Université Saad Dahlab de Blida

Faculté des Sciences
Département d'informatique

Mémoire présenté par :

BOUKHICH Mohamed Yazid
et
CHERFAOUI Moussa

En vue d'obtenir le diplôme de Master
Domaine : Mathématique et Informatique.
Filière : Informatique.
Spécialité : Informatique.
Option : Ingénierie du logiciel.

Sujet :

Combinaison de la convergence et de la diversité dans l'optimisation multi-objective évolutionnaire

Soutenue le: 10/10/2013 devant le jury composé de :

Mme. BENSETTITI	Présidente.
Mr. AIT AKKACHE Mustapha	Promoteur.
Mr. OULD AISSA	Examinateur.
Mr. BOUGHERARA	Examinateur.

Année : 2012/2013

Remerciement

En premier lieu, nous remercionsAllah le tout puissant qui nous a donné la volonté, le courage et toute la force durant toutes ces années d'études ainsi de pouvoir réaliser ce modeste travail.

Tous nos remerciements s'adressent à nos parents pour leurs sacrifices dès le début, Ainsi leurs soutiens pour surmonter les moments difficiles que nous avons vécus lors de notre parcours universitaire.

Notre profonde gratitude et nos sincères remerciements vont à notre promoteurM. AIT AKKACHEMustapha, pour nous avoir proposé ce sujet et de nous avoir dirigé et encouragé ainsi soutenu tout au long de la réalisationde ce travail.

Nous remercions infiniment tous nos amis de l'université Saad Dahleb de nous avoir soutenus durant toute cette période.
Merci à tous.

Dédicace

Je dédie ce modeste travail

A mes chers parents que dieu le tout puissant les protége.

A toute la famille Cherfaoui

A mon frère Toufik et ma sœur Lamia et ma petite nièce Israa

A tous mes amis

À mon binôme Yazid

A toute la promotion de Master de département d'informatique 2012/2013

Je dédie ce modeste travail

Moussa

Dédicaces

A mes parents qui ont été la source du bonheur et joie dans ma vie pour

leur soutient tout au long de mes études .

A mes sœurs.

A mon frère.

A mes neveux Amine, Fouad ,Akram, YOUCEF.

A mes grands parents .

A mes tantes et oncles.

A mes beaux frères.

A Mehdi , Karim , Leila , Zakaria , rayan.

A tous mes chers amis.

A mon binôme Moussa.

A tous ceux que j'ai eu le plaisir de connaitre durant toute ma vie.

Je dédie ce modestetravail

Yazid

Résumé

Résumé

Le but de ce travail est de concevoir un outil qui combine la convergence et la diversité dans l'optimisation multi-objective évolutionnaire à travers l'utilisation d'une forme décontractée de la dominance de Pareto dite la ϵ-dominance [2].

Le travail consiste dans un premier temps à étudier la ϵ-dominance ainsi que son utilisation commeune stratégie d'archivagedans les algorithmesévolutionnairesmulti- objectifs,puis à faire une série de simulations en utilisant les fonctions tests de Zitzler, Deb et Thiele[26]. Ces simulations permettent d'étudier la stabilité des résultats de l'algorithme obtenu par rapport aux paramètres d'initialisation.

Mots clés

Algorithmes évolutionnaires ; Archivage ; Convergence ; Diversité ; Dominance de Pareto ; Elitisme ; ϵ-dominance ; Lexico-Max-Ordering ; Optimisation multi-objective.

Summary

The aim of this work was to design a tool that combines the convergence and diversity in multi-objective evolutionary optimization using a relaxed form of Pareto dominance called the ϵ-dominance [2].

This work is a first step to study the ϵ-dominance and its use as an archiving strategy in multi-objective evolutionary algorithms, then to make aseriesof simulations using the test functions of Zitzler, Deb and Thiele [26]. These simulations are used to study the stability of the results obtained from the algorithm, compared with initialization parameters.

Keywords:

Archiving ; Convergence ; Diversity ; Elitist ; Evolutionary algorithms ; ϵ-dominance ; Lexico-Max-Ordering ; multi-objective optimization ; Pareto dominance.

Sommaire

Introduction générale..... 1

I. Chapitre I : Optimisation multi-objective

I.1Introduction..... 3
I.2L'optimisation mono-objective..... 3
I.3L'optimisation multi-objective..... 4
I.4Dominance et optimalité de Pareto..... 4
I.4.1Concept de dominance et solutions Pareto-optimales..... 5
I.5Les relations dérivées de la dominance..... 9
I.5.1 Optimalité au sens lexicographique..... 10
I.5.2 Optimalité au sens Max-ordering (optimalité maximale)..... 10
I.5.3 Optimalité lexicographique max-ordering..... 10
I.6 la ϵ–Dominance..... 12
I.7 Vecteur idéal et vecteur de nadir..... 14
I.8La convexité..... 15
I.9Conclusion..... 16

II.ChapitreII : Evolution artificielle

II.1 Introduction..... 17
II.2 Algorithmes évolutionnaires..... 17
II.3Principe d'un algorithme évolutionnaire..... 18
II.4 Représentation d'un individu..... 20
II.4.1 Le principe..... 20
II.4.2 La représentation binaire..... 21
II.4.3 La représentation réelle..... 21
II.5 Initialisation de la population..... 21
II.6 Darwinisme artificiel..... 21
II.6.1 Sélection..... 21
II.6.2 Remplacement..... 23
II.7 Operateurs de variation..... 24
II.7.1 Le croisement..... 24
*Le croisement binaire..... 24
*Le croisement réel..... 26
II.7.2lamutation..... 27
*La mutation binaire..... 27
*La mutation réelle..... 28
II.8 Algorithmes évolutionnaires multi-objectif..... 29
II.8.1 (NSGA- II) ElitistNondominatedSortingGeneticAlgorithm..... 29
II.8.2 (M.O.G.A)La méthode Multiple Objective GéneticAlgorithm..... 32
II.9 Conclusion..... 34

III. Chapitre III :AEMO ET Problème test

III.1 Introduction..... 35
III.2 ϵ-MOEA(ϵ -Domination based Multi-Objective EvolutionnaryAlgorithm)..... 35
III.3 ϵ-MOEA-LMO(ϵ-Domination based Multi-Objective Evolutionnary Algorithm Lexicographique Max Ordering)..... 39
III.4 Problèmes tests..... 40
III.4.1 ZDT1..... 42
III.4.2 ZDT2..... 42
III.4.3 ZDT3..... 43
III.4.4 ZDT4..... 43
III.4.5ZDT6..... 44
III.5 Mesure de performance..... 45
III.5.1 Métriques exactes..... 45
III.5.2 Métriques aveugles..... 46
III.6Conclusion..... 48

IV. Chapitre IV : Réalisation et tests

IV.1Introduction..... 49
IV.2 Résultats expérimentaux..... 49
IV.2.1 Conditions expérimentales..... 49
IV.3 Implémentation de ϵ-MOEA..... 49
IV.4 Comparaison entre ϵ-MOEA etϵ-MOEA-LMO..... 50
IV.5 Comportement de ϵ-MOEA-LMO par rapport à l'incorporation de connaissance en initialisation..... 61
IV.6 Comportement de ϵ-MOEA par rapport à l'incorporation de connaissance en initialisation..... 63
IV.7 Comparaison par rapport à l'initialisation entre l' ϵ-MOEA-LMO etϵ-MOEA..... 65
IV.8 Temps d'exécution..... 68
IV.9Conclusion..... 69

Conclusion générale..... 70

A Implémentation et réalisation..... 71

Liste des figures

I.1 Exemple de dominance..... 5
I.2 Représentation des solutions dans le plan $f1, f2$.....6
I.3 Les solutions et leurs rangs de Pareto.....6
I.4 Différents opérateurs de reproduction et front de Pareto 8
I.5 Optimalité Locale 8
I.6 Le théorème du contact 9
I.7 Les solutions de Pareto, Lex-MO et Max-ordering.....12
I.8 Grille uniforme avec 400 boxes (capacité maximum de 20 points).....14
I.9 Concepts de dominance (gauche) et de ϵ-dominance (droite) 14
I.10Vecteur idéal Z^* et vecteur de Nadir Z^{nad}.....15
I.11Exemple d'ensemble convexe et ensemble non convexe 15

II.1 Cycle d'un AE.....19
II.2Sélection par roulette.....22
II.3 Croisement binaire à un point.....25
II.4Croisement binaire à trois points 25
II.5 Le croisement uniforme.....26
II.6 Mutation binaire.....28
II.7Fonction de densité de probabilité de création d'un enfant en utilisant l'opérateur de mutation polynomiale.....28
II.8 Principe de NSGA-II.....30
II.9 Principe de crowding.....31
II.10 Exemple de fonction d'efficacité.....33

III.1 Concept de vecteur d'identification.....38
III.2 Principe de fonctionnement de ϵ-MOEA.....39
III.3 Solution deZDT1.....42
III.4 Solution de ZDT2.....43
III.5 Solution de ZDT3.....43
III.6 Solution de ZDT4.....44
III.7 Solution de ZDT6.....45
III.8 Exemple d'application de la métrique (Hypervolume).....47
III.9 Métrique de recouvrement des deux ensembles.....48

IV.1Front de Pareto de ϵ-MOEApour le problème ZDT1.....50
IV.2Les valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT1.....52
IV.3 Les valeurs de SCM et SP en fonction de ϵ pour 10 000 itérations pour ZDT1.....52
IV.4 Front ZDT1 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000.....53
IV.5 Les valeurs de SCM et SP en fonction de ϵ pour 5 000 itérations pour ZDT2.....54

IV.6Les valeurs de SCM et SP en fonction de ϵ pour 10 000 itérations pour ZDT2.....54
IV.7 Front ZDT2 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations..55
IV.8 Les valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT3.....56
IV.9 Les valeurs de SCM et SP en fonction de ϵ pour 10000 itérations pour ZDT3.....56
IV.10Front ZDT3 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations56
IV.11Les valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT4.....58
IV.12Les valeurs de SCM et SP en fonction de ϵ pour 10000 itérations pour ZDT4.....58
IV.13Front ZDT4 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations58
IV.14 Les valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT6.....59
IV.15 Les valeurs de SCM et SP en fonction de ϵ pour 10000 itérations pour ZDT6.....60
IV.16Front ZDT4de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations.60
IV.17SCM et SP entre ϵ-MOEA-LMO et ϵ-MOEA pour ZDT2.....63
IV.18 SCM et SP entre ϵ-MOEA et ϵ-MOEAI pour ZDT3.....65
IV.19 Front ZDT1 de ϵ-MOEA et ϵ-MOEAI après 3000 itérations.....65
IV.20 SCM et SP de ϵ-MOEA-LMOI et deϵ-MOEAI pour ZDT4.....67
IV.21 Front ZDT2 de ϵ-MOEA-LMOI et ϵ-MOEAI après 1500 itérations 68

