République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département de Génie Electrique

MEMOIRE

Présenté au **L**aboratoire de **C**ommande des **P**rocessus
En vue de l'obtention du titre de

Magister

Spécialité : Automatique
Option : **A**utomatique des **S**ystèmes **I**ndustriels

Par

LABDELAOUI Hilal

Ingénieur de l'ENP

THÈME

Algorithmes Evolutionnaires Multiobjectifs : Application pour la Commande d'un Réseau Electrique Multimachine

Soutenu le 10 Juin 2007 devant le jury composé de :

O. MAHMOUDI	Professeur à l'ENP	Président
F. BOUDJEMA	Professeur à l'ENP	Rapporteur
D. BOUKHETALA	Maître de Conférences à l'ENP	Rapporteur
M. TADJINE	Professeur à l'ENP	Examinateur
M. TEGUAR	Maître de Conférences à l'ENP	Examinateur
L. NEZLI	Maître de Conférences à l'ENP	Examinateur

Mémoire préparé au sein du Laboratoire de Commande des Processus de l'ENP
10, Avenue Hassen Badi, BP 182, 16200 El-Harrach, Alger
www.enp.edu.dz

ملخص

في هذا العمل، تم اقتراح خوارزمية تطوُرية جديدة، متعددة الأهداف، و تطبيقها في حل مسألة تحكم في شبكة كهربائية متعددة الآلات. الخوارزمية المقترحة تحت اسم ε-GMOEA، تقوم على دمج تقنيات مستعملة من طرف ثلاث طرق تطوُرية أخرى معروفة و هي ε-MOEA، NSGA-II و SPEA2. بهدف تقييم قدرات ε-GMOEA، نقوم بمقارنة هذه الأخيرة بالنسبة إلى الخوارزميات الأخرى في كيفية حل جملة من المسائل النظرية، باستعمال بعض معايير النجاعة. نقوم فيما بعد باستخدام الخوارزمية في مسألة إنشاء متحكم لا مركزي لشبكة كهربائية متكونة من ثلاث آلات. هذا المتحكم طُور على أساس شراكة ما بين تقنية التراجع و التحكم بواسطة المنطق الغامض.

كلمات مفتاحية: الخوارزمية التطوُرية المتعددة الأهداف، الشبكة الكهربائية المتعددة الآلات، التحكم اللامركزي، التراجع، التحكم بواسطة المنطق الغامض.

Résumé

Dans ce travail, un nouvel algorithme évolutionnaire multiobjectif est proposé et appliqué à un problème de commande d'un réseau électrique multimachine. L'algorithme proposé, nommé ε-GMOEA, est basé sur une combinaison de certaines techniques utilisées par trois autres approches évolutionnaires multiobjectifs très connues, à savoir ε-MOEA, NSGA-II et SPEA2. Dans le but d'évaluer les performances de ε-GMOEA, celui-ci est comparé avec les autres méthodes sur une série de problèmes de test, en utilisant quelques métriques de performance. L'algorithme est ensuite utilisé pour la synthèse d'une commande décentralisée d'un réseau électrique à trois machines. La commande a été développée sur la base d'une association de la technique du backstepping et la commande par logique floue.

Mots clés : Algorithme évolutionnaire multiobjectif, réseau électrique multimachine, commande décentralisée, backstepping, commande par logique floue.

Abstract

In this work, a new multiobjective evolutionary algorithm is proposed and applied to a multimachine power system control problem. The proposed algorithm, called ε-GMOEA, is based on a combination of some techniques used by three other well known algorithms, namely ε-MOEA, NSGA-II and SPEA2. In order to assess the performance of ε-GMOEA, it is compared with the other methods on several test problems, using some performance metrics. The algorithm is then used for the design of a decentralized control scheme of a three machines power system. The used control is based on an association of the backstepping technique and fuzzy logic control.

Keywords: Multiobjective evolutionary algorithm, multimachine power system, decentralized control, backstepping, fuzzy logic control.

Remerciements

Je tiens tout d'abord à exprimer ma profonde reconnaissance à Mr BOUDJEMA Farès et à Mr BOUKHETALA Djamel, qui ont assuré la direction de ce mémoire, pour leur suivi et leurs conseils judicieux. Qu'ils trouvent ici l'expression de mon profond respect.

Je remercie Mr MAHMOUDI Oulhadj d'avoir bien voulu accepter la présidence du jury de ce mémoire, ainsi que Mr TADJINE Mohamed, Mr TEGUAR Madjid et Mr NEZLI Lazhari, qui ont accepté d'être membres de ce jury.

Je remercie également Mr BENHAMOUDA Anis, Mr LOUCIF Abdelhalim, Mr KHERROUBA Nabil et Mr MADAOUI Chafaa, pour leur aide et leur soutien constant tout au long des étapes de ce travail.

Les derniers remerciements vont à ma famille pour tout ce qu'elle a fait pour moi et sans laquelle rien de ce travail n'aurait été réalisé.

A mes cher parents
A ma sœur et mes frères

Table des matières

Chapitre 1
Introduction

Chapitre 1

Introduction

1.1 Motivation

La plupart des problèmes réels impliquent la considération de plusieurs critères d'optimisation. En automatique, la conception et la synthèse d'un régulateur mettent en évidence un ensemble de critères de performance, ou objectifs. En théorie comme en pratique, cela représente un problème ouvert qui peut être traité de plusieurs façons. Généralement, les objectifs considérés ne peuvent être satisfaits d'une façon simultanée, vu la nature multiobjectif des problèmes traités. Par exemple, le temps de montée et le dépassement sont deux objectifs en conflit, la tentative d'amélioration de l'un deux impliquera une détérioration de l'autre. Par conséquent, il existe un certain compromis entre ces deux objectifs, qui peut être recherché en utilisant une méthode d'optimisation multiobjectif.

Un grand nombre de méthodes ont été développées pour tenter d'apporter une réponse satisfaisante aux problèmes d'optimisation multiobjectifs. Parmi lesquels, on peut citer le recuit simulé, la recherche Tabou, les essaims de particules, les systèmes immunitaires et les algorithmes évolutionnaires. Ces méthodes sont des heuristiques capables d'approcher efficacement les solutions optimales d'un problème quelconque avec un temps de calcul raisonnable et en ayant un minimum d'adaptations à réaliser pour ce problème.

Dans ce travail, nous nous intéressons aux algorithmes évolutionnaires multiobjectifs. Ces derniers se caractérisent par une flexibilité remarquable, leur permettant de traiter des problèmes variés et complexes. Ils ont été appliqués à une variété de problèmes de commande et d'identification des systèmes, aussi bien pour le cas des systèmes linéaires que pour le cas des systèmes non linéaires. Leur apport en matière de conception et de synthèse est toujours apprécié de nos jours, et des travaux de recherches continuent toujours à être effectués sur l'amélioration des performances de ces algorithmes.

1.2 Objectifs et contributions

Notre but et principale contribution dans ce travail est le développement d'une nouvelle méthode d'optimisation multiobjectif. Cette méthode est basée sur une combinaison de certaines techniques utilisées par d'autres méthodes de résolution bien connues et validées par la communauté scientifique. Cette méthodologie devra être applicable à divers types de problèmes, et doit donc être un outil souple et robuste.

En l'absence d'autres facteurs, tels que la préférence de certains objectifs par rapport à d'autres ou la présence de contraintes sur les objectifs ou sur les paramètres du problème, la tâche d'un algorithme évolutionnaire multiobjectif quelconque sera de trouver la meilleure approximation possible des solutions optimales du problème posé. Par conséquent, des métriques de performance s'avèrent nécessaires pour l'évaluation des capacités de telles approches. Dans ce travail, nous utilisons quelques mesures qui ont été déjà proposées et utilisées dans la littérature, et nous proposons une nouvelle mesure, qui permet de donner une caractérisation plus robuste d'un certain aspect de performance.

L'utilisation de la méthode évolutionnaire développée ici est illustrée dans la résolution d'un problème d'optimisation multiobjectif pour la commande d'un réseau électrique multimachine. Une nouvelle technique de commande décentralisée des réseaux électriques multimachines est d'abord proposée, puis appliquée en optimisant ses paramètres, sur un cas particulier d'un réseau à trois machines.

1.3 Organisation du document

Ce travail est organisé de la façon suivante. Le chapitre qui suit présente d'une façon générale les algorithmes évolutionnaires et met l'accent sur les algorithmes génétiques, un cas particulier de ces algorithmes, très connus et utilisés dans la communauté scientifique. Le troisième chapitre traite les problèmes d'optimisation multiobjectifs et offre une description détaillée du fonctionnement de quelques célèbres méthodes évolutionnaires multiobjectifs. Le quatrième chapitre présente notre approche d'optimisation multiobjectif. L'évaluation des performances de cette approche est faite conjointement avec les trois autres méthodes présentées au Chapitre 3, en utilisant une série de problèmes de test et en se basant sur certains critères de performance. Le cinquième chapitre décrit l'application de l'algorithme proposé sur un problème de commande d'un réseau électrique multimachine. Pour finir, nous exposons les conclusions finales de notre travail et les perspectives futures.

Chapitre 2
Les Algorithmes Evolutionnaires

Chapitre 2

Les Algorithmes Evolutionnaires

2.1 Introduction

Les Algorithmes Evolutionnaires (AEs) sont des méthodes de recherche stochastique, globale, itérative et parallèle. Leurs concepts sont basés sur des métaphores biologiques qui font appel à des théories de l'évolution et la sélection naturelle (bien que cette vue n'est pas toujours soutenue de nos jours). On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les Algorithmes Génétiques (Holland, 1975 ; Goldberg 1989), les Stratégies d'Evolution (Rechenberg, 1973 ; Schwefel, 1981) et la Programmation Evolutionnaire (Fogel, 1991). Toutefois, ces algorithmes se différencient uniquement par leur manière de représenter l'information et par leur façon de faire évoluer les solutions d'une itération à l'autre.

Ce chapitre a pour but d'introduire les principes de base du fonctionnement des algorithmes évolutionnaires. La Section 2.3 sera particulièrement consacrée aux algorithmes génétiques, qui sont en fait les méthodes évolutionnaires les plus connues et utilisées jusqu'à présent. La Section 2.4 révèle, en général, les différents avantages et inconvénients des AEs. Enfin, nous terminons par une revue générale des applications antérieures et potentielles des algorithmes évolutionnaires dans le domaine de l'automatique.

2.2 Concepts et principes de base

L'idée originale dans la conception des algorithmes évolutionnaires était de faire "évoluer" une population d'individus représentant des solutions candidates au problème posé, à travers un certain nombre de générations, dans le but d'obtenir des solutions de plus en plus satisfaisantes. L'amélioration dans la qualité des solutions est une conséquence directe de l'application des opérateurs de variation (ou opérateurs génétiques), à chaque génération, sur la population d'individus.

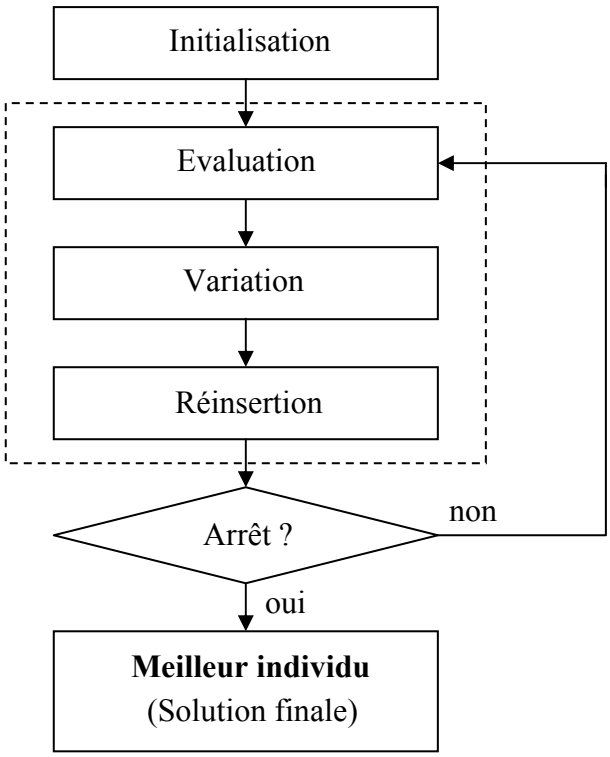


Figure 2.1 : Schéma fonctionnel d'un algorithme évolutionnaire standard.

La Figure 2.1 montre les différentes étapes du fonctionnement d'un algorithme évolutionnaire de base (problème mono-objectif). On commence par la génération d'une première population d'individus (généralement d'une façon aléatoire), et probablement l'initialisation de certains paramètres nécessaires. Ensuite, les individus sont évalués à la base d'un certain critère d'évaluation, qui n'est rien d'autre que la formulation mathématique du problème traité. Cela veut dire que le "meilleur" individu de la population (on dit aussi le plus adapté) est celui qui optimise (minimise ou maximise) le plus ce critère. Dans le reste de ce travail, on considère toujours un problème de minimisation.

Une fois les individus évalués, ceux les plus adaptés auront en principe plus de chance d'être sélectionnés par rapport à ceux plus faibles, afin de contribuer à la génération de nouveaux individus, ou descendants. Les individus sélectionnés formeront donc une nouvelle population, dite intermédiaire, à qui on appliquera ensuite les opérateurs de variation : le croisement et/ou la mutation. Cette variation effectuée sert à exploiter les "données génétiques" contenues dans la population courante, et également à explorer de plus l'espace de recherche. Enfin, la procédure de réinsertion consiste à former une nouvelle population d'individus, à partir de ceux récemment générés (descendants) et probablement des individus de l'ancienne population (parents).

Les étapes ainsi décrites (évaluation, sélection, croisement, mutation et réinsertion) sont répétées à chaque génération, jusqu'à la satisfaction d'un certain critère d'arrêt (généralement,

un nombre maximum de générations atteint). La solution finale du problème traité est représentée par le meilleur individu de la dernière génération.

Un algorithme évolutionnaire quelconque est donc caractérisé par sa façon de représenter l'individu, son schéma de sélection, les types de ses opérateurs génétiques utilisés ainsi que sa stratégie de réinsertion (ou de remplacement). Par exemple, un algorithme génétique (AG) utilise une représentation binaire ou réelle des paramètres et emploie comme opérateurs de variation, le croisement et la mutation. Alors que les stratégies d'évolution (SE) manipulent toujours des paramètres réels et insistent plutôt sur le rôle de la mutation. Quant à la programmation évolutionnaire (PE), elle était originellement basée sur l'évolution d'une population d'automates finis, mais elle a été développée ensuite pour pouvoir travailler sur des réels. La PE ne fait appel qu'à l'opérateur de mutation.

Dire qu'un AE est meilleur qu'un autre, cela dépendra fortement du problème traité. En fait, il n'existe pas vraiment un algorithme qui donne des performances optimales sur toutes les instances de problèmes. Dans ce chapitre, on a choisit les algorithmes génétiques pour être étudiés car ce sont les algorithmes qui possèdent le plus grand degré de flexibilité et de robustesse, pour résoudre des problèmes variés et complexes.

2.3 Les algorithmes génétiques

Les algorithmes génétiques ont été inventés et développés par John Holland durant les années soixante et soixante-dix. Ses recherches sur les processus d'adaptation des systèmes naturels ont permis la découverte de cette nouvelle technique de recherche. Les fondements mathématiques des AGs ont été exposés plus tard par David Goldberg (1989).

L'implémentation d'un algorithme génétique sur une plateforme numérique passe par plusieurs étapes. La première étape concerne la définition et la formulation mathématique du problème posé, soit l'élaboration d'un certain critère de minimisation, appelé fonction objectif, en fonction des variables de décision du problèmes (paramètres du problème), ainsi que la spécification de leur domaine de variation (espace de recherche).

La deuxième étape est le codage de l'individu, c'est-à-dire la formulation des fonctions de codage et de décodage qui doivent être significatives vis-à-vis de la définition de l'espace de recherche. Le codage de l'individu doit traduire le plus précisément possible les caractéristiques de la solution envisageable du problème. Cette étape est d'une importance primordiale, car elle influe grandement sur la qualité des résultats ainsi que sur la conduite et le temps de convergence de l'algorithme. Le codage est le point clé du fonctionnement des algorithmes génétiques, et plus généralement de celui des algorithmes évolutionnaires.

Les autres étapes concernent les opérateurs de variation et la procédure de réinsertion. En fait, on doit choisir les types des opérateurs utilisés pour la sélection, le croisement et la

mutation, ainsi que les valeurs des différents paramètres qui peuvent y intervenir. Tous ces points seront abordés avec plus de détails dans les prochaines sections.

2.3.1 La population

Les AGs opèrent sur un certain nombre de solutions potentielles qui forment la population. Le nombre d'individus qui définit la taille de la population dépend de la nature et de la complexité du problème, par exemple du nombre de variables de décision et de fonctions objectifs. Pour un problème mono-objectif, une population typique est composée d'environs 30 à 100 individus.

Dans le fonctionnement des 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 où la fonction objectif est définie. Le phénotype est utilisé dans l'étape de l'évaluation. Le génotype, quant à lui, est une représentation du phénotype à un niveau "inférieur", et c'est à ce niveau que l'individu est manipulé par l'AG (c'est-à-dire, l'application des opérateurs génétiques). Le phénotype est donc codé dans le génotype, originellement sous forme de caractères de bits (chaînes de bits), mais plus généralement sous forme de n'importe quelle structure de données convenable [15].

Dans le cas du codage binaire, chaque variable de décision dans l'ensemble des paramètres est codée sous forme d'une concaténation de bits. Le nombre de bits utilisé est choisi selon la précision voulue. Cependant, le codage binaire standard possède quelques inconvénients. Par exemple, avec ce codage une petite variation du génotype peut entraîner une grande modification dans la valeur du phénotype, ce qui signifie que ce type codage ne reflète pas convenablement le comportement des variables de décision. Pour cela, on a introduit le codage binaire en Gray, qui a la particularité d'avoir un seul bit de différence entre deux valeurs adjacentes.

D'autres stratégies de codage alternatives, telles que les représentations réelles et entières, sont de plus en plus utilisées de nos jours. En fait, le codage réel est généralement préconisé dans le cas des problèmes continus, du fait qu'il est plus simple à utiliser et gagne en temps d'exécution. Dans ce cas, un individu sera simplement représenté par un vecteur de valeurs réelles.

2.3.2 L'évaluation

Les individus sont évalués via la fonction objectif qui définit le problème. Par conséquent, chaque individu sera caractérisé par une certaine valeur de coût, qui lui soit propre. L'adaptation d'un individu, appelée fitness, est alors tirée de son coût en prenant en considération tous les autres individus de la population. Notons que la fitness peut être à

minimiser ou à maximiser. Donc, la valeur de coût est une caractéristique “intrinsèque” de l’individu, tandis que la valeur de fitness lui est une caractéristique “relative”.

La relation coût-fitness concerne principalement le processus de sélection. Elle doit donc être une relation monotone sur le sous-ensemble des réels non négatifs. Différentes techniques pour le calcul de fitness ont été déjà proposées, mais peu d’entre elles sont toujours utilisées de nos jours (voir la section suivante).

2.3.3 La sélection

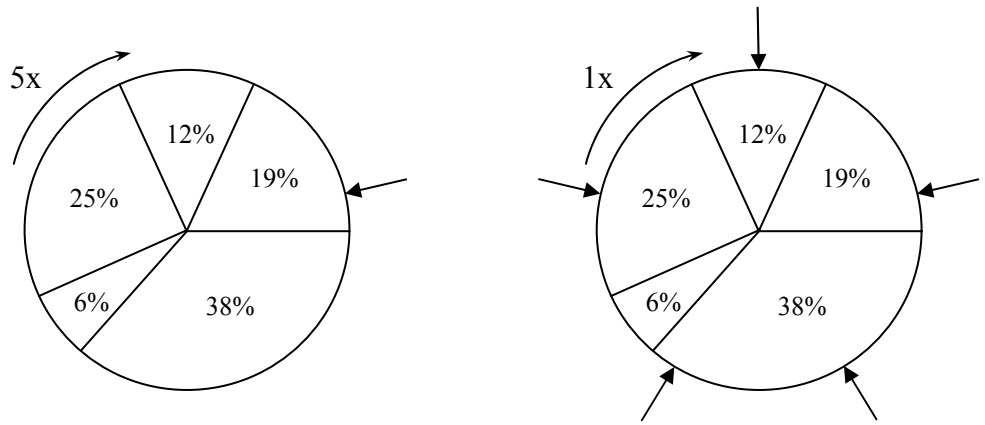
La sélection conditionne la capacité d’un individu à se reproduire. Elle détermine également la nouvelle topologie génétique qui servira de modèle pour la génération de nouveaux individus. Le résultat est une population intermédiaire constituée de copies des individus de la population des parents. C’est donc un outil d’exploitation des données génétiques contenues dans la population courante.

Parmi les techniques de sélection qui existent aujourd’hui, plusieurs d’entre elles utilisent le mécanisme de la roulette de loterie. Dans ce cas, chaque individu de la population occupe une certaine section de roue proportionnelle à sa fitness (à maximiser). La probabilité de sélection d’un individu quelconque dépend du nombre de lancers de la roulette ainsi que du nombre et la disposition des pointeurs (Figure 2.2).

Dans la sélection par échantillonnage stochastique avec remplacement, la roulette est lancée un nombre de fois égal au nombre d’individus à sélectionner, et à chaque lancer on sélectionne l’individu désigné par le pointeur (Figure 2.2-a).

Une autre méthode semblable à celle-ci est la sélection par échantillonnage stochastique sans remplacement, dans laquelle la roulette est lancée une seule fois, avec un nombre de pointeurs égal au nombre d’individus à sélectionner (Figure 2.2-b). Il est clair que cette deuxième méthode est plus sûre et plus efficace que la première, car elle permet de prendre une meilleure “représentation génétique” de la population, tout en favorisant les individus les plus adaptés.

Une autre méthode de sélection qui est très utilisée de nos jours est la sélection par tournoi. Dans cette méthode, un certain nombre d’individus (typiquement 2 ; tournoi binaire) sont tirés aléatoirement de la population et sont amenés à participer à un tournoi à base de fitness, où le vainqueur sera sélectionné (c'est-à-dire, celui qui a la plus grande ou la plus petite valeur de fitness). Cette procédure est répétée jusqu’à avoir le nombre de sélectionnés souhaité, en réintégrant à chaque fois les individus participant au dernier tournoi dans la population avant le prochain tirage. L’avantage de ce schéma de sélection est qu’il ne nécessite plus un calcul de fitness supplémentaire, puisque le tournoi peut être effectué en utilisant directement les valeurs de la fonction objective (valeurs de coût).



(a) (b)

Figure 2.2: Sélection par échantillonnage stochastique ; avec remplacement (a) et sans remplacement (b).

2.3.4 Le croisement

Le croisement est l'opérateur qui va permettre le "brassage" des caractères génétiques de la population. Cet opérateur va générer par exemple deux descendants, en effectuant un mélange de gènes de deux parents. Donc, les caractéristiques de deux parents, ou plus, peuvent être héritées par un même descendant, ce qui va donc permettre d'accélérer considérablement le processus de recherche. C'est pour cette raison que le croisement est souvent accompli avec une grande probabilité, généralement entre 0.7 et 1. Donc, le croisement est considéré à la fois comme étant un outil d'exploitation et d'exploration des données génétiques. Notons encore que le croisement peut être effectué en utilisant plus que deux parents.

Le croisement multi-point est le type de croisement le plus utilisé dans le cas d'un codage binaire. Il consiste à choisir aléatoirement des points de croisement situés sur le génotype des parents, puis d'échanger les portions génétiques résultantes, entre ces parents. La Figure 2.3 donne un exemple d'un croisement binaire en un seul point.

Dans le cas du codage réel, plusieurs types d'opérateurs peuvent être utilisés. Un de ces types qui est très employé ces dernières années, est le croisement binaire simulé ou SBX (Simulated Binary Crossover), développé par Kalyanmoy Deb (1995) [11]. En fait, cet opérateur ne fait qu'imiter le principe de fonctionnement du croisement binaire en un seul point. Les étapes suivantes donne la procédure de création de deux solutions descendantes ($x_i^{(1,t+1)}$ et $x_i^{(2,t+1)}$) à partir de deux solutions parentes ($x_i^{(1,t)}$ et $x_i^{(2,t)}$) :

Etape 1 : Choisir un nombre aléatoire $u_i \in [0,1)$.

Etape 2 : Calculer un facteur de dispersion β_i, égal au rapport de la différence absolue des valeurs des descendants et celle des valeurs des parents :

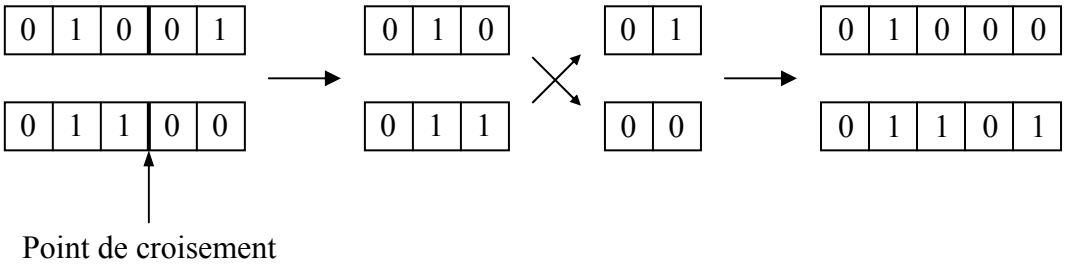


Figure 2.3 : Croisement en un seul point.

$$\beta_i = \left| \frac{x_i^{(1,t+1)} - x_i^{(2,t+1)}}{x_i^{(1,t)} - x_i^{(2,t)}} \right|. \tag{2.1}$$

En utilisant la fonction de la densité de probabilité donnée ci-dessous, l'abscisse β_{qi} est obtenue de sorte que la surface limitée par la courbe de la fonction et les lignes verticales passant par 0 et β_{qi} soit égale à u_i.

$$\mathrm{P}(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c}, & \text{si } \beta_i \le 1, \\ 0.5(\eta_c + 1)\dfrac{1}{\beta_i^{\eta_c + 2}}, & \text{sinon.} \end{cases} \tag{2.2}$$

où η_c est un nombre réel positif, appelé l'indice de distribution du croisement. Celui-ci doit être défini par l'utilisateur. Une valeur élevée de η_c augmente la probabilité de générer des descendants proches de leurs parents (dans l'espace des décisions), et vice versa. En utilisant l'Equation (2.2) on calcule β_{qi} en fonction de u_i comme suit :

$$\beta_{qi} = \begin{cases} (2u_i)^{\frac{1}{\eta_c + 1}}, & \text{si } u_i \le 0.5, \\ \left(\dfrac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c + 1}}, & \text{sinon.} \end{cases} \tag{2.3}$$

Etape 3 : Calculer les descendants comme suit :

$$\begin{cases} x_i^{(1,t+1)} = 0.5\left[(1+\beta_{qi})x_i^{(1,t)} + (1-\beta_{qi})x_i^{(2,t)}\right], \\ x_i^{(2,t+1)} = 0.5\left[(1-\beta_{qi})x_i^{(1,t)} + (1+\beta_{qi})x_i^{(2,t)}\right]. \end{cases} \tag{2.4}$$

Dans ce cas, les descendants seront symétriques par rapport à la moyenne des parents et pour une valeur de η_c fixe ils auront un écart proportionnel à celui de leurs parents. A partir de l'Equation (2.4), on obtient :

$$x_i^{(2,t+1)} - x_i^{(1,t+1)} = \beta_{qi}\,(x_i^{(2,t)} - x_i^{(1,t)})\,. \tag{2.5}$$

Ce qui veut dire que si deux parents sont éloignés l'un de l'autre (comme c'est le cas dans les populations initiales, où des solutions sont générées d'une façon aléatoire), alors les descendants le seront aussi (exploration). Par contre, lorsque les solutions tendent à converger sous l'effet des opérateurs génétiques (donc les parents deviennent de plus en plus proches entre eux), des solutions distantes ne seront plus permises et la recherche sera concentrée vers des régions restreintes (exploitation).

