (la commande).

w : est un bruit sur le moment.

L'analyse des éléments finis nous a donné les incertitudes suivantes :

$$0.09 \le k \le 0.4 \quad \text{et} \quad 0.0038 \le f \le 0.04$$

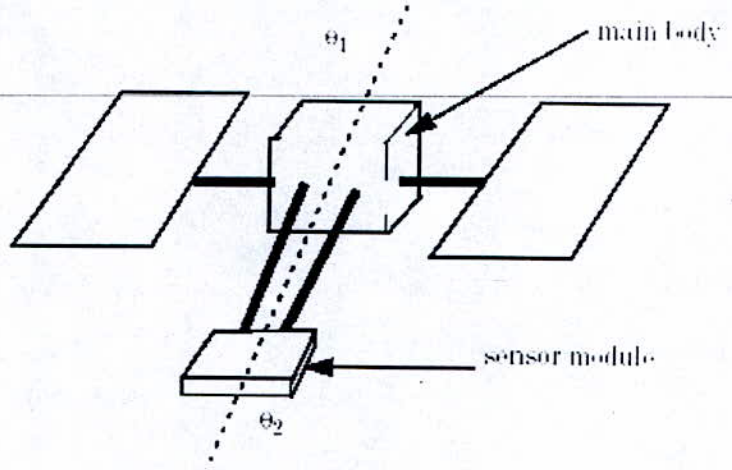


Figure 3.7: Satellite.

Objectifs de la commande

En se basant sur le modèle de synthèse pour un retour d'état via l'approche mixte H_2 / H_∞ , nous avons

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & J_1 & 0 \\ 0 & 0 & 0 & J_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k & k & -f & f \\ k & k & f & -f \end{bmatrix} + \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} (w + T) \qquad (3.27)$$

$$z_\infty = \theta_2$$

$$z_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} T = \begin{bmatrix} \theta_1 \\ \theta_2 \\ T \end{bmatrix} \qquad (3.28)$$

Le but de la commande est la synthèse d'un retour d'état robuste $T = Kx$ qui minimise l'influence du bruit w sur la position angulaire θ_2, ce but peut être réalisé en considérant (3.24) et (3.25).

Implémentation : un AG multiobjectif a été lancé pour achever l'ensemble Pareto des régulateurs optimaux au sens H_2 / H_∞, une population de 100 individus ayant chacun une longueur de 32 bits a été évolué pendant 50 génération avec $p_c = 0.7, p_m = 10^{-3}$, $\gamma \in [10^{-9}.1]$

Résultats et discussion

Les résultats obtenus montrent clairement le compromis qui existe entre les performances H_2 (énergie, variance) et les performance H_∞ (robustesse et rejet de perturbation) ,(voir Figure 3.8). Par ailleurs, nous constatons l'apparition des modes oscillants pour les régulateurs obtenus (retour d'état H_∞) (voir Figure 3.9). Le choix du régulateur approprié dépend des préférences (énergie, robustesse désirée, réponse désirée), ces dernières dépendent à leur tour de la tâche effectuée par le système.

Enfin, un test de robustesse a été effectué, nous constatons la robustesse des régulateurs obtenus vis à vis des variations paramétriques (voir Figure 3.9).

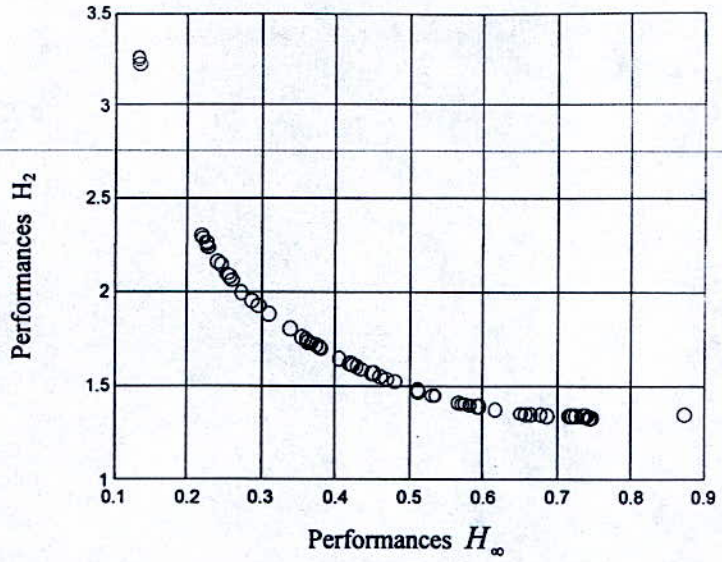


Figure 3.8: Le compromis entre les performances H_2 et H_∞.

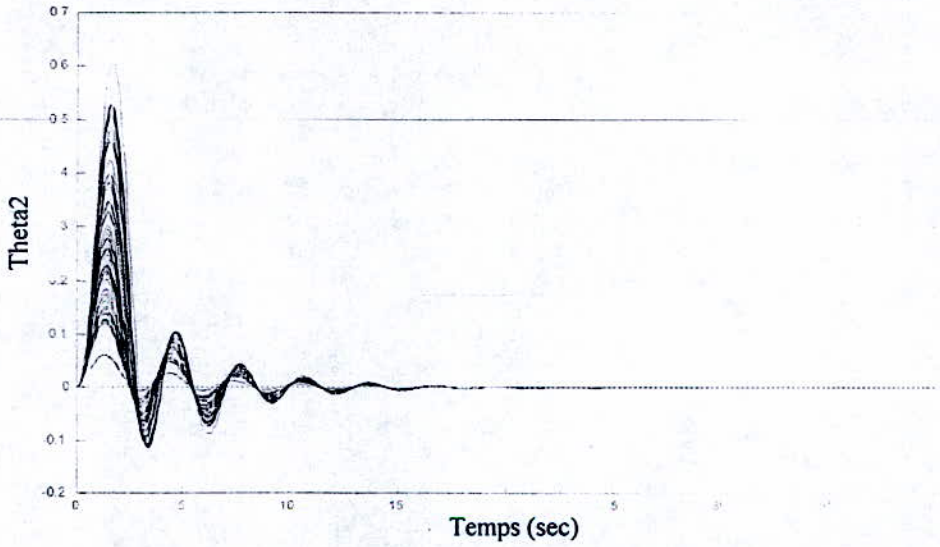


Figure 3.9: effet de perturbation sur θ_2 pour tout les régulateurs.