A.1 Interface principale de l'application.....71
A.2 Graphe de la mesure de performance SCM.....72

Liste des tableaux

I.1 Les valeurs des objectifs et les Sorts.....11
IV.1 Comparaison entre ϵ-MOEAet ϵ-MOEA-LMOpour le problème ZDT1.....51
IV.2 Comparaison entre ϵ-MOEAet ϵ-MOEA-LMOpour le problème ZDT2.....53
IV.3 Comparaison entre ϵ-MOEAet ϵ-MOEA-LMOpour le problème ZDT3..... 55
IV.4 Comparaison entre ϵ-MOEAet ϵ-MOEA-LMOpour le problème ZDT4..... 57
IV.5 Comparaison entre ϵ-MOEAet ϵ-MOEA-LMOpour le problème ZDT6..... 59
IV.6 Comparaison entreϵ-MOEA-LMO et ϵ-MOEA-LMOIpour ZDT1..... 61
IV.7 Comparaison entreϵ-MOEA-LMO et ϵ-MOEA-LMOIpour ZDT2..... 61
IV.8 Comparaison entreϵ-MOEA-LMO et ϵ-MOEA-LMOIpour le problème..... 62
IV.9 Comparaison entreϵ-MOEA-LMO et ϵ-MOEA-LMOIpour ZDT4..... 62
IV.10 Comparaison entreϵ-MOEA-LMO et ϵ-MOEA-LMOIpour ZDT6..... 62
IV.11 Comparaison entreϵ-MOEA et ϵ-MOEAI pour ZDT1.....63
IV.12 Comparaison entreϵ-MOEA et ϵ-MOEAI pour ZDT2.....63
IV.13 Comparaison entreϵ-MOEA et ϵ-MOEAI pour ZDT3.....64
IV.14 Comparaison entreϵ-MOEA et ϵ-MOEAI pour ZDT4.....64
IV.15 Comparaison entreϵ-MOEA et ϵ-MOEAI pour ZDT6.....64
IV.16 Comparaison entreϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT1.....65
IV.17 Comparaison entreϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT2.....66
IV.18 Comparaison entreϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT3.....66
IV.19 Comparaison entreϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT4.....66
IV.20 Comparaison entreϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT6.....67
IV.21 Temps d'exécution des différents algorithmes.....68

Introduction générale

L'optimisation est l'une des branches les plus importantes des mathématiques appliquées modernes et de nombreuse recherches, à la fois pratiques et théoriques, lui sont consacrées.

Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes appliqués au traitement d'images, à la conception de systèmes, la planification des taches, …etc. Ces problèmes sont souvent multi objectifs. De manière générale, plusieurs critères contradictoires doivent être satisfaits simultanément. La notion de solution unique disparait pour les problèmes de ce type, au profit de la notion de compromis entre les divers objectifs. L'étude des compromis à donner lieu à la définition des solutions optimales au sens de Pareto.

Les algorithmes évolutionnaires multi-objectifs(AEMO) sont maintenant reconnus comme une technique particulièrement adaptée à la recherche de la surface de Pareto, puisqu'ils travaillent sur une population de solutions-candidats, à la différence de la plupart des méthodes traditionnelles qui manipulent en général un point unique de l'espace de recherche. Cette particularité des algorithmes évolutionnaires rend possible l'obtention d'un ensemble de Pareto approché en un seul essai de l'algorithme. De plus, l'utilisateur n'est pas obligé de définir des paramètres subjectifs supplémentaires comme les poids relatifs des critères d'optimisation. Par ailleurs, ce type d'algorithmes peut être appliqué aussi bien aux problèmes discrets que continus ou mixtes.

Le but de ce rapport, est l'étude des méthodes d'optimisation multi-objectives évolutionnaires, qui ont reçu un intérêt croissant ces dernières années.

Dans un premier temps, nous avons étudié et mis en œuvre l'algorithme **ϵ-MOEA** proposé par K. Deb [30]. Cet algorithme combine la convergence et la diversité et permet à l'utilisateur de fixer à priori, la cardinalité du front de Pareto recherché.

Dans **ϵ-MOEA** l'espace des objectifs est partagé en boxes et pour chaque boxe l'algorithme permet au plus à une seule solution de submerger et Cela grâce à une double sélection.

- La première est relative aux boxes et elle est basée sur la notion de ϵ-dominance, une forme décontractée de la dominance de Pareto.

- la seconde repose sur une procédure combinant la dominance de Pareto et la distance euclidienne par rapport aux solutions retenues dans les boxes.

Dans un deuxième temps, nous étudierons la possibilité de remplacement de la procédure utilisée pour les solutions dans les boxes par une autre technique fondée sur la dominance Lexco-Max_Ordering, une dérivée de la dominance de Pareto [24].

Ensuite, nous nous sommes intéressés au comportement de l'algorithme **ϵ-MOEA** par rapport à l'intégration de connaissances lors de son initialisation sur le vrai front de Pareto du problème à résoudre.

L'organisation de ce mémoire est la suivante,

-Le premier chapitre (Chapitre I) est consacré à la présentation de l'optimisation multi-objectifs, et des concepts fondamentaux comme la dominance de Pareto, la ϵ-dominance et la dominance lexico_Max_Ordering sur lesquelles se porte notre intérêt.

-Dans le deuxième chapitre (Chapitre II) nous trouvons un état de l'art sur les algorithmes évolutionnaires multi objectif (EMOA).

-Dans le troisième chapitre (Chapitre III) nous présentons l'algorithme **ϵ-MOEA**, ainsi que le **ϵ-MOEA** modifié, les fonctions tests sur lesquelles nous appliquons nos algorithmes ainsi que les mesures utiles pour la comparaison de nos algorithmes.

-Le dernier chapitre (Chapitre IV) contient nos expérimentations numériques pour la comparaison des deux versions de l'algorithme ainsi que leur comportement vis-à-vis d'initialisation contenant des informations sur le vrai front de Pareto.

Chapitre 1

Optimisation multi-objective

I.1Introduction

La principale difficulté rencontrée en optimisation mono-objectif vient du fait que modéliser un problème sous la forme d'une équation unique peut être une tâche difficile et peut aussi biaiser la modélisation.

L'optimisation multi-objectif autorise des degrés de liberté qui manquaient en optimisation mono-objectif. Cette souplesse n'est pas sans conséquences sur la démarche à suivre dans la recherche de l'optimum aux problèmes posés par le multi objectif. Cette recherche ne nous donnera plus une solution unique, mais une multitude de solutions. Ces solutions sont appelées ***solutions de Pareto*** et l'ensemble de solutions obtenu à la fin de la recherche est la ***surface de compromis***.

I.2 L'optimisation Mono-objective [1]

Un problème d'optimisation mono-objectif se définit comme la recherche du minimum ou du maximum (l'optimum) d'une fonction donnée $f: D \rightarrow Y \subseteq R$. Le domaine Y doit être un sous-ensemble des nombres réels (Y ⊆ R) et le domaine D de f est appelé l'espace de décision. Dans la suite de ce mémoire, nous considérerons que tous nos problèmes d'optimisation sont à minimiser, car le principe de dualité utilisé dans le contexte d'optimisation, permet de transformer la maximisation en la minimisation en multipliant la fonction objective par -1.

Nous pouvons aussi trouver des problèmes d'optimisation pour lesquels les variables de la fonction à optimiser sont contraintes à évoluer dans une certaine partie de l'espace de recherche. Dans ce cas, il s'agit alors d'une forme particulière de ce qui forme un problème d'optimisation sous contraintes.

En mathématiques, un problème d'optimisation se présentera sous la forme suivante:

Minimiser	$f(\vec{x})$	(fonction à optimiser).
avec	$\vec{g}(\vec{x}) \leq 0$	(m contraintes d'inégalité).
et	$\vec{h}(\vec{x}) = 0$	(p contraintes d'égalité).

Avec : $\vec{x} \in R^n, \vec{g}(\vec{x}) \in R^k$ et $\vec{h}(\vec{x}) \in R^p$.

Ici le vecteur $\vec{x}$ représente les variables de décision et les vecteurs $\vec{g}(\vec{x})$ et $\vec{h}(\vec{x})$ représentent respectivement m contraintes d'inégalité et p contraintes d'égalité.

I.3L'optimisation multi-objective [1][5][26]

Lors de la modélisation d'un problème, nous cherchons souvent à satisfaire plusieurs objectifs. Par exemple, nous recherchons un système performant et qui consomme peu. Dans ce cas, nous parlons de problème d'optimisation multi-objective (ou problème d'optimisation multicritère), il peut ressembler à la forme suivante :

$$\begin{aligned} &\text{Minimiser} f_m(\vec{x}) && \text{m=1,..., M} \\ &g_j(\vec{x}) \geq 0 && \text{j=1, ..., k} \\ &h_j(\vec{x}) = 0 && \text{j=1,..., L} \\ &x_i^l \leq x_i \leq x_i^u \end{aligned}$$

Le vecteur $\vec{x} = (x_1, \ldots, x_n)$ et le vecteur de n variables de décision. Les x_i^l et x_i^u sont respectivement les bornes inferieure et supérieure de la variable x_i. Ces bornes définissent l'espace de décision D.

Si une solution $\vec{x}^*$ ne satisfait pas au moins une des K contraintes d'inégalité ou une des L contraintes d'égalité, elle est dite solution infaisable. L'ensemble des solutions faisable constitue l'espace faisable de l'espace de recherche noté $S(S \subset D)$.

Dans le cas multi-objectif, le concept d'optimum est différent de celui du cas mono-objectif. En effet, il ne s'agit plus ici de la recherche d'un unique optimum global, mais plutôt d'une surface de solutions qui offrent un bon «compromis » entre les différents objectifs. Pour bien comprendre la notion d'optimalité dans le cas multi-objectif, nous introduisons les définitions suivantes, basées sur les travaux de Pareto.

I.4Dominance et optimalité de Pareto

Les fondements mathématiques pour l'optimisation multi objectif qui considère des critères contradictoires d'une manière équitable a été posée par Vilfre do Pareto il ya 110 ans [24]. La notation de l'optimalité au sens de Pareto est fortement basée sur la définition de la domination.

I.4.1Concept de dominance et solutions Pareto-optimales

Définition 1 Dominance[5]

La solution $x^{(i)} \epsilon S$ domine la solution $x^{(j)} \epsilon S$ (notée $x^{(i)} \succ x^{(j)}$) si les conditions suivantes sont vérifiées :

1) $f_m(x^{(i)}) \leq f_m(x^{(j)}) \forall m \epsilon \{1, .., M\}$.

2) $\exists m \epsilon \{1, .., M\}$ tq : $f_m(x^{(i)}) < f_m(x^{(j)})$.

La solution $x^{(i)} \epsilon S$ domine **faiblement** la solution $x^{(j)} \epsilon S$ (on note $x^{(i)} \succcurlyeq x^{(j)}$)si :

$f_m(x^{(i)}) \leq f_m(x^{(j)}) \forall m \epsilon \{1, .., M\}$.