2.3.5 La mutation

La mutation consiste à altérer aléatoirement le codage d'un individu. Son rôle est de faire émerger de nouveaux génotypes en explorant des zones de l'espace de recherche, qui pourraient ne pas être visitées par simple application de l'opérateur de croisement. Ainsi la mutation lutte contre la "dérive" génétique. Habituellement, seulement une très petite partie du génotype est modifiée par mutation, permettant ainsi l'héritage de la plupart des caractéristiques génétiques des parents. Ceci peut être réalisé en utilisant des faibles taux de mutation, de l'ordre de 0.01.

L'effet de la mutation sur un génotype binaire est illustré à la Figure 2.4. Dans ce cas, la mutation provoque l'inversion de la valeur du troisième bit. Etant donné que l'application de l'opérateur de mutation est uniforme sur une population entière d'individus, il est possible qu'un génotype quelconque soit muté en plusieurs points.

Dans le case d'un codage réel, l'opérateur de mutation consiste à ajouter un pas de mutation aléatoire aux variables constituant le génotype d'un individu. Ce pas est généralement petit mais sa taille optimale est très difficile à déterminer car elle est fonction de la nature du problème envisagé. Dans la mutation polynomiale (Polynomial Mutation) proposée par Deb (1996) [10], les étapes suivantes exposent la procédure de mutation d'une solution parente $x_i^{(1,t+1)}$ pour donner une solution descendante $y_i^{(1,t+1)}$.

Etape 1 : Choisir un nombre aléatoire $u_i \in [0,1)$.

Etape 2 : Calculer le paramètre δ_i en utilisant la fonction de la densité de probabilité :

$$\mathrm{P}(\delta) = 0.5\,(\eta_m + 1)\,(1 - |\delta|)^{\eta_m}\,, \tag{2.6}$$

soit

$$\delta_i = \begin{cases} (2u_i)^{\frac{1}{\eta_m+1}} - 1, & \text{si } u_i < 0.5 \\ 1 - \left[2\,(1-u_i)\right]^{\frac{1}{\eta_m+1}}, & \text{sinon} \end{cases} \tag{2.7}$$

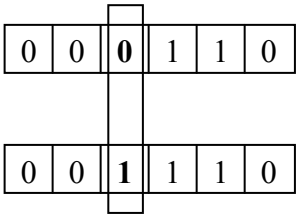


Figure 2.4 : Mutation binaire.

Etape 3 : Obtenir le descendant en utilisant :

$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + (x_i^u - x_i^l)\,\delta_i\,, \tag{2.8}$$

où x_i^u et x_i^l désignent respectivement la valeur minimale et maximale que peut prendre la variable x. La forme de la densité de probabilité est directement liée à l'indice de distribution η_m, qui doit être défini au préalable par l'utilisateur.

2.3.6 La réinsertion

La réinsertion des descendants dans la population courante peut se faire par plusieurs manières. Dans le cas le plus simple, la population entière des parents est remplacée inconditionnellement par celle des descendants. Ceci est connu par le remplacement générationnel. Il est clair qu'avec ce type de remplacement, la "pression sélective" ou le biais par rapport au meilleur individu, est nettement réduite (elle dépend également des techniques employées pour le calcul de fitness et pour la sélection). En plus, il n'y a pas de garantie que le meilleur individu soit préservé d'une génération à l'autre. Pour remédier à ces problèmes, on a introduit le modèle générationnel élitiste, dans lequel la nouvelle population est composée des meilleurs individus de l'ensemble des parents et de leurs descendants, ce qui va donc permettre de conserver le meilleur individu à travers les générations.

Dans le cas d'un algorithme génétique dit stationnaire, un nombre minimal de descendants (typiquement un ou deux) sont produits par croisement et mutation. Ces descendants sont alors évalués et probablement réinsérés dans la population des parents, en remplaçant :

- Des membres aléatoires 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 avoir lieu :

- Inconditionnellement.

- Seulement si les descendants sont meilleurs que les individus qu'ils vont remplacer.
- D'une manière probabiliste, dépendant des fitness des descendants par rapport à celles des individus qu'ils vont remplacer.

Généralement, les AGs stationnaires sont meilleurs que leurs équivalents générationnels en termes de qualité de solutions obtenues, car ils exploitent plus rapidement les nouvelles solutions générées par variation génétique. La pression sélective est plus élevée dans les AGs stationnaires, mais leur demande en espace mémoire est plus faible. Par ailleurs, les AGs stationnaires manipulant des populations de tailles petites, perdent en général la diversité génétique d'une façon relativement rapide, et ceux qui manipulent des populations de tailles assez grandes entraînent vraisemblablement une augmentation du coût de calcul et une baisse de la vitesse de convergence.

2.4 Avantages et inconvénients des AEs

L'avantage crucial d'un algorithme évolutionnaire, en particulier l'AG, sur les autres techniques de recherche, est sa grande flexibilité. En effet, l'AE est une méthode de recherche directe, il n'existe donc aucune restriction théorique sur la fonction d'évaluation. Des réponses de simulation et même humaines aux solutions, sont permises. En plus, il n'y a pas de restrictions sur la représentation d'une solution. N'importe quel type de données peut être employé, à condition d'utiliser des opérateurs de variation appropriés. Par ailleurs, l'AE est une méthode de recherche globale et stochastique, par conséquent il offre plus de robustesse aux cas multimodaux, non continus, bruités, dynamiques, etc.

Les inconvénients des AEs peuvent être résumés en trois points. Tout d'abord, ils n'offrent aucune garantie de trouver un point optimal en un temps fini, mais cela est vrai pour toutes les méthodes d'optimisation globales. Deuxièmement, leur base théorique reste insuffisante, même si quelques travaux de recherche et des théories ont pu être développés dans ce contexte. Enfin, le réglage des paramètres est largement inspiré du essai/erreur sauf pour les stratégies d'évolution qui sont auto-adaptatives.

2.5 Applications des AEs en automatique

Les algorithmes évolutionnaires sont utilisés, depuis une vingtaine d'années, pour résoudre de nombreux problèmes d'ingénierie. Dans le domaine de l'automatique, ils sont utilisés par exemple pour la conception et la synthèse des régulateurs, l'identification des systèmes, l'analyse de la stabilité robuste et le diagnostic des défaillances. Dans certains cas, ils sont employés comme étant le seul moyen de conception et de synthèse. Dans d'autres cas, ils sont combinés avec d'autres techniques intelligentes ou métaheuristiques, pour former un ensemble d'outils complémentaires.

Lors de la conception d'un régulateur, il faut tenir compte de plusieurs critères de performance, tels que la stabilité, la rapidité et la robustesse du système commandé. Chacun de ces critères dépend évidemment de la structure et des paramètres du régulateur. Néanmoins, cette dépendance ne peut être exprimée en termes de formules mathématiques. En plus, il existe généralement un compromis à considérer parmi les critères qui sont contradictoires. Par conséquent, il est difficile de choisir parmi toutes les valeurs admissibles des paramètres d'un régulateur celles qui correspondent, au mieux, aux performances désirées. Pour résoudre ces problèmes en utilisant un AE, on peut coder la structure et/ou les paramètres du régulateur dans un seul génotype d'un individu, et définir une ou plusieurs fonctions objectifs qui correspondent aux critères de performance choisis.

Les algorithmes génétiques restent à nos jours les méthodes évolutionnaires les plus utilisés dans le domaine de la commande. Ils ont été utilisés par exemple pour obtenir les valeurs des paramètres des régulateurs PID, LQG, H_∞, à structure variable, et autres. Dans la commande floue, ils sont utilisés pour générer la base des règles floues et/ou ajuster les fonctions d'appartenance associées. Dans le cas de la commande par réseaux de neurones, les AGs peuvent être utilisés pour obtenir les poids et/ou l'architecture du réseau neuronal. Par ailleurs, les algorithmes génétiques ont été appliqués à l'identification linéaire et non linéaire des systèmes discrets ou continus, en cherchant la structure du modèle, ou bien ses paramètres ou encore les deux à la fois. Pour plus de détails et d'exemples sur les applications antérieures et potentielles des AEs en automatique, voir la référence [14].

2.6 Conclusion

Dans ce chapitre, une présentation générale des algorithmes évolutionnaires et une description détaillée du fonctionnement des algorithmes génétiques, sont données. Les AGs sont des méthodes de recherche flexibles et robustes, qui conviennent en particulier aux problèmes impliquant des variables de types différents et/ou des objectifs difficiles. Les applications des AGs en automatique sont variées et très prometteuses. Toutefois, leur utilisation en mode on-line (temps réel) est toujours conditionnée par l'habilité du système à se répondre au niveau d'exploration demandé par l'AG.

Parmi les divers champs d'application des algorithmes évolutionnaires, l'optimisation multiobjectif constitue actuellement un axe de recherche très important et très utile pour la résolution de beaucoup de problèmes d'optimisation réels. Le prochain chapitre introduit ainsi les algorithmes évolutionnaires multiobjectifs.

Chapitre 3

Optimisation Multiobjectif par Algorithmes Evolutionnaires

Chapitre 3

Optimisation Multiobjectif par Algorithmes Evolutionnaires

3.1 Introduction

La plupart des problèmes d'optimisation réels consistent à optimiser non pas un, mais plusieurs critères qui doivent être satisfaits simultanément et qui, la plupart du temps, sont en plus contradictoires. Traditionnellement, ces critères sont agrégés en une seule fonction objectif et traités de la même manière qu'un problème mono-objectif, mais cette approche possède beaucoup d'inconvénients et ne répond certainement pas à la nature multiobjectif des problèmes traités. Par ailleurs, une seule et unique solution finale trouvée ne sera vraisemblablement pas suffisante pour satisfaire aux différents objectifs posés dans un problème quelconque. De ce fait, on est amené à utiliser de nouveaux outils et concepts mathématiques qui vont nous permettre de considérer et de manipuler chaque objectif, ou critère, d'une façon indépendante des autres.

Le présent chapitre est dédié à l'optimisation multiobjectif par moyens d'algorithmes évolutionnaires. Nous présentons tout d'abord un ensemble de concepts et définitions liés aux problèmes d'optimisation multiobjectifs, ainsi qu'une revue générale sur les approches de résolution multiobjectifs basées sur les méthodes évolutionnaires. L'optimisation sous contraintes est introduite sans beaucoup de détails dans la Section 3.2, tout en restant liée au cas multiobjectif. Ensuite, nous révélons quelques points clés et essentiels dans la recherche multiobjectif, qui sont en fait à l'origine du grand succès qu'ont connu les algorithmes évolutionnaires multiobjectifs (AEMOs) durant ces toutes vingt dernières années. Pour terminer, un éventail qualitatif de méthodes évolutionnaires multiobjectifs est présenté, en essayant d'apporter un regard critique sur chacune d'elles. Ces méthodes présentent quelques caractéristiques fonctionnelles de base et sont largement utilisées dans la littérature.

3.2 L'optimisation multiobjectif

L'optimisation multiobjectif, dite aussi multicritère ou vectorielle, est définie comme étant le problème de trouver :

> Un vecteur de décisions satisfaisant les contraintes imposées (s'il y en existe) et optimisant une fonction vectorielle dont les composantes sont les fonctions objectifs du problème posé. Ces dernières traduisent mathématiquement les critères de performance et sont généralement en conflit les unes par rapport aux autres. Par conséquent, le terme "optimiser" signifie plutôt déterminer une solution qui donne le meilleur compromis possible entre les valeurs de l'ensemble des fonctions objectifs.

Mathématiquement, cette définition peut être traduite sous forme d'une optimisation simultanée de m composantes $f_k, k = 1,\ldots,m$ d'une certaine fonction vectorielle $\mathbf{f}$, probablement non linéaire, d'une variable de décision générale x dans un univers U où :

$$\mathbf{f}(\mathrm{x}) = (f_1(\mathrm{x}),\ldots,f_m(\mathrm{x})) . \tag{3.1}$$

Généralement, le problème ne possède pas une solution unique, parfaite, mais plutôt un ensemble de solutions alternatives, non dominées, connu sous le nom de l'ensemble Pareto-optimal. En supposant un problème de minimisation, la dominance et l'optimalité au sens de Pareto sont alors définies comme suit :

Définition 3.1 (Dominance au sens de Pareto) Un vecteur $\mathbf{u} = (u_1,\ldots,u_m)$ domine $\mathbf{v} = (v_1,\ldots,v_m)$ si et seulement si $\mathbf{u}$ est partiellement inférieur à $\mathbf{v}$ ($\mathbf{u} \prec \mathbf{v}$), c'est-à-dire :

$$\forall i \in \{1,\ldots,m\}, u_i \le v_i \quad \wedge \quad \exists i \in \{1,\ldots,m\}, u_i < v_i . \tag{3.2}$$

Définition 3.2 (Optimalité au sens de Pareto) Une solution $\mathrm{x}_\mathrm{u} \in U$ est dite Pareto-optimale si et seulement si il n'existe aucune autre solution $\mathrm{x}_\mathrm{v} \in U$ pour laquelle $\mathbf{v} = \mathbf{f}(\mathrm{x}_\mathrm{v}) = (v_1,\ldots,v_m)$ domine $\mathbf{u} = \mathbf{f}(\mathrm{x}_\mathrm{u}) = (u_1,\ldots,u_m)$.

Les solutions Pareto-optimales sont dites aussi efficaces, non dominées et non inférieures. Les vecteurs objectifs correspondants sont dits simplement non dominés. L'ensemble de tous les vecteurs non dominés est appelé l'ensemble non dominé ou la surface de compromis ou encore le front non dominé, du problème [15].

La Figure 3.1 illustre le concept de la dominance au sens de Pareto ainsi que celui de la Pareto-optimalité. Dans ce cas, uniquement les vecteurs A et C sont non dominés, ils constituent de ce fait le front non dominé. La surface de Pareto (ou le front de Pareto), quant à elle, est indiquée par le trait gras à l'extrémité de l'espace des objectifs réalisables. Rappelons que les deux fonctions f_1 et f_2 sont à minimiser.

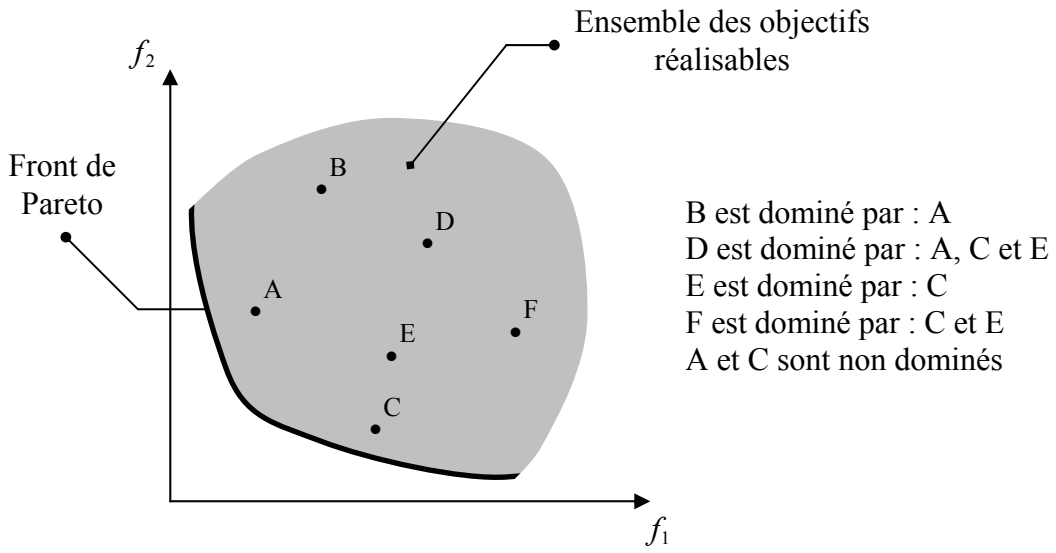


Figure 3.1 : Exemple illustratif de la notion d'optimalité au sens de Pareto.

3.2.1 La décision dans l'optimisation multiobjectif

L'utilisation de la notion d'optimalité de Pareto n'est qu'une première étape pour la résolution d'un problème d'optimisation multiobjectif. Le choix final d'un bon compromis parmi toutes les alternatives non inférieures trouvées ne dépend pas uniquement du problème posé, mais aussi des préférences subjectives d'un agent de décision ; le décideur. Ainsi, la solution finale du problème posé sera le résultat d'un processus d'optimisation et un autre de décision.

Selon l'ordre dans lequel ces deux processus sont envisagés, on distingue trois grandes classes de méthodes de résolution différentes [15] :

Les méthodes a priori (Décideur → Recherche)

Le décideur exprime ces préférences en combinant les différents objectifs dans une seule fonction de coût. Dans ce cas, le décideur est supposé connaître a priori le poids de chaque objectif afin de les mélanger dans une seule fonction unique. Ce qui revient à résoudre un problème mono-objectif. Cependant dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction de coût, soit par manque d'expérience ou d'informations, soit parce que les différents objectifs sont non commensurables (c'est-à-dire exprimés dans des unités différentes).

Les méthodes a posteriori (Recherche → Décideur)

Le décideur prend sa décision d'après un ensemble de solutions calculées au préalable par un solveur (ou optimiseur). Dans ce cas, la qualité de la décision dépend du choix de la méthode de résolution. Car celle-ci va devoir donner un ensemble de résultats le plus représentatif de l'espace des objectifs efficaces. Le choix du décideur est souvent influencé par la forme (degré de convexité) du front non dominé trouvé.

Les algorithmes évolutionnaires, de part leur principe de fonctionnement, s'apparentent à ce type de technique. Nous présentons plus loin dans ce chapitre, différentes implémentations permettant de modifier l'algorithme évolutionnaire mono-objectif, afin qu'il prenne en considération l'aspect multiobjectif.

Les méthodes progressives ou interactives (Recherche ↔ Décideur)

Dans ces méthodes, les processus de décision et d'optimisation sont alternés. Par moment, le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi d'une façon interactive le compromis entre ses préférences et le résultat. Ces méthodes exigent une connaissance approfondie de la part du décideur des différents outils utilisés.

3.2.2 L'optimisation sous contraintes

La solution d'un problème réel peut être contrainte par un certain nombre de restrictions imposées sur les variables de décision ou sur les objectifs du problème. Généralement, les contraintes appartiennent à l'une des deux catégories suivantes :

Contraintes de définition

Elles expriment le domaine de définition de la fonction objectif. Dans le domaine de la commande, la stabilité en boucle fermée en est un exemple, car la plupart des mesures de performance ne sont pas définies pour les systèmes instables.

Contraintes de préférence

Elles imposent des restrictions supplémentaires sur les solutions du problème, d'après des connaissances préalables. Une marge de stabilité donnée, par exemple, exprime une préférence (subjective) du concepteur.

Les contraintes peuvent être exprimées en termes d'inégalités du type :

$$f(\mathrm{x}) \leq g \,, \tag{3.3}$$

où f est une fonction réelle de la variable x, et g une valeur constante. L'inégalité peut être également stricte ($<$ au lieu de $\leq$). Une solution qui ne satisfait, au moins une contrainte, est dite non faisable ou non réalisable.

D'une façon générale, le problème d'optimisation sous contraintes est celui de minimiser une fonction multiobjectif ($f_1, f_2, \ldots, f_m$) d'une certaine variable de décision x dans un univers U, en présence de q conditions impliquant x et éventuellement s'exprimant sous forme d'un vecteur d'inégalité du type :

$$(f_1(\mathrm{x}), f_2(\mathrm{x}), \ldots, f_q(\mathrm{x})) \leq (g_1, g_2, \ldots, g_q)\,. \tag{3.4}$$

De nombreuses approches dans le cadre des AEs ont été développées pour traiter ce genre de problèmes. En fait, beaucoup d'entre elles utilisent le principe de pénalité, où chaque solution de la population se voit attribuée une valeur de pénalité proportionnelle à son degré de violation des contraintes, et qui sera ajoutée ensuite à la valeur de fitness de cette solution. D'autres approches, en revanche, tentent de rechercher des solutions faisables, par exemple, en "réparant" des solutions non faisables.

Une autre approche de prise en compte des contraintes qui est très simple à utiliser, a été proposée par Deb (1999). Cette approche fait appel à un schéma de sélection par tournoi binaire qui tient compte de la violation ou non des contraintes, par les deux solutions concurrentes. Dans ce cas, la définition de la dominance est modifiée comme suit [9] :

Définition 3.3 (Dominance avec contraintes) Une solution x_u domine une solution x_v si et seulement si l'une des conditions suivantes est satisfaite :

- La solution x_u est faisable est la solution x_v ne l'est pas.
- Les deux solutions x_u et x_v sont non faisables, mais x_u possède un degré de violation de contraintes plus faible que celui de x_v.
- Les deux solutions x_u et x_v sont faisables et la solution x_u domine x_v (ou le vecteur $\mathbf{u}$ domine $\mathbf{v}$).

L'utilisation de ce principe de dominance avec contraintes permet de concentrer la recherche, dans un premier temps, vers des régions de l'espace des solutions réalisables, puis de la faire évoluer vers le front de Pareto, tout en restant limitée par les contraintes imposées.

3.3 Points clés dans la recherche multiobjectif

Avec leur parallélisme inhérent, les algorithmes évolutionnaires sont capables de trouver une multitude de solutions Pareto-optimales en un seul run (exécution). Pour ce faire, ils doivent employer certains mécanismes leur permettant d'orienter la recherche vers le front Pareto-optimal, et également de maintenir la diversité au sein de la population, afin d'éviter une convergence prématurée et assurer une bonne distribution des solutions non dominées trouvées. Dans ce qui suit, nous présentons trois de ces mécanismes ; une sélection Pareto, l'élitisme et le maintien de la diversité [3].

3.3.1 Un mécanisme de sélection Pareto

Une sélection Pareto est une sélection qui utilise la relation de dominance au sens de Pareto pour affecter des "rangs" aux individus de la population, faisant apparaître ainsi la notion de front. Goldberg est le premier à avoir proposé une technique de ce genre, appelée ranking. Dans cette technique, initialement, tous les individus non dominés de la population reçoivent

le rang 1 et sont retirés temporairement de la population. Puis, les nouveaux individus non dominés reçoivent le rang 2 avant d'être à leur tour retirés. Le processus s'itère tant qu'il reste des individus dans la population. La fitness de chaque individu correspond à son rang dans la population (ici la fitness est à minimiser). Ainsi, l'évaluation d'un individu ne dépend pas uniquement de lui-même, mais aussi du reste de la population.

3.3.2 L'élitisme

L'élitisme est une technique qui permet de conserver les meilleurs individus dans les générations futures. Une des premières implémentations de ce mécanisme dans un algorithme génétique mono-objectif a été présentée par De Jong (1975). L'élitisme permet, certes, d'améliorer les performances des algorithmes pour des fonctions unimodales, mais au risque d'entraîner une convergence prématurée pour d'autres fonctions multimodales.

Dans le cadre des problèmes multiobjectifs, plusieurs techniques ont été proposées pour conserver les individus non dominés d'une génération à l'autre, tout en essayant de maintenir une meilleure caractérisation possible de l'ensemble non dominé trouvé. Certains chercheurs ont choisi d'employer une population externe d'individus (archive), dans laquelle est stocké le meilleur ensemble de points non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et les individus stockés continuent toujours à pouvoir être choisis par l'opérateur de sélection. L'incorporation de l'élitisme dans le fonctionnement des AEMOs a permis d'améliorer nettement les performances de ces derniers, tant sur le plan de la convergence que celui de la distribution des solutions non dominées.

3.3.3 Le maintien de la diversité

Maintenir un certain degré de diversité dans la population d'un algorithme évolutionnaire consiste à éviter que la population ne converge prématurément vers une zone particulière de l'espace de recherche ou de l'espace des objectifs. En effet, s'il n'existe pas un mécanisme de "contrôle" de la diversité, les opérateurs de sélection et de remplacement vont privilégier trop vite certains individus meilleurs à cette étape de la recherche rendant difficile, sinon impossible, la découverte de nouvelles solutions optimales. Dans le cas des problèmes multiobjectifs, converger prématurément signifie être focalisé sur une partie du front Pareto, ou pire sur un front local.

De nombreuses techniques ont été développées pour remédier à ce problème telles que le sharing, le crowding et la technique de réinitialisation. Néanmoins, elles sont toutes basées sur le principe de réglage de la pression sélective, dans lequel on cherche à privilégier certains individus afin de garder une population plus diversifiée. Ceci peut être fait en faisant, par exemple, pénaliser les individus qui se ressemblent (d'après une certaine mesure de ressemblance) laissant ainsi plus de chance aux individus isolés d'être sélectionner.

3.4 Quelques algorithmes évolutionnaires performants

Récemment, beaucoup de recherches ont été menées sur l'application des AEs aux problèmes d'optimisation multiobjectifs. Celles-ci ont permis de mettre en évidence l'intérêt primordial d'utiliser des méthodes d'optimisation basées sur les concepts de Pareto et de la population. Dans ce qui suit, nous présentons trois algorithmes représentatifs résolvant des problèmes d'optimisation multiobjectifs. Ces trois algorithmes ont donné lieu à des travaux récents et nombreux, illustrant leurs originalités et leurs performances sur de nombreuses instances de problèmes théoriques et réels.

3.4.1 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II a été proposé par Deb *et al*. (2000) [9], comme une amélioration de NSGA (1995), un algorithme original basé sur la technique du ranking introduite par Goldberg. Le nouvel algorithme tente d'éliminer les trois difficultés associées au premier. Il incorpore ainsi l'aspect d'élitisme, ne requière aucun paramètre du sharing et utilise une implémentation plus rapide de la procédure du ranking. Comparativement à NSGA, NSGA-II obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de Deb, ce qui fait de cet algorithme un des plus utilisés aujourd'hui.

L'algorithme NSGA-II peut être décrit comme suit. Initialement, une population de parents P_0 est crée aléatoirement. La procédure du ranking est appliquée à cette population, où chaque solution lui est assignée une valeur de fitness égale à son niveau de non dominance (1 est le meilleur niveau). Donc, la fitness est à minimiser ici. Les opérateurs de sélection par tournoi binaire, de croisement et de mutation sont utilisés pour créer une population de descendants Q_0 de taille N. A partir de là, la procédure de NSGA-II continue, pour une génération t, en suivant les étapes ci-après :

Etape 1 : Combiner la population des parents avec celle des descendants pour donner R_t ; $R_t = P_t \bigcup Q_t$.