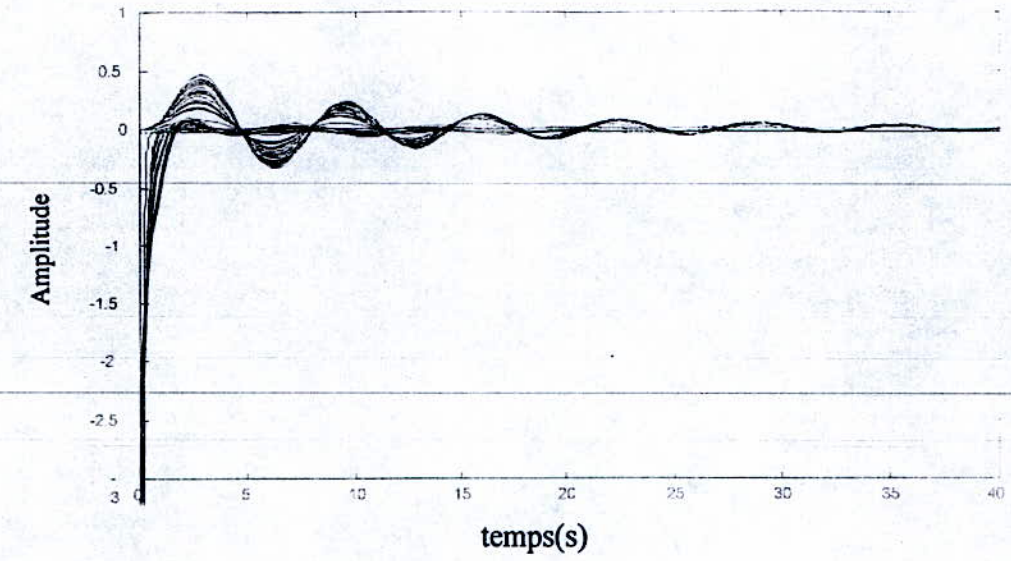


Figure 3.10: Signaux de commande engendrés par les régulateurs optimaux.

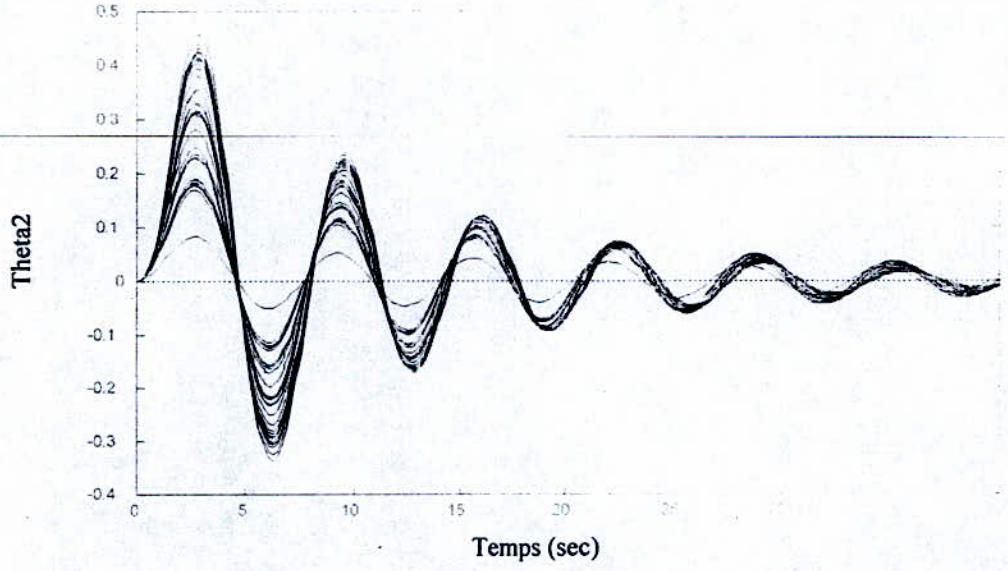


Figure 3.11: Test de robustesse.

3.4 Identification non linéaire par Algorithmes génétiques

L'identification des systèmes est l'un des problèmes classiques de l'automatique. Elle peut être définie comme un processus expérimental pour la construction d'un modèle mathématique à partir des observations d'entrée et de sortie du système. En d'autres termes, c'est l'art de créer une description mathématique d'un système dynamique inconnu.

Un système dynamique vu comme un concept mathématique, peut être considéré comme une structure qui reçoit une entrée $u(t)$ à chaque instant t, et génère une sortie $y(t)$. Dans la plupart des cas, la sortie $y(t)$ dépend non seulement de $u(t)$, mais aussi de l'histoire passée des entrées, et donc celle du système. Un système peut donc être représenté comme il est illustré dans la Figure 3.12, où le système est excité par la variable d'entrée **u** et la perturbation **e**, et **y** est la variable de sortie du système.

Dans ce sens, l'identification des systèmes linéaires a été largement abordée (Ljung, 1987; Söderströmand Stoica, 1989) où le processus d'identification se concentre sur la détermination de l'ordre et les paramètres associés des termes impliqués dans une description (mathématique) sous forme d'une fonction linéaire dynamique. Cependant, la plupart des systèmes réels sont non-linéaires, et les modèles linéaires ne peuvent en aucun cas refléter complètement et précisément le comportement dynamique exposé par ces systèmes. L'identification des systèmes non linéaires est un problème plus complexe que son équivalent linéaire puisqu'on doit aussi identifier quels termes non linéaires sont impliqués.