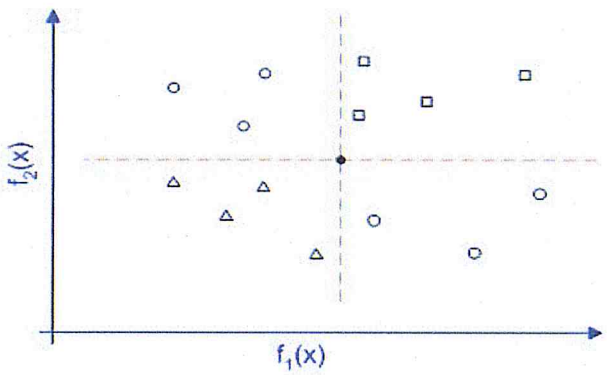
Remarque : Si le problème d'optimisation est un problème de maximisation, les symboles $\leq$ *et* $<$ dans les ingalités ci-dessus seront remplacés respectivement par $\geq$ *et* $>$.

La Figure I.1 illustre un exemple graphique des définitions mentionnées ci-dessus, le point noir:

– domine chacun des carrés,

– est dominé par chacun des triangles, et

– est équivalent aux anneaux au sens de la dominance.



FigureI.1- exemple de dominance[5]

Exemple : pour mieux assimiler la relation de dominance, nous allons proposer un exemple : Nous considérons un problème à deux objectifs : maximiser $f_1$et minimiserf_2. L'ensemble des solutions est représenté dans le planf_1, f_2. (Voir **Figure I.2**).

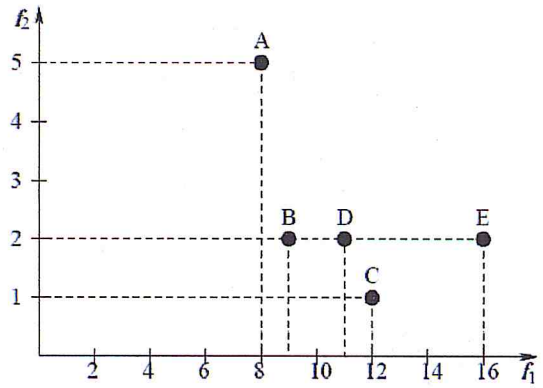


Figure I.2 -Représentation des solutions dans le plan *f*1, *f*2.[1]

Les deux pointes (E et C) de la **Figure I.2** ***représentent l'ensemble des points non dominés***. Nous pouvons établir un classement des solutions en fonction du ***rang de domination***. Ces points (E et C) sont donc des solutions **optimales au sens de Pareto** de rang1, car ils dominent tous les autres points mais ne se dominent pas entre eux. La **Figure I.3** représente les différents points et leur rang.

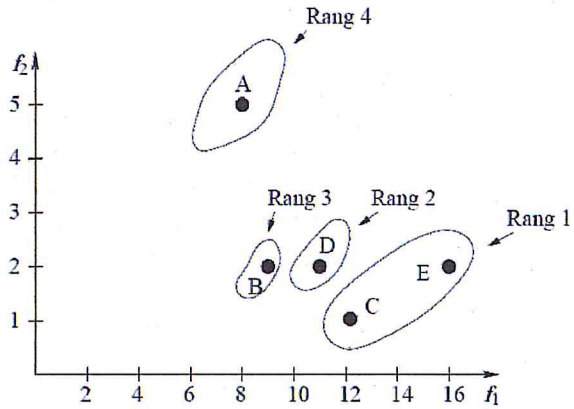


Figure I.3 - les solutions et leurs rangs de Pareto [1]

Voici ci-dessous **l'algorithme1.1**qui permet l'assignation du rang de Pareto, la variable N désigne le nombre de points de l'ensemble sur lequel, nous effectuons les comparaisons.

Algorithmel.1 : Assignation du front de Pareto.[1]

RangCourant=1 M=N

While N ≠ 0 do

For i=1 to m dolf $X_i$est non dominé

Rang(X_i,t)=RangCourant End For

For =1 to m do If Rang(X_i,t)=RangCourant

Ranger X_i dans une population temporaire

N=N-1 End ForRangCourant=RangCourant+1

M=N

End While

Définition 2 Ensemble de Pareto [5][27]

L'ensemble de Pareto global S* du problème d'optimisation multi-objectif est l'ensemble de points tels qu'aucun autre point de l'espace faisable S ne les domine, c'est-à-dire :

$$S^*=\{p \in S \mid \nexists\, q \in S : q \succ p\}$$

L'image de l'ensemble de Pareto dans l'espace des critères est appelée la surface dePareto (ou le front de Pareto dans le cas de problème bi-objectif).

Définition 3 Front de Pareto[11]

Soit F l'image de l'ensemble réalisable X dans l'espace des objectifs, Le Front de Pareto ND (F) de $f(X) = (f_1, f_2)(X)$ est défini comme suit :

$$ND(F)=\{\vec{v} \in F \mid \nexists\, \vec{u} \in F, \vec{u} < \vec{v}\}$$

Le Front de Pareto est aussi appelé ***l'ensemble des solutions efficace*** ou ***la surface des compromis.***

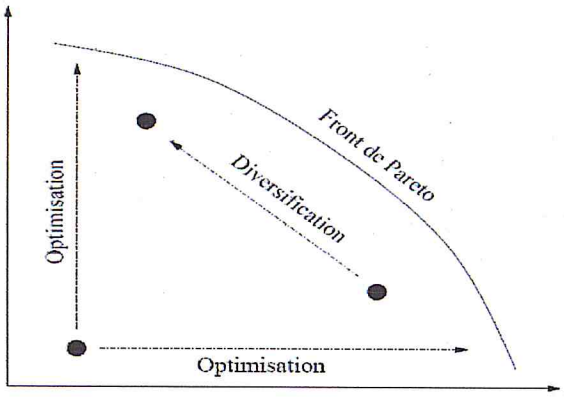


Figure I.4-Différents opérateurs de reproduction et front de Pareto[1]

Définition 4 L'optimalité locale au sens de Pareto[1][4][11]

Un vecteur de décision $\vec{u} \in X$ est dit Pareto localement optimal ***SSi*** :

Pour un $\delta > 0$ fixé :

$$\nexists \vec{v} \in X, \vec{f}(\vec{v}) \in B(\vec{f}(\vec{u}),\delta) \text{ et} \vec{v} \succ \vec{u}$$

Où $B(\vec{f}(\vec{u}),\delta)$ représente une boule de centre $\vec{f}(\vec{u})$ et de rayon δ.

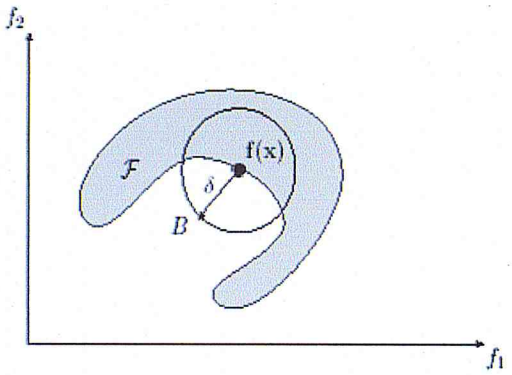


Figure I.5- Optimalité Locale [11]

Définition 5 L'optimalité globale au sens de Pareto [1] [4] [11]

Soit F l'image de l'ensemble réalisable X dans l'espace des objectifs. Un vecteur de décision $\vec{u} \in X$ est dit Pareto globalement optimal Ssi $\nexists \vec{v} \in X, \vec{v} \succ \vec{u}$, dans ce cas $\vec{f}(\vec{u}) \in F$ est appelé : Solutionefficace.

La différence entre cette définition et celle de l'optimalité locale tient dans le fait que nous ne considérons plus une restriction de l'ensemble R^n.

Une version graphique de la précédente définition utilise le théorème de contact.

Définition 6 Cône négatif [1] [11]

Un cône négatif est défini dans R^k de la manière suivante :

$$C^{-}=\{ \vec{x} \mid \vec{f}(\vec{x}) \in R^k et\ \vec{f}(\vec{x}) \leq 0\}$$

Nous disons aussi qu'un vecteur $\vec{x}$ est optimal au sens de Pareto pour un problème d'optimisation multi-objective donné si :

$$(C^{-} + \vec{x}) \cap F = \{\vec{x}\}$$

Où F désigne l'espace de solutions réalisables.

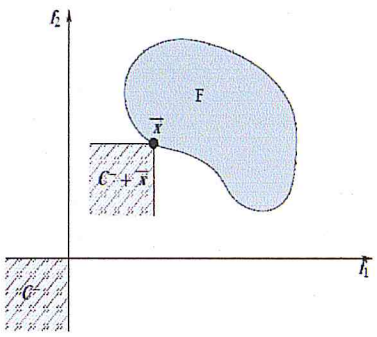


Figure I.6- Le théorème du contact[1]

I.5 Les relations dérivées de la dominance

La relation de dominance n'offre pas de degrés de liberté dans sa définition. Par exemple, il n'est pas possible d'inclure dans la définition de la relation de dominance une préférence d'un

objectif par rapport à un autre. C'est pour contrecarrer ce manque de flexibilité que des relations dérivées de la relation de dominance ont été développées. Les solutions que permettent de trouver ces relations dérivées de la dominance sont toutes optimales au sens de Pareto. La grande différence rencontrée avec ces relations est que l'ensemble des solutions obtenu avec ces relations est un sous-ensemble de l'ensemble des solutions obtenu avec la relation de dominance de Pareto.

I.5.1 Optimalité au sens lexicographique [16]

Une solution $\overrightarrow{x^*} \in R^n$ est optimale au sens lexicographique :

$$\overrightarrow{x^*} \succcurlyeq_{lex} \vec{x}. \forall \vec{x} \in R^n - \{\overrightarrow{x^*}\}$$

Si $\vec{x}, \vec{y} \in R^n$,nous disons que $\vec{x} \succcurlyeq_{lex} \vec{y}$ s'il existe une valeur d'index q^* telle que $f_q(\vec{x}) = f_q(\vec{y})$ pour $q = 1, \ldots., (q^* - 1)$ et $f_q^*(\vec{x}) < f_q^*(\vec{y})$. Les relations entre $f_q(\vec{x})$ et $f_q(\vec{y})$ pour $q \geq q^*$ ne sont pas prises en compte car nous nous arrêtons à l'indice q^* (c'est le premier indice pour lequel $f_q(\vec{x}) < f_q(\vec{y})$). Cette comparaison lexicographique des objectifs est notée par :

$$(f(\vec{x})) \leq_{lex} (f(\vec{y}))$$

Cette définition implique que l'utilisateur a rangé par ordre d'importance les différents objectifs. La comparaison entre les deux solutions se fera dans l'ordre de classement des objectifs.