Etape 2 : Appliquer la procédure du ranking à R_t et identifier les différents fronts F_i, $i = 1, 2, \ldots,$ etc.

Etape 3 : Créer une nouvelle population $P_{t+1} = \Phi$. Poser $i = 1$. Tant que $|P_{t+1}| + |F_i| < N$, faire $P_{t+1} = P_{t+1} \bigcup F_i$ et $i = i + 1$.

Etape 4 : Inclure dans P_{t+1} les $(N - |P_{t+1}|)$ individus de F_i les mieux répartis au sens de la distance de crowding.

Etape 5 : Créer une population de descendants Q_{t+1} à partir de P_{t+1} en utilisant les opérateurs de sélection, de croisement et de mutation. L'opérateur de sélection par tournoi binaire sera légèrement modifié afin qu'il prenne en considération la distance de crowding. Cet opérateur,

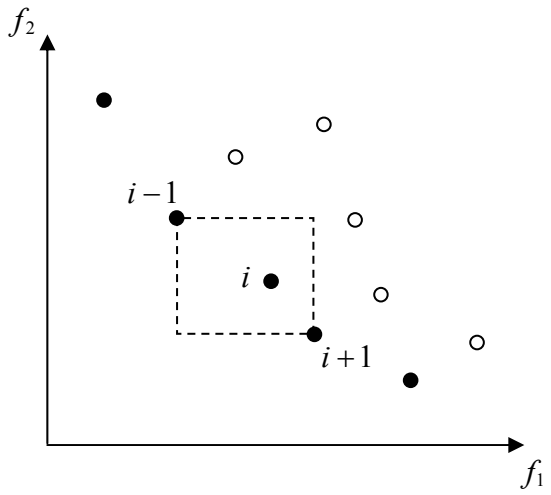


Figure 3.2 : Calcul de la distance de crowding dans NSGA-II**.**

nommé Crowded Tournament Selection, compare tout d'abord les deux individus en compétition selon leurs rangs de non dominance. Dans le cas où ils ont le même rang (c'est-à-dire qu'ils sont non dominés entre eux), alors celui qui a la plus grande distance de crowding est sélectionné.

La distance de crowding est une notion fondamentale dans le fonctionnement de NSGA-II. En effet, non seulement elle intervient dans le renouvellement de la population des parents P_{t+1}, mais elle permet aussi de maintenir une certaine diversité parmi les individus de la population se situant sur un même front. La distance de crowding d_i d'un point particulier i sert comme une estimation de la densité des solutions au voisinage de ce point. Elle se calcule en fonction du périmètre de l'hypercube ayant comme sommets les points les plus proches de i sur chaque objectif. Sur la Figure 3.2, la distance de crowding du point i est égale à la circonférence du rectangle formé par les deux point $i-1$ et $i+1$.

En conclusion, NSGA-II est un algorithme à la fois simple et efficace qui permet surtout de maintenir un grand degré de diversité dans la population, ce qui lui permettra en fait d'éviter plus facilement les dérives vers les fronts locaux. Toutefois, la répartition des solutions sur le front non dominé devient de plus en plus instable (problème de détérioration) pour des populations de tailles petites.

3.4.2 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 (2001) [37] est la seconde version de l'algorithme SPEA [35] qui a été développé initialement par Zitzler et Thiele (1998). SPEA2 diffère de son prédécesseur en trois points. Tout d'abord, il utilise une méthode de calcul de fitness améliorée qui tient compte, pour chaque individu, du nombre d'individus qu'il domine et par lesquels il est dominé. Deuxièmement, il incorpore une méthode d'estimation de densité basée sur le calcul du voisinage d'un point. Enfin, une nouvelle méthode de troncation est appliquée à l'archive,

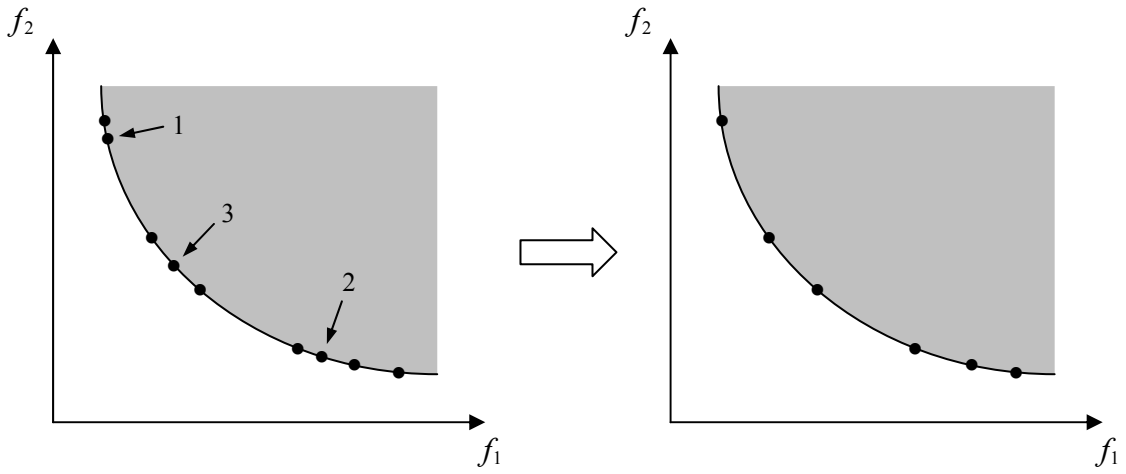


Figure 3.3 : Exemple de la procédure de troncation dans SPEA2, $\overline{N} = 6$.

afin de garantir la préservation des solutions extrêmes sur le front non dominé. Le fonctionnement de l'algorithme SPEA2 commence par la génération d'une population initiale P_0 et la création d'une archive vide $\overline{P}_0 = \Phi$, puis pour une génération t, on suit les étapes suivantes :

Etape 1 : Calculer les valeurs de fitness des individus dans P_t et $\overline{P}_t$ (voir plus loin).

Etape 2 : Copier tous les individus non dominés de P_t et $\overline{P}_t$ dans $\overline{P}_{t+1}$. Si $\left|\overline{P}_{t+1}\right| > \overline{N}$ (où $\overline{N}$ est la taille de l'archive) alors réduire $\overline{P}_{t+1}$ en utilisant l'opérateur de troncation, sinon, si $\left|\overline{P}_{t+1}\right| < \overline{N}$ alors remplir $\overline{P}_{t+1}$ par les meilleurs individus dominés de $\overline{P}_t$ et $\overline{P}_{t+1}$.

Etape 3 : Créer une nouvelle population P_{t+1} à partir de $\overline{P}_{t+1}$ en utilisant les opérateurs de sélection par tournoi binaire, de croisement et de mutation.

Pour le calcul de fitness, chaque individu i dans l'archive $\overline{P}_t$ et la population P_t, lui sera assigné une valeur $S(i)$ appelée force, égale au nombre d'individus qu'il domine :

$$S(i) = \left| \{ j \mid j \in P_t \bigcup \overline{P}_t \wedge i \prec j \} \right|. \tag{3.5}$$

La fitness d'un individu i est donnée par :

$$R(i) = \sum_{j \in P_t \cup \overline{P}_t,\, j \prec i} S(j). \tag{3.6}$$

Cela veut dire que la fitness d'un individu quelconque est déterminée par les forces des individus qu'ils domine. Notons que la fitness est encore à minimiser.

Par ailleurs, SPEA2 utilise une adaptation de la technique du "$k^{\text{ème}}$ plus proche voisin", où la densité en chaque point est une fonction décroissante de la distance entre ce point et son $k^{\text{ème}}$ plus proche voisin. L'inverse de la distance de la $k^{\text{ème}}$ solution voisine est pris comme la

mesure de densité. Plus exactement, les distances entre chaque individu i et tous les autres individus $j \in P_t \bigcup \overline{P}_t$ sont calculées dans l'espace des objectifs et sont stockées dans une liste. Cette liste est ensuite ordonnée dans l'ordre croissant, et le $k^{\text{ème}}$ élément correspond à la distance recherchée $\sigma_i^{(k)}$. Généralement, on prend $k = \sqrt{N + \overline{N}}$. La densité des solutions autour d'un point i est caractérisée alors par la quantité suivante :

$$D(i) = \frac{1}{\sigma_i^{(k)} + 2}. \tag{3.7}$$

La valeur de fitness définitive est donnée par :

$$F(i) = R(i) + D(i). \tag{3.8}$$

Ces valeurs sont utilisées lors de l'étape de mise à jour de l'archive $\overline{P}_t$. D'abord, tous les individus non dominés, c'est-à-dire ceux qui ont des valeurs de F inférieures à 1, sont copiés dans l'archive de la génération suivante. Dans le cas où le nombre d'individus non dominés est plus grand que la taille de l'archive, alors on procède à une technique de troncation qui s'applique d'une façon itérative jusqu'à avoir $\left|\overline{P}_t\right| = \overline{N}$. Pour cela, à chaque itération un individu i est supprimé de l'archive tel que $i \leq_d j, \forall\, j \in \overline{P}_{t+1}$ avec :

$$\begin{aligned} i \leq_d j \Leftrightarrow\ & \forall\, 0 < k < \left|\overline{P}_{t+1}\right| :\ \sigma_i^k = \sigma_j^k \quad \vee \\ & \exists 0 < k < \left|\overline{P}_{t+1}\right| :\ \left[(\forall\, 0 < l < k : \sigma_i^l = \sigma_j^l) \ \wedge\ \sigma_i^k < \sigma_j^k\right], \end{aligned} \tag{3.9}$$

où σ_i^k désigne la distance au $k^{\text{ème}}$ plus proche voisin de i dans $\overline{P}_{t+1}$. En d'autres termes, l'individu qui a la petite distance par rapport à un autre individu dans l'archive est choisi à ce stage. Si plusieurs individus ont une même distance minimale, alors on considère le deuxième niveaux des plus petites distances, et ainsi de suite. La Figure 3.3 illustre l'application de cette procédure dans le cas d'une archive de taille $\overline{N} = 6$. Les individus supprimés, au nombre de trois, sont indiqués selon leur ordre de suppression.

SPEA2 est un algorithme qui demande beaucoup de calculs, notamment lors de la procédure de troncation. En revanche, son schéma de calcul de fitness lui permet d'orienter la recherche d'une façon équilibrée. En plus, il offre une distribution beaucoup plus stable et uniforme que celle fournie par NSGA-II. SPEA2 est toujours considéré comme un modèle de référence pour beaucoup de travaux de recherche actuels.

3.4.3 ε-Multi-Objective Evolutionary Algorithm (ε-MOEA)

Deb *et al*. ont proposé un algorithme évolutionnaire stationnaire (2003) [12], basé sur le concept de la ε-dominance introduit par Laumanns *et al*. [22]. L'espace de recherche est dans

ce cas divisé en hypercubes, et la diversité des solutions non dominées est maintenue en faisant occuper chaque hypercube par une seule solution au maximum.

Le concept de la ε-dominance est une nouvelle formulation de la définition standard de la dominance au sens de Pareto. Ce concept permet à la fois une distribution plus stable des solutions non dominées et une convergence accentuée. Quelques définitions concernant ce concept sont données ici [22] :

Définition 3.4 (ε-dominance) Un vecteur $\mathbf{u}$ ε-domine $\mathbf{v}$ ($\varepsilon > 0$) si et seulement si $\mathbf{u} - \varepsilon$ domine $\mathbf{v}$ ($\mathbf{u} \prec_\varepsilon \mathbf{v}$), c'est-à-dire :

$$(\mathbf{u} - \varepsilon) \prec \mathbf{v}. \tag{3.10}$$

La définition de la ε-dominance n'est en fait pas unique. Celle présentée ici est basée sur une approximation additive du vecteur des objectifs. Une autre définition qui utilise une approximation multiplicative peut être faite comme suit :

$$\mathbf{u}/(1+\varepsilon) \prec \mathbf{v}. \tag{3.11}$$

Définition 3.5 (ε-Front Pareto approximé) Soit $F \in \Re^{+m}$ un ensemble de vecteurs et ($\varepsilon > 0$). Un ensemble $F_\varepsilon \subseteq F$ est dit ε-Front Pareto approximé, si chaque vecteur $\mathbf{u} \in F$ est ε-dominé par au moins un vecteur $\mathbf{v} \in F_\varepsilon$, c'est-à-dire :

$$\forall \mathbf{u} \in F : \exists \mathbf{v} \in F_\varepsilon / \mathbf{v} \prec_\varepsilon \mathbf{u}. \tag{3.12}$$

Bien évidemment, l'ensemble F_ε n'est pas unique. On dénote par $P_\varepsilon(F)$ l'ensemble de tous les ε-Fronts Pareto approximés de F.

Définition 3.6 (ε-Front Pareto) Soit $F \in \Re^{+m}$ un ensemble de vecteurs et ($\varepsilon > 0$). Un ensemble $F_\varepsilon^* \subseteq F$ est dit ε-Front Pareto si F_ε^* est un ε-Front Pareto approximé de F, c'est-à-dire $F_\varepsilon^* \in P_\varepsilon(F)$, et F_ε^* contient uniquement des objectifs non dominés de F. L'ensemble de tous les ε-Fronts Pareto de F est dénoté par $P_\varepsilon^*(F)$. Si F représente l'espace des objectifs réalisables, alors $P_\varepsilon^*(F)$ contiendra uniquement des fronts Pareto-optimaux.

ε-MOEA commence par la création d'une population initiale P_0. L'archive E_0 est constitué de solutions ε-non dominées de P_0 (au sens additif). Ensuite, une solution de P_t et une autre de E_t sont choisies pour la reproduction. Pour choisir une solution de P_t, deux solutions sont sélectionnées aléatoirement et sont comparées entre elles au sens usuel de Pareto. Si une solution domine l'autre alors elle est choisie, sinon, on choisit aléatoirement une parmi les deux. Pour choisir une solution de l'archive E_t, on utilise simplement une sélection aléatoire. Les deux solutions choisies de P_t et E_t donneront lieu à une solution descendante (par croisement et mutation), notée c.

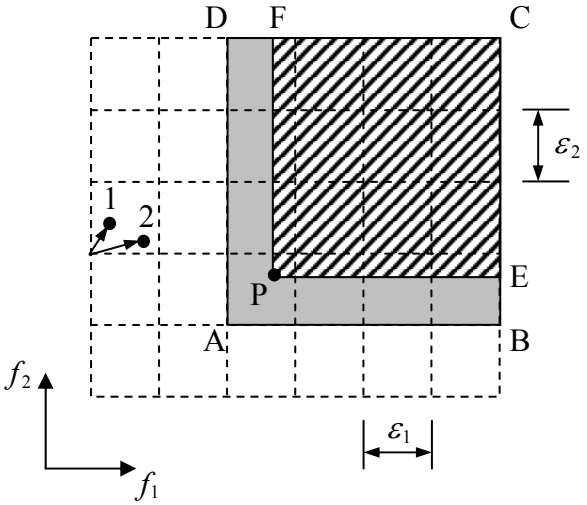


Figure 3.4 : Illustration du concept de ε-dominance.

Pour son inclusion dans l'archive, la solution c est comparée avec chaque membre de l'archive au sens de la ε-dominance. Pour cela, on associe à c et à chaque solution de l'archive un vecteur d'identification $\mathbf{B} = (B_1, B_2, \ldots, B_m)$ calculé de la façon suivante :

$$B_j(\mathbf{u}) = \lfloor (u_j - f_j^{\min}) / \varepsilon_j \rfloor, \tag{3.13}$$

où $\lfloor \ \rfloor$ dénote la partie entière d'un nombre réel. $f_j^{\min}$ désigne la valeur minimale possible de la fonction objectif f_j, et ε_j est la valeur de ε pour cette fonction. Les vecteurs d'identification divisent l'espace des objectifs en hypercubes, chacun ayant une longueur égale à ε_j suivant le $j^{\text{ème}}$ objectif. La Figure 3.4 montre que la solution P ε-domine la totalité de la région ABCDA, tandis qu'avec la relation de dominance standard P domine uniquement la région PECFP. Le vecteur d'identification de P n'est rien d'autre que les coordonnées du point A dans l'espace des objectifs.

Si le vecteur d'identification d'un membre quelconque a de l'archive domine celui de c, ce qui veut dire que la solution c est ε-dominée par a, alors la solution c est rejetée. Par ailleurs, si $\mathbf{B}(c)$ domine $\mathbf{B}(a)$, alors la solution c est acceptée et la solution a est supprimée de l'archive. Dans le cas où c est ε-non dominée avec tous les membres de l'archive, deux situations peuvent être alors considérées. Premièrement, si la solution c possède le même vecteur B qu'un autre membre de l'archive (c'est-à-dire qu'elles appartiennent au même hypercube), alors elles sont comparées tout d'abord en utilisant la relation de dominance usuelle. Si la solution c domine le membre de l'archive, ou encore, elle n'est pas dominée par ce membre mais elle est plus proche au vecteur $\mathbf{B}$ (en termes de distance euclidienne), alors elle est retenue. Les solutions 1 et 2 illustrent ce dernier cas. Elles occupent le même hypercube et sont non dominées entre elles, mais la solution 1 est plus proche du vecteur B, elle est donc retenue et la solution 2 est supprimée de l'archive. Dans le deuxième cas, où la

solution descendante ne partage avec aucun membre de l'archive son vecteur d'identification, alors elle est également ajoutée à l'archive.

Pour l'inclusion de la solution c dans la population P_t, on doit la comparer avec tous les membres de cette population. Si la solution c domine un ou plusieurs membres de P_t, alors elle remplace l'un d'entre eux (choisi aléatoirement). Autrement, si c est dominée par au moins un membre de P_t, elle est rejetée. Enfin, si c est non dominée avec tous les membres de la population, elle remplace alors un d'entre eux d'une façon aléatoire, de sorte que la taille de P_t reste inchangée.

En conclusion, ε-MOEA est un algorithme fortement élitiste qui privilégie nettement les solutions non dominées. Par conséquent, il offre beaucoup de rapidité à la convergence de l'algorithme, tout en essayant de garder une bonne distribution des solutions non dominées. Toutefois, cette stratégie de recherche risque de faire perdre, d'une façon spontanée, la diversité à la population ou même à l'archive.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les algorithmes évolutionnaires comme des méthodes de recherche très efficaces pour résoudre des problèmes d'optimisation multiobjectifs. Nous avons vu les différentes techniques et modifications qui peuvent être apportées à un AE mono-objectif afin de l'adapter à la recherche d'un ensemble de solutions Pareto-optimales, et ce en un seul run.

Les trois algorithmes évolutionnaires multiobjectifs illustrés ici ont démontré, sans aucun doute, leur fort potentiel à trouver des solutions approchées satisfaisantes pour un grand nombre de problèmes théoriques et réels (en particulier NSGA-II et SPEA2). Ces derniers sont toujours considérés comme des techniques de référence solides, en raison de leur originalité et de leur simplicité. Cependant, chacun de ces algorithmes, comme tout autre AEMO, manifeste toujours un certain compromis entre les différentes performances attendues de lui (convergence, diversité, temps d'exécution, etc). Avec l'introduction du concept de la ε-dominance, on a pu améliorer ce compromis à un certain degré.

Chapitre 4

ε-GMOEA : Un Algorithme Evolutionnaire Amélioré

Chapitre 4

ε-GMOEA : Un Algorithme Evolutionnaire Amélioré

4.1 Introduction

Les différentes méthodes de résolution présentées au chapitre précédent tentent d'approcher le mieux possible la surface de Pareto. L'algorithme ε-MOEA, en particulier, intègre un mécanisme de remplacement très robuste qui permet à la fois une convergence affinée et une bonne distribution des solutions non dominées. Quant à NSGA-II, il utilise un calcul de fitness simple et qui offre beaucoup de flexibilité au fonctionnement de l'algorithme.

Dans ce chapitre, nous proposons une nouvelle méthode évolutionnaire multiobjectif, appelée ε-GMOEA (ε-Generational Multi-Objective Evolutionary Algorithm), qui est basée essentiellement sur l'algorithme ε-MOEA. Cette méthode tente de tirer profit des différents points forts de ε-MOEA et également de NSGA-II. Une description détaillée de son implémentation est donnée dans la section suivante. Dans la Section 4.3, nous présentons une série de problèmes de test ainsi que quelques critères de performance nécessaires pour l'évaluation d'un AEMO en général. La Section 4.4 donne les résultats expérimentaux des tests conduits à titre de comparaison entre les performances des approches NSGA-II, SPEA2, ε-MOEA et ε-GMOEA.

4.2 Présentation de l'algorithme ε-GMOEA

ε-GMOEA est un algorithme évolutionnaire générationnel qui utilise le même principe de stockage des solutions non dominées que l'algorithme ε-MOEA. En plus, il intègre un mécanisme de sélection basé sur la procédure du ranking de NSGA-II. Dans ce qui suit, on donne les différents détails de son fonctionnement.

4.2.1 Description

ε-GMOEA commence par la création d'une population initiale P_0 de taille N. L'archive E_0 est une copie de P_0. Ensuite, pour une génération t, on a les étapes suivantes :

Etape 1 : Calculer les valeurs de fitness des individus de E_t (voir plus loin).

Etape 2 : Créer une nouvelle population P_{t+1} à partir de E_t en utilisant les opérateurs de sélection, de croisement et de mutation.

Etape 3 : Mettre à jour l'archive E_{t+1} en considérant les individus de P_{t+1} et E_t. Dans cette étape, on identifie tout d'abord les individus non dominés de l'union P_{t+1} et E_t. Si le nombre d'individus résultant est inférieur à N alors on choisit les N meilleurs individus (d'après leur rang de non dominance) de l'ensemble P_{t+1} plus E_t, pour former la nouvelle archive E_{t+1}. Dans le cas contraire, c'est-à-dire lorsqu'il existe plus de N individus non dominés dans P_{t+1} et E_t, on applique alors la procédure d'archivage de ε-MOEA (voir Chapitre 3), dans laquelle on considère plutôt la population P_{t+1} comme étant l'archive et les solutions de E_t comme étant des nouvelles solutions récemment crées.

Le test d'inclusion des solutions de E_t dans P_{t+1} se fait d'une façon itérative et le contenu final de P_{t+1} est copié dans E_{t+1}. Tout comme ε-MOEA et à l'opposé de SPEA2, la taille de l'archive dans ε-GMOEA n'est pas fixée a priori. En effet, cette taille dépend de la valeur du paramètre $\varepsilon_i, i = 1,\ldots,m$.

Pour le calcul des fitness des individus de E_t, on utilise une méthode basée sur l'agrégation des objectifs se situant dans un certain voisinage de l'espace des objectifs. Ce dernier est divisé en hypercubes de largeurs l_j, $j = 1,\ldots,m$. Ensuite, on identifie l'ensemble des points P_i, dits "pivots", pour lesquels l'hypercube correspondant contient au moins un vecteur de E_t (voir Figure 4.1, dans ce cas les points P_i sont des points d'intersection des lignes horizontales et verticales qui divisent l'espace des objectifs). Ensuite, les fitness des pivots P_i sont calculées en utilisant la procédure du ranking plus une estimation de densité, qui est fonction du nombre d'objectifs appartenant aux hypercubes correspondant aux P_i :

$$F(P_i) = r(P_i) + \frac{d(P_i)}{N}, \tag{4.1}$$

où $r(P_i)$ désigne le rang de non dominance de P_i, et $d(P_i)$ est le nombre d'objectifs appartenant à l'hypercube correspondant au point P_i. Ces valeurs de fitness seront utilisées ensuite dans l'étape de sélection, dans laquelle on sélectionne d'abord un pivot, en utilisant un schéma de sélection par tournoi binaire, parmi l'ensemble des pivots existants (le nombre de pivots est inférieur ou égal à la taille de l'archive). Ensuite, un individu est choisi aléatoirement parmi ceux qui partagent l'hypercube du pivot sélectionné. Notons que la fitness des pivots est à minimiser. La Figure 4.1 donne un exemple du calcul de fitness des

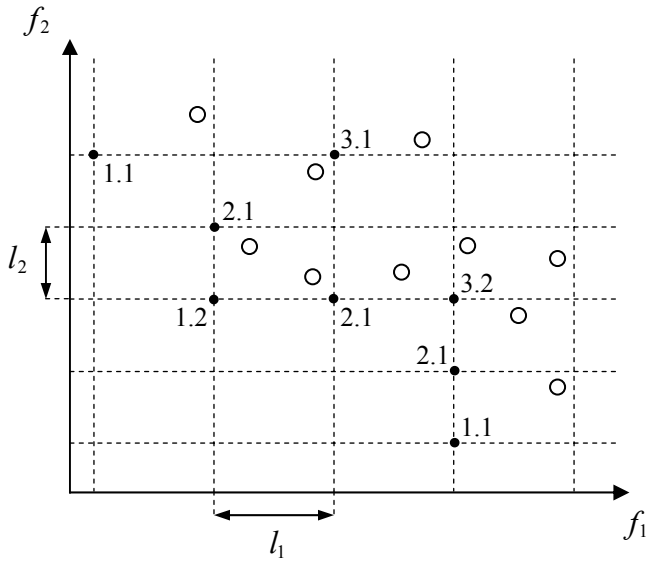


Figure 4.1 : Calcul de fitness dans ε-GMOEA.

pivots dans le cas d'une archive constituée de 10 individus (cercles vides). Les pivots, au nombre de 8, sont montrés par des points.

Concernant les largeurs l_j, $j = 1,\ldots,m$, on va se limiter dans cette étude à utiliser une simple relation, faisant intervenir un paramètre λ, qui en fait joue un rôle déterminant dans la conduite et la convergence de l'algorithme ε-GMOEA. La relation des l_j est la suivante :

$$l_j = \frac{f_j^{\max} - f_j^{\min}}{\lambda}, \tag{4.2}$$

avec

$$\lambda = \lfloor (\lambda_f - \lambda_i).(t/T)^{\alpha} + \lambda_i \rfloor, \tag{4.3}$$

où λ_i et λ_f désignent les valeurs de λ choisies pour la première et la dernière génération, respectivement. T est la nombre total de générations et t désigne l'indice de génération. Quant à α, c'est un paramètre qui doit permettre le contrôle du degré de diversité en fonction de l'indice de génération t.

4.2.2 Discussion

D'après la description précédente, on peut dire que le développement de ε-GMOEA a été largement inspiré de celui de ε-MOEA, de NSGA-II et également de SPEA2. En effet, d'abord il utilise une archive externe (SPEA2 et ε-MOEA) dans laquelle sont stockés les meilleurs individus obtenus jusqu'à une génération donnée. La sélection se fait à partir de cette archive (SPEA2) en utilisant une méthode basée sur l'agrégation des objectifs avec

l'application du ranking (NSGA-II) sur les points pivots qui en résultent. Enfin, il intègre le mécanisme d'archivage basé sur la relation de ε-dominance (ε-MOEA).