3.4.1 La procédure d'identification [Rod 1999]

Un rappel général sur la procédure d'identification est illustré dans la Figure 3.13. Ce processus se compose des étapes suivantes (Norton, 1986; Söderström et Stoica, 1989):

1 L'expérimentation Cette étape est en relation avec le choix du signal d'entrée. Le système est alors excité et ses sorties sont observées après chaque période d'échantillonnage.

2 Caractérisation du système Ceci concerne le choix d'une classe générale appropriée pour laquelle le modèle à utiliser pour l'identification appartient. Dans cette étape, une structure de représentation spécifique est sélectionnée dans cette classe.

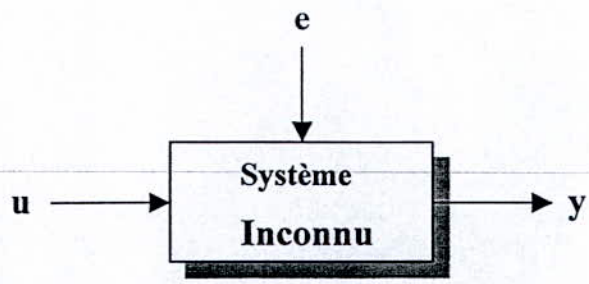


Figure 3.12: Un système avec entrée **u**, sorite **y** et perturbation.

3 Sélection d'un modèle Un modèle est choisi à partir des modèles candidats appartenant à la structure de représentation des modèles utilisée pour l'identification.

4 Estimation des paramètres Les paramètres associés au modèle sélectionné sont calculés en utilisant un algorithme d'estimation. La plupart des algorithmes d'estimation travaillent en minimisant un certain critère de performance (par exemple la somme des carrés des erreurs).

5 Validation du modèle Le modèle identifié est testé ici, généralement, en utilisant un nouvel ensemble de données, dans le but de vérifier si le modèle choisit décrit adéquatement l'ensemble des données.

Dans cette procédure, si le modèle identifié ne satisfait pas le critère d'adaptation et le modèle de validation, alors un nouveau modèle sera choisi, et ses paramètres estimés jusqu'à ce que les critères d'adaptation et de validation soient satisfaits.

3.4.2 Représentation des modèles *NARMAX*

Leontaritis et Billings (1985) ont introduit le modèle *NARMAX* (Non-linear AutoRegressive Moving Average with eXogenous inputs) qui n'est rien d'autre qu'une extension de la description *ARMAX* pour la représentation des systèmes non-linéaires. Ce modèle est utilisé pour identifier, une large classe de systèmes non linéaires. Il est donné par une fonction non-linéaire F de la sortie $y(k)$, l'entrée $u(k)$ et probablement d'une perturbation $e(k)$. D'où.

$$y(k) = F(y(k-1),\ldots, y(k-n_y), u(k-1),\ldots, u(k-n_u), e(k-1),\ldots, e(k-n_e)) + e(k) \qquad (3.29)$$

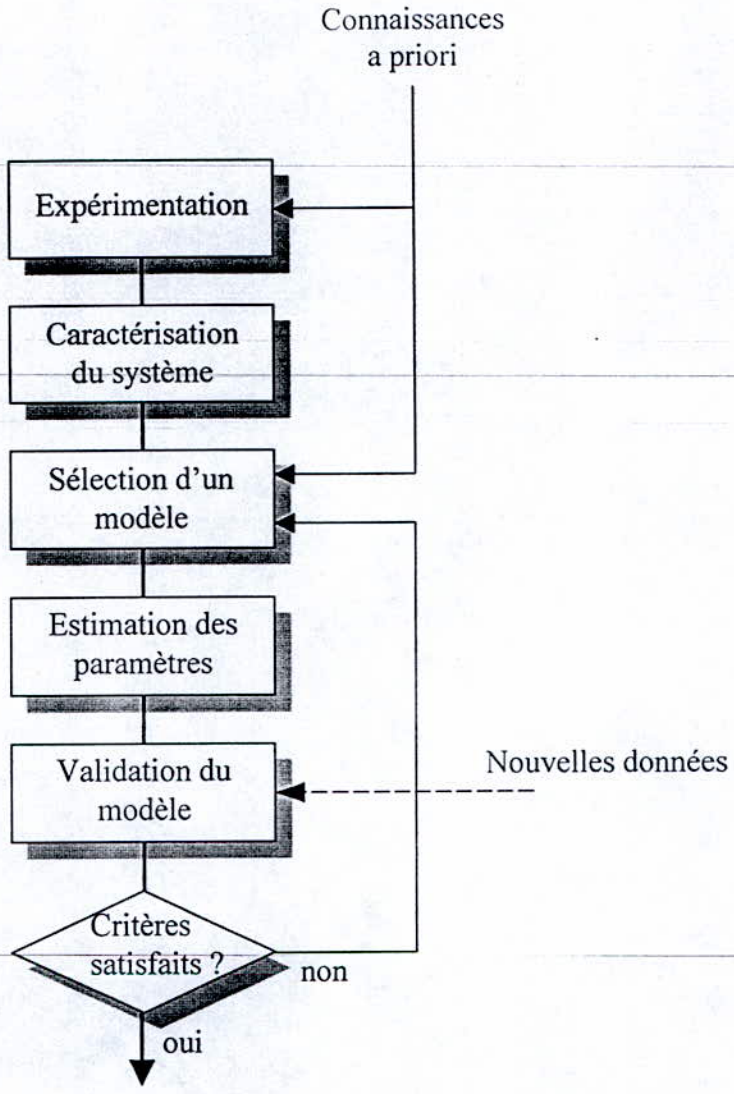


Figure 3.13: Etapes de la procédure d'identification.