I.5.2 Optimalité au sens Max-ordering (optimalité maximale)[16]

Une solution $\vec{x} \in R^n$ est max-ordering optimale si la valeur du pire(le max) objectif est aussi petite que possible, c'est à dire :

$$Max f_q(\vec{x}) \leq Max f_j(\vec{y})$$
$$q \in \{1, \ldots, n\} \quad j \in \{1, \ldots, n\} \text{ et } \vec{y} \in R^n$$

I.5.3 Optimalité lexicographique max-ordering [17]

Définition1 Pour tout élément $\vec{x} \in R^n$ $Sort(\vec{x}) = (Sort_1(\vec{x}), \ldots, Sort_n(\vec{x}))$ est défini comme étant le vecteur contenant les composantes de $\vec{x}$ dans un ordre décroissant, c'est-à-dire$Sort_1(\vec{x}) \geq \cdots \geq Sort_n(\vec{x})$.

Définition2 Une solution faisable $\vec{x} \in R^n$ est dite solution lexicographique max-ordering (Lex-MO) si le vecteur des objectifs associe à$\vec{x}$ est lexicographiquement minimal par rapport

$$Sort\ (f(\vec{x})) \leq_{lex} Sort\ (f(\vec{y}))\ \forall\ \vec{y} \in R^n$$

à$Sort\ (f(\vec{x}))$ où $(f = f_1, f_2, \dots, f_n)$qui représente des fonctions objectives. Donc

L'algorithme suivant exprime la méthode pour tirer un optimum au sens Lex-MO

Algorithme I.1 : Tirer l'optimum Lex-MO[17]

For j=1,…,N
$F_j = Sort(F_j(X))$
Min=1 ;
i=1;
while(i<N) do
Begin
if($F_j <_{lex} F_{min}$) $thenmin = i$;
$i = i + 1$;
End
Opt={aj:$F_j = F_{min}$}

Où N représente le nombre de la population.

Exemple [16]

L'exemple ci-dessous montre la relation entre la dominance Lex-MO et celle de Pareto. Nous considérons un problème dont l'ensemble réalisable de la solution est $F = \{a, b, c, d, e\}$, supposant que les valeurs de la fonction objective et les vecteurs Sort sont présentés dans le **Tableau I.1**

F	$f(F)$	$Sort(f(F))$
a	(1,3,8,2,4)	(8,4,3,2,1)
b	(4,3,8,1,1)	(8,4,3,1,1)
c	(7,5,4,6,1)	(7,6,5,4,1)
d	(3,7,4,6,5)	(7,6,5,4,3)
e	(4,7,5,6,5)	(7,6,5,5,4)

Tableau I.1 - Les valeurs des objectifs et de Sort [16]

Il est à noter que $a, b, c\ et\ d$ sont des solutions Pareto. La solution optimalelexicographiquement est évidemment a. L'ensemble de solutions de max-ordering est $c, d\ et\ e$.Alors que c est la solution unique de Lex-MO ainsi est une solution de Pareto et Max-ordering optimale.

L'avantage de cet exemple est de conclure que les solutions Lex-MO sont toujours des solutions optimales au sens de Pareto et optimales au sens max ordering[16].

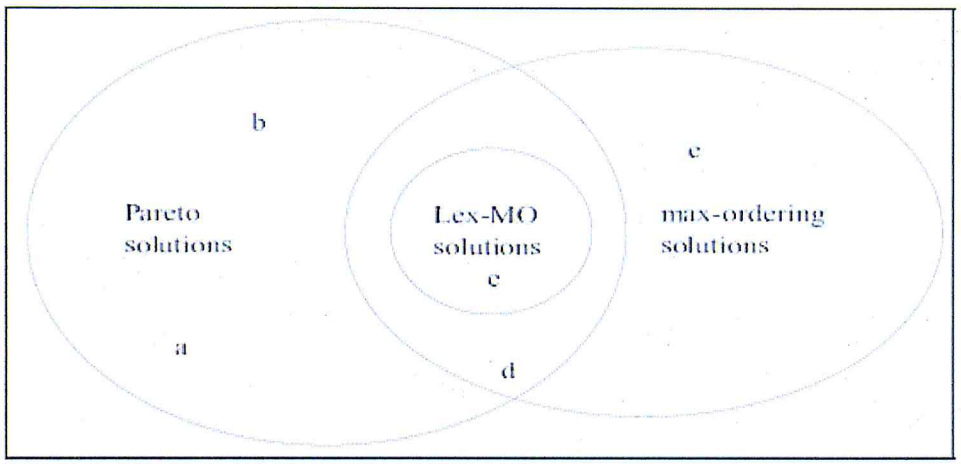


Figure I.7- Les solutions de Pareto, Lex-MO et Max-ordering[16]

Discussion

La dominance lex-MO combine à la fois les dominances de Pareto, max-ordering et lexicographique. Et d'après la **Figure1.7**, nous concluons que :

- Les solutions Pareto optimales ne sont pas toutes des solutions Lex-MO optimales.
- L'ensemble des solutions max-ordering optimales comportent ou moins une solution Pareto optimale(dans notre exemple la solution d).

-L'ensemble des solutions Lex-MO sont des solutions optimales à la fois au sens de Pareto et de max-ordering.

I .6 la ϵ-Dominance [2][3]

La ϵ-dominance est une forme de dominance décontractée et il existe deux schémas différents pour sa mise en œuvre: l'approche additive et l'approches multiplicative. Comme nos objectifs sont à minimiser, un vecteur$\vec{z} \in R^m$et$\epsilon > 0$ est donné.

Pour les schémas additif $\vec{z}$est dit ϵ-domine tous les points de l'ensemble :

$$\{\vec{v} \in R^m : z_i - \epsilon \leq v_i, Pour\ tout\ i = 1, \dots, m\}$$

Alors que pour le schéma multiplicatif, $\vec{z}$est dit ϵ-domine tous les points de l'ensemble :

$$\{\vec{v} \in R^m : z_i(1 - \epsilon) \leq v_i, Pour\ tout\ i = 1, \dots, m\}$$

Bien que les définitions ci-dessus supposent la même valeur de ϵpour tous les objectifs, ilpeut être facilement généralisé en considérant une valeur différente pour chaque objectif.

Afin de faire ce la, il suffit de prendre un ϵ_i pour chaque $i \in \{1,2,..,m\}$ sans perte de généralité, noussupposons que $1 \leq f_i \leq$ K, pour tout i.Les deux méthodes génèrent une hyper-grille dans l'espace des fonctions objectives avec$\left(\frac{K-1}{\epsilon}\right)^m$ boxes dans le système additif et$\left(\frac{-\log K}{\log(1-\epsilon)}\right)^m$ pour le multiplicatif. Comme la ϵ-dominance permet de prendre un seul point non ϵ-dominé dans chaque boxe, ces grilles peuvent accueillir un maximum de $\left(\frac{K-1}{\epsilon}\right)^{m-1}$ points non ϵ-dominé pour la méthode additive et$\left(\frac{-\log K}{\log(1-\epsilon)}\right)^{m-1}$ points non ϵ-dominé pour la méthode multiplicative.

Si T est le nombre de solutions souhaitées sur le front de Pareto pour notre problème d'optimisation, il suffit de prendre pour le régime additif :

$$\epsilon = \frac{(k-1)}{T^{\frac{1}{m-1}}}$$

Et pour le régime multiplicatif :

$$\epsilon = 1 - K^{-T^{\left(\frac{1}{1-m}\right)}}$$

Cela va générer dans les deux cas une hyper-grille avec une capacité maximale de Tpoints non ϵ-dominé.

La ϵ-dominance est un mécanisme qui permet l'approximation de surfaces (courbes) par un nombre fini de points en créant une hyper-grille dans leurs espaces. La **FigureI.8** représente le résultat de la ϵ-dominance pour la courbe$x^2 + y^2 = 1$.

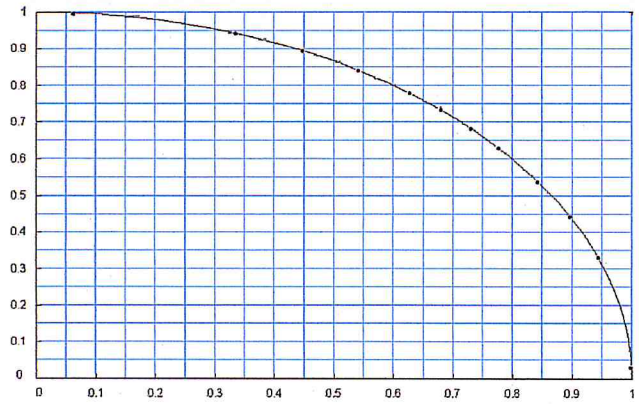


Figure I.8 - Grille uniforme avec 400 boxes (capacité maximum de 20 points)[2]

Remarque

Si $\epsilon \longrightarrow 0$, nous nous retrouvons dans les mêmes conditions de la dominance de Pareto (voir **FigureI.9**).

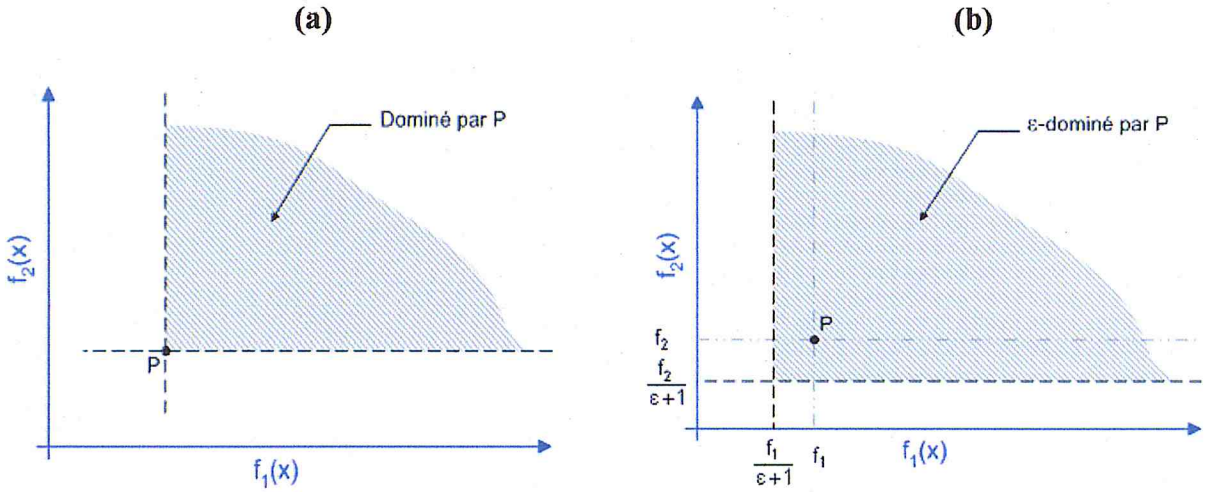


Figure I.9 – Concepts de dominance (gauche) et de ϵ-dominance (droite)[5]

I.7 Vecteur idéal et vecteur de nadir [1][11]