Les différences qui existent entre ε-GMOEA et ses prédécesseurs peuvent être résumées en deux points principaux. Premièrement, dans le calcul de fitness la procédure du ranking est appliquée aux points pivots et non directement aux individus de l'archive. Cette stratégie devra permettre le contrôle de la vitesse de convergence de l'algorithme par rapport à son degré de diversité. Ceci est directement lié au choix des valeurs du paramètre λ dans l'Equation (4.2). En effet, des valeurs élevées de λ permettent d'accélérer la convergence de l'algorithme mais en dépit d'une diminution de la diversité, et vice versa. Ce point de vue peut être soutenu en se référant au principe général de la ε-dominance.

Dans cette étude, on s'est limité à utiliser la relation de λ donnée par l'Equation (4.3), qui donne à l'utilisateur la possibilité de contrôler le degré de diversité de la recherche en fonction de l'indice de génération. Ainsi l'utilisateur peut choisir entre une recherche rapide qui accentue la convergence de l'algorithme et une recherche lente mais qui assure une meilleure distribution des solutions non dominées. Toutefois, comme on va le voir plus loin, des valeurs très petites de λ peuvent, dans certains cas, offrir de très bonnes convergences et diversités à la fois.

La deuxième différence qui existe entre ε-GMOEA et les autres est que ce dernier intègre un schéma particulier pour la mise à jour de l'archive lui permettant de conserver à la fois, des solutions ε-non dominées et éventuellement d'autres solutions supplémentaires, nécessaires pour le maintien de la diversité au sein de l'archive. L'idée était de faire inclure les solutions de l'ancienne archive dans la nouvelle population plutôt que le contraire (comme c'est le cas dans ε-MOEA). Le but de cela est de permettre à des solutions se situant dans un même hypercube d'être tout de même sélectionnées ensemble, afin d'entrer dans la nouvelle archive.

4.3 Evaluation

Dans le but d'évaluer les performances de l'algorithme proposé ici, nous allons le tester sur une série de problèmes théoriques, en vue de le comparer avec les trois autres algorithmes étudiés auparavant, à savoir NSGA-II, SPEA2 et ε-MOEA. Ces problèmes de test sont exprimés sous forme de problèmes de minimisation de certaines fonctions mathématiques, conçues soigneusement pour représenter plusieurs types de difficultés aux algorithmes évolutionnaires multiobjectifs.

La mesure de performance d'un AEMO est elle-même multiobjectif. Par exemple, il faut caractériser l'approche de la surface de Pareto, ainsi que la distribution des solutions non dominées par rapport à cette surface. L'objectif général d'un AEMO sera donc de trouver un

Tableau 4.1 : Fonctions de test à deux objectifs.

Problème	n	Bornes	Fonctions objectifs
FON	8	$[-4,4]$	$f_1(\mathrm{x}) = 1-\exp(-\sum_{i=1}^{n}(x_i - 1/\sqrt{n})^2)$ $f_2(\mathrm{x}) = 1-\exp(-\sum_{i=1}^{n}(x_i + 1/\sqrt{n})^2)$
SCH	10	$[-4,4]$	$f_1(\mathrm{x}) = \sum_{i=1}^{n} x_i^2$ $f_2(\mathrm{x}) = \sum_{i=1}^{n}(x_i - 2)^2$
KUR	3	$[-10^3,10^3]$	$f_1(\mathrm{x}) = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$ $f_2(\mathrm{x}) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5\sin x_i^3)$
ZDT1	30	$[0,1]$	$f_1(\mathrm{x}) = x_1$ $f_2(\mathrm{x}) = g(\mathrm{x})[1-\sqrt{x_1 / g(\mathrm{x})}]$ $g(\mathrm{x}) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$
ZDT2	30	$[0,1]$	$f_1(\mathrm{x}) = x_1$ $f_2(\mathrm{x}) = g(\mathrm{x})[1-(x_1 / g(\mathrm{x}))^2]$ $g(\mathrm{x}) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$
ZDT3	30	$[0,1]$	$f_1(\mathrm{x}) = x_1$ $f_2(\mathrm{x}) = g(\mathrm{x})[1-\sqrt{x_1 / g(\mathrm{x})} - \frac{x_1}{g(\mathrm{x})}\sin(10\pi x_1)]$ $g(\mathrm{x}) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$, $i = 2,\ldots,n$	$f_1(\mathrm{x}) = x_1$ $f_2(\mathrm{x}) = g(\mathrm{x})[1-\sqrt{x_1 / g(\mathrm{x})}]$ $g(\mathrm{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$
ZDT6	10	$[0,1]$	$f_1(\mathrm{x}) = 1-\exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(\mathrm{x}) = g(\mathrm{x})[1-(f_1(\mathrm{x}) / g(\mathrm{x}))^2]$ $g(\mathrm{x}) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$

ensemble de solutions (individus de la population) qui représentera le mieux le front de Pareto. Dans ce qui suit, nous présentons les fonctions de tests et les métriques de performance utilisées pour l'évaluation des AEMOs précédents.

4.3.1 Fonctions de test

La série de fonctions de test utilisée est divisée en deux parties. La première concerne des problèmes impliquant l'optimisation de deux fonctions objectifs. La deuxième partie traite des problèmes à trois objectifs.

Dans le Tableau 4.1 sont décrites les fonctions de test bi-objectifs à utiliser [9]. Elles représentent les problèmes : FON (Fonseca et Fleming, 1995), SCH (problème généralisé de

Tableau 4.2 : Fonctions de test à trois objectifs.

Problème	*n*	Bornes	Fonctions objectifs
DTLZ1	7	[0,1]	$f_1(\mathrm{x}) = \frac{1}{2} x_1 x_2 (1 + g(\mathrm{x_m}))$ $f_2(\mathrm{x}) = \frac{1}{2} x_1 (1 - x_2)(1 + g(\mathrm{x_m}))$ $f_3(\mathrm{x}) = \frac{1}{2}(1 - x_1)(1 + g(\mathrm{x_m}))$ $g(\mathrm{x_m}) = 100[5 + \sum_{x_i \in \mathrm{x_m}} [(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]]$ $\mathrm{x_m} = \{x_i,\ i = m, \cdots, n\}$
DTLZ3	12	[0,1]	$f_1(\mathrm{x}) = (1 + g(\mathrm{x_m}))\cos(x_1 \frac{\pi}{2})\cos(x_2 \frac{\pi}{2})$ $f_2(\mathrm{x}) = (1 + g(\mathrm{x_m}))\cos(x_1 \frac{\pi}{2})\sin(x_2 \frac{\pi}{2})$ $f_3(\mathrm{x}) = (1 + g(\mathrm{x_m}))\sin(x_1 \frac{\pi}{2})$ $g(\mathrm{x_m}) = 100[10 + \sum_{x_i \in \mathrm{x_m}} [(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]]$ $\mathrm{x_m} = \{x_i,\ i = m, \cdots, n\}$

Schaffer, 1985), KUR (Kursawe, 1990) et la série des problèmes ZDT (Zitzler *et al*., 2000). Le problème FON possède un front de Pareto non convexe et ses solutions optimales sont données par $x_1 = x_2 = \cdots x_n \in [-1/\sqrt{n}, 1/\sqrt{n}]$. SCH est un problème qui possède un front de Pareto convexe avec des solutions optimales telles que $x_1 = x_2 = \cdots x_n \in [0, 2]$. Le problème KUR est plus compliqué et possède un front de Pareto discontinu et non convexe. Il est également caractérisé par un large domaine de définition des variables et son ensemble optimal est difficile à trouver [8]. Par ailleurs, les problèmes ZDT1, ZDT2 et ZDT3 impliquent un nombre élevé de paramètres et ont respectivement un front de Pareto convexe, non convexe et discontinu. Les fronts Pareto-optimaux sont donnés par $g = 1$. Le problème ZDT4, quant à lui, possède 21^9 fronts locaux et devient de ce fait un bon exemple pour tester l'habilité d'un AEMO à traiter des cas de multimodalité. Le front de Pareto est également donné par $g = 1$. Enfin, la fonction ZDT6 présente deux difficultés qui résultent de la non uniformité de l'espace des objectifs. Premièrement, les solutions Pareto-optimales sont distribuées d'une façon non uniforme sur le front de Pareto. Deuxièmement, la densité des solutions diminue en général aux approches de ce même front. Celui-ci, non convexe, est obtenu en posant encore $g = 1$.

Afin de démontrer les capacités des AEMOs à traiter des problèmes de dimensions plus élevées, on considère deux autres problèmes de la série DTLZ (Deb *et al*., 2001) impliquant l'optimisation de trois fonctions objectifs (Tableau 4.2) [13]. Le premier problème, DTLZ1, possède un front de Pareto linéaire obtenu pour $x_i = 0.5,\ i = m, \ldots, n$. Ce front est décrit par l'équation suivante :

$$\sum_{i=1}^{3} f_i = 0.5 . \tag{4.4}$$

La difficulté avec ce problème est de devoir converger vers le front Pareto-optimal tout en évitant (11^5-1) fronts locaux.

Par ailleurs, le problème DTLZ3 contient (3^{10}-1) fronts locaux, tous parallèles au front Pareto-optimal. Celui-ci, obtenu pour $x_i = 0.5,\ i = m,\ldots,n$, est donné par l'équation :

$$\sum_{i=1}^{3} f_i^2 = 1. \tag{4.5}$$

4.3.2 Métriques de performance

Différentes métriques de performance ont été proposées dans le cadre de l'optimisation multiobjectif. Certaines d'entre elles sont des métriques relatives, qui comparent deux ensembles en utilisant l'un d'entre eux comme référence. D'autres métriques, absolues, évaluent un ensemble sans avoir besoin d'autres points ou ensemble de référence.

Pour comparer deux ensembles (ou surfaces) de compromis, il faut tenir compte de plusieurs aspects, notamment : la qualité (la proximité par rapport au front de Pareto théorique), la distribution (est-ce que toutes les parties du front de Pareto sont découvertes), la répartition (est-ce que les points sont répartis de manière homogène sur le front). Il est difficile, voir impossible, de prendre en considération tous ces paramètres au travers d'une seule valeur numérique. C'est pourquoi il est courant d'utiliser plusieurs mesures (ou métriques) pour tester tel ou tel aspect de l'ensemble [3]. Dans ce qui suit, nous présentons quelques mesures permettant d'évaluer les performances d'un AEMO en se basant sur le front de Pareto théorique, déjà connu pour toutes les instances de problèmes de test considérées ici.

Une des métriques les plus connues et utilisées dans la littérature est la métrique de convergence *GD* (Generational Distance, Van Veldhuizen, 1999) [7] qui calcule l'écart moyen entre le front expérimental constitué par l'ensemble non dominé F trouvé par l'algorithme et le front théorique F^* :

$$GD = \frac{1}{|F|} \sum_{\mathbf{u} \in F} \min\{\| \mathbf{u} - \mathbf{u}^* \|_2, \mathbf{u}^* \in F^*\}, \tag{4.6}$$

où $\| \|_2$ représente la distance euclidienne calculée dans l'espace des objectifs. L'inconvénient avec cette métrique est qu'elle ne tient pas compte des divergences locales entre les solutions non dominées trouvées le long du front expérimental. Cela est généralement dû au problème de détérioration partielle auquel sont exposées les différentes méthodes évolutionnaires multiobjectifs.

D'autre part, Xue propose une autre technique de caractérisation plus significative [33], dans laquelle on différencie entre les solutions non dominées trouvées selon leur contribution

à l'image générale du front non dominé par rapport au front de Pareto. Pour cela, au lieu de calculer les distances des solutions non dominées au front Pareto-optimal, on calcule plutôt la distance des solutions Pareto-optimales au front expérimental trouvé. Pour chaque solution dans le front de Pareto, F^*, on mesure sa distance par rapport à F :

$$d_i = \min_{\mathbf{u} \in F} \| \mathbf{u}_i^* - \mathbf{u} \|_2 . \tag{4.7}$$

On peut trouver que certaines solutions non dominées ne sont pas impliquées dans le calcul de ces distances car aucune solution de F^* possède une distance minimale par rapport à ces solutions. On peut donc identifier l'ensemble des solutions de F qui contribuent réellement au calcul des d_i. Cet ensemble, noté M est défini comme suit :

$$M = \bigcup_{\mathbf{u}_k^* \in F^*} \{ \mathbf{u} \mid \| \mathbf{u}_k^* - \mathbf{u} \|_2 = \min_{\mathbf{u}_i \in F} \| \mathbf{u}_k^* - \mathbf{u}_i \|_2 \} . \tag{4.8}$$

Par ailleurs, il est possible d'avoir des solutions Pareto-optimales qui ont une distance minimale au front F par rapport à une même solution non dominée. Par suite, on peut regrouper les distances minimales de l'ensemble des solutions Pareto-optimales, d'après leurs solutions non dominées communes $\mathbf{u}_m, m \in \{1, \cdots, |M|\}$. Ces groupes sont notés $D_1, D_2, \cdots, D_{|M|}$, où le groupe $D_m, m \in \{1, \cdots, |M|\}$ contient les distances minimales associées à $\mathbf{u}_m$. On peut alors définir l'ensemble D_{min} comme suit :

$$D_{min} = \bigcup_{i=1,\cdots,|M|} \{\min(D_i)\}, \tag{4.9}$$

qui est un ensemble contenant les distances minimales de chacun des groupes $D_1, D_2, \cdots, D_{|M|}$. Xue définit ainsi la mesure de performance $\tilde{d}_{min}$ comme étant le médian de l'ensemble D_{min}. L'utilisation du médian est dû au fait qu'il peut y avoir des solutions extrémes dans le front non dominé. Dans de ce cas, le médian fournit une estimation plus robuste et plus significative. Une autre mesure qui concerne la distribution des solutions non dominées est la suivante :

$$\bar{d}_q = \frac{1}{|F_q^*|} \sum_{\mathbf{u}^* \in F_q^*} \min \| \mathbf{u}^* - \mathbf{u} \|_2 , \tag{4.10}$$

où F_q^*, qui est un sous-ensemble de F^*, contient le quart des solutions Pareto-optimales ayant les distances les plus larges par rapport à F.

Le calcul de ces deux métriques est illustré à la Figure 4.2. Dans ce cas, le front expérimental F, composé de solutions non dominées, est représenté par des cercles. Le front de Pareto F^*, originellement représenté par une courbe continue, est discrétisé en un

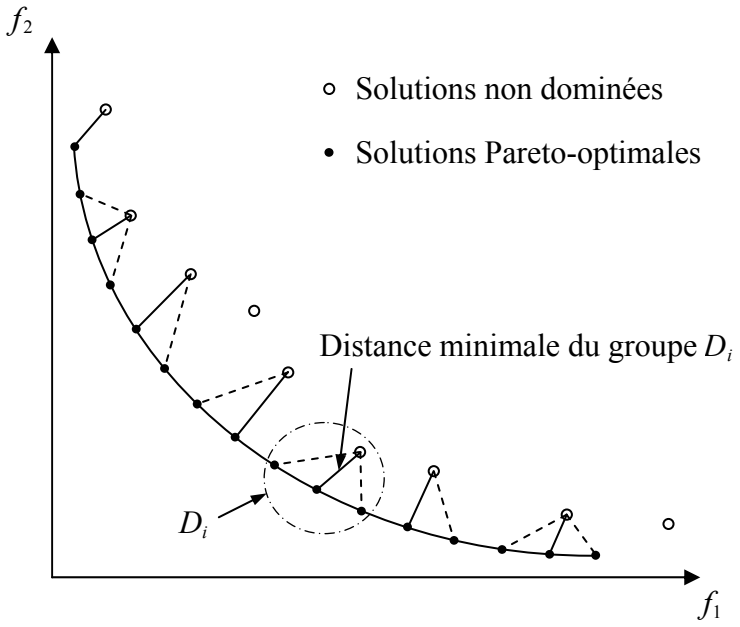


Figure 4.2 : Exemple illustratif pour le calcul des métriques $\widetilde{d}_{min}$, $\overline{d}_q$ et $\overline{d}_{max}$.

ensemble de points équidistants appartenant à la courbe théorique. Le nombre de points Pareto-optimaux détermine la précision des différentes mesures calculées. Pour chaque solution de F^*, est calculée sa distance minimale par rapport à F. On remarque que certaines solutions non dominées ne sont pas impliquées dans le calcul de ces distances. L'ensemble D_{min}, quant à lui, est formé par les distances minimales représentées par un trait solide dans chaque groupe D_i. Son médian, $\widetilde{d}_{min}$, mesure donc la convergence vers le front Pareto-optimal.

Toutefois, on remarque que la métrique $\overline{d}_q$ ne fournit pas une mesure robuste quant à la distribution des solutions non dominées par rapport au front Pareto-optimal. En effet, l'Equation (4.10) montre que cette mesure dépend sensiblement du nombre de solutions Pareto-optimales choisies pour représenter le front de Pareto. Un nombre élevé de solutions Pareto-optimales risque en fait d'augmenter l'influence de quelques régions où la distance entre le front non dominé et celui de Pareto est relativement plus large par rapport à d'autres régions. Pour remédier à ce problème, on propose ici une nouvelle métrique qui permet d'avoir une caractérisation beaucoup plus robuste de l'aspect de distribution. Cette métrique, notée $\overline{d}_{max}$, est définie comme suit :

$$\overline{d}_{max} = \frac{1}{|M|}\sum_{i=1}^{|M|}\{\max(D_i)\}. \tag{4.11}$$

C'est une approche qui est moins sensible au nombre de solutions Pareto-optimales utilisées pour représenter le front de Pareto, et permet également d'avoir une idée sur la convergence des solutions non dominées trouvées.

Tableau 4.3 : Valeurs des paramètres ε et λ_i .

Problème	$\varepsilon_i, i = 1,\cdots,m$	λ_i
FON	0.007	30
SCH	0.02	30
KUR	0.035	20
ZDT1	0.0075	30
ZDT2	0.0076	3
ZDT3	0.0026	5
ZDT4	0.0075	15
ZDT6	0.0068	1
DTLZ1	0.035	15
DTLZ3	0.06	20

4.4 Résultats expérimentaux

Dans cette section, nous présentons les résultats expérimentaux des quatre algorithmes évolutionnaires étudiés dans ce travail. Les résultats décrits ici sont obtenus en réalisant une série de dix runs indépendants pour chacune des fonctions de test présentées dans la section précédente. Pour l'évaluation des performances des AEMOs, les métriques GD, $\widetilde{d}_{min}$ et $\overline{d}_{max}$ sont utilisées, en calculant leurs valeurs moyennes et déviations standard pour toutes les instances des dix runs effectués.

Les AEMOs sont exécutés en utilisant une population d'une taille égale à 100 (même taille pour l'archive de SPEA2), et ce pour toute la série de tests. Le nombre de générations est de 200 pour tous les problèmes à deux objectifs, 300 et 500 pour DTLZ1 et DTLZ3, respectivement. Notons que tous les algorithmes évolutionnaires utilisent le même nombre d'évaluations des fonctions objectifs, qui est égal au produit du nombre d'individus et du nombre de générations. Par ailleurs, on utilise comme opérateurs de variation le croisement binaire simulé et la mutation polynomiale (voir Chapitre 2) avec des indices de distribution de 15 et 20, respectivement. La probabilité de croisement est égale à 1 et le taux de mutation à $1/n$, où n est le nombre de variables de décision dans chaque problème.

Concernant les valeurs du paramètre ε, elles ont été choisies de telle façon à obtenir un nombre final de solutions ε-non dominées approximativement égal au nombre d'individus de la population. Certes, ces valeurs ne sont pas connues a priori et doivent être posées par essai/erreur. Ces valeurs sont données dans le Tableau 4.3. Dans le même tableau, sont données également les valeurs du paramètre λ_i de l'algorithme ε-GMOEA. Un choix également fait par essai/erreur. Les deux autres paramètres, à savoir λ_f et α, sont fixés à 100 et 20, respectivement. Ces valeurs reflètent l'idée de vouloir assurer d'abord une bonne diversité dans l'archive pour un nombre maximal de générations avant de laisser progressivement plus de liberté et d'affinité à la recherche.

Tableau 4.4: Comparaison des performances des AEMOs. Les meilleurs résultats sont indiqués en caractère gras.

Problème	AEMO	*GD*		$\widetilde{d}_{min}$		$\overline{d}_{max}$	
		Moy ($\times 10^{-3}$)	Dév std ($\times 10^{-4}$)	Moy ($\times 10^{-3}$)	Dév std ($\times 10^{-4}$)	Moy ($\times 10^{-3}$)	Dév std ($\times 10^{-4}$)
FON	NSGA-II	5.15	3.1	4.60	2.7	10.86	3.1
	SPEA2	4.39	2.2	3.96	3.2	9.37	1.2
	ε-MOEA	3.35	3.7	**2.21**	2.6	10.75	4.6
	ε-GMOEA	**2.88**	2.2	2.51	2.7	**9.16**	2.4
SCH	NSGA-II	16.37	13.6	14.53	15.8	46.97	8.5
	SPEA2	13.28	5.3	11.98	8.3	**41.36**	9.7
	ε-MOEA	9.68	10.9	**5.89**	6.5	53.73	42.4
	ε-GMOEA	**8.44**	4.4	7.26	4.6	44.47	24.3
KUR	NSGA-II	11.30	17.1	7.71	12.2	79.33	31.6
	SPEA2	9.44	11.3	6.89	10.4	**64.92**	22.6
	ε-MOEA	9.37	48.8	5.43	9.4	89.14	258.2
	ε-GMOEA	**5.41**	5.8	**4.82**	5.1	67.69	14.5
ZDT1	NSGA-II	1.69	2.4	1.26	2.0	9.19	2.2
	SPEA2	1.66	2.6	1.47	3.1	**7.87**	0.9
	ε-MOEA	0.60	0.7	0.59	0.6	8.24	1.0
	ε-GMOEA	**0.43**	0.6	**0.41**	0.6	8.16	0.4
ZDT2	NSGA-II	1.55	4.8	1.27	4.2	9.36	2.6
	SPEA2	1.69	3.3	1.53	3.3	**8.04**	1.6
	ε-MOEA	0.50	3.2	0.48	3.1	8.21	2.2
	ε-GMOEA	**0.23**	0.2	**0.22**	0.2	8.13	1.2
ZDT3	NSGA-II	0.74	1.0	0.52	0.6	11.33	3.0
	SPEA2	0.82	1.7	0.60	1.1	9.91	0.9
	ε-MOEA	0.59	0.8	0.49	0.5	10.07	11.1
	ε-GMOEA	**0.28**	0.6	**0.24**	0.5	**9.56**	2.0
ZDT4	NSGA-II	5.96	28.3	5.91	30.0	11.43	18.3
	SPEA2	5.55	38.8	5.42	40.0	10.12	30.0
	ε-MOEA	3.37	19.3	3.42	21.1	11.04	15.6
	ε-GMOEA	**2.85**	20.2	**2.88**	20.9	**9.68**	12.6
ZDT6	NSGA-II	19.94	33.1	16.79	32.1	19.43	30.6
	SPEA2	25.61	39.1	22.98	44.2	25.16	40.3
	ε-MOEA	6.54	11.4	6.37	11.7	8.72	9.4
	ε-GMOEA	**1.66**	0.7	**0.93**	0.8	**5.26**	1.6
DTLZ1	NSGA-II	3.36	7.1	3.88	7.1	43.35	9.6
	SPEA2	4.55	19.6	3.77	6.2	35.00	5.9
	ε-MOEA	2.75	5.3	3.51	5.0	30.20	4.0
	ε-GMOEA	**2.62**	1.1	**3.40**	1.2	**29.81**	5.3
DTLZ3	NSGA-II	23.07	63.7	22.83	66.8	127.32	33.5
	SPEA2	282.89	5539.7	17.74	80.4	**100.03**	24.3
	ε-MOEA	9.81	25.4	10.03	26.1	111.42	39.6
	ε-GMOEA	**5.79**	8.0	**5.99**	7.3	109.26	32.3

Tableau 4.5 : Comparaison des valeurs de $|GD - \widetilde{d}_{min}|$.

Problème	NSGA-II ($\times 10^{-4}$)	SPEA2 ($\times 10^{-4}$)	ε-MOEA ($\times 10^{-4}$)	ε-GMOEA ($\times 10^{-4}$)
FON	5.5	4.3	11.4	**3.7**
SCH	18.4	13.0	37.9	**11.8**
KUR	35.9	25.5	39.4	**5.9**
ZDT1	4.3	1.9	**0.1**	0.2
ZDT2	2.8	1.6	0.2	**0.1**
ZDT3	2.2	2.2	1.0	**0.4**
ZDT4	0.5	1.3	0.5	**0.3**
ZDT6	31.5	26.3	**1.7**	7.3
DTLZ1	**5.2**	7.8	7.6	7.8
DTLZ3	2.4	2651.5	2.2	**2.0**

Dans le Tableau 4.4, se récapitulent les résultats obtenus des tests effectués. Nous remarquons que l'algorithme ε-GMOEA présente les meilleures résultats de convergence (métrique GD) sur toutes les instances de test, suivi toujours de ε-MOEA. La même remarque peut être faite pour la métrique $\widetilde{d}_{min}$, à l'exception des deux problèmes FON et SCH où ε-MOEA s'impose. Par ailleurs, du même point de vue, l'algorithme NSGA-II semble être similaire à SPEA2 dans l'ensemble des tests. Une vue générale des résultats de convergence montre clairement la supériorité des algorithmes utilisant le concept de la ε-dominance.

Concernant la métrique $\overline{d}_{max}$, on peut dire qu'il y a une légère diminution des performances de ε-GMOEA et ε-MOEA, ce qui est évident vu la manière par laquelle la mise à jour de l'archive est effectuée avec ces deux algorithmes. En effet, on peut dire que la technique de stockage employée par ces deux algorithmes est sensible aux degrés de convexité du front Pareto-optimal. Les performances de ε-GMOEA et ε-MOEA concernant cette métrique sont donc limitées. Par contre, la méthode de troncation employée par l'algorithme SPEA2 lui procure une meilleure distribution des solutions non dominées, ce que témoignent déjà les résultats obtenus.

Une autre caractérisation de la convergence des solutions est la valeur absolue de la différence entre les métriques GD et $\widetilde{d}_{min}$. Cette mesure permet d'avoir une idée sur l'homogénéité de la disposition des solutions non dominées le long du front expérimental. D'autre part, elle donne une estimation statistique de la régularité de distribution des solutions non dominées par rapport au front de Pareto, étant donné que la métrique GD utilise la moyenne des distances tandis que $\widetilde{d}_{min}$ se sert du médian. Les valeurs de cette mesure sont données dans Tableau 4.5. On remarque que ε-GMOEA se distingue encore une fois dans la majorité des cas. Des résultats typiques des performances des quatre algorithmes testés sont illustrés sur les Figures (4.3) à (4.12). Les fronts non dominés, montrés par des points, correspondent aux médians des valeurs de la métrique GD. Les fronts de Pareto sont représentés pour chaque fonction de test par un trait continu.