Les entiers positifs n_y, n_u et n_e sont les ordres dynamiques correspondants, et $\{e(k)\}$ est une séquence indépendante de moyenne nulle. En pratique, la fonction non-linéaire F peut être approximée, par exemple, par un polynôme de degré l, ainsi on obtient la représentation suivante

$$y(k) = \theta_0 + \sum_{i_1=1}^{n} \theta_{i_1} x_{i_1}(k) + \sum_{i_1=1}^{n} \sum_{i_2=i_1}^{n} \theta_{i_1 i_2} x_{i_1}(k) x_{i_2}(k) + \cdots$$

$$+ \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}}^{n} \theta_{i_1 \cdots i_l} x_{i_1}(k) \cdots x_{i_l}(k) + e(k) \qquad (3.30)$$

où $n = n_y + n_u + n_e$, et tous les θ représentent des coefficients scalaires, et $x(k)$ représente les termes en y, u, ou e avec retard ($y(k-1), ..., u(k-1), ..., e(k-1)$).

L'équation appartient clairement au modèle de régression linéaire suivant

$$y(k) = \sum_{i=1}^{M} \theta_i \, p_i(k) + \xi(k) \qquad k = 1, \ldots, N \tag{3.31}$$

où N est la longueur de la séquence des données, $p_i(k)$ représente tous les termes possibles linéaires, non-linéaires et combinés de l'entrée, de la sortie et du bruit, de degré maximal égal à l, et M est le nombre total de ces termes. $\xi(k)$ est l'erreur résiduelle et θ_i sont des paramètres inconnus à estimer. L'équation peut être aussi écrite sous la forme matricielle

$$\begin{aligned} \mathbf{y} &= [\mathbf{p}_1 \cdots \mathbf{p}_M]\boldsymbol{\theta} + \boldsymbol{\xi} \\ &= \mathbf{P}\boldsymbol{\theta} + \boldsymbol{\xi} \end{aligned} \tag{3.31}$$

Le modèle en polynôme donné au-dessus est donc non-linéaire en entrée, en sortie et en termes de bruit, mais il est toutefois linéaire vis à vis des paramètres. Le vecteur des paramètres peut être estimé en minimisant la norme Euclidienne $\|\mathbf{y} - \boldsymbol{\theta}\mathbf{P}\|$, c'est à dire en résolvant par exemple un problème de Moindres Carrées (MC). Dans notre cas, nous utiliserons pour l'estimation des paramètres la méthode d'estimation orthogonale.

3.4.3 La sélection des termes [Fon 1995]

La recherche d'une structure de modèle convenable, comme décrit plus haut, peut être vue comme un problème de sélection de sous-ensemble de termes non-linéaires. En général, on préfère avoir une structure de modèle avec le moins de termes possible, cependant, l'augmentation du nombre de termes avec le degré de non-linéarité et les ordres dynamiques est très rapide. Une structure de sept termes est généralement acceptable et donne dans beaucoup de cas des résultats satisfaisants.

Dans ce qui suit, le choix d'un codage convenable pour les individus et les opérateurs associés sont discutés.

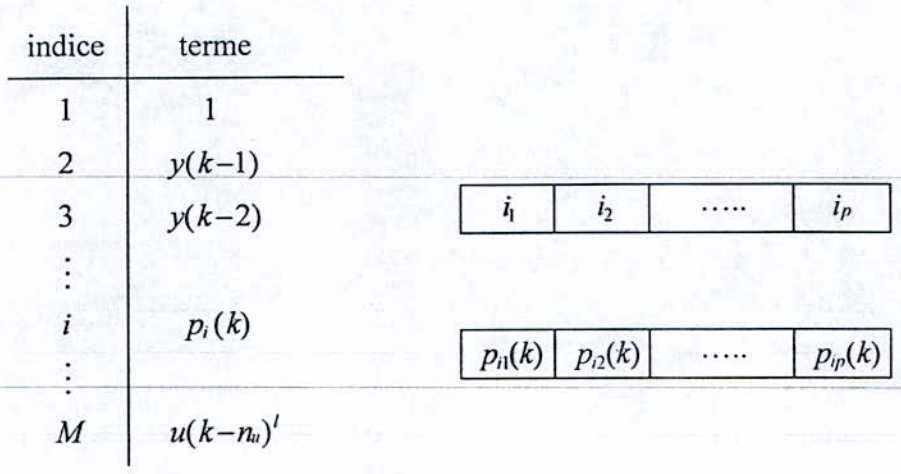


Figure 3.14: Codage des chromosomes.

3.4.3.1 Codage des chromosomes

Lucasius et Kateman ont proposé une méthode de représentation des sous-sembles de termes convenant parfaitement pour la recherche de sous-ensembles de petites longueurs. Une table contenant tous les termes possibles d'un problème donné (d'après les valeurs de n_u, n_y, n_e et l) est établie, chaque terme possible correspondant à ce problème est représenté par son indice dans cette table (voir Figure 3.14). Les individus de la population sont donc représentés par des sous-ensembles de p termes parmi l'ensemble global de tous les termes possibles (p est la longueur de l'individu).

3.4.3.2 Les opérateurs génétiques

Les opérateurs génétiques pour le problème de sélection des termes devraient garantir la génération de modèles valides à chaque instant, de préférence sans introduction de redondance. La redondance existe quand il y a plusieurs représentations différentes d'un même individu, c'est à dire, un individu qui possède au moins deux gènes identiques (voir Figure 3.15). Une description brève du fonctionnement des deux opérateurs de croisement et de mutation est donnée au-dessus, de légères modifications ont été introduites dans le but d'éliminer toute possibilité de redondance.