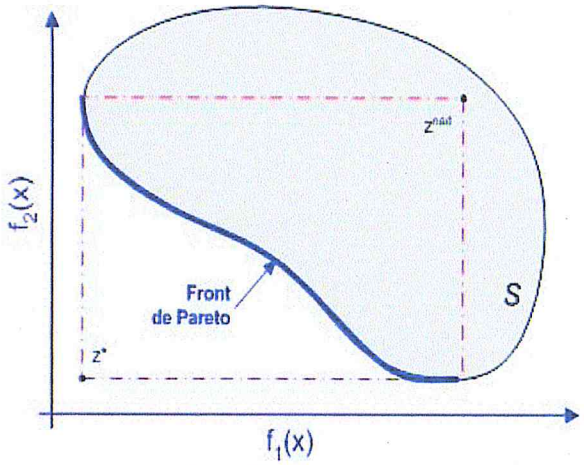
Le vecteur idéal Z^*du problème d'optimisation multi-objectif est le vecteur de l'espace des critères (objectifs) dont chaque composante Z_m est la solution du problème de minimisation de la fonction $f_m$sous les contraintes du problème d'optimisation, Les coordonnées de ce point sont obtenues en optimisant chaque fonction objective séparément. ***Le vecteur idéal est utilisé dans beaucoup de méthodes d'optimisation comme point de référence.***

A la différence du vecteur idéal qui représente les bornes inférieures de chaque objectif dans l'espace faisable, le vecteur de Nadir Z^{nad}correspond à leurs bornes supérieures sur la surface de Pareto et non pas dans tout l'espace faisable (Voir **Figure 1.10**).Les coordonnées de ce point correspondent aux pires valeurs obtenues par chaque fonction objective lorsque l'on restreint l'espace des solutions à la surface de compromis. ***Le vecteur de Nadir sert à***

restreindre l'espace de recherche. Il est utilisé dans certaines méthodes d'optimisation interactives.

Le vecteur idéal et le vecteur de Nadir sont en particulier utilisés de la façon suivante :

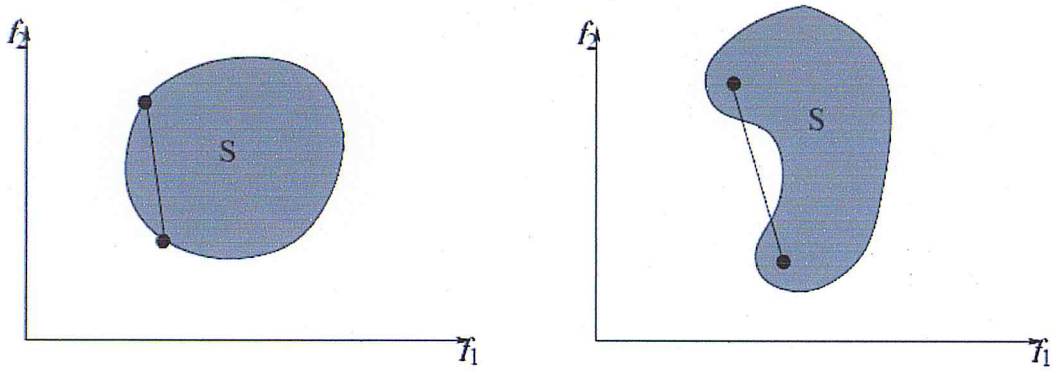
$$f_m^{norm} = \frac{f_m - Z_m^*}{Z_m^{nad} - Z_m^*}$$



FigureI.10 - Vecteur idéal Z^* et vecteur de Nadir Z^{nad}[1][11]

I.8 La convexité [1][11]

Un ensemble S est convexe si, étant données deux points distincts quelconques de cet ensemble, le segment qui relie ces deux points est continu dans l'ensemble S. L'exemple d'ensemble convexe et un exemple d'ensemble non convexe sont représentés dans la **Figure I.11**



(a) Un ensemble convexe (b) un ensemble non convexe

Figure I.11 -Exemple d'ensemble convexe et ensemble non convexe[1][11]

I.9 Conclusion :

Dans ce premier chapitre nous avons parlé de l'optimisation multi-objective et nous avons remarqué que résoudre un problème d'optimisation multi-objectif n'est pas un processus

simple. Pour cela, nous avons consacré le deuxième chapitre pour définir les algorithmes évolutionnaire et leur participation à résoudre les problèmes d'optimisation multi-objective.

Chapitre II

Evolution artificielle

II.1Introduction

Les algorithmes d'optimisation peuvent être généralement divisés en deux classes de base: les algorithmes probabilistes et les algorithmes déterministes.

Les algorithmes déterministes sont utilisés si les relations entre les caractéristiques des solutions possibles et leurs utilité sont directes et si l'espace de recherche peut être exploré efficacement pour un problème donné. Si ces relations ne sont pas si évidentes ou trop compliquées, ou si la dimension de l'espace de recherche est très élevée, il devient plus difficile de résoudre le problème avec une méthode déterministe. Essayer de le résoudre serait un résultat possible dans l'énumération exhaustive de l'espace de recherche, qui n'est pas faisable même pour des problèmes relativement faibles.

Les algorithmes stochastiques donnent une solution au bout d'un temps d'exécution plus court. Les résultats obtenus en les utilisant sont approximatifs (ils ne peuvent pas tout simplement être l'optimum global). D'autre part, une solution un peu inférieure à la meilleure possible est meilleure que celle qui nécessite un temps excessif pour être trouvée. Ces algorithmes et plus particulièrement les métaheuristiques combinent des fonctions objectives ou des heuristiques d'une manière abstraite et sans utiliser de connaissances plus approfondies dans leur structure. Cette combinaison est souvent exécutée stochastiquement en utilisant des statistiques obtenues à partir des échantillons provenant de l'espace de recherche ou fondées sur un modèle de certains phénomènes naturels ou d'un processus physique. Par exemple, le recuit simulé détermine quelle solution candidate sera évaluée prochainement. Selon le facteur de probabilité de Boltzmann, des configurations atomiques du métal fondu qui se solidifie, les algorithmes évolutionnaires, copient le comportement de l'évolution naturelle et traitent les solutions candidates en tant que personnes qui sont en concurrence dans un environnement virtuel. Dans ce chapitre, nous présentons les algorithmes évolutionnaires, leur comportement et les concepts de base.

II.2 Algorithmes évolutionnaires [5]

Les Algorithmes Evolutionnaires (AE) sont des méthodes d'optimisation permettant de s'attaquer d'une façon différente à la résolution numérique de problèmes difficiles grâce à leur robustesse et leur souplesse. C'est leur capacité à travailler sur des espaces de recherche non standards qui leur ouvre les perspectives les plus originales. Bien que simplistes du point de vue d'un biologiste, ces algorithmes sont suffisamment complexes pour fournir des mécanismes de recherche adaptatifs et robustes.

Les algorithmes d'évolution ont un mécanisme de base commun. Celui-ci consiste à faire évoluer une population par transformation aléatoire de certains de ses éléments et l'application du principe de la sélection naturelle. Plusieurs techniques ont été élaborées dont les principales sont les suivantes : **Algorithmes Génétiques, Stratégies d'évolution, Programmation évolutionnaire** et **Programmation Génétique**.

Bien que les applications des algorithmes évolutionnaires soient diverses et variées (c'est ainsi qu'ils ont donné de bons résultats dans différents domaines : optimisation de forme en mécanique des solides et en aérodynamique, recherche opérationnelle et génie industriel, automatique, ...), l'étude mathématique de ces algorithmes reste encore bien limitée face à leur complexité théorique et il a fallu attendre les années 90 pour que des démonstrations complètes et rigoureuses de convergence en probabilité soient établies. Néanmoins, ces résultats théoriques sont difficilement exploitables dans la pratique.

II.3 Principe d'un algorithme évolutionnaire [29]

Le processus d'optimisation évolutionnaire commence par l'étape de l'initialisation : un nombre fini d'individus p choisis généralement par tirage aléatoire uniforme dans l'espace de recherche D forme la population initiale P_0. Après évaluation de la population initiale (calcul de la performance), certains individus (les plus performants) sont choisis lors de l'étape de la sélection. L'application des opérateurs de variation (croisement et mutation) permet de créer un nouvel ensemble d'individus, appelé "population d'enfants" (à noter que cette étape est toujours stochastique). Ces enfants vont être évalués à leurs tour et combinés avec leur parents afin de décider lesquels d'entre eux vont remplacer certains parents pour faire partie de la génération suivante, il s'agit de l'étape de remplacement. La Figure **II.1** illustre le principe général de fonctionnement d'un algorithme évolutionnaire (AE).

Algorithmell.2 : Algorithme Evolutionnaire [5]

1. Initialisation :*Initialiser le compteur des générations t=0.

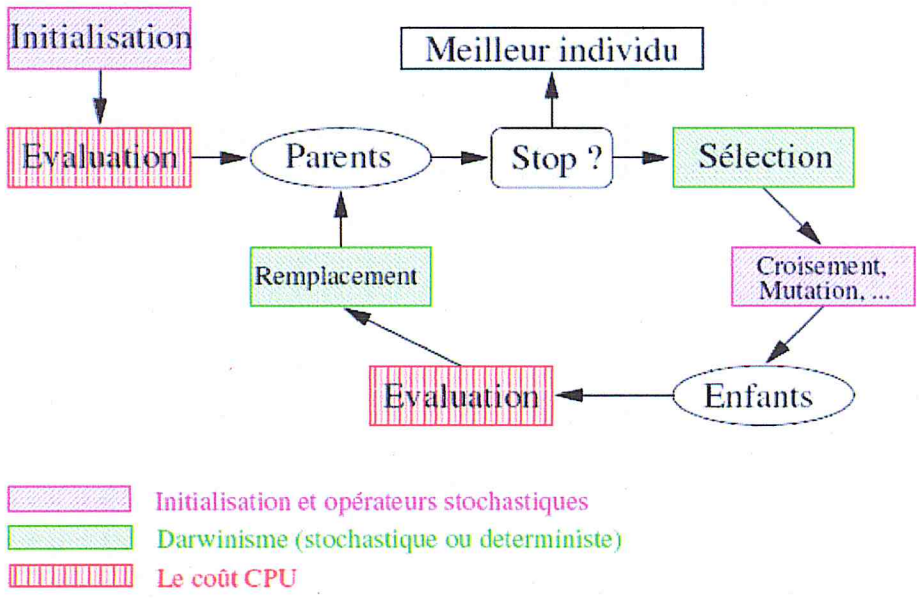
*Initialiser aléatoirement une population P(t) de taille N fixée.

*Evaluer chaque individu de P(t).

2. Sélectionner les individus les plus performants (au sens de F) de la population, les recopier pour former une nouvelle population de même taille.
3. Créer une nouvelle population en appliquant les opérateurs de variation :

*Opérateur de croisement : recombiner des parties de deux individus pour en obtenir deux nouveaux.