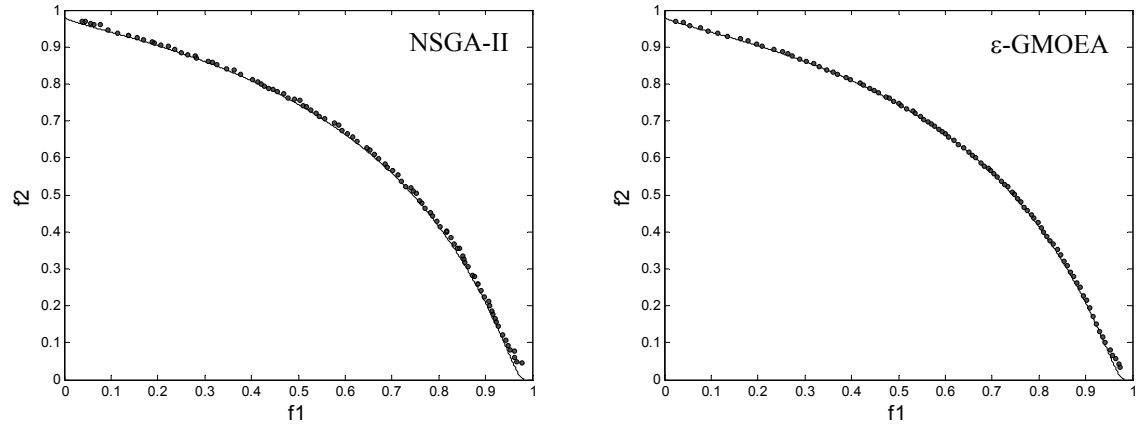


Figure 4.3 : Fronts expérimentaux pour le problème FON.

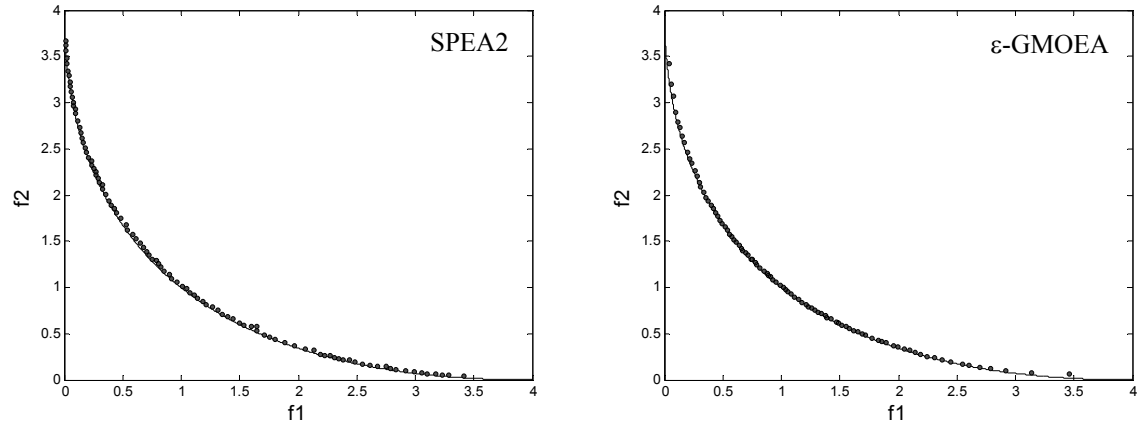


Figure 4.4 : Fronts expérimentaux pour le problème SCH.

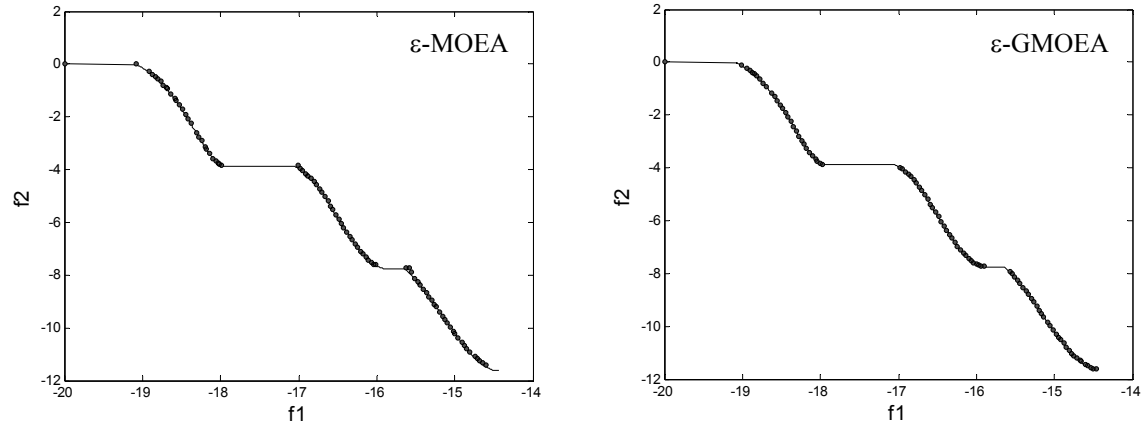


Figure 4.5 : Fronts expérimentaux pour le problème KUR.

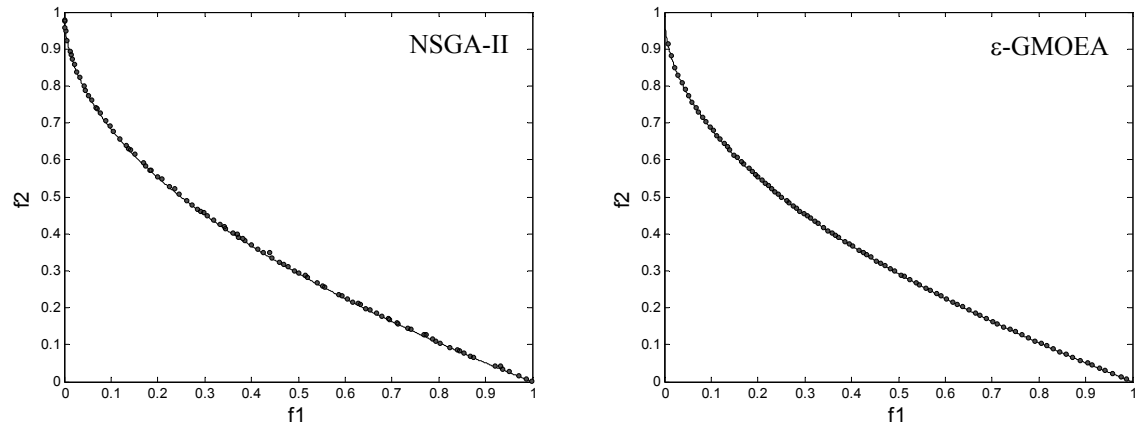


Figure 4.6 : Fronts expérimentaux pour le problème ZDT1.

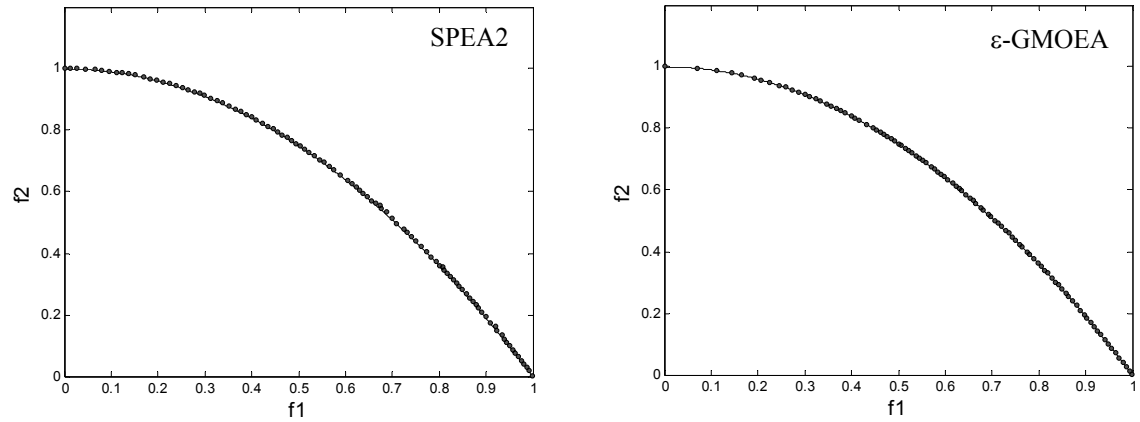


Figure 4.7 : Fronts expérimentaux pour le problème ZDT2.

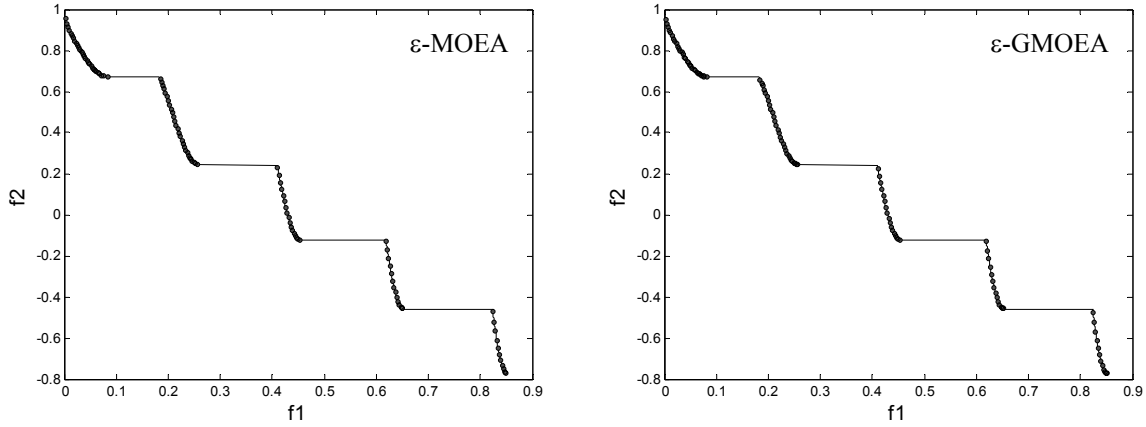


Figure 4.8 : Fronts expérimentaux pour le problème ZDT3.

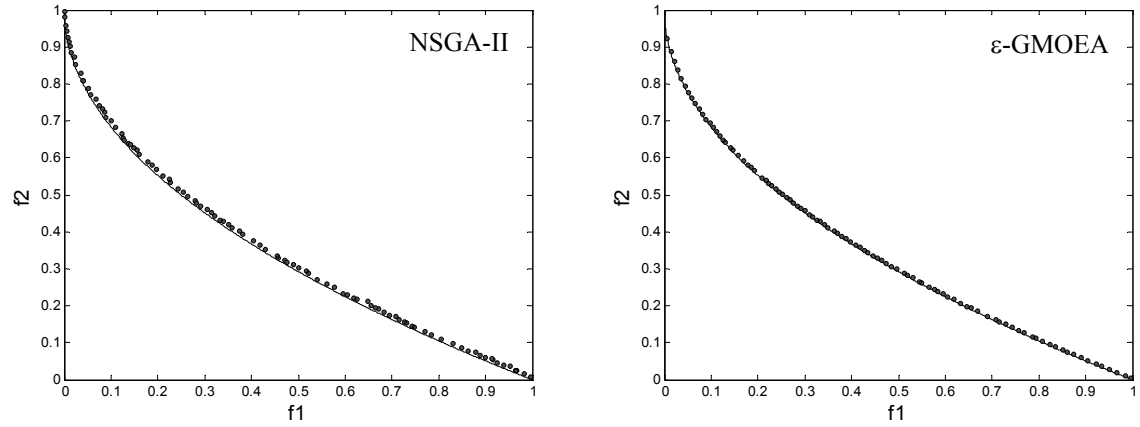


Figure 4.9 : Fronts expérimentaux pour le problème ZDT4.

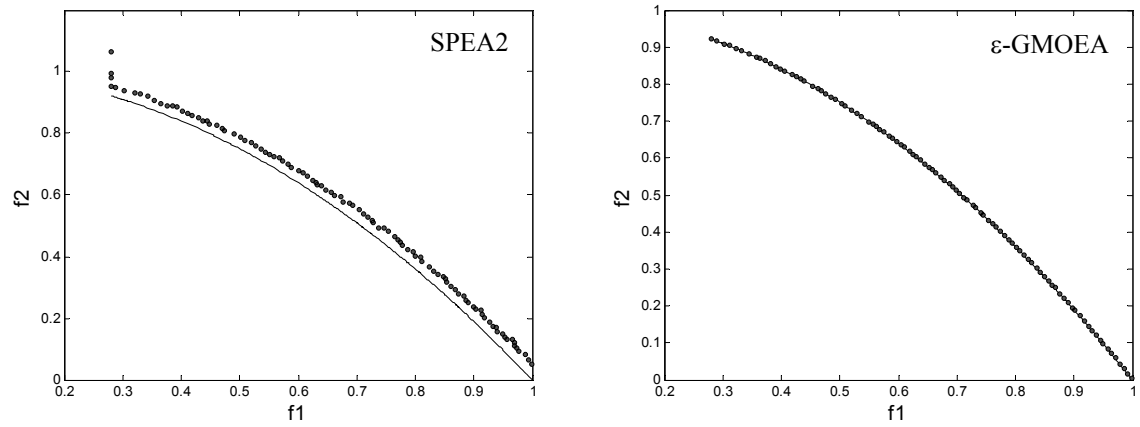


Figure 4.10 : Fronts expérimentaux pour le problème ZDT6.

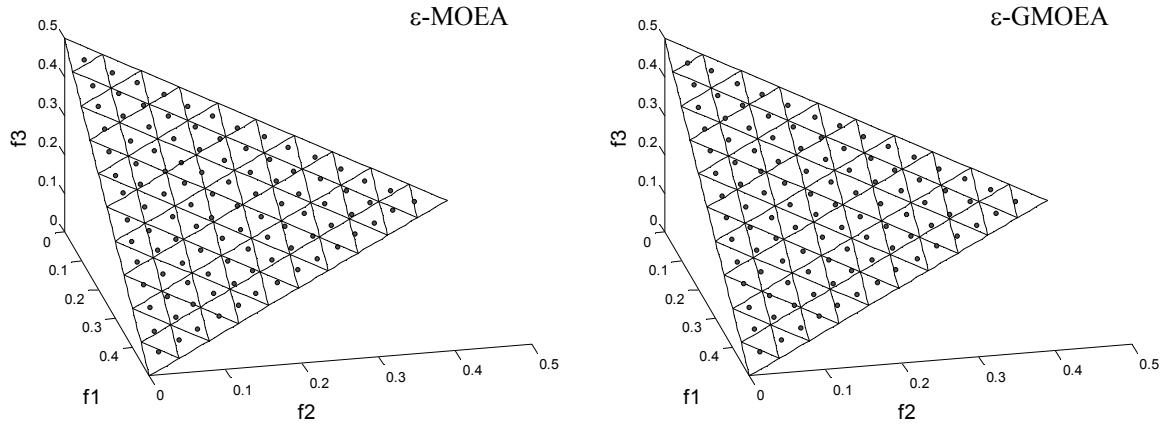


Figure 4.11 : Fronts expérimentaux pour le problème DTLZ1.

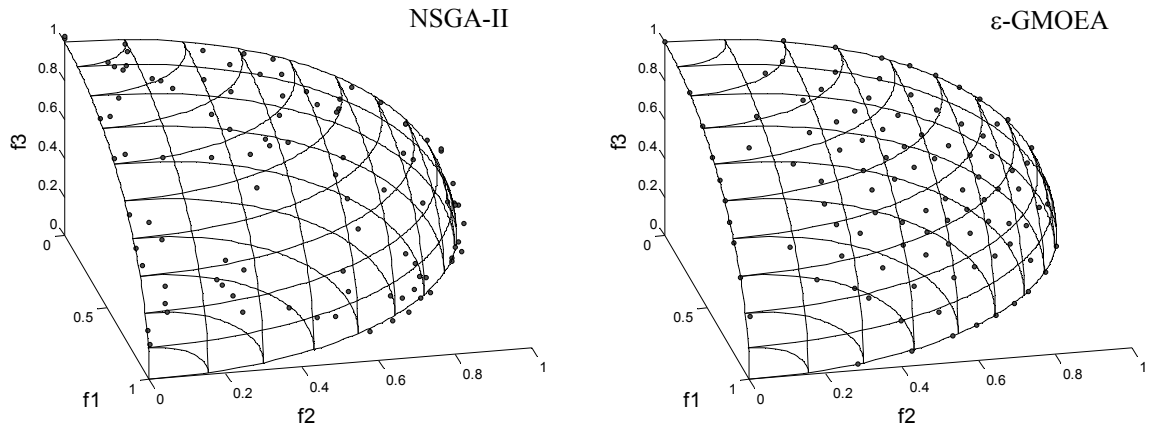


Figure 4.12 : Fronts expérimentaux pour le problème DTLZ1.

4.5 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle implémentation d'un algorithme évolutionnaire multiobjectif, nommée ε-GMOEA, en se basant sur les trois algorithmes évolutionnaires présentés au Chapitre 3. La méthode ε-GMOEA exploite les différents avantages et points forts de ses prédécesseurs et ajoute quelques particularités afin d'aboutir à des résultats meilleurs.

Les performances de ε-GMOEA ont été validées sur un ensemble de dix instances de test, largement évoquées dans la littérature. Les résultats expérimentaux ont montré que les algorithmes se basant sur le concept de la ε-dominance permettent d'obtenir de meilleurs résultats du point de vue convergence, mais n'assurent pas forcément une bonne distribution des solutions non dominées. D'autre part, bien que ε-GMOEA se montre très performant, il a toujours l'inconvénient de devoir choisir les valeurs du paramètre λ d'une façon manuelle. Dans cette étude, nous avons utilisé une relation qui lie ce paramètre à l'indice de génération, dans le but de contrôler le degré de diversité par rapport au nombre total de générations. Afin d'illustrer encore une fois les performances de ε-GMOEA, ce dernier est utilisé dans le prochain chapitre, pour la résolution d'un problème d'optimisation multiobjectif réel, tiré du domaine de l'automatique.

Chapitre 5

Application à un Problème de Commande d'un Réseau Electrique Multimachine

Chapitre 5

Application à un Problème de Commande d'un Réseau Electrique Multimachine

5.1 Introduction

Les réseaux électriques sont des systèmes complexes décrits généralement par des modèles mathématiques non linéaires et de dimension élevée. Ils sont caractérisés par plusieurs unités de génération d'énergie électrique, géographiquement éloignées, et reliées entre elles par des lignes de transport qui acheminent l'énergie électrique vers les consommateurs. Un des problèmes majeurs qui concerne les réseaux électriques est le maintien de leur stabilité transitoire, suite à des perturbations rapides et sévères. Ces perturbations allant, le plus souvent, jusqu'à dépasser les techniques de commande linéaires conventionnelles. Par ailleurs, vu la nature interconnectée des réseaux électriques, la décentralisation de la commande s'avère une approche plus simple et plus efficace.

Dans ce chapitre, nous proposons une nouvelle commande décentralisée des réseaux électriques multimachines basée sur l'association de la technique du backstepping et la commande par logique floue. Cette approche fera ensuite l'objet d'une synthèse multiobjectif en utilisant l'algorithme ε-GMOEA, dans le but de démontrer l'intérêt d'utiliser de telles méthodes pour l'optimisation des paramètres d'un régulateur selon les différents objectifs posés. Dans la section suivante, on présente notre approche de réglage pour le cas général des réseaux électriques multimachines, en mettant l'accent sur le point clé de l'association backstepping/commande floue. La Section 5.3 donne les détails d'implémentation de l'algorithme ε-GMOEA et les résultats de simulation de l'optimisation multiobjectif des paramètres de réglage, pour le cas particulier d'un réseau électrique à trois machines.

5.2 Conception d'une commande décentralisée par logique floue et backstepping pour un réseau électrique multimachine

La logique floue, de part sa simplicité et son efficacité, a été largement adoptée pour la conception de stabilisateurs de réseaux électriques (PSS ; Power System Stabilizers). En effet, c'est avant tout un outil qui permet de concevoir des régulateurs efficaces et robustes en utilisant de simples règles floues, sans avoir à connaître le modèle mathématique précis du système à commander. Ceci met beaucoup d'importance et d'intérêt sur le choix des régulateurs flous pour le cas des systèmes à paramètres incertains ou bien des systèmes complexes avec des termes d'interconnexion.

Pour synthétiser un régulateur flou, on doit d'abord choisir les variables d'entrée et de sortie de ce régulateur et déterminer la plage de leur variation. Ensuite, on construit les sous-ensembles flous dans chaque plage de variation et définir les fonctions d'appartenance associées. Puis, on doit choisir les méthodes d'inférence et de défuzzification, et enfin construire la base des règles floues. Ces règles peuvent être obtenues à l'aide de connaissances préalables sur le comportement du système pour plusieurs points de fonctionnement. La structure du régulateur flou est donc simple et en même temps efficace.

Dans le but d'améliorer les performances des régulateurs flous, ces derniers sont parfois combinés avec d'autres techniques de commande pour éliminer certaines difficultés associées à l'une ou à l'autre de ces techniques. Ils ont été également combinés avec des techniques intelligentes et métaheuristiques pour former un ensemble d'outils complémentaires visant à optimiser les performances générales du système de commande.

La présente section est consacrée à la conception d'une commande décentralisée pour le maintien de la stabilité transitoire d'un réseau électrique multimachine. Cette commande est basée essentiellement sur la logique floue, mais qui nécessite un apport déterminant de la part de la technique du backstepping. Dans ce qui suit, nous présentons tout d'abord le modèle dynamique du réseau électrique multimachine. Ensuite, nous donnons les détails d'implémentation de la commande proposée.

5.2.1 Modèle dynamique du réseau électrique

Considérons un réseau électrique composé de n générateurs, reliés entre eux par des lignes de transport. En tenant compte de la commande du circuit d'excitation du générateur et celle de la puissance mécanique (vannage de la turbine), le modèle dynamique de chaque générateur peut alors être donné comme suit [17] [34] :

Equations mécaniques du générateur

$$\dot{\delta}_i = \omega_i , \tag{5.1}$$

$$\dot{\omega}_i = -\frac{D_i}{2H_i}\omega_i + \frac{\omega_0}{2H_i}(P_{mi} - P_{ei}), \tag{5.2}$$

où δ_i désigne l'angle électrique (ou de charge) du générateur, en rad. ω_i : l'erreur de la vitesse rotorique, en rad/sec, et ω_0 la vitesse de synchronisme. P_{mi} : la puissance mécanique fournie au générateur, en p.u. P_{ei} : la puissance électrique active, en p.u. H_i : la constante d'inertie, en sec. D_i : le coefficient d'amortissement, en p.u.

Dynamiques électriques du générateur

$$\dot{E}'_{qi} = \frac{1}{T'_{d0i}}(E_{fdi} - E_{qi}), \tag{5.3}$$

$$\dot{E}_{fi} = \frac{1}{T_{ai}}[K_{ai}(V_{tri} - V_{ti} + V_{si}) - E_{fi}], \tag{5.4}$$

avec E'_{qi} : la f.e.m transitoire sur l'axe q, en p.u. E_{qi} : la f.e.m sur l'axe q, en p.u. E_{fi} : la f.e.m d'excitation, en p.u. T_{d0i} : la constante transitoire de court-circuit sur l'axe d, en sec. V_{ti} : la tension terminale du générateur, en p.u. V_{tri} : la tension terminale de référence, en p.u. V_{si} : la commande du circuit d'excitation, en p.u. K_{ai} : le gain du régulateur automatique de tension (AVR). T_{ai} : la constante de temps du régulateur de tension, en sec.

Commande du vannage de la turbine

$$\dot{P}_{mi} = \frac{1}{T_{mi}}(-P_{mi} + P_{GVi}), \tag{5.5}$$

$$\dot{P}_{GVi} = \frac{1}{T_{si}}(-\frac{1}{R_i\omega_0}\omega_i - P_{GVi} + P_{ci}), \tag{5.6}$$

où P_{GVi} est le vannage de vapeur du générateur, en p.u. T_{mi} : la constante de temps de la turbine, en sec. P_{ci} : la commande de la puissance mécanique de la turbine, en p.u. T_{si} : la constante de temps du gouverneur de vitesse, en sec. R_i : la constante de régulation du générateur.

Equations électriques

$$E_{qi} = E'_{qi} + (x_{di} - x'_{di})I_{di}, \tag{5.7}$$

$$I_{qi} = \sum_{j=1}^{n} E'_{qj}(B_{ij}\sin\delta_{ij} + G_{ij}\cos\delta_{ij}), \tag{5.8}$$

$$I_{di} = \sum_{j=1}^{n} E'_{qj}(G_{ij}\sin\delta_{ij} - B_{ij}\cos\delta_{ij}), \tag{5.9}$$

$$P_{ei} = E'_{qi}\, I_{qi}, \tag{5.10}$$

$$Q_{ei} = E'_{qi} I_{di}, \tag{5.11}$$

$$V_{qi} = E'_{qi} - x'_{di} I_{di}, \tag{5.12}$$

$$V_{di} = x'_{di} I_{qi}, \tag{5.13}$$

$$V_{ti} = \sqrt{V_{qi}^2 + V_{di}^2}, \tag{5.14}$$

avec x_{di} : la réactance sur l'axe d, en p.u. x'_{di} : la réactance transitoire sur l'axe d, en p.u. I_{qi} : la composante du courant sur l'axe q, en p.u. I_{di} : la composante du courant sur l'axe d, en p.u. $Y_{ij} = G_{ij} + jB_{ij}$: l'élément de la $i^{\text{ème}}$ ligne et la $j^{\text{ème}}$ colonne de la matrice d'admittance réduite du réseau, en p.u. Q_{ei} : la puissance électrique réactive, en p.u. V_{qi} : la composante de la tension terminale sur l'axe q, en p.u. V_{di} : la composante de la tension terminale sur l'axe d, en p.u. $\delta_{ij} = \delta_i - \delta_j$.

Le modèle dynamique donné ici comporte en tout six variables d'état, c'est-à-dire δ_i, ω_i, E'_{qi}, E_{fi}, P_{mi} et P_{GVi}. Les deux entrées de commande sont V_{si} et P_{ci}. La première entrée, V_{si}, agit sur le circuit d'excitation du générateur qui intègre également le régulateur automatique de tension (Figure 5.1). Elle influe principalement sur la tension terminale du générateur, mais aussi sur la vitesse rotorique ω_i et l'angle électrique δ_i. L'entrée de commande P_{ci}, de sa part, permet le contrôle de la puissance mécanique de la turbine à travers le vannage P_{GVi}, allant jusqu'à modifier la vitesse ω_i, et par suite l'angle δ_i. La commande P_{ci} agit sur le gouverneur de vitesse du générateur (Figure 5.2).

La stabilité d'un réseau électrique est définie comme étant l'aptitude de celui-ci à fonctionner au voisinage du synchronisme (c'est-à-dire, $\omega_i = 0, i = 1,\ldots,n$), lorsqu'il est sollicité par une ou plusieurs perturbations qui peuvent survenir sur ce réseau. La perturbation crée un déséquilibre entre la production et la consommation d'énergie électrique dans le réseau. Ce déséquilibre induit la variation de l'énergie cinétique provoquant ainsi l'évolution des angles électriques des générateurs et accompagnée par des oscillations dynamiques [23].