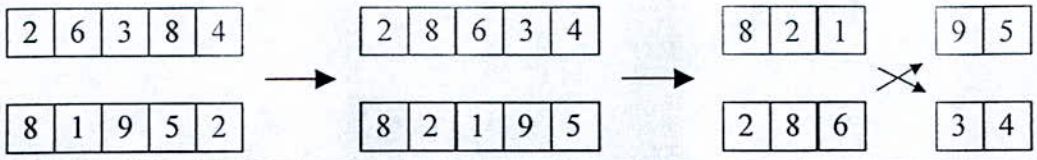


Figure 3.15: Opérateur de croisement pour le problème d'identification.

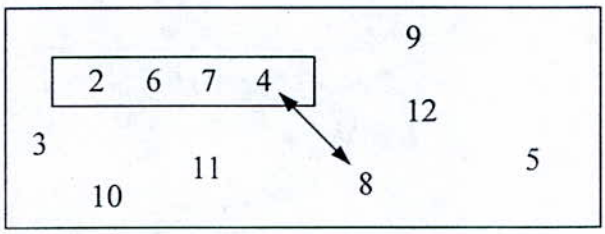


Figure 3.16: Opérateur de mutation pour le problème d'identification.

Croisement Il se fait en éloignant tout d'abord les gènes identiques des deux parents, puis en croisant les deux portions de gènes non identiques restantes. Par conséquent, la redondance est éliminée et si les parents ne sont pas identiques la création de nouvelle descendance est alors assurée.

Mutation Cet opérateur teste chaque élément du sous-ensemble, et le remplace par un élément quelconque sélectionné aléatoirement à partir du complémentaire de ce sous-ensemble, avec une certaine probabilité (voir Figure 3.16).

3.4.4 Application à un simple processus de Wiener [Rod 1999]

Dans cette section nous appliquons la méthode d'identification non-linéaire par les AGs comme elle a été décrite plus haut sur un simple processus de Wiener, ce processus est un cas particulier des systèmes qui sont linéaires en entrée et non-linéaires en sortie. L'équation différentielle de la partie dynamique du processus de Wiener est

$$\dot{v}(t) + v(t) = u(t) \tag{3.33}$$

et la partie statique non-linéaire est donnée par

$$y(k) = 2 + v(k) + v^2(k) \tag{3.34}$$

Le processus décrit au-dessus est excité par un signal aléatoire gaussien bruité. Ce signal comporte 200 points de données dont les 100 premiers points sont utilisés dans l'estimation des paramètres des modèles, tandis que les 100 autres points sont utilisés pour la validation du modèle final. Les maximums des ordres dynamiques et le degré de non-linéarité sont $n_u=n_y=5$, $n_e=0$ et $l=3$.

Un AG est considéré avec une population de 50 individus et est lancé pour 50 générations. La sélection utilisée est la sélection SUS, les taux de croisement et de mutation sont $p_c=0.7$, $p_m=0.1$, le *generation gap* est égal à 0.9 et on utilise enfin une réinsertion élitiste. L'évaluation des individus se fait via une fonction objective qui n'est rien d'autre que la variance des résidus et est donnée comme suit

$$\hat{\sigma}_\varepsilon^2(\theta_p) = \sum_{k=1}^{N} \varepsilon^2(k) = \sum_{k=1}^{N} \left[y(k) - \hat{y}(k/\theta_p) \right]^2 \tag{3.35}$$

où

$$\varepsilon(k) = y(k) - \hat{y}(k/\theta_p) \qquad k = 1,\ldots..,N \tag{3.36}$$

sont les erreurs prédictives ou résidus, associés au vecteur des paramètres θ_p et $\hat{y}(k/\theta_p)$ est la sortie prédictive à un pas correspondant à ces mêmes paramètres, N est le nombre de points de données utilisé pour l'estimation, soit 100 dans notre cas.

La Figure 3.17 montre l'évolution de la variance des résidus correspondant au meilleur individu à chaque génération. On remarque que l'AG a convergé vers une solution optimale dont la variance est égale à 4.33×10^{-5}. Cette solution est représentée par le meilleur individu de la population à la dernière génération, le modèle associé à cet individu est le suivant

$$\begin{aligned} y(k) = {} & a_1 + a_2 u(k-2) u^2(k-4) + a_3 u(k-1) u(k-5) y(k-3) + a_4 u(k-1) y^2(k-2) + \\ & a_5 y(k-1) + a_6 u(k-4) u(k-5) y(k-4) + a_7 u(k-1) u(k-2) y(k-3) \end{aligned} \tag{3.37}$$