*Opérateur de mutation : modifier aléatoirement un individu.
4. Evaluer la fitness de chaque individu de la nouvelle population.
5. Remplacer certains individus de l'ancienne population par les meilleurs individus de la nouvelle population.
6. Si la condition d'arrêt n'est pas vérifiée, aller à l'étape 2, sinon, retourner le meilleur individu de P(t).



FigureII.1 - Cycle d'un AE[26]

Génotypevsphénotype :[26]

L'espace de recherche des AE, appelé aussi, en termes génétiques, l'espace des génotypes, peut être différent de l'espace des solutions, appelé l'espace des phénotypes sur

lequel est défini la performance. Par exemple, un génotype peut être un vecteur réel, et représenter une structure physique complexe dans l'espace des phénotypes.

La fonction de codage qui transforme un phénotype en un génotype doit être injective : à chaque phénotype correspond un seul génotype. La performance du génotype est celle du phénotype correspondant, L'initialisation de la population se fait, généralement, d'une façon aléatoire dans l'espace des génotypes. Le choix des opérateurs de croisement et de mutation dépend du codage des individus, Par contre, les opérateurs de sélection et de remplacement sont indépendants de la représentation génotypique, puisqu'ils utilisent uniquement la performance des individus.

Exploitation vs exploration [26]

A chacune des étapes de l'algorithme décrit ci-dessus, il est important de réaliser un compromis entre l'exploitation et l'exploration. Avec ces deux notions, nous introduisons un dilemme aussi important que difficile à résoudre lors de l'utilisation des AE.

Le choix de l'exploitation revient à effectuer une recherche locale dans le voisinage de meilleurs individus. L'idée est d'exploiter efficacement l'information obtenue précédemment par les meilleures solutions pour spéculer sur la position de nouveaux points avec l'espoir d'améliorer la performance.

Toutefois, l'exploitation toute seule ne permet pas de préserver la diversité génétique. La population devient alors homogène et dans ce cas, l'évolution d'une population risque de se résumer à l'évolution d'un seul individu dominant (convergence prématurée).

Le choix de l'exploration consiste à diriger la recherche de façon à préserver une population diversifiée. L'exploration de nouvelles régions de l'espace de recherche permet d'introduire dans la population de l'information innovatrice. Cependant, l'excès d'exploration conduit une quasi recherche aléatoire qui empêche la convergence.

Il faut donc maintenir un équilibre entre l'exploitation de bonnes solutions rencontrées et l'exploration des zones inconnues de l'espace de recherche S pour garantir l'efficacité de l'algorithme.

II.4 Représentation d'un individu

II .4.1 Le principe [5]

La représentation (codage) des individus doit :

- être manipulable par des opérateurs de variation (comme les mutation, croisement, sélection).
- minimiser l'épistasie (indépendance des gènes entre eux).
- assurer une transition simple et efficace vers l'espace de recherche.

II .4.2 La représentation binaire [26]

Le codage binaire est le cadre général des AE traditionnels. Chaque individu A est représenté par un vecteur binaire(ou chaine de bits), où chaque élément prend la valeur 0 ou 1. A= $(a_1, \dots, a_l) \in \{0,1\}^1$, où l est la taille de vecteur (nombre de bits).

II .4.3 La représentation réelle [26]

Avec la représentation réelle, l'espace de recherche est réel R^n(variable non bornées) ou une partie de l'espace réel $S \subseteq R^n$ (variables bornées).

Cette représentation a été introduite initialement pour les stratégies d'évolution [31], mais son utilisation s'est étendue rapidement aux autres types d'algorithmes évolutionnaires. Pour des problèmes d'optimisation dans l'espace réel, l'espace de génotype s'identifie à l'espace des phénotypes : I=$\vec{x} = (x_1, \dots, x_n) \in R^n$.

II.5 Initialisation de la population

La procédure d'initialisation est la plus naïve parmi les autres procédures. Elle consiste à générer des individus de manière aléatoire dans l'espace de recherche S en veillant éventuellement à ce que les individus produits respectent les contraintes. En codage binaire chaque chromosome prend ces valeurs dans $\{0,1\}$ avec équiprobabilité.

II.6 Darwinisme artificiel [5]

La partie darwiniste de l'algorithme évolutionnaire possède deux étapes :

- la reproduction afin de sélectionner les parents qui vont se reproduire.
- le remplacement.

II.6.1 Sélection [1] [5] [26]

La sélection est un opérateur essentiel dont le principe consiste à permettre aux meilleurs individus d'une population de se reproduire. Le réglage de ce mécanisme est déterminant dans le comportement de l'algorithme d'évolution : un excès de sélection conduit à une perte de diversité et une insuffisance peut mener à une marche aléatoire (pas de

dans la littérature un nombre important de stratégies de sélection plus ou moins adaptées au problèmequ'elles traitent. Nous présentons ici la procédure de sélection la plus fréquemment rencontrée.

*** Sélection par roulette (roulette wheelSelection)**[5]

Elle consiste à représenter sur une roulette chacun des individus de la population $P(t) = \{X_1, \dots, X_2\}$par une section qui est proportionnelle à leur fitness. Ensuite, nous lançons N fois la roulette et nous sélectionnons chacun des gagnants. Autrement dit, laprobabilité de sélectionner l'individu $X_i$de la population est:

$$\frac{F(x_i)}{\sum_{j\in\{1,..N\}} F(x_j)}$$

L'espérance $n_i$de nombre de copies d'un élément $X_i$de la population courante estdonnée par l'expression:

$$n_i = \frac{N.F(x_i)}{\sum_{j\in\{1,..N\}} F(x_j)}$$

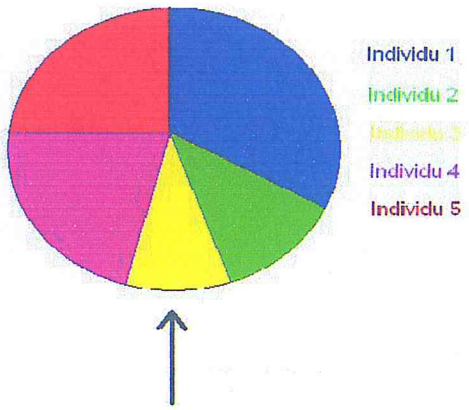


Figure II.2 -Sélection par roulette[12]

Cette méthode de sélection favorise les meilleurs individus, mais les mauvais ont toutde même des chances d'être sélectionnés. Par contre, le coût d'exécution et la variancesont élevés. La perte de diversité est aussi possible car le nombre de copies obtenuesdesmeilleurs individus (voir uniquement du meilleur) peut représenter l'ensemble dela prochaine population.

*** Sélection par rang** [8]

La sélection par rang est une variante du système de roulette. Elle s'agit de trier la population par fitness. A chaque chromosome est associé un rang en fonction de sa position, de 1 à *N* (du plus mauvais au meilleur). Ensuite, le principe est le même que pour la roulette mais les proportions sont liées au rang. Dans cette stratégie tous les individus gardent une bonne chance d'être sélectionnés (ce qui évite les convergences prématurées). Par contre, la convergence est plus lente car les bons individus se distinguent moins bien du lot.

*** Sélection par tournoi** [5]

La sélection par tournoi n'utilise aussi que des comparaisons entre les individus et ne nécessite même pas de tri de la population. Elle possède un paramètre T, qui est la taille de tournoi. Pour sélectionner un individu, nous tirons T uniformément de la population et nous sélectionnons d'une manière déterministe le meilleur de ces T individus. Au cours d'une génération, il y a autant de tournois que d'individus à sélectionner. Cette méthode est caractérisée par une pression de sélection en général plus forte que les méthodes proportionnelles (pour qu'un individu peu performant puisse être sélectionné. Il faut que ses adversaires soient encore moins bons que ce dernier). De plus, elle est la moins chère en termes de coût d'exécution et facilement paramétrable par la valeur de T.

II.6.2 Remplacement [5][26]

Diverses stratégies de remplacement peuvent être utilisées. Le principe consiste à remplacer l'ancienne population par une nouvelle, obtenue après application d'opérateurs de variation. Dans les algorithmes génétiques standards, le remplacement est générationnel, c'est à dire que la population des enfants remplace purement et simplement la population parente. Néanmoins, il existe d'autres stratégies de remplacement dont :

- **Le remplacement d'un pourcentage** des individus de l'ancienne génération par les meilleurs enfants.
- **Le remplacement systématique du plus mauvais individu.**
- **Le remplacement aléatoire** (en faisant attention à minimiser une stratégie de recherche cohérente).
- **Le remplacement déterministe.**

Le remplacement déterministe [26] est utilisé typiquement dans les stratégies d'évolution. Son caractère purement déterministe lui donne un rôle clef dans l'évolution car il guide la recherche vers les zones des meilleurs individus. Il opère en sélectionnent les μ, $(1 < \mu \leq \gamma)$, meilleures solutions parmi :

- L'union de μ parents et γ enfants : schéma appelé $(\mu + \gamma)$-ES.
- L'ensemble de γ enfants : schéma appelé (μ, γ)-ES.

Le remplacement $(\mu + \gamma)$ élitiste, garantit une amélioration monotone de performance de la population, mais il s'adapte mal à un éventuel changement d'environnement. Par contre, avec un remplacement (μ, γ), nous pouvons perdre les meilleurs individus, mais l'algorithme est plus flexible avec les changements d'environnement.

II.7 Opérateurs de variation

Les opérateurs de variation sont les opérateurs de mutation et de croisement qui ont pour but de produire de nouvelles générations d'individus à partir de ceux qui ont été sélectionnés.

II.7.1 Le croisement [5]

Le croisement est analogue à une reproduction sexuée en s'appuyant sur le principe que les enfants héritent des qualités de leurs parents. La forme standard de l'opérateur de croisement est $C : E * E \rightarrow E * E$, qui croise avec une certaine probabilité $P_c (O \leq P_c \leq 1)$, deux parents $(p_1, p_2) \in E * E$. D'autres formes de croisement sont possibles comme celle où un seul enfant est produit par plusieurs parents. Parmi les différents types majeurs de croisement, il y a:

(*)Le croisement binaire

Croisement 1-point :

C'est le croisement le plus simple et le plus classique dans les algorithmes génétiques. Il consiste à sélectionner aléatoirement un point de coupure dans chacun des deux parents p_1 et p_2 et construire deux nouveaux individus (enfants) e_1 et e_2 en échangeant leurs gènes de part et d'autre de ce point.