L'objectif du réglage sera donc de maintenir, principalement, le synchronisme des générateurs du réseau. D'autre part, on doit assurer le réglage des angles électriques δ_i, $i = 1,\ldots,n$. Par ailleurs, les tensions terminales des générateurs doivent être maintenues à leurs valeurs d'équilibre (ou de consigne), ce qui va être assuré grâce aux régulateurs automatiques de tension (voir l'Equation (5.4)). La sortie du AVR dépend uniquement de la tension terminale et agit sur le circuit d'excitation du générateur. Certes, ce type de régulateurs de tension est classique et ses performances peuvent être facilement améliorées en utilisant d'autres techniques de réglage plus récentes. Néanmoins, pour des raisons de simplicité, nous allons adopter ce type de régulateurs.

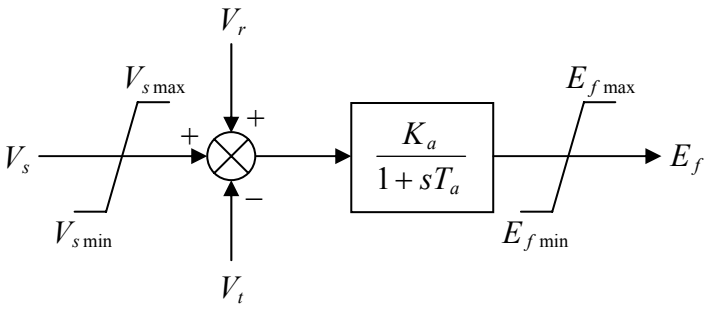


Figure 5.1: Schéma d'excitation du générateur.

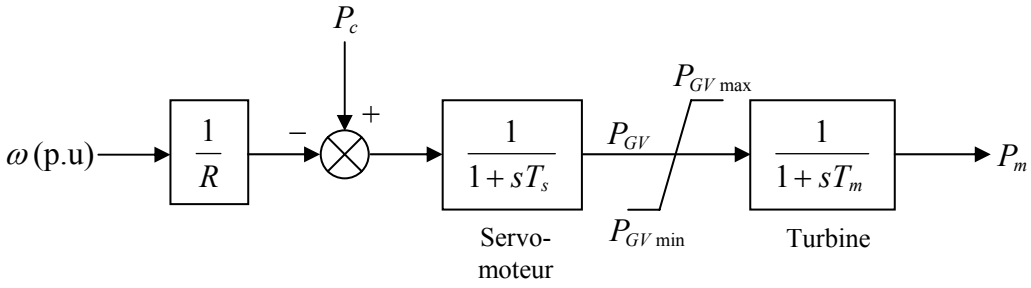


Figure 5.2: Schéma du gouverneur de vitesse.

5.2.2 Conception de la commande

Dans ce qui suit, nous présentons notre approche de réglage par logique floue et backstepping. Nous commençons tout d'abord par donner la nouvelle forme du modèle dynamique du réseau électrique, suite à l'application du bouclage linéarisant direct (BLD). Ce bouclage va permettre de rendre le modèle du réseau partiellement linéaire pour un large domaine d'opération, augmentant ainsi la robustesse des régulateurs synthétisés et en renforçant la décentralisation de la commande.

Néanmoins, en appliquant le bouclage linéarisant direct, on ne peut éviter tout de même la restitution de certains résidus non linéaires. Il nous est donc nécessaire d'adopter une technique de réglage adaptative, comme la logique floue, afin de garantir un comportement efficace du système de commande. En appliquant le bouclage linéarisant direct, on obtient le modèle modifié suivant :

$$
\begin{aligned}
&\dot{\delta}_i = \omega_i ,\\
&\dot{\omega}_i = -\frac{D_i}{2H_i}\omega_i - \frac{\omega_0}{2H_i}\Delta P_i ,\\
&\Delta\dot{P}_i = -\frac{1}{T'_{d0}}\Delta P_i + \frac{1}{T'_{d0}} v_{fi} + \gamma_i(\delta, \omega, E_f, P_m, P_{GV}) ,\\
&\dot{E}_{fi} = \frac{1}{T_{ai}}[K_{ai}(V_{tri} - V_{ti} + V_{si}) - E_{fi}] ,
\end{aligned}
\qquad (5.15)
$$

$$\dot{P}_{mi} = \frac{1}{T_{mi}}(-P_{mi} + P_{GVi}),$$

$$\dot{P}_{GVi} = \frac{1}{T_{si}}(-\frac{1}{R_i \omega_0}\omega_i - P_{GVi} + P_{ci}),$$

où

$$\Delta P_i = P_{ei} - P_{mi}, \tag{5.16}$$

$$\gamma_i(\delta, \omega, E_f, P_m, P_{GV}) = E'_{qi} \sum_{j=1}^{n} \dot{E}'_{qj} B_{ij} \sin(\delta_i - \delta j) - E'_{qi} \sum_{j=1}^{n} E'_{qj} B_{ij} \cos(\delta_i - \delta_j)\omega_j - \dot{P}_{mi}, \tag{5.17}$$

$$v_{fi} = I_{qi} E_{fi} - (x_{di} - x'_{di}) I_{qi} I_{di} - P_{mi} - T'_{d0i} Q_{ei} \omega_i. \tag{5.18}$$

v_{fi} étant la nouvelle entrée de commande, relie l'ancienne commande V_{si}, à travers la f.e.m d'excitation E_{fi}, à la nouvelle variable d'état ΔP_i. Par ailleurs, γ_i est un terme d'interconnexion qui est une quantité non linéaire complexe. Néanmoins, pour des raisons de décentralisation, cette quantité est considérée comme étant inconnue pour la $i^{\text{ème}}$ unité de génération, étant donné qu'elle ne peut être reconstruite à partir des variables locales de cette unité.

Commençons maintenant à concevoir la commande. On remarque tout d'abord que pour commander l'angle électrique δ_i, on doit agir sur les deux variables d'entrée v_{fi} et P_{ci}. Cependant, vu qu'il existe plusieurs dynamiques entre ces deux entrées et la variable à commander δ_i (en particulier pour l'entrée P_{ci}), la commande floue seule risque de ne pas agir d'une façon efficace et robuste vis-à-vis des différentes perturbations qui peuvent nuire à la stabilité du réseau électrique. Pour cela, nous allons faire appel à la technique du backstepping, dans le but de ramener la variable δ_i à un point "conceptuellement" plus proche des deux variables de commande v_{fi} et P_{ci}. Ainsi, on peut améliorer les performances du régulateur flou, en ayant moins de difficultés à le synthétiser.

La technique du backstepping est une procédure récursive qui combine le choix de la fonction de Lyapunov avec la synthèse de la loi de commande. Elle consiste à transformer le problème général de la synthèse de commande en une série de problèmes de synthèse pour des systèmes réduits, assurant ainsi la stabilité du système global d'une façon récursive. A chaque étape de la procédure, on augmente l'ordre du système réduit considéré et on génère une commande virtuelle afin d'assurer la stabilité de ce système. Cette procédure est répétée jusqu'à ce que la commande réelle apparaisse à la dernière étape [19].

Pour appliquer cette technique à notre problème de commande, on doit commencer par définir la première variable d'erreur pour la $i^{\text{ème}}$ unité de génération :

$$z_{i1} = \Delta\delta_i = \delta_i - \delta_{i0}\,, \tag{5.19}$$

et en considérant ω_i comme étant une commande virtuelle pour z_{i1}, on peut assurer la stabilité asymptotique de z_{i1} en ayant :

$$\omega_i = -c_{i1}\Delta\delta_i\,, \tag{5.20}$$

où $c_{i1} > 0$ est une constante à choisir. En effet, si on considère la fonction de Lyapunov suivante :

$$V_1 = \frac{1}{2}\sum_{i=1}^{n} z_{i1}^2\,. \tag{5.21}$$

on aura

$$\dot{V}_1 = \sum_{i=1}^{n} z_{i1}\dot{z}_{i1} = -\sum_{i=1}^{n} c_{i1} z_{i1}^2 < 0\,. \tag{5.22}$$

Par suite, on définit la nouvelle variable d'erreur :

$$z_{i2} = \omega_i + c_{i1}\Delta\delta_i\,, \tag{5.23}$$

et en voyant ΔP_i comme étant une nouvelle commande virtuelle, on peut assurer la stabilité asymptotique de z_{i2} en choisissant :

$$\Delta P_i = \frac{2H_i}{\omega_0}\left[\left(-\frac{D}{2H_i} + c_{i1} + c_{i2}\right)\omega_i + c_{i1}c_{i2}\Delta\delta_i\right], \tag{5.24}$$

avec $c_{i2} > 0$ une autre constante à choisir. Cette une expression qui dépend donc de $\Delta\delta_i$ et également de ω_i. On peut s'assurer du bon choix de cette expression en considérant la fonction de Lyapunov augmentée suivante :

$$V_2 = V_1 + \frac{1}{2}\sum_{i=1}^{n} z_{i2}^2\,. \tag{5.25}$$

sa dérivée est donnée par :

$$\begin{aligned}\dot{V}_2 &= -\sum_{i=1}^{n} c_{i1} z_{i1}^2 + \sum_{i=1}^{n} z_{i2}\left[-\frac{D_i}{2H_i}\omega_i - \frac{\omega_0}{2H_i}\Delta P_i + c_{i1}\omega_i\right].\\ &= -\sum_{i=1}^{n} c_{i1} z_{i1}^2 - \sum_{i=1}^{n} c_{i2} z_{i2}^2 < 0.\end{aligned} \tag{5.26}$$

Cela veut dire que ΔP_i doit, en tant que commande virtuelle, assurer la stabilité asymptotique de la variable z_{i2}, qui à son tour, doit entraîner la convergence de z_{i1} vers zéro. Définissons maintenant la dernière variable d'erreur :

$$z_i = \Delta P_i - \frac{2H_i}{\omega_0}\left[\left(-\frac{D}{2H_i} + c_{i1} + c_{i2}\right)\omega_i + c_{i1}c_{i2}\Delta\delta_i\right], \tag{5.27}$$

et c'est la variable pour laquelle on doit finalement agir sur les deux entrées de commande v_{fi} et P_{ci} afin de la faire converger et maintenir à zéro. Ceci peut être fait en utilisant un régulateur flou du type Mamdani, qui aura comme entrées la variable z_i et sa variation instantanée dz_i, et comme sorties l'entrée de commande v_{fi} ainsi que la variation de référence de P_{GVi}, soit dP_{GVi}. En négligeant la dynamique du régulateur automatique de tension, la commande réelle u_{fi} peut alors être confondue avec E_{fi} dans l'Equation (5.18), et par suite exprimée de la façon suivante :

$$V_{si} = \frac{1}{I_{qi}K_{ai}}\left[v_{fi} + (x_d - x'_d)I_{qi}I_{di} + P_{mi} + T'_{d0i}Q_{ei}\omega_i\right], \tag{5.28}$$

où K_{ai} est ajouté pour compenser l'effet du gain du stabilisateur de tension. Par ailleurs, la commande P_{ci} est donnée par :

$$P_{ci} = dP_{GVi} + P_{GVi}\,. \tag{5.29}$$

Donc, le régulateur flou va pouvoir se limiter à réguler uniquement la variable z_i sans tenir compte des autres variables (δ_i, ω_i et ΔP_i). Ses deux sorties vont réagir principalement sur la différence ΔP_i.

En examinant les Equations (5.5), (5.6) et (5.15) on peut voir l'influence de v_{fi} et P_{ci} sur le comportement général de l'état ΔP_i, et par suite construire une table de règles floues qui exprime notre expertise de réglage. Dans cette table, donnée dans le Tableau 5.1, les deux variables d'entrée z_i et dz_i sont reliées aux variables de sorties v_{fi} et dP_{GVi} à travers 25 règles floues du type IF-THEN, en considérant 5 sous-ensembles flous pour chacune des deux entrées et 7 pour les sorties. Les fonctions d'appartenance associées aux sous-ensembles flous sont illustrées à la Figure 5.4.

La Figure 5.3 résume la structure de la commande proposée au niveau d'une seule unité de génération. Cette commande exploite l'avantage de flexibilité des deux approches du backstepping et de la commande par logique floue. Par ailleurs, elle dispose d'un seul régulateur flou qui permet donc de coordonner les actions des deux variables de commande. En plus, c'est une technique qui est simple à implémenter et ne se sert que des mesures locales de chaque unité de génération.

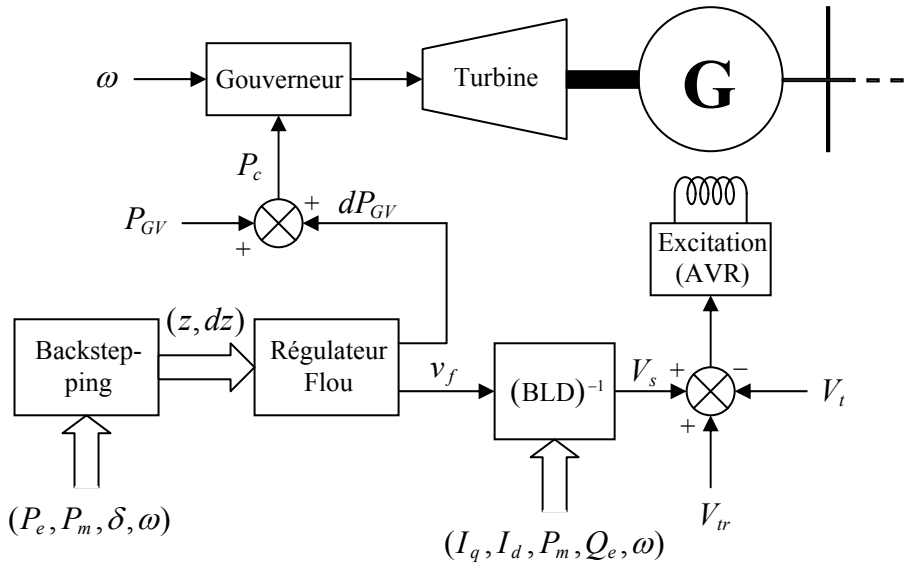


Figure 5.3 : Schéma de commande dans une unité de génération.

Dans le but d'illustrer et d'optimiser les performances de cette commande, nous présentons dans la prochaine section une approche évolutionnaire pour la synthèse multiobjectif des paramètres de cette commande, pour le cas particulier d'un réseau électrique à trois machines.

5.3 Optimisation multiobjectif des paramètres de réglage

La commande précédente fait intervenir trois types de paramètres ; ceux du backstepping, du régulateur flou et enfin les gains des stabilisateurs de tension. Las paramètres du backstepping contrôlent la rapidité de convergence des variables à commander, leur influence sur les performances du système de commande est généralement systématique.

Par contre, la logique floue a été originellement formulée comme une technique de simulation du raisonnement humain, qui s'appuie sur des connaissances ou des données inexactes ou imprécises. L'un des inconvénients majeurs de cette technique réside dans le choix optimal des paramètres des fonctions d'appartenance des sous-ensembles flous ainsi que celui de la table des règles floues. Habituellement, ce choix se fait par essai/erreur, en se basant sur des connaissances fournies par un expert humain et de certaines caractéristiques du système commandé. Cela va induire dans la plupart des cas un fonctionnement sous optimal du régulateur flou.

Plusieurs approches de conception, permettant de spécifier les différents paramètres dont dépend un régulateur flou, ont été décrites dans la littérature. Elles sont pour la plupart basées sur un apprentissage qui permet de définir de façon itérative le meilleur jeu de paramètres pour une structure donnée du régulateur. Jusqu'à présent, les chercheurs se sont

Tableau 5.1 : Base des règles du régulateur flou pour la sortie v_{fi}. La sortie dP_{GVi} est déduite par symétrie des sous-ensembles de v_{fi}.

dz_i \ z_i	NB	NS	Z	PS	PB
NB	PB	PB	PM	PS	Z
NS	PB	PM	PS	Z	NS
Z	PM	PS	Z	NS	NM
PS	PS	Z	NS	NM	NB
PB	Z	NS	NM	NB	NB

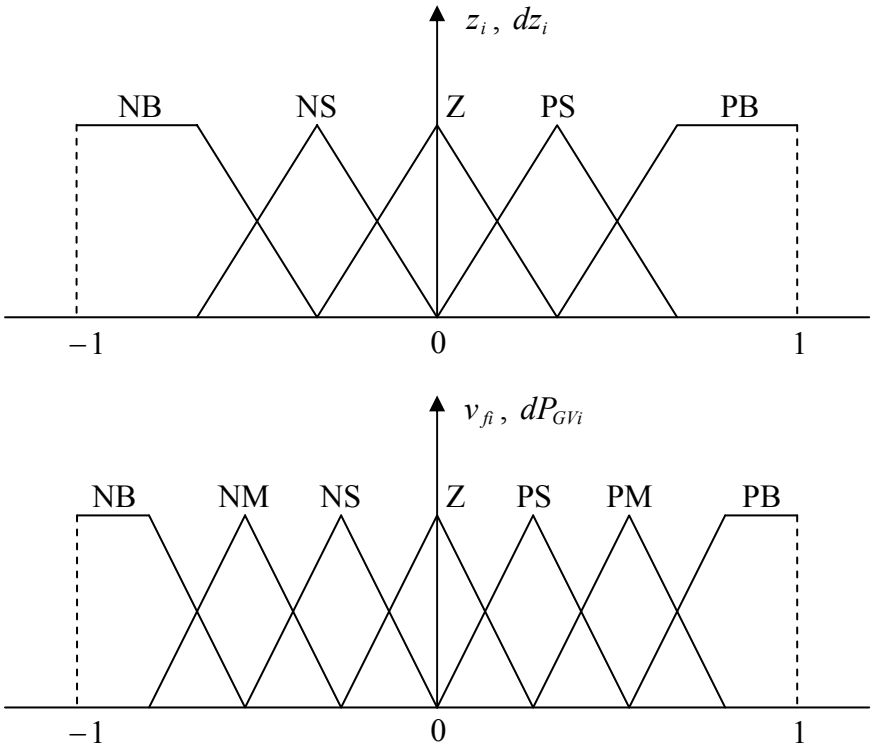


Figure 5.4 : Formes des fonctions d'appartenance du régulateur flou.

concentrés principalement sur l'optimisation des paramètres des fonctions d'appartenance, des règles floues ou encore les deux à la fois [32].

Par ailleurs, les gains des régulateurs de tension (AVR) permettent d'amplifier et de réformer les signaux de commande établis à partir de l'erreur qui existe entre la valeur instantanée de la tension terminale et sa valeur de référence, ajoutée à cela le signal généré à partir de la deuxième sortie du régulateur flou. Le rôle du signal d'excitation du générateur est donc double. D'une part, il permet le régale de la tension terminale, d'autre part il contribue à la régulation de l'angle et la vitesse rotoriques du générateur. Il est clair qu'avec ce type de réglage, un compromis existera toujours entre les deux parties de l'action de commande.

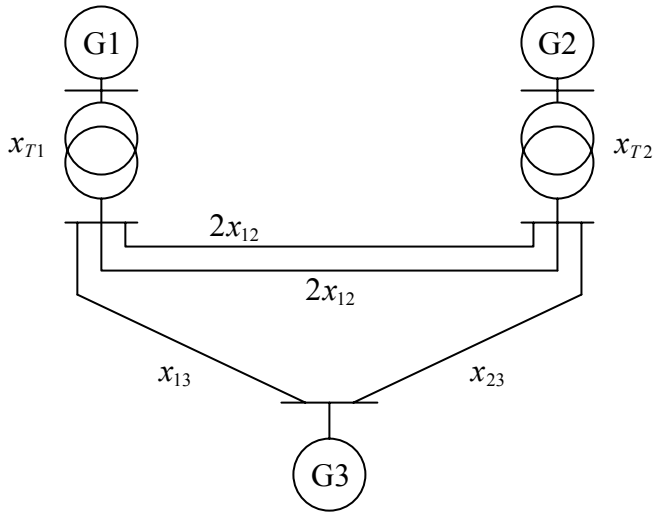


Figure 5.5 : Réseau électrique à trois machines.

5.3.1 Implémentation de l'algorithme évolutionnaire

Dans le but de simplifier les simulations, on considère le réseau électrique montré sur la Figure 5.5 [34]. Ce réseau est composé de trois machines génératrices dont la troisième est considérée comme étant un nœud infini. Les paramètres de ce réseau sont donnés dans le Tableau 5.2.

Le problème posé est de trouver des valeurs optimales pour les paramètres de la commande précédente, dans le but d'obtenir un bon compromis entre les différents critères de performance. Ces derniers peuvent être résumés en deux critères principaux, traduisant respectivement les objectifs de réglage de la variable z_i et de la tension terminale V_{ti} :

$$\begin{cases} f_1(\mathrm{x}) = \sum_{i=1}^{2} \int_0^{t_f} |z_i| \, dt, \\ f_2(\mathrm{x}) = \sum_{i=1}^{2} \int_0^{t_f} |V_{ti} - V_{ti0}| \, dt, \end{cases} \tag{5.30}$$

où x est le vecteur de décision qui contient les solutions recherchées par l'algorithme évolutionnaire, c'est-à-dire les paramètres de la commande. Néanmoins, en conduisant des tests sur l'influence des paramètres du backstepping sur le comportement général du système commandé, nous avons remarqué que ces derniers dépendent fortement du type et de la localisation de la perturbation considérée sur le réseau électrique. De ce fait, ces paramètres seront fixés a priori et ne figureront donc pas dans le codage de l'individu.

Il est à noter que le problème original était de considérer chaque objectif de réglage pour chaque machine d'une façon indépendante de l'autre machine. Toutefois, étant donné la

Tableau 5.2 : Paramètres du réseau à trois machines.

Paramètre	G1	G2
x_d (p.u)	1.863	2.36
x'_d (p.u)	0.657	0.719
x_T (p.u)	0.129	0.127
T'_{d0} (sec)	6.9	7.96
H (sec)	4	5.1
D (p.u)	5	3
T_a (sec)	0.02	0.02
T_m (sec)	0.35	0.35
T_s (sec)	0.1	0.1
R	0.05	0.05
x_{12} (p.u)	0.7	
x_{13} (p.u)	0.93	
x_{23} (p.u)	0.9	
ω_0 (rad/sec)	314.159	
$[E_{f\min}, E_{f\max}]$	[-6,6]	
$[V_{s\min}, V_{s\max}]$	[-0.2, 0.2]	
$[P_{GV\min}, P_{GV\max}]$	[0.1, 1.1]	

ressemblance qui existe entre les deux machines du réseau considéré, nous nous permettrons dans ce cas d'adopter les critères agrégés précédents, mais à condition de considérer un cas de simulation dans lequel les deux machines ont un comportement relativement similaire.

L'individu de la population doit donc contenir uniquement les paramètres des fonctions d'appartenance des régulateurs flous ainsi que les gains des régulateurs de tension. La Figure 5.6 donne un schéma de codage des fonctions d'appartenance des variables d'entrée et de sortie de chaque régulateur flou, en considérant deux paramètres pour chacune des variables d'entrée et de sortie.

Les paramètres des entrées sont x_1 et x_2 pour la variable z et x_3 et x_4 pour dz. Les gains de normalisation de z et dz sont fixés à 2 et 1, respectivement. Par contre, ceux des sorties sont considérés comme des paramètres à rechercher, soient x_5 pour v_f et x_7 pour dP_{GV}. Par ailleurs, x_6 et x_8 permettent de coder les fonctions d'appartenance de v_f et dP_{GV}, respectivement. Dans ce cas, nous considérons pour des raisons de simplicité que chacune des fonctions d'appartenance de NM à PM chevauche la fonction adjacente de 50 %. Enfin, x_9 est le paramètre qui représente le gain du régulateur de tension, K_a.

Ainsi, l'individu de la population contiendra 9 variables de décision pour chaque machine du réseau, soit un total de 18 variables. Les domaines de variation des paramètres x_i, $i = 1,\ldots,9$, sont donnés dans le Tableau 5.3. Le choix de ces domaines est fait de façon à assurer de bons signaux de commande.

Tableau 5.3: Domaines de variation des variables de décision.

Paramètre	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Min	0.05	0.05	0.005	0.005	1	0.01	1	0.01	200
Max	0.2	0.2	0.05	0.05	50	0.1	50	0.1	400

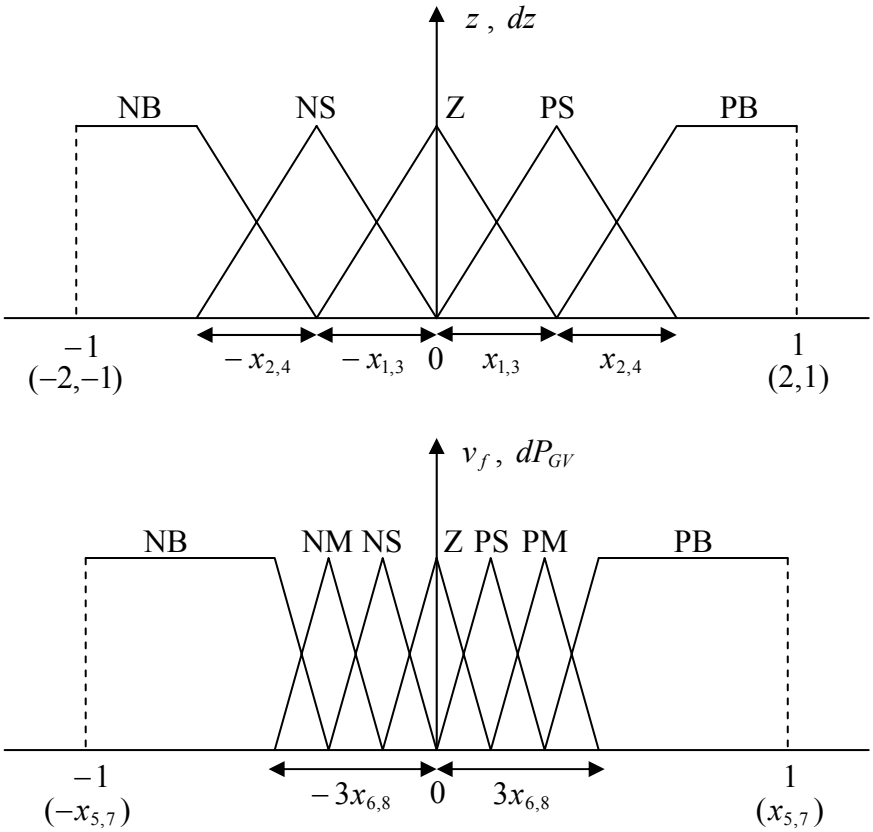


Figure 5.6 : Codage des fonctions d'appartenance du régulateur flou.

5.3.2 Résultats de simulation

L'algorithme ε-GMOEA est utilisé pour résoudre le problème d'optimisation précédent, en considérant une population de 100 individus et un nombre total de 150 générations. Les opérateurs de croisement binaire simulé et de mutation polynomiale sont utilisés en ayant comme indices de distribution, 15 et 20 respectivement. La probabilité de croisement est égale à 1 et le taux de mutation à 1/18. Par ailleurs, les paramètres λ_i, λ_f et α sont fixés respectivement à 10, 100 et 5. Enfin, on choisit $\varepsilon_i = 2.10^{-4}$, $i = 1,2$ dans le but d'obtenir approximativement 100 solutions ε-non dominées.