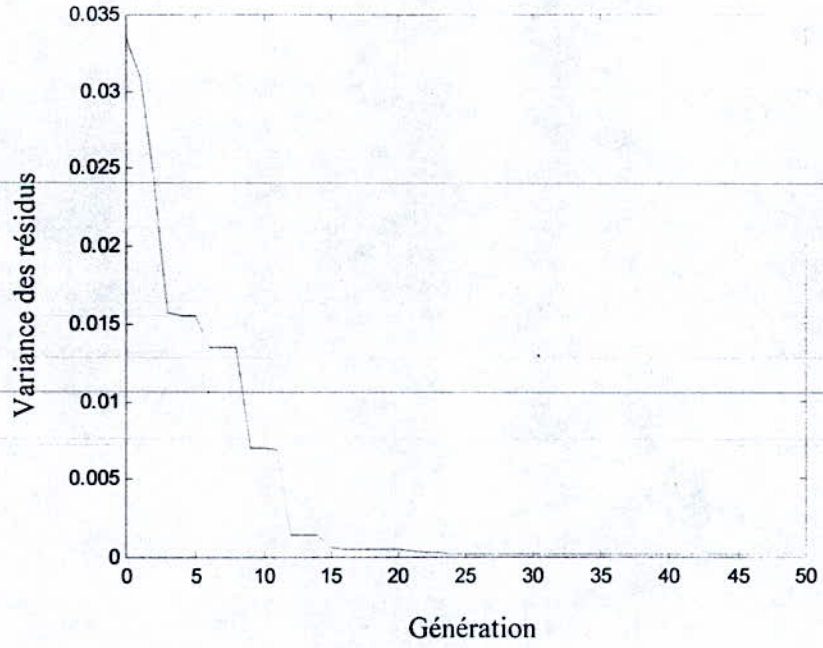
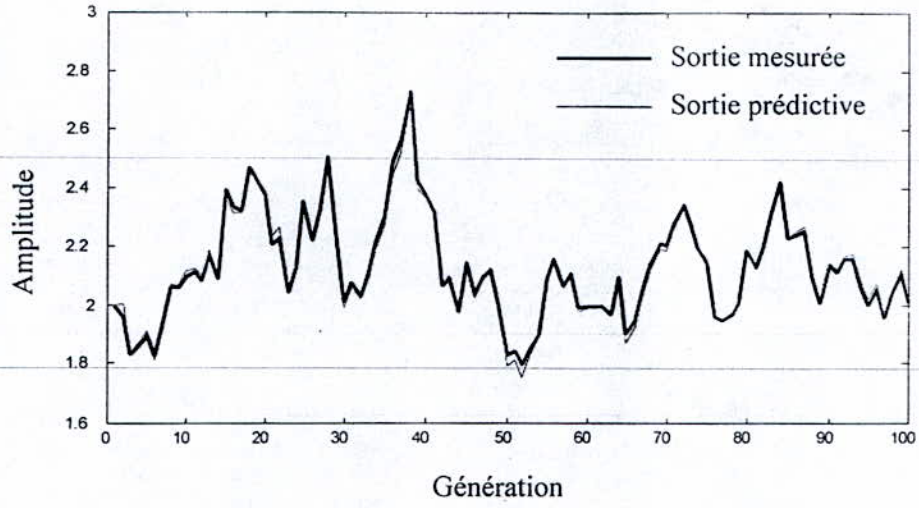
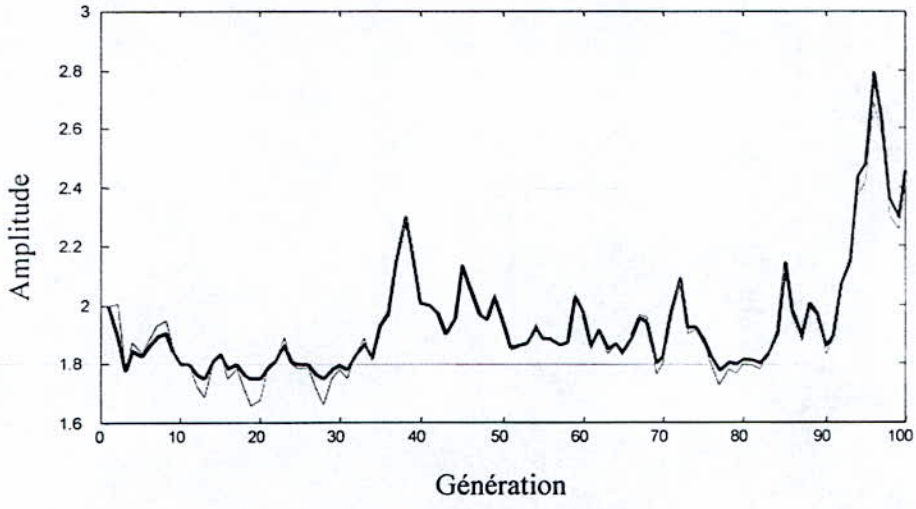


Figure 3.17: Evolution de la variance des résidus du meilleur individu.

La Figure 3.18 donne les sorties mesurées et prédictives du processus de Wiener considéré, pour les deux types de données d'estimation et de validation. On remarque que le modèle trouvé reflète très significativement la dynamique du système bien qu'il comporte seulement sept termes, sans oublier que l'AG a été lancé pour 50 générations seulement et évoluant sur une population de taille 50. En effet, les résultats obtenus sont spécifiques à la configuration de l'AG faite plus haut, en outre, ces résultats ne sont pas définitifs du fait qu'ils varient d'un essai à un autre de l'AG.



(a)



(b)

Figure 3.18: Sorties mesurées et prédictive. (a) Avec les données d'estimation (b) Avec les données de validation.

3.5 Conclusion

Dans ce chapitre, nous avons prouvé la grande capacité des AGs à affronter des problèmes de commande et d'identification des systèmes. L'optimisation des paramètres d'une loi de commande, obtenue via l'approche de backstepping, avec les AGs était considérée et les résultats obtenus étaient satisfaisants du point de vue performances, ce qui valide certainement l'efficacité des AGs dans la synthèse off-line indirecte des loi de commandes. L'implémentation d'un algorithme génétique multiobjectif pour la synthèse des régulateurs mixtes H_2 / H_∞ a montré la capacité des AGs de résoudre des problèmes d'optimisation

multiobjectif, le résultat est un ensemble de régulateurs, et le régulateur final est le sujet du choix de l'utilisateur selon les performances préférées. Enfin, l'utilisation des AGs pour l'identification des systèmes de type *NARMAX* a abouti à une structure de modèle très significative, ceci prouve sans aucun doute la flexibilité des AGs de résoudre des problèmes très complexes, tels que l'identification non-linéaire.

Conclusion générale

Dans ce travail, nous avons présenté l'aspect théorique et pratique des algorithmes génétiques. Les applications effectuées ont montré la grande efficacité et utilité des AGs dans le domaine général de l'optimisation, ainsi que le domaine de la commande et d'identification des systèmes. Par ailleurs, l'utilisation d'une approche multiobjectif s'avère une méthode nécessaire et indispensable pour la résolution des problèmes de commande et d'identification. En effet, la plupart des applications de commande et' d'identification sont des problèmes multiobjectif qui présentent des objectifs souvent en concurrence. Les AGs ont prouvé leur efficacité pour affronter de tels problèmes.