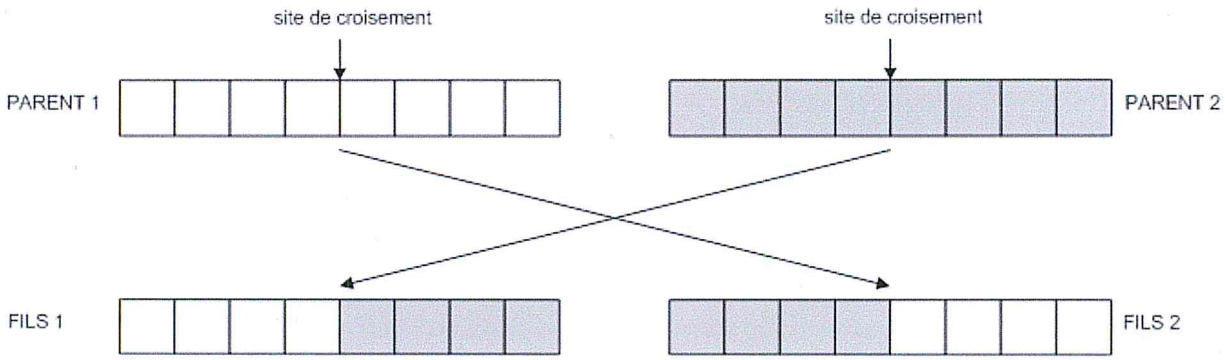


Figure II.3 - Croisement binaire à un point [5]

Le fait de choisir un seul point de croisement biaise l'effet de cross-over. S'il est choisi proche d'une extrémité du chromosome, les enfants seront presque identiques aux parents et s'il est choisi au milieu, ils en seront très différents.

Croisement multi-points [5]

Le croisement multi-points évite ce problème en considérant les chromosomes comme circulaires plutôt que linéaires et en les choisissant en k points de coupure. La figure II.4, présente un exemple de croisement multipoints avec k=3.

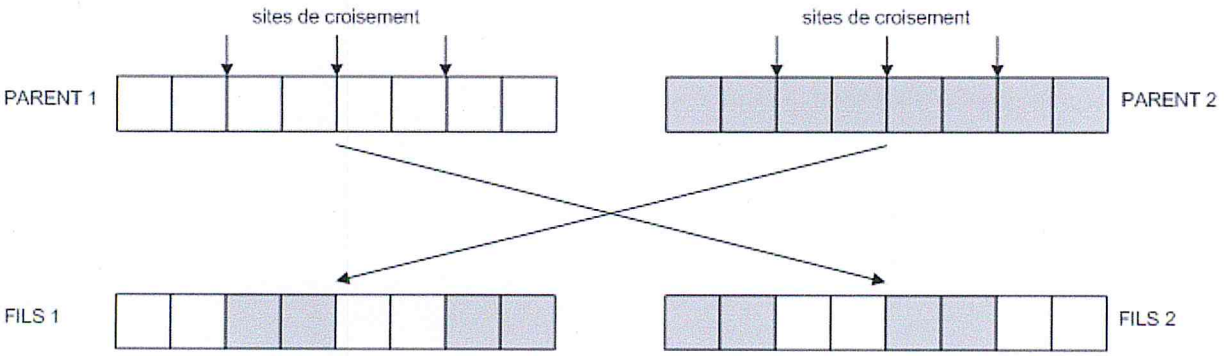


Figure II.4 - Croisement binaire à trois points [5]

Croisement uniforme

Le croisement uniforme utilise un masque binaire généré aléatoirement, de la même taille que les chromosomes pour indiquer à chaque locus le parent qui fournira le gène (voir **Figure II.5**).

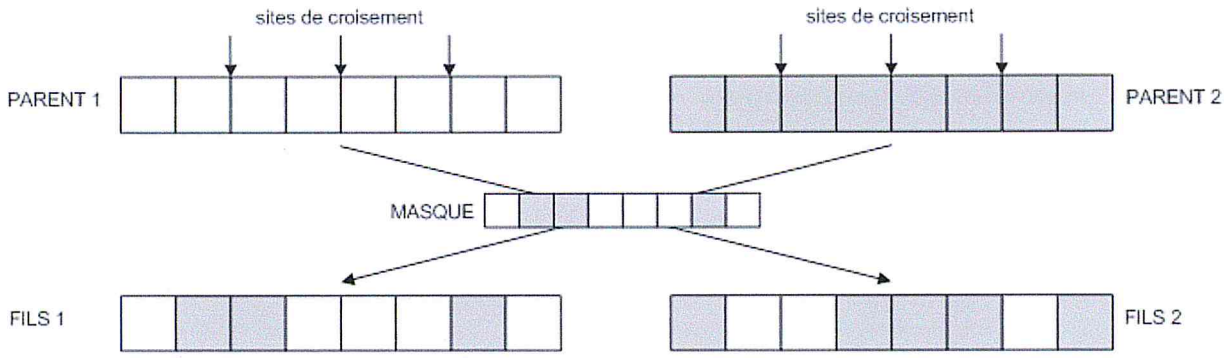


Figure II.5 - Le croisement uniforme [5]

D'autres opérateurs de croisement existent. Ils peuvent soit amener des modifications à ceux présentés ci-avant, soit être spécifiques à une classe de problèmes, mais ils obéissent néanmoins à un principe commun, l'échange d'information entre individus.

(*)Le croisement réel

Le croisement standard [5]

Le croisement réel standard est très proche de celui décrit pour le codage binaire dans la section précédente. Il ne se différencie du croisement binaire que par la nature des éléments qu'il altère. Bien évidemment, ce ne sont plus des bits qui sont échangés de part et d'autre du point de croisement mais des valeurs réelles.

Le croisement binaire simulé (SBX) [9]

Le croisement binaire simulé (*Simulated Binary Crossover)* reproduit les mécanismes du croisement standard à un point utilisé, lorsque les variables objets sont représentées sous la forme de chaînes binaires. A partir de deux parents $p_{1(i)}$et $p_{2(i)}$, ce croisement génère deux enfants $c_{1(i)}$et $c_{2(i)}$ par l'intermédiaire de la relation suivante :

$$\begin{Bmatrix} c_{1(i)} = 0{,}5\left[(1+B)p_{1(i)} + (1-B)p_{2(i)}\right] \\ c_{2(i)} = 0{,}5\left[(1-B)p_{1(i)} + (1+B)p_{2(i)}\right] \end{Bmatrix}$$

Où B représente un facteur de dispersion défini par :

$$B = \begin{cases} (2u)^{\frac{1}{\mu+1}} \; Si \; u < 0{,}5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\mu+1}} \; ailleurs \end{cases}$$

Où u est une variable aléatoire uniformément répartie dans l'intervalle $[O, 1]$ et μ un paramètre réel non négatif qui caractérise la forme de la distribution des enfants par rapport aux parents. De fortes valeurs de μ engendrent de fortes probabilités de retrouver un enfant dans le voisinage local des parents. Tout comme les stratégies d'évolution, le croisement SBX possède des propriétés intéressantes d'auto-adaptation.

II.7.2 la mutation [5]

L'idée générale de la mutation est la modification (avec une certaine probabilité P_m, $0 \leq P_m \leq 1$) d'un ou plusieurs gènes de l'individu sélectionné, afin d'introduire de la variabilité dans la population. Cet opérateur agit de E dans E. Il peut :

- favoriser l'exploitation (si l'individu muté est proche de l'individu original).
- favoriser l'exploration (si l'individu muté est éloigné de l'individu original).

La mutation apporte aux algorithmes évolutionnaires la propriété d'ergodicité de parcours d'espace et la réintroduction de la diversité perdue. Parmi les différents types majeurs de mutation, il y a:

(*) La mutation binaire : La mutation binaire est une modification aléatoire de la valeur d'un gène qui se produit avec une probabilité fixée P_m par individu. Les mutations les plus utilisées sont :

1-bit mutation : elle consiste à choisir une position uniformément dans un individu et changer la valeur du bit correspondant (voir **Figure. II.6**).

$\frac{c}{l}$ **mutation**: Il s'agit de changer la valeur du bit de chaque position indépendamment avec une probabilité $\frac{c}{l}$, où l est la taille de l'individu (nombre de bits) et $c > 0$.

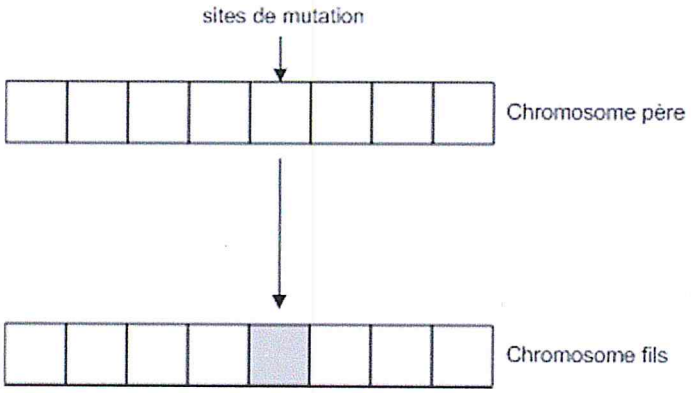


Figure II.6 - Mutation binaire [5]

(*)La mutation réelle

La mutation Polynomiale [32]

La mutation génère un enfant à partir d'un parent selon l'équation 1. δ_k, elle-même est donnée par les équations 2 ou 3 selon que le nombre aléatoire r_k ($0 \leq r_k \leq 1$) est respectivement inférieur ou supérieur à 0,5. η_m, qui est l'indice de distribution pour la mutation. Les termes $parent_k^{sup}$ et $parent_k^{inf}$ sont respectivement les limites supérieures et inférieures des intervalles de variation associés au paramètre d'indice k.

$$enfant_k = parent_k + (parent_k^{sup} - parent_k^{inf})\ \delta_k \text{.....Equation 1.}$$

$$\delta_k = (2r_k)^{\frac{1}{\eta_m+1}} - 1, si\ r_k < 0{,}5 \text{ ..... Equation 2.}$$

$$\delta_k = 1 - [2(1-r_k)]^{\frac{1}{\eta_m+1}}, si\ r_k \geq 0{,}5 \text{.....Equation 3.}$$

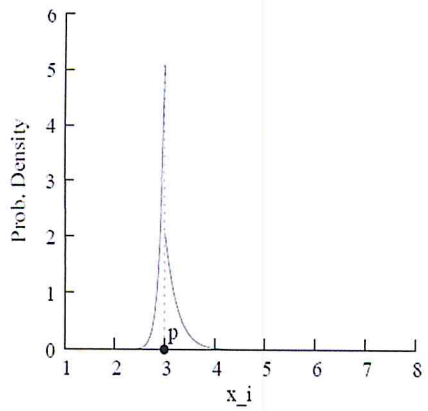


Figure II.7 - Fonction de densité de probabilité de création d'un enfant en utilisant l'opérateur de mutation polynomiale [10]

II.8 Algorithmes évolutionnaires multi-objectifs

Depuis une trentaine d'années, le domaine de l'optimisation évolutionnaire multi-objectif connaît une évolution importante. A cet effet, nous distinguons deux groupes d'algorithmes, ceux du premier groupe sont dits *non Pareto* par rapport à leur principe de fonctionnement et ils sont maintenant dépassés. Ceux du second groupe sont dits *de Pareto* et ils sont très utilisés actuellement. Parmi les algorithmes du dernier groupe, nous avons choisi de présenter les algorithmes suivants :

II.8.1 (NSGA- II) Elitist Nondominated Sorting Genetic Algorithm [5]

Dans cet algorithme, le double objectif convergence-diversité est atteint, d'une part, par l'utilisation d'un schéma du calcul de la performance qui préfère les solutions non-dominées et d'autre part, par l'application d'une technique de mesure de densité des solutions du même front non-dominé [15]. Dans le NSGA-II, la population des enfants $Q_t$est d'abord créée à partir de la population des parents P_t. Ensuite, elles sont réunies en ensemble $R_t = P_t \cup Q_t$, qui est trié selon le principe de dominance : l'ensemble R est divisé en un nombre de classes distinctes R_j selon la façon suivante : tous les individus non-dominés de R appartiennent à l'ensemble R_1. Tous les éléments non-dominés de R/R_1 sont placés dans l'ensemble R_2 et ainsi de suite jusqu'à trier toute la population. Ainsi nous obtenons r sous-ensembles tels que:

$$R = \coprod_{j=1}^{r} R_j \text{ et } R_1 \prec R_2 \prec \cdots \prec R_r$$

Notons qu'entre deux solutions de la même classe, aucune ne peut être considérée meilleure que l'autre par rapport à tous les objectifs du problème. Le nombre total de classes, noté r, dépend de la population R.