Pour le calcul des fonctions objectifs de l'Equation (5.30), on considère un cas de perturbation sur les puissances mécaniques fournies aux deux générateurs du réseau. Cette perturbation est représentée par un accroissement brutal et permanent (en échelon) de la puissance mécanique d'une valeur de 0.1 p.u, appliqué à partir de l'instant $t = 0.1$ sec. Le temps final de simulation est $t_f = 1$ sec. Le point de fonctionnement considéré est le suivant :

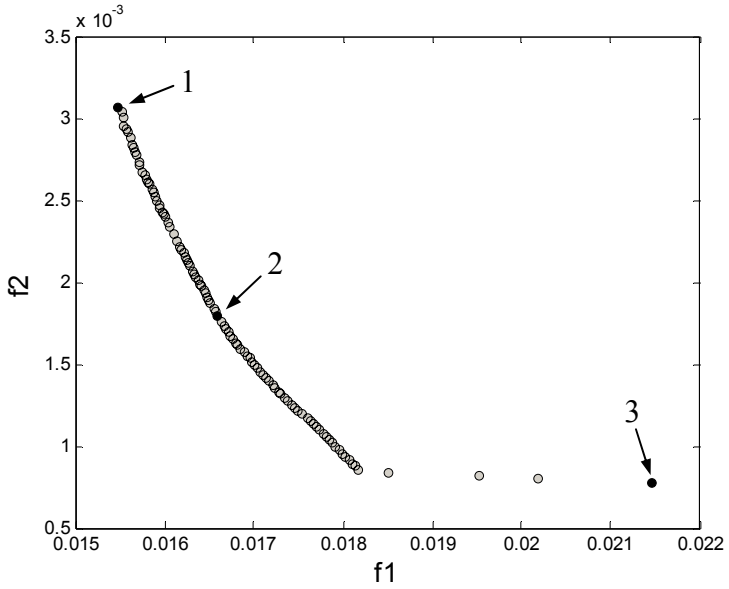


Figure 5.7 : Front expérimental obtenu.

$$\begin{aligned} &\delta_{10} = 54.12°, \quad P_{m10} = 0.56(\text{p.u}), \quad V_{t10} = 1(\text{p.u}), \\ &\delta_{20} = 55.16°, \quad P_{m20} = 0.56(\text{p.u}), \quad V_{t20} = 1(\text{p.u}). \end{aligned} \tag{5.31}$$

Par ailleurs, les paramètres du backstepping sont posés comme suit :

$$c_{ij} = 5, \quad i, j = 1, 2. \tag{5.32}$$

L'exécution de l'algorithme évolutionnaire a aboutit à un total de 102 solutions ε-non dominées. La Figure 5.7 montre le front non dominé trouvé. Celui-ci est globalement convexe et reflète clairement le compromis qui existe entre l'objectif de réglage de la variable z et celui de la tension terminale V_t. On remarque aussi qu'il existe un point de cassure sur ce front. Par ailleurs, on constate que la distribution des solutions ε-non dominées n'est pas globalement homogène, et ce à cause de la méthode d'archivage utilisée par ε-GMOEA. D'autre part, on peut dire que le premier objectif représente lui-même un certain compromis entre les performances de réglage des deux variables δ et ω.

Sur la Figure 5.8, sont illustrées les réponses de z, δ et V_t pour le cas de perturbation considéré, en utilisant les paramètres de trois solutions de l'archive finale de ε-GMOEA (Figure 5.7). Ces solutions représentent les deux points extrémaux du front expérimental trouvé en plus du point médian. Les paramètres de ces points sont donnés dans le Tableau 5.4. Nous pouvons ainsi confirmer tout d'abord la robustesse du régulateur synthétisé. En effet, l'angle électrique et la tension terminale des deux générateurs du réseau reviennent à leurs valeurs d'équilibre en 2 secondes. D'autre part, nous pouvons voir l'effet du compromis sur le comportement de z, et V_t.

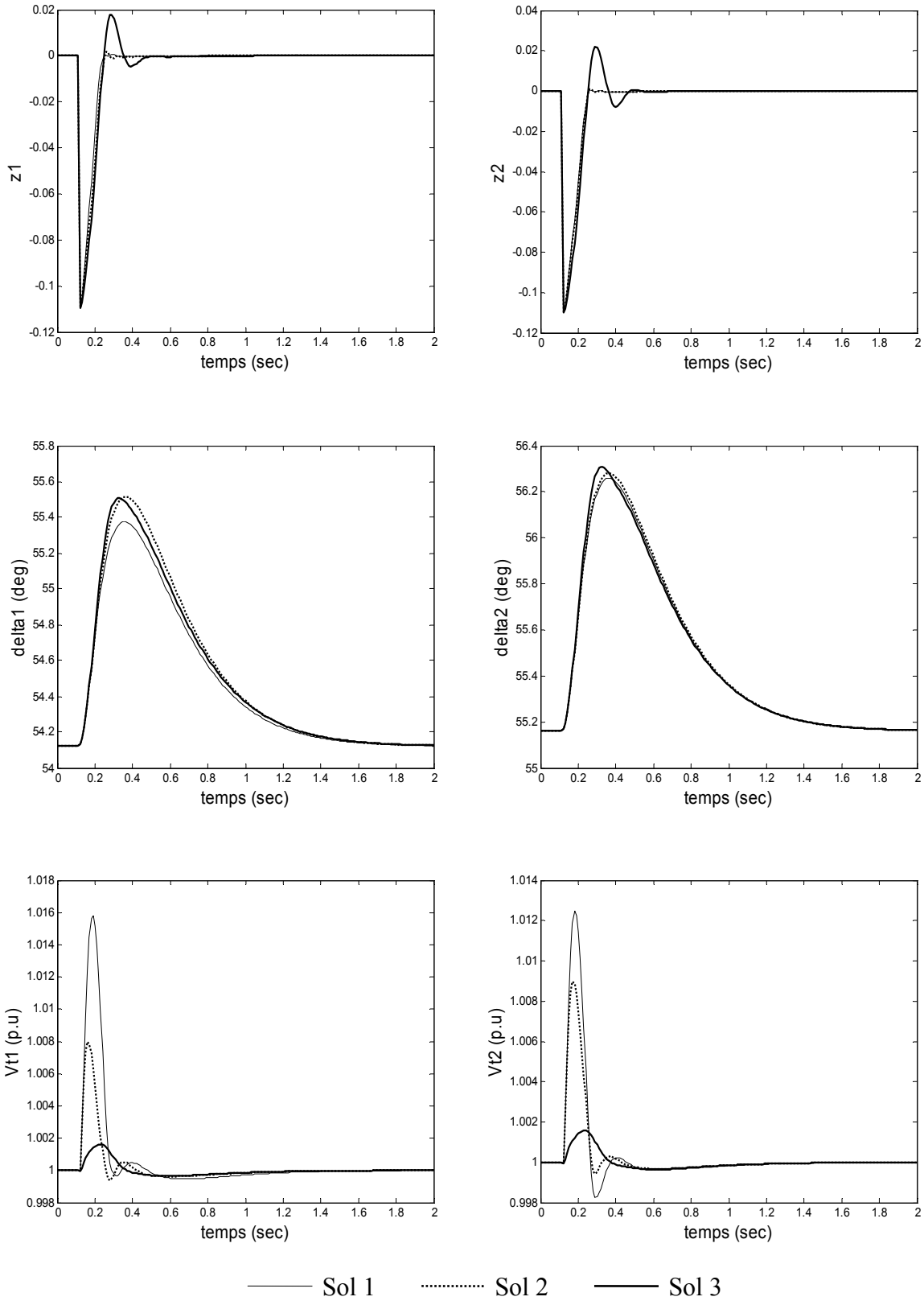


Figure 5.8 : Réponses de z , δ et V_t . Perturbation sur les puissances mécaniques (+0.1 p.u).

Tableau 5.4 : Paramètres des trois solutions de l'archive correspondant respectivement à la valeur minimale, médiane et maximale de la fonction objectif f_1.

Paramètre		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Sol 1	G1	0.050	0.050	0.026	0.044	49.93	0.095	49.56	0.094	203.6
	G2	0.050	0.123	0.026	0.005	34.75	0.098	49.95	0.098	342.7
Sol 2	G1	0.050	0.054	0.024	0.047	46.27	0.016	49.74	0.094	400.0
	G2	0.050	0.126	0.026	0.006	25.97	0.093	49.93	0.098	400.0
Sol 3	G1	0.072	0.050	0.026	0.036	1.031	0.097	21.12	0.096	399.9
	G2	0.060	0.110	0.025	0.007	1.053	0.097	44.63	0.041	399.9

Dans le but de valider ces résultats et de tester encore la robustesse du régulateur synthétisé, nous présentons dans ce qui suit une série de tests qui mettent en évidence plusieurs aspects du réglage. Pour cela, nous allons considérer deux autres points de fonctionnement qui reflètent un déséquilibre de charge entre les deux générateurs du réseau.

Le premier point de fonctionnement est donné par :

$$\begin{aligned} &\delta_{10} = 47.37°, \quad P_{m10} = 0.4\,(\text{p.u}), \quad V_{t10} = 1(\text{p.u}), \\ &\delta_{20} = 67.14°, \quad P_{m20} = 0.8\,(\text{p.u}), \quad V_{t20} = 1.02\,(\text{p.u}). \end{aligned} \tag{5.33}$$

Le deuxième point de fonctionnement est le suivant :

$$\begin{aligned} &\delta_{10} = 72.10°, \quad P_{m10} = 0.85\,(\text{p.u}), \quad V_{t10} = 1.02\,(\text{p.u}), \\ &\delta_{20} = 57.32°, \quad P_{m20} = 0.5\,(\text{p.u}), \quad V_{t20} = 1(\text{p.u}). \end{aligned} \tag{5.34}$$

Ces deux points de fonctionnements sont alors utilisés dans deux cas de test. Le premier cas est un court-circuit triphasé appliqué au début d'une des deux lignes reliant le générateur 1 au générateur 2, à l'instant $t = 0.1$ sec. La durée de ce court-circuit est de 0.15 sec, ce qui implique qu'il y aura l'ouverture de la ligne à l'instant $t = 0.25$ sec. Le deuxième cas est une perturbation de +0.15 p.u, à $t = 0.1$, sur la puissance mécanique fournie au générateur 1.

Les résultats de simulation de ces tests sont illustrés sur les Figures 5.9 à 5.16, en utilisant les paramètres des trois solutions du Tableau 5.3. Nous pouvons ainsi confirmer tout d'abord l'efficacité et la robustesse de la commande proposée. L'angle de charge, la vitesse rotorique et la tension terminale des générateurs reviennent à leurs valeurs initiales, et les signaux de commande se stabilisent à leurs nouvelles valeurs d'équilibre. La stabilité du réseau électrique est donc maintenue tout en assurant un bon réglage des variables à commander. Par ailleurs, on remarque qu'il existe toujours un compromis entre les performances de z et celles de V_t. Par contre, le compromis entre la tension terminale V_t et l'angle électrique δ ainsi que la vitesse ω n'est pas toujours maintenu, ce qui est évident vu la manière par laquelle ont été formulées les fonctions objectifs de ce problème.

5.4 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode de réglage pour le maintien de la stabilité transitoire des réseaux électriques multimachines. Cette méthode est basée sur une association de la technique du backstepping et de la commande par logique floue. Ainsi, on a pu exploiter d'une façon coordonnée et efficace, l'avantage de flexibilité que présente ces deux techniques. En effet, le backstepping permet non seulement d'assurer une stabilité asymptotique des variables à commander, mais aussi d'éliminer certaines difficultés qui concerne la commande floue, telle que la construction de la table des règles. L'idée était de faire rapprocher l'objectif du réglage des variables δ et ω aux variables d'entrée v_f et P_c, par l'intermédiaire d'une nouvelle variable z, établie à partir des mesures locales de l'unité de génération, en utilisant la technique du backstepping.

En second lieu, nous avons présenté une approche évolutionnaire pour l'optimisation multiobjectif des performances de réglage, pour un cas particulier d'un réseau électrique à trois machines. L'algorithme ε-GMOEA est utilisé dans le but de trouver un compromis entre les objectifs de réglage de la variable z, commandée par le régulateur flou, et la tension terminale des générateurs V_t, dont le réglage est assuré par un régulateur de tension classique. Les résultats obtenus ont montré, premièrement, que la commande proposée a permis de maintenir efficacement la stabilité du réseau électrique considéré, et ce pour plusieurs types de perturbation. Par ailleurs, le compromis qui existe entre les performances de z et V_t a été toujours maintenu, ce qui confirme donc le bon usage de l'approche évolutionnaire multiobjectif pour la résolution du problème posé.

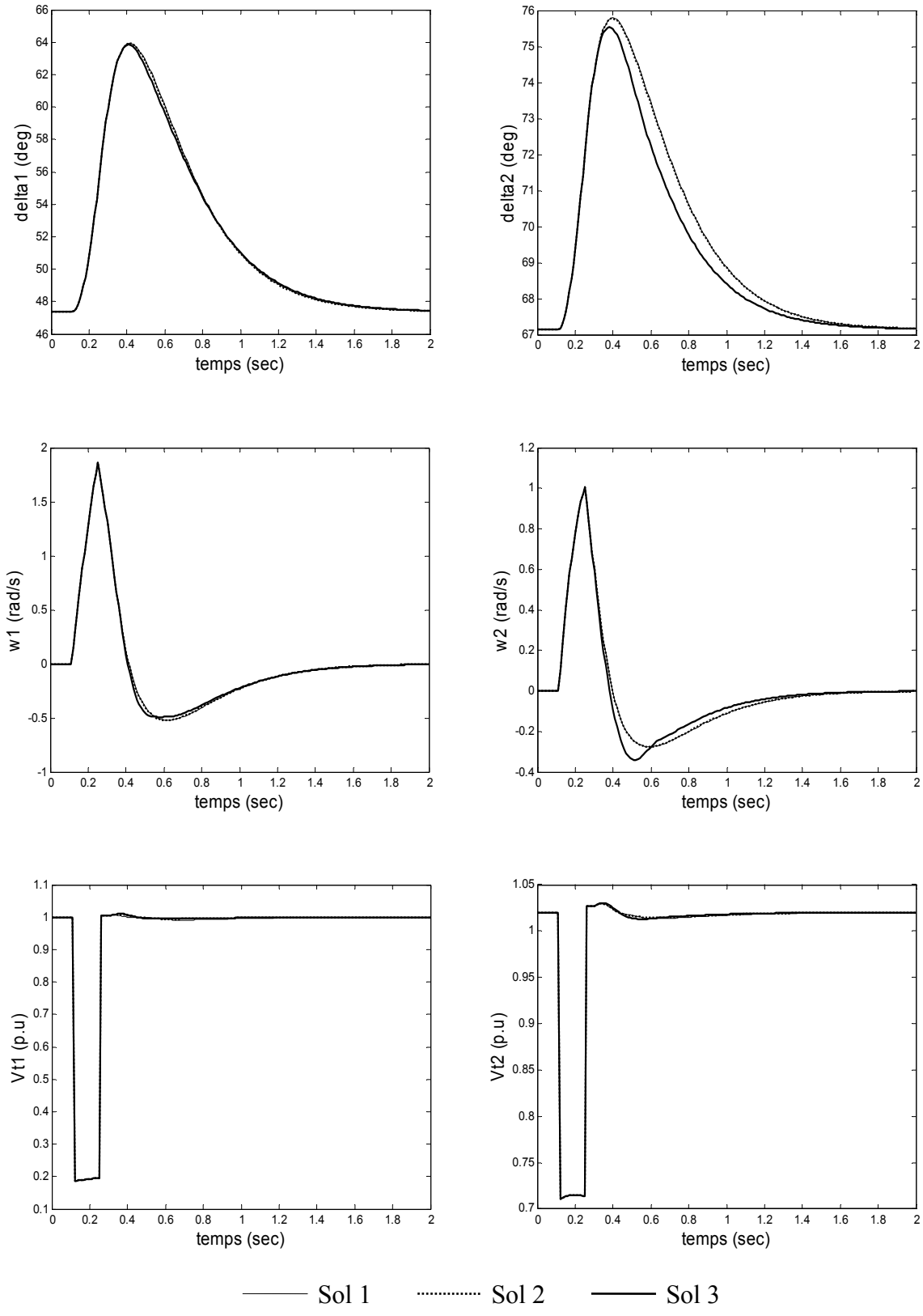


Figure 5.9 : Réponses de δ , ω et V_t . 1er cas, 1er point de fonctionnement.

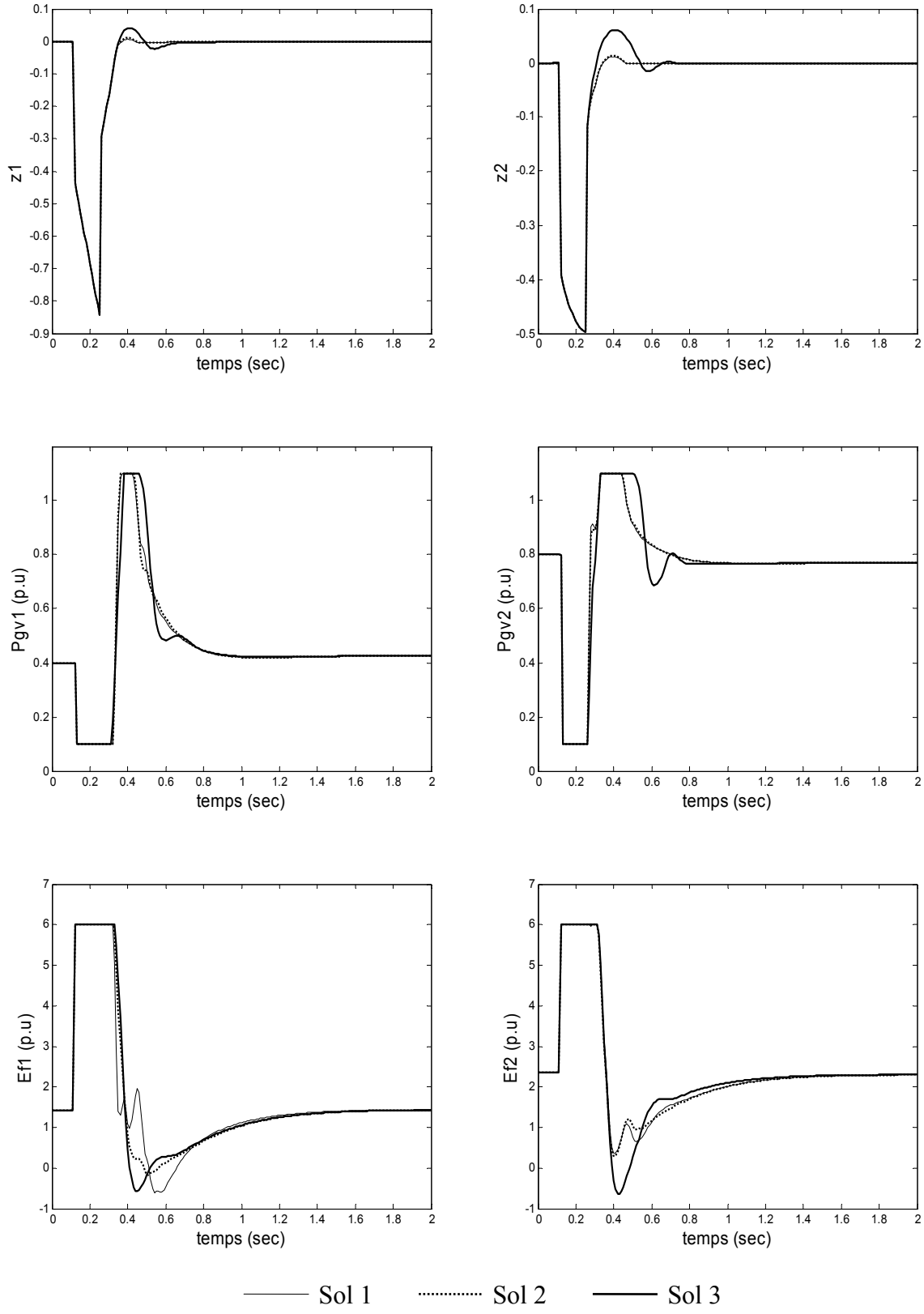


Figure 5.10 : Réponses de z , P_{GV} et E_f . 1er cas, 1er point de fonctionnement.

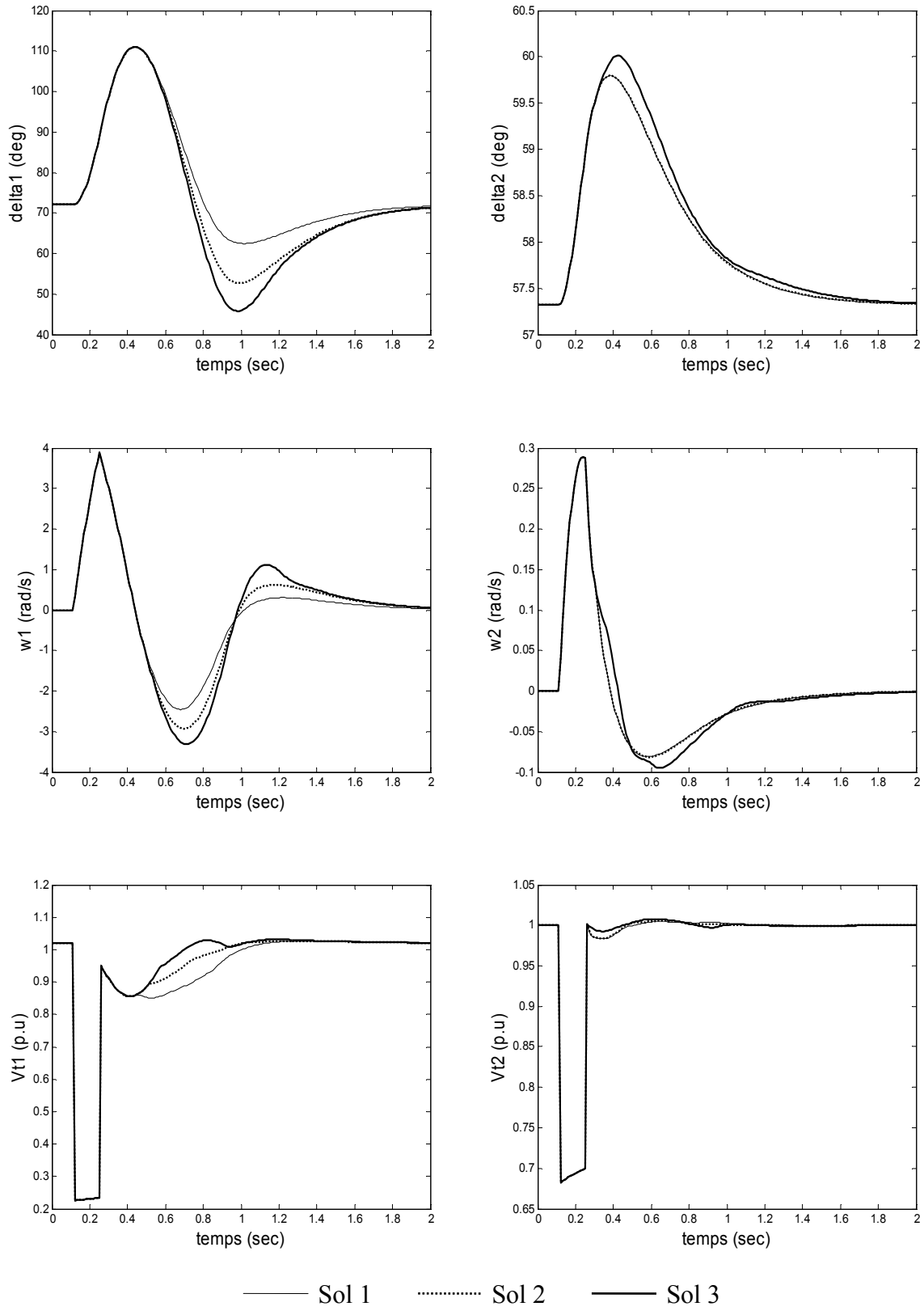


Figure 5.11 : Réponses de δ , ω et V_t . 1er cas, 2ème point de fonctionnement.

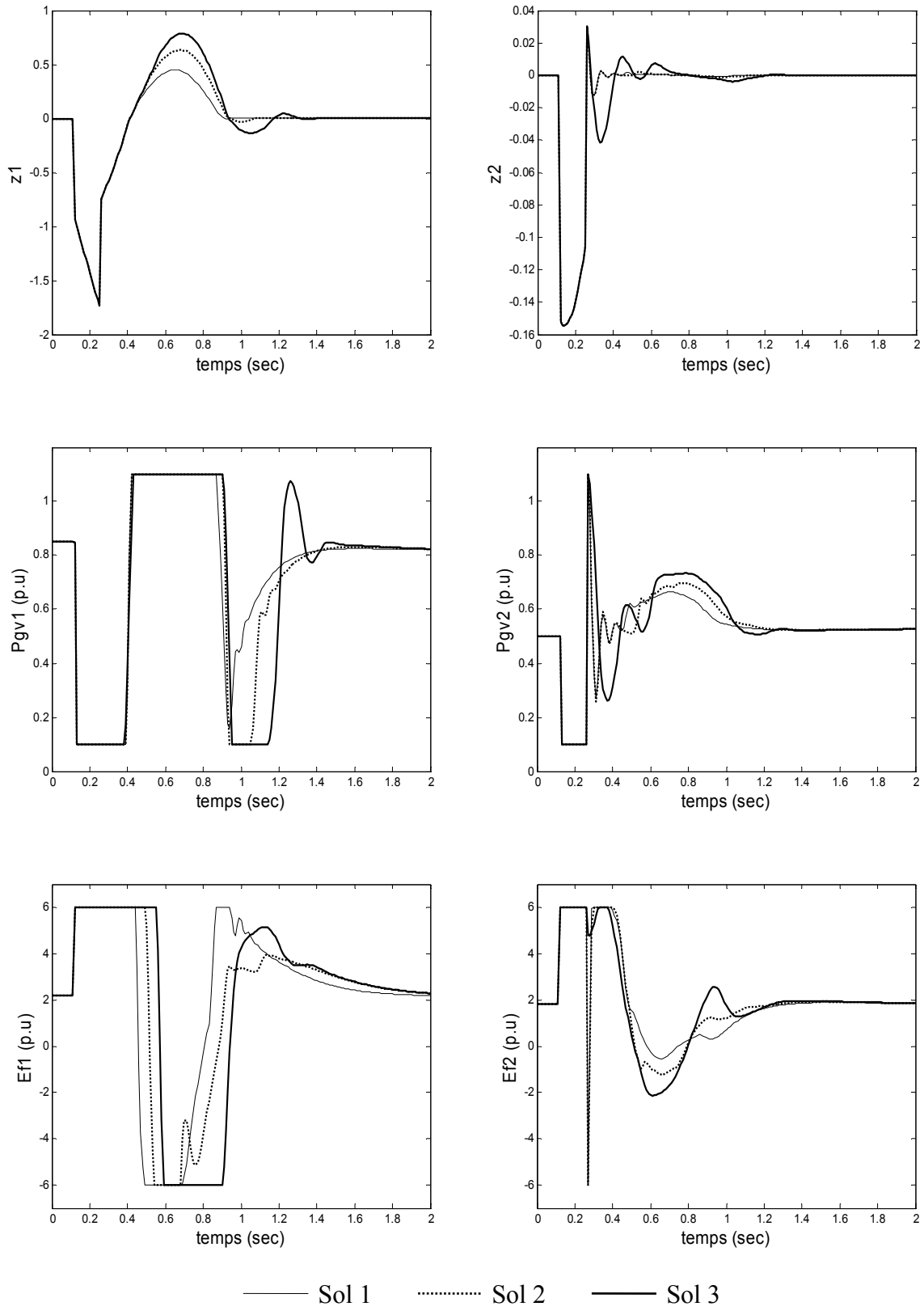


Figure 5.12 : Réponses de z , P_{GV} et E_f . 1er cas, 2ème point de fonctionnement.