L'optimisation des paramètres d'une loi de commande obtenue à partir de la technique de Backstepping a été réalisée avec un AG. Les résultats obtenus étaient très satisfaisants du point de vue performances du système commandé. Le problème était relativement simple, il suffisait de calculer la loi de commande stabilisante afin de coder les paramètres de cette loi dans un individu, et de préciser un espace de recherche approprié.

Par ailleurs, nous avons réalisé une optimisation multiobjectif via l'approche H_2 et H_∞ en vue d'obtenir des régulateurs qui assurent à la fois des performances nominales, et une robustesse vis à vis des incertitudes. L'ensemble des régulateurs obtenu a reflété très largement les solutions non-dominées du problème considéré. Par conséquent, l'utilisateur peut choisir parmi touts ces solutions suivant ses préférences concernant les performances du système commandé.

Enfin, l'utilisation d'un AG pour le choix d'une structure convenable pour le problème d'identification non-linéaire a montré la capacité des AGs de traiter des problèmes qui manipulent des données assez complexes. Un codage particulier des chromosomes et des opérateurs génétiques légèrement modifiés ont finalement permis de traiter très efficacement le problème de la sélection des termes.

Références bibliographiques

[And 2001] J. Anderson. Multiobjective Optimisation in Engineering Design, Application to Fluid Power Systems. Linköpings Universitet, Sweden, 2001.

[Bac 1996] T. Bäck. Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York, 1996.

[Ber 2001] A. Berro. Optimisation Multiobjectif et Stratégie d'Evolution en Environnement Dynamique. Thèse de Doctorat, Université des Sciences Sociales Toulouse I, 2001.

[Bor 1998] Bor-Sen Chen. A Structure-Specified Hinf Optimal Control Design. IEEE Transactions on Control Systems Technology, Vol. 6, No. 6, November 1998.

[Chi 1994] Ching-Fang. Advanced Control Systems Design. PTR Prentice Hall, 1994.

[Deb 1999] K. Deb. Multiobjective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, Evolutionary Computation, 1999.

[Deb 2000] K. Deb. S. Agrawal, S. Pratap, A. Meyarivan, Constrained Test Problems for Multiobjective Genetic Optmization, 2000.

[Fle 2001] P. J. Fleming, R. C. Purshouse. Genetic Algorithms in Control Systems Engineering. Research Report No. 789, University of Sheffield, 2001.

[Fon 1993] C. M. Fonseca, P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Proceeding of the Fifth International Conference on Genetic Algorithms, San Mateo, California, 1993.

[Fon 1995] C. M. Fonseca. Multiobjective Genetic Algorithms with Application to Control Engineering Problems. PhD thesis, University of Sheffield, 1995.

[Gha 1998] S. Gharbi, F. Hadj Miloud. Commande Adaptative Décentralisée par la technique de Backstepping, Application en Robotique. Thèse de PFE, ENP, 1998.

[Gol 1992] D. E. Goldberg, J. J. Richardson. Genetic Algorithm with Sharing for Multimodal Function Optimization, Genetic Algorithms and their Applications. Lawrence Erlbaum Associates, Hillsdales, 1992.

[Gol 1994] D. E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts, 1989.

[Hoc 2001] A. Hocine, D. Mazouni. Utilisation des Algorithmes Génétiques en Identification et en Commande des Systèmes. Thèse de PFE, ENP, 2001.

[Hol 1975] **J. H. Holland**. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.

[Jai 1997] S. Jain, F. Khorrami. Decentralized Adaptive Control of a Class of Large-Scale Interconnected Non-linear Systems. IEEE transaction on automatic control, vol 42, no 2, pp 136-154, 1997.

[Lan 1988] J. D. Landau. Identification et Commande des Systèmes, Hermès, Paris, 1988.

[Mah 1995] S. W. Mahfoud. Niching Methods for Genetic Algorithms. Illigal TR. N° 95001, University of Illinois, Urbana, May. 1995.

[Maz 1999] S. Maza. Etude de l'interaction entre l'identification et commande robuste, théorie et applications. Thèse de PFE, 1999.

[Mit 1998] M. Mitchell. An Introduction to Genetic Algorithms. The MIT press, Massachusetts Institute of Technology, 1998.

[Ren 1995] J. M. Renders. Algorithmes Génétiques et Réseaux de Neurones, Application à la commande des processus. Hermès, Paris, 1995.

[Rod 1999] K. Rodriguez. Multiobjective Evolutionary Algorithms In Non-Linear System Identification. PhD Thesis, University of Sheffield, 1999.

[Whi 1997] J. F. Whidborne. Multi-Objective Robust Control System Design. King's College London Strand, London WC2R 2LS, UK.

[Who 1997] M. S. Whorton. High Performance, Robust Control of Flexible Space Structures. PhD thesis, Georgia Institute of Technology, 1997.

[Zit 1998] E. Zitzler, L. Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. TIK-Report n° 43, 1998.

[Zit 1999] E. Zitzler, K. Deb, L. Thiele. Comparison of Multiobjective Evolutionary, Algorithms: Empirical Results. TR. N° 70, Swiss Federal Institute of Technology, Zurich, 1999.

Annexe

Modèle dynamique du robot manipulateur Puma 560

Le modèle dynamique du robot Puma 560 peut être représenté par un système d'équations différentielles non-linéaires du second ordre qui peut s'écrire sous la forme matricielle suivante:

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + G(q) = u - mJ^T(q)\left[J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} + g\right]$$

Le robot Puma 560 se compose de trois sous-systèmes (trois articulations), chaque sous-système peut être représenté par les équations d'état suivantes

$$\begin{cases} \dot{z}_{i,1} = z_{i,2} \\ \dot{z}_{i,2} = \dfrac{1}{I_i} u_i + \phi_i \end{cases} \qquad i = 1, 2, 3$$

où

I_1 =2.57, I_2=6.79, I_3=1.16

$$\phi_i = \dot{z}_{i,2} - \frac{1}{I_i}\left[(M+mJ^TJ).[\dot{z}_{1,2}\ \dot{z}_{2,2}\ \dot{z}_{3,2}]^T + (B+mJ^T\dot{J}).[z_{1,2}\ z_{2,2}\ z_{3,2}]^T + G + mJ^Tg\right]$$

Les matrices M, B, G, J sont données comme suit

Matrice d'inertie M

$$m_{11} = 2.57 + 1.38C_2^2 + 0.3S_2S_3 + 0.744C_2S_{23}$$

$$m_{12} = m_{21} = 0.69S_2 - 0.134C_{23} + 0.0238C_2$$

$$m_{13} = m_{31} = -0.134C_{23} - 0.00397S_{23}$$

$$m_{22} = 6.79 + 0.744S_3$$

$$m_{23} = m_{32} = 0.333 + 0.372S_3 - 0.011C_3$$

$$m_{33} = 1.16$$

Matrice B

$$b_{11} = (-2.76S_2C_2 + 0.744C_{223} + 0.6S_2C_3 - 0.0213(1 - 2S_2S_3))\,\dot{q}_2$$

$$b_{12} = (0.69C_2 + 0.134S_{23} - 0.0238S_2)\dot{q}_2 + (0.267S_{23} - 0.00758C_{23})\dot{q}_3$$

$$b_{13} = (0.744C_2C_{23} + 0.6S_2C_3 + 0.022S_{23}C_2 - 0.0213(1 - 2S_2S_3))\dot{q}_1 +$$

$$0.5(0.0267S_{23} - 0.00758C_{23})\dot{q}_3$$

$$b_{21} = -0.5(-2.76S_2C_2 + 0.744C_{223} + 0.6S_2C_3 - 0.0213(1 - 2S_2S_3))\dot{q}_1$$

$$b_{22} = (0.022S_3 + 0.744C_3)\dot{q}_3$$

$$b_{23} = 0.5(0.022S_3 + 0.744C_3)\dot{q}_3$$

$$b_{31} = -0.5(0.744C_2C_{23} + 0.6S_2C_3 + 0.022S_{23}C_2 - 0.0213(1 - 2S_2S_3))\dot{q}_1$$

$$b_{32} = -0.5(0.022S_3 + 0.744C_3)\dot{q}_2$$

$$b_{33} = 0$$

Vecteur gravitationnel G

$$g_1 = 0$$

$$g_2 = -37.2C_2 - 8.4S_{23} + 1.02S_2$$

$$g_3 = -8.4S_{23} + 0.25C_{23}$$

Matrice Jacobinne J

$$J_{11} = -S_1 (a_2 C_2 + a_3 C_{23}) - (d_2 + d_3) C_1 - d_4 S_1 S_{23}$$

$$J_{12} = -C_1 (a_2 S_2 + a_3 S_{23}) - d_4 C_1 C_{23}$$

$$J_{13} = -a_3 C_1 S_{23} + d_4 C_1 C_{23}$$

$$J_{21} = -C_1 (a_2 C_2 + a_3 C_{23}) - (d_2 + d_3) S_1 + d_4 C_1 S_{23}$$

$$J_{22} = -S_1 (a_2 S_2 + a_3 S_{23}) + d_4 S_1 C_{23}$$

$$J_{23} = -a_3 S_1 S_{23} + d_4 S_1 C_{23}$$

$$J_{31} = 0$$

$$J_{32} = -(a_2 C_2 + a_3 C_{23}) - d_4 S_{23}$$

$$J_{33} = -a_3 C_{23} - d_4 S_{23}$$

Où

$a_2 = 0.4319$

$a_3 = -0.0203$

$d_2 = 0.2435$

$d_3 = -0.0934$

$d_4 = 0.4331$

ملخص :

في هذا للعمل قمنا بدراسة الخوارزميات الجينية و تطبيقاتها في مجالي التعرف اللاخطي. في البداية قمنا بدراسة نظرية شاملة حول الخوارزمية الجينية ثم قمنا بتطبيقها في مجالي التحكم و التعرف.في البداية طبقت للحصول على العوامل المثالية لقانون تحكم محصل عليه بتقنية الرجوع المرحلي ثم طبقت في مجال التحكم H_∞/H_2 أخيرا قمنا بتطبيق الخوارزميات الجينية كطريقة للتعرف اللاخطي.

الكلمات المفتاحية: الخوارزميات الجينية، الرجوع المرحلي، التحكم بواسطة H_∞/H_2 ، التعرف اللاخطي.

Résumé Dans ce travail, nous avons étudié les Algorithmes Génétiques ainsi que leur utilisation en commande et identification des systèmes. Dans un premier cas, ils ont été appliqués à un problème d'optimisation des paramètres d'une loi de commande, obtenue via l'approche de backstepping, qui est appliquée à un robot manipulateur de type PUMA 560. Dans le deuxième cas, un AG a été utilisé pour trouver l'ensemble des régulateurs, obtenus via l'approche H_2/H_∞ et satisfaisants les performances H_2/H_∞. Finalement, un AG a été appliqué dans la procédure de sélection de termes dans le problème d'identification non-linéaire.

Mots clés: Algorithme Génétique, commande H_2/H_∞, identification non-linéaire, backstepping.

Abstract

In this work, we have studied the Genetic Algorithms, and their applications in control engineering. First, they have been applied for the optimisation of control law's parameters, obtained by backstepping technique and then applied to a robot of type PUMA 560.

Secondly, they have been applied to obtain a set of mixed controllers which achieve both H_2 and H_∞ performances. Finally, a GA has been used in the procedure of terms selection for the non-linear identification problem.

Key words: Genetic Algorithm, H_2/H_∞ control, non-linear system identification, backstepping.