Quand toute la population est triée, La population suivante P(t+1) est remplie par les solutions des sous-ensembles non-dominé de R(t) l'un après l'autre en commençant, évidement, par le premier front. Pour choisir les solutions qui vont survivre du front dont seulement une partie peut être placée dans la population suivante, une mesure de la densité des solutions dans l'espace de critères, dite distance de surpeuplement, (*crowding distance*, en anglais), est utilisée (voir **Figure II.9**).

La **Figure II.8** exprime le principe de fonctionnement de cet algorithme :

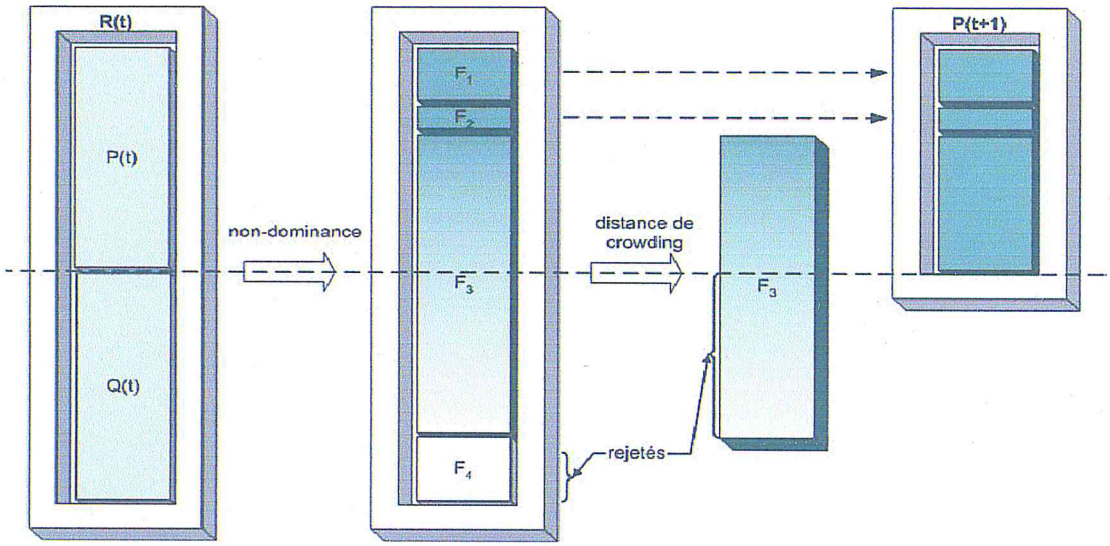


Figure II.8 - Principe de NSGA-II [5]

Algorithmell.3 : Une itération de NSGA-II [5]

1. Créer la population des enfantsQ_tà partir de la population des parentsP_t.
Q_t= générer (P_t).
2. Réunir les populations des parents et des enfants R_t= P_t∪$Q_t$et trier l'ensemble qui en résulte en sous-ensembles $F_i$tels que :

$$R_t = \coprod_{i=1}^{r} F_i \qquad et\ F_1 < F_2 < \cdots < F_r$$

3. Initialiser :
– la nouvelle population $P_{t+1} = \{\}$
– le compteur des sous-ensembles non-dominés i = 1.

4. Tant que $|P_{t+1}|+|F_i|<N$ faire :$P_{t+1} \leftarrow P_{t+1} \cup F_i$
$i \leftarrow i + 1$

5.Ordonner l'ensemble F_i selon les distances du surpeuplement (crowding-distance) et inclure les $N - |P_{t+1}|$solutions ayant les valeurs de distances les plus grandes dans la populationP_{t+1}.

Pour le calcul de la distance de surpeuplement, nous utilisons l'algorithme qui suit :

Algorithmell.4: Calcul des distances de surpeuplement [5]

1. Initialiser :
– le nombre d'individus de$F$.
– les distances de surpeuplement $d_i = 0$ pour toute solution i de F.
– le compteur d'objectifs m = 1.

2. Réordonner l'ensemble F de façon que les valeurs def_m sur ses éléments diminuent.
Notons $I^m = sort_{[f_m,>]}(F)$ le vecteur des indices, c'est-à-dire I_i^m dénote l'indice de la solution i dans la liste ordonnée selon l'objectif m.
3. Mettre à jour la valeur de d_i
– Pour chaque solution i telle que $2 \leq I_i^m \leq (l-1)$,

$$d_i \leftarrow d_i + \frac{f_m^{I_i^m+1} - f_m^{I_i^m-1}}{f_m^{max} - f_m^{min}}$$

– sur les extrémités de F,autrement dit si $I_i^m = 1\ ou\ I_i^m = l$, $d_i = \infty$.
4. Si m = M, la procédure est terminée, sinon incrémenter le compteur d'objectifs $m \leftarrow m + 1$ et retourner à l'étape 2

La distance de surpeuplement, d_i correspond au semi-périmètre du cuboïde dont les vertexes sont les voisins les plus proches de l'individu i (voir **Figure II.9).**

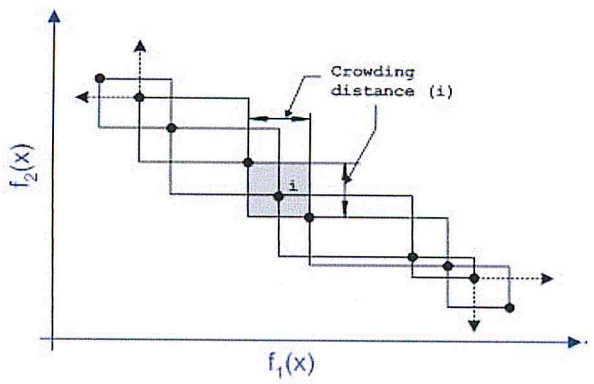


Figure II.9 - Principe de crowding [5]

Complexité de NSGA-II [26]

L'étape 1 de NSGA-II consiste à trier la population de taille $2N$ en sous-ensembles non dominés, ce qui exige $O\ (MN^2)$ opérations. L'étape 4 utilise les valeurs de distance de surpeuplement de toutes les N solutions de la population P_t, dont le calcul coûte $O\ (MNlogN)$. La complexité totale de NSGA-II est alors $O\ (MN^2)$, où M est le nombre d'objectifs et N la taille de la population.

Avantage [26]

Les stratégies de préservation de la diversité employée dans NSGA-II n'exigent aucun paramètre à fixer. Notons que rien n'empêche de calculer les distances de surpeuplement dans l'espace de décision si cela est considéré comme plus adapté au problème.

Inconvénient [26]

Dans certaines générations, toute la population est continue dans le premier front non-dominé. A ce stade, des solutions Pareto optimales situées dans une région très « peuplée » de l'espace de recherche, peuvent être éliminées en laissant la place à des solutions non dominées dans la population courante et qui peuvent être non Pareto optimales. Le NSGA- II exige le regroupement de la population de taille 2N.

II.8.2 (M.O.G.A.) La méthode Multiple Objective Génetic Algorithm [1]

Ici, le "rang" d'un individu (numéro d'ordre qui permet de classer un individu par rapport aux autres) est donné par le nombre d'individus qui dominent l'individu considéré. Par exemple, si l'on considère un individu x_i à la génération t qui est dominé par $p_i^{(t)}$ individus, le rang de l'individu considéré est donné par :

$$rang(x_i, t) = 1 + p_i^{(t)}$$

A tous les individus non dominés nous affectons le rang 1. Les individus dominés se voient donc affectés à un rang important (une forte pénalité donc). Pour le calcul de l'efficacité, nous pouvons suivre les étapes suivantes :

1. Classer les individus en fonction de leur rang.
2. Affecter une efficacité à un individu en interpolant à partir du meilleur (rang 1) jusqu'au plus mauvais (rang *n*), en utilisant une fonction *f* (*rang*). Cette fonction est le plus souvent linéaire (voir **Figure II.10**).

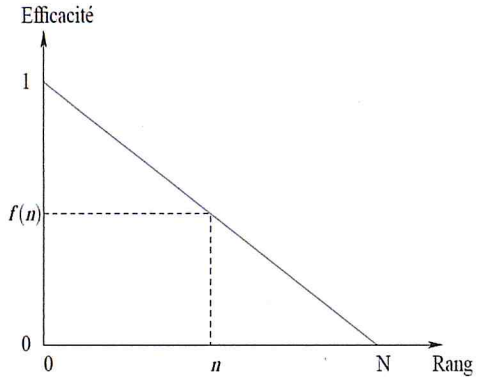


Figure II.10 - Exemple de fonction d'efficacité [1]

Le pseudo-code de cette méthode est représenté dans **l'algorithmeII.5**.

AlgorithmeII.5 : Algorithme MOGA [1]

```
Initialisation de la population
Evaluation des fonctions objectives
Assignation d'un rang basé sur la dominance
Assignation d'une efficacité à partir du rang
Pour i=1 jusqu'à G
Sélection aléatoire proportionnelle à
l'efficacité
Croisement
Mutation
Evaluation des fonctions objectives
Assignation d'un rang basé sur la dominance
Assignation d'une efficacité à partir du rang
FinPour.
```

Discussion [1]

L'inconvénient de cette méthode est qu'elle ne permet pas dans certains cas, d'obtenir une diversité dans la représentation des solutions. Par contre, la méthode NSGA II, permet d'assurer une bonne présentation et traite cette difficulté.

Pour ce type de méthode, le nombre de comparaisons effectuées par génération s'écrit :

$$N.(N-1).M$$

Où N correspond au nombre de points dans la population et M au nombre de fonctions objectives.

Alors, la complexité totale de MOGA est