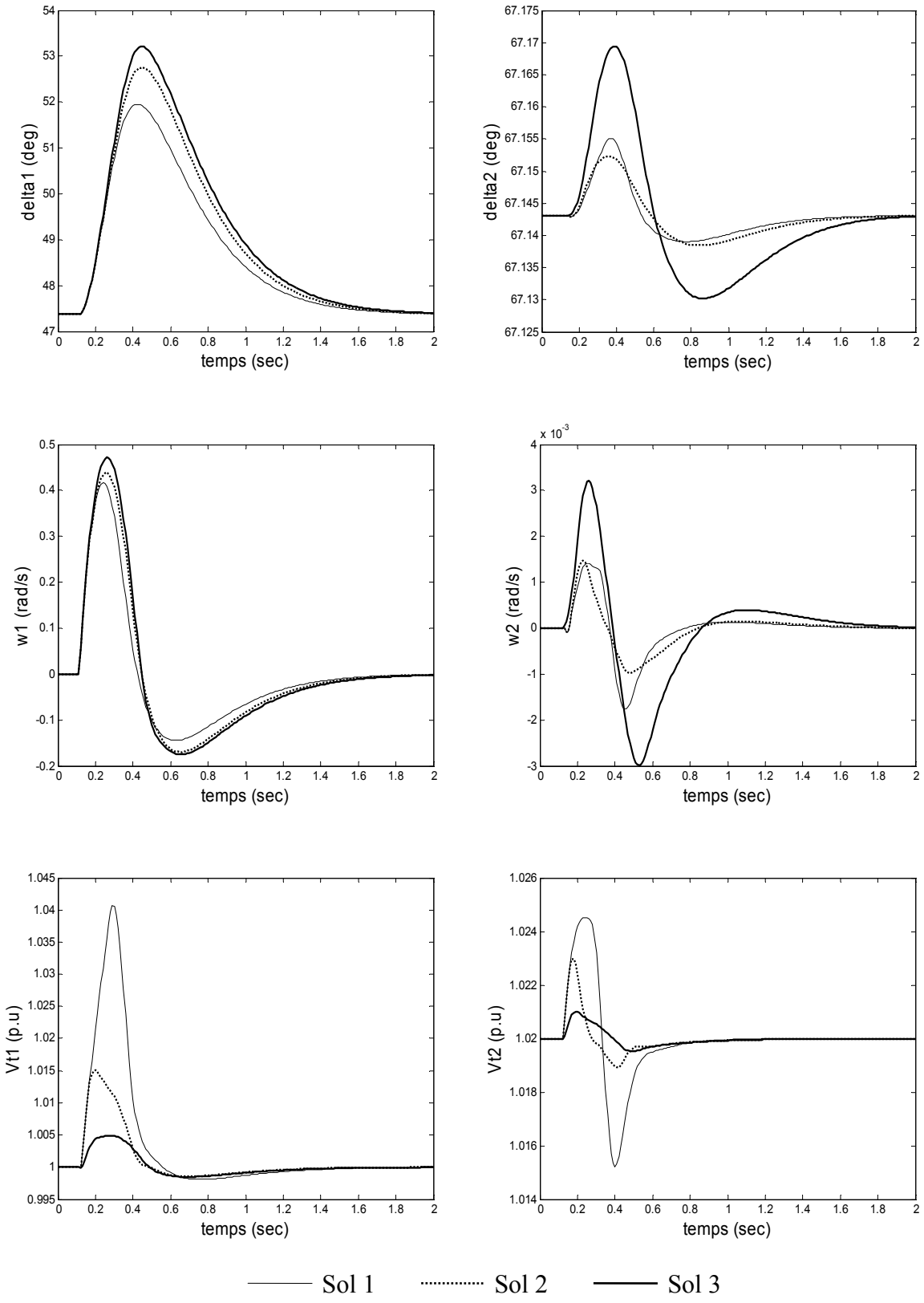


Figure 5.13 : Réponses de δ , ω et V_t . 2ème cas, 1er point de fonctionnement.

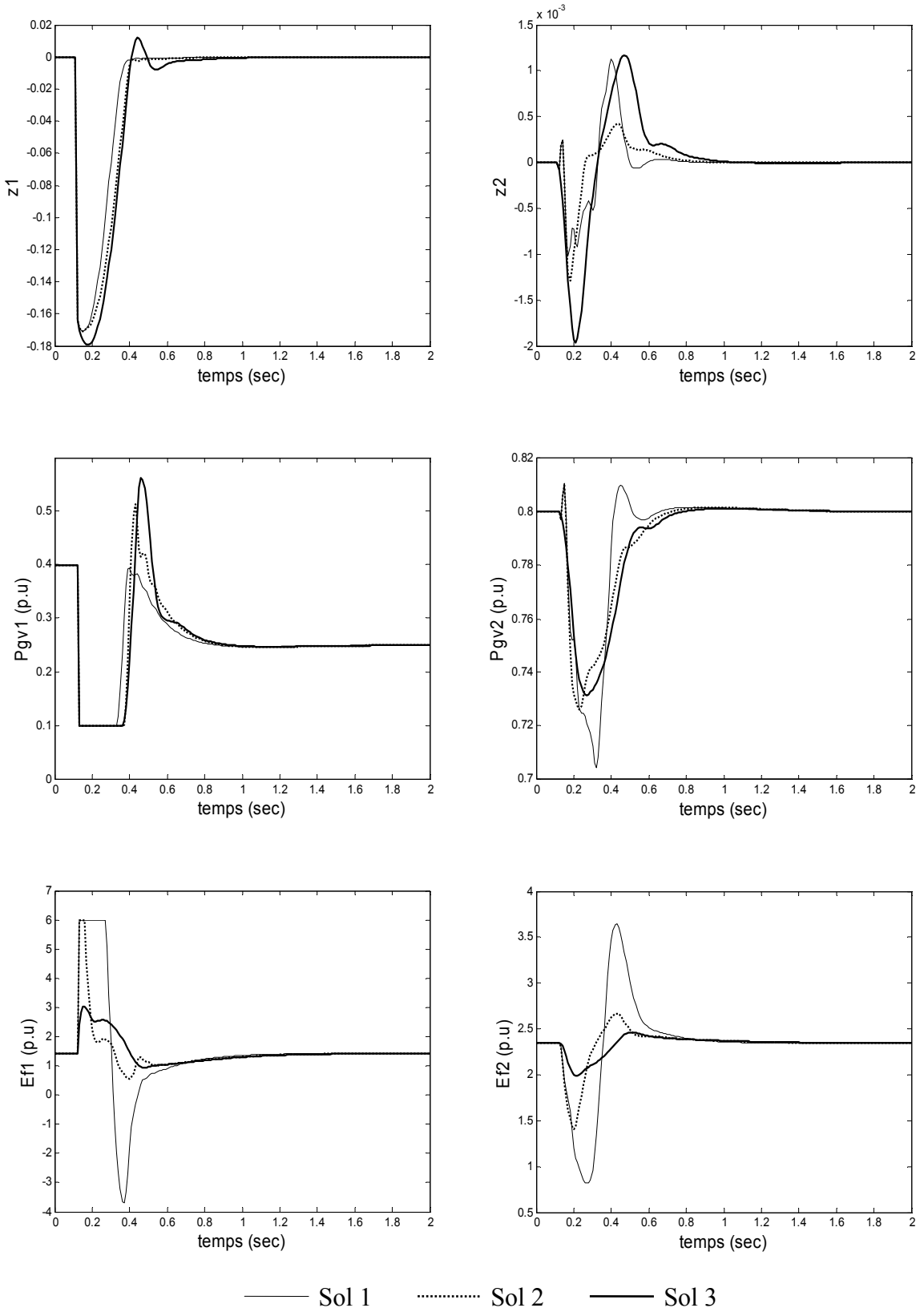


Figure 5.14 : Réponses de z , P_{GV} et E_f . 2ème cas, 1er point de fonctionnement.

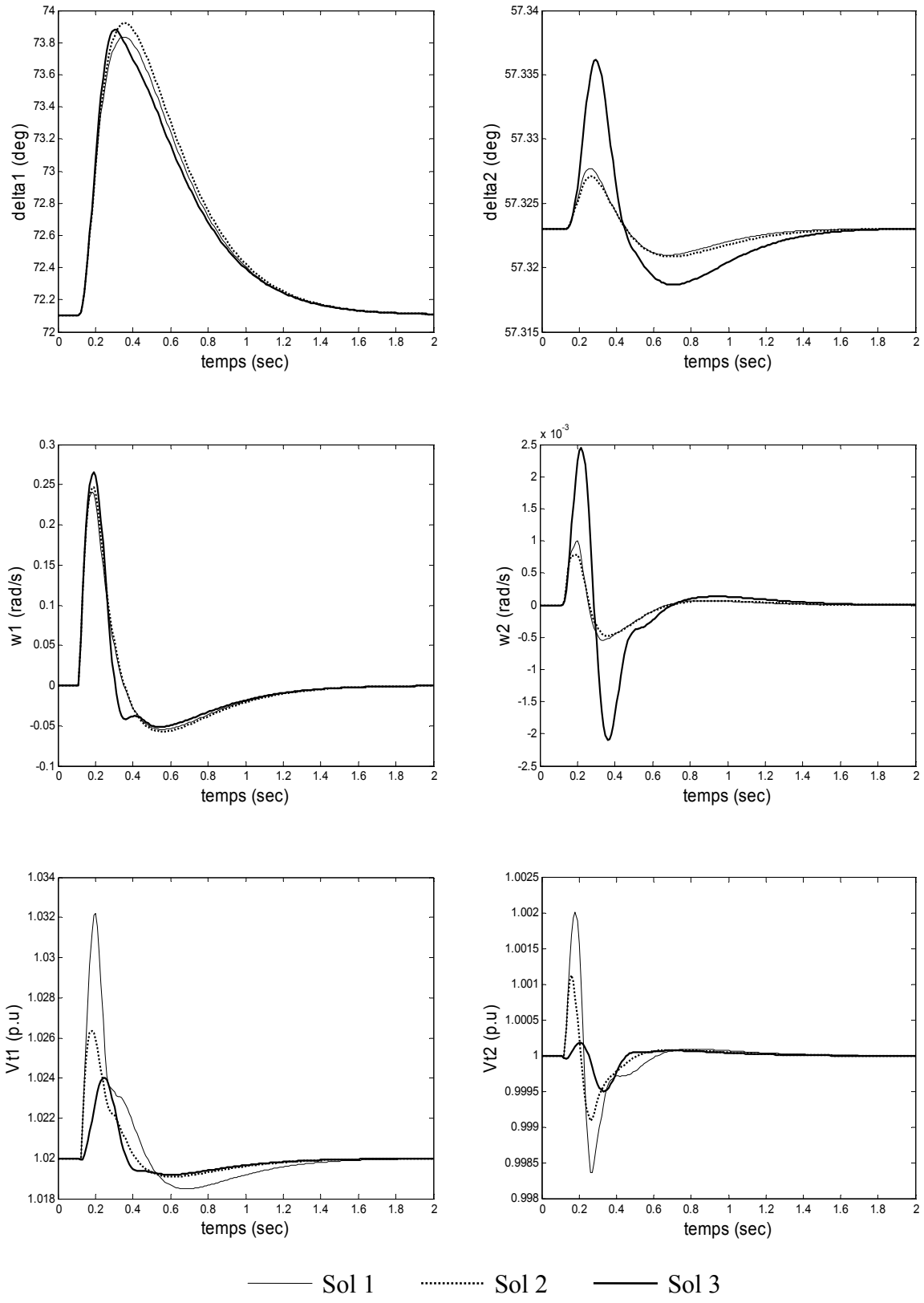


Figure 5.15 : Réponses de δ , ω et V_t . 2ème cas, 2ème point de fonctionnement.

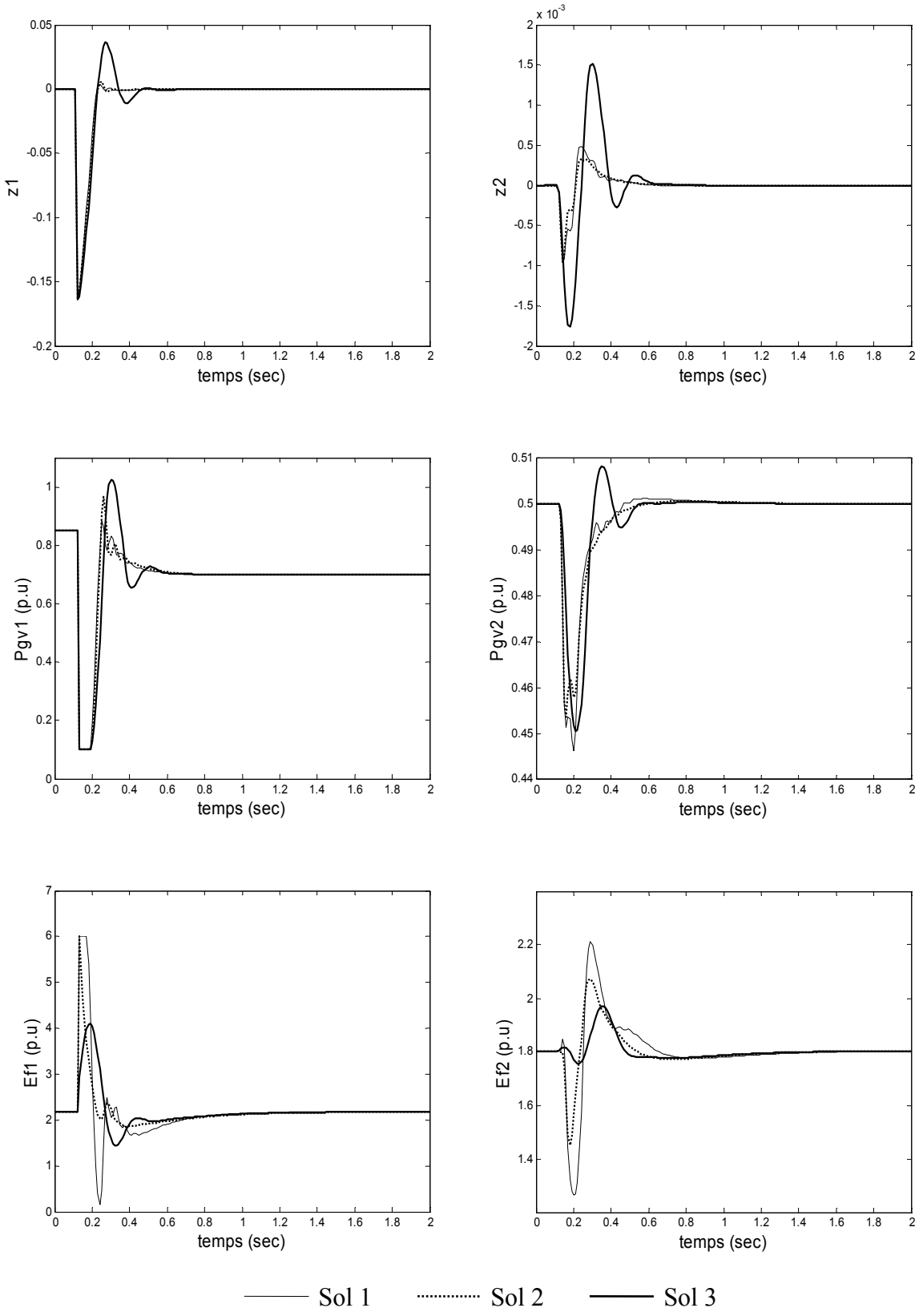


Figure 5.16 : Réponses de z , P_{GV} et E_f . 2ème cas, 2ème point de fonctionnement.

Chapitre 6
Conclusion

Chapitre 6

Conclusion

6.1 Synthèse

Dans ce travail, nous avons abordé et étudié les algorithmes évolutionnaires multiobjectifs avec application à un problème de commande d'un réseau électrique multimachine. Ainsi, plusieurs contributions ont été apportées. D'abord, nous avons revu les algorithmes évolutionnaires d'un point de vue mono-objectif en prenant comme exemple d'étude les algorithmes génétiques, toujours considérés comme des méthodes de référence dans le domaine du calcul évolutionnaire. En effet, ces derniers présentent des caractéristiques particulières leur permettant d'avoir des avantages de flexibilité, d'efficacité et de robustesse, pour traiter une variété de problèmes théoriques et réels.

L'optimisation multiobjectif est présentée ensuite dans le Chapitre 3. Le concept de Pareto est ainsi introduit et des définitions concernant l'optimisation multiobjectif en général sont données. Dans le même chapitre, sont décrits trois des algorithmes évolutionnaires multiobjectifs les plus connus et utilisés jusqu'à présent, à savoir NSGA-II, SPEA2 et ε-MOEA. Ces algorithmes se sont montrés capables de donner des réponses satisfaisantes aux problèmes d'optimisation multiobjectifs, dans différents domaines d'application.

Dans le Chapitre 4, nous avons proposé ε-GMOEA, un algorithme évolutionnaire multiobjectif amélioré qui imite de chacun des algorithmes précédents des caractéristiques avantageuses, afin d'aboutir à des performances meilleures. Les tests conduits à titre de comparaison entre ces différents algorithmes ont révélé que les deux approches ε-GMOEA et ε-MOEA, se basant sur le concept de la ε-dominance, permettent d'obtenir de meilleures performances du point de vue convergence. Néanmoins, leur répartition des solutions non dominées n'est pas toujours bonne. En plus, le choix des valeurs du paramètre ε n'est pas évident et la taille de leurs archives n'est encore pas limitée.

ε-GMOEA est utilisé ensuite dans le Chapitre 5 pour la synthèse multiobjectif des paramètres d'une commande décentralisée d'un réseau électrique à trois machines. La commande a été d'abord développée sur la base d'une association de la technique du backstepping et de la commande par logique floue. Les résultats obtenus ont montré que ε-GMOEA a permis de trouver un compromis entre les deux objectifs de réglage considérés. La performance et le comportement du régulateur témoignent bien de la validité de la méthode de conception proposée.

6.2 Perspectives

Les perspectives de notre travail portent aussi bien sur le plan de l'optimisation multiobjectif que sur le plan de la commande décentralisée des réseaux électriques. Les premières perspectives concernent l'amélioration de l'algorithme ε-GMOEA, et d'une façon générale les AEMOs. On peut envisager pour cela d'utiliser des techniques d'adaptation des paramètres d'un AEMO, comme la taille de la population, les paramètres des opérateurs génétiques et le critère d'arrêt. Une meilleure distribution des solutions ε-non dominées est préférable dans le cas des approches se basant sur le concept de la ε-dominance, en plus d'une limitation de la taille de leur archive. D'autre part, des métriques de performance plus robustes et plus significatives sont toujours bénéfiques pour l'évaluation des AEMOs. Par ailleurs, on peut penser à étudier d'autres métaheuristiques capables de donner de meilleures performances, notamment pour des problèmes difficiles et avec contraintes.

Pour la commande des réseaux électriques, on peut envisager d'abord une extension de l'application de la commande décentralisée, proposée dans ce travail, pour des cas de réseaux électriques plus larges. Par ailleurs, une technique de commande avancée de la tension terminale des générateurs devra améliorer les performances du réglage, notamment lors de la présence d'une coordination entre les différents blocs de régulation dans une même unité de génération. Pour le réglage flou, l'utilisation des méthodes d'optimisation multiobjectifs pour l'optimisation simultanée des fonctions d'appartenance et des règles floues et éventuellement d'autres paramètres, est encore recommandée.

Références bibliographiques

Références bibliographiques

[1] Abdel-Magid, Y. and M. Abido. Optimal Multiobjective Design of Robust Multimachine Power System Stabilizers Using Genetic Algorithms. *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1125–1132, 2003.

[2] Andreoiu, A. *On Power System Stabilizers: Genetic Algorithm Based Tuning and Economic Worth as Ancillary Services*. PhD thesis, Chalmers University of Technology, Sweden, 2004.

[3] Barichard, V. *Approches Hybrides pour les Problèmes Multiobjectifs*. Thèse de doctorat, Ecole Doctorale d'Angers, France, 2003.

[4] Befekadu, G., O. Govorun, and I. Erlich. Multi-Objective Optimization and Online Adaptation Methods for Robust Tuning of PSS Parameters. In *Proceedings of the International Symposium on Modern Electric Power Systems (MEPS'06)*, pp. 187–192, Wroclaw, Poland, 6–8 Sept, 2006.

[5] Chabane, Y. *Commande Hybride Logique Floue-Algorithmes Génétiques pour l'Amortissement de Perturbations dans les Réseaux Electriques*. Mémoire de Magister, Ecole Nationale Polytechnique, 2005.

[6] Chuang, Y. and C. Wu. A Damping Constant Limitation Design of Power System Stabilizer Using Hybrid Differential Evolution. *Journal of Marine Science and Technology*, vol. 14, no. 2, pp. 84–92, 2006.

[7] Day, R. *Explicit Building Block Multiobjective Evolutionary Computation: Methods and Applications*. PhD thesis, Air Force Institute of Technology, USA, 2005.

[8] Deb, K. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[9] Deb, K., S. Agrawal, A. Pratap and T. Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. Technical Report No 2000001. Kanpur: Indian Institute of Technology Kanpur, India, 2000.

[10] Deb, K. and M. Goyal. A Combined Genetic Adaptive Search (geneAS) for Engineering Design. *Computer Science and Informatics*, vol. 26, no. 4, pp. 30–45, 1996.

[11] Deb, K and A. Kumar. Real-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems. *Complex Systems*, vol. 9, no. 6, pp. 431–454, 1995.

[12] Deb, K., M. Mohan and M. Shikhar. A Fast Multiobjective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. Technical Report No 2003002. Kanpur: Indian Institute of Technology Kanpur, India, 2003.

[13] Deb, K., L. Thiele, M. Laumanns and E. Zitzler. Scalable Multiobjective Optimization Test Problems. In *Proceedings of Congress on Evolutionary Computation (CEC-2002), IEEE Service Center*, vol. 1, pp. 825–830, 2002.

[14] Fleming, P. and R. Purshouse. Genetic Algorithms in Control Systems Engineering. Research Report No. 789. Department of Automatic Control and Systems Engineering, University of Sheffield, UK, 2001.

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

[16] Guo, Y., D. Hill and Y. Wang. Global Transient Stability and Voltage Regulation for Power Systems. *IEEE Transactions on Power Systems*, vol. 16, no. 4, pp. 678–688, 2001.

[17] Guo, Y., D. Hill and Y. Wang. Nonlinear Decentralized Control of Large-Scale Power Systems. *Automatica*, vol. 36, pp. 1275–1289, 2000.

[18] Hu, W., S. Mei, Q. Lu, T. Shen and A. Yokoyama. Nonlinear Adaptive Decentralized Stabilizing Control of Multimachine Systems. *Applied Mathematics and Computation*, vol. 133, no. 2, pp. 519–532, 2002.

[19] Khalil, H. *Nonlinear Systems*. Second Edition, Prentice-Hall, 1996.

[20] Kundur, P. *Power System Stability and Control*. McGraw-Hill, Inc., 1994.

[21] Laumanns, M. *Analysis and Applications of Evolutionary Multiobjective Optimization Algorithms*. PhD thesis, Swiss Federal Institute of Technology, Switzerland, 2003.

[22] Laumanns, M., L. Thiele, K. Deb and E. Zitzler. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[23] Melahi, A. *Commande Décentralisée par Logique Floue des Processus Complexes: Application aux Réseaux Electriques*. Mémoire de Magister, Ecole Nationale Polytechnique, 2001.

[24] Menniti, D., A. Burgio, N. Sorrentino and A. Pinnarelli. Damping Oscillations Improvement by Fuzzy Power System Stabilizers Tuned by Genetic Algorithm. In *14th Power System Computation Conference (PSCC'02)*, 24–28 June, Sevilla, Spain, 2002.

[25] Mostaghim, S. *Multiobjective Evolutionary Algorithms: Data Structures, Convergence and Diversity*. PhD thesis, University of Paderborn, Germany, 2004.

[26] Okou, A. *Commande Non Linéaire Adaptative des Réseaux Electriques Multimachines*. Thèse de doctorat, Université de Québec, Canada, 2002.

[27] Passino, K and S. Yurkovich. *Fuzzy Control*. Addison Wesley Longman, Inc., 1998.

[28] Pothiya, S., I. Ngamroo, S. Runggeratigul and P. Tantaswadi. Design of Optimal Fuzzy Logic based PI Controller using Multiple Tabu Search Algorithm for Load Frequency Control. *International Journal of Control, Automation, and Systems*, vol. 4, no. 2, pp. 155–164, 2006.

[29] Psillakis, H. and A. Alexandridis. A New Excitation Control for Multimachine Power Systems, I: Decentralized Nonlinear Adaptive Control Design and Stability Analysis. *International Journal of Control, Automation, and Systems*, vol. 3, no. 2 (special edition), pp. 278–287, 2005.

[30] Tran, K. *An Improved Multiobjective Evolutionary Algorithm with Adaptable Parameters*. PhD thesis, Nova Southeastern University, USA, 2006.

[31] Villalobos-Arias, M. *Analysis of Optimization Heuristics for Multiobjective Problems*. PhD thesis, CINVESTAV-IPN, Mexico, 2005.

[32] Wu, W. *Synthèse d'un Contrôleur Flou par Algorithme Génétique : Application au Réglage Dynamique des Paramètres d'un Système*. Thèse de doctorat, Université de Lille 1, France, 1998.

[33] Xue, F. *Multiobjective Differential Evolution: Theory and Applications*. PhD thesis, Rensselaer Polytechnic Institute, USA, 2004.

[34] Zhu, C., R. Zhou and Y. Wang. A New Decentralized Nonlinear Voltage Controller For Multimachine Power Systems. *IEEE Transactions on Power Systems*, vol. 13, no. 1, pp. 211–216, 1998.

[35] Zitzler, E. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology, Switzerland, 1999.

[36] Zitzler, E., M. Laumanns and S. Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. Metaheuristics for Multiobjective Optimization, Lecture Notes in Economics and Mathematical Systems, Springer, vol. 535, pp. 3–37, 2004.

[37] Zitzler, E., M. Laumanns and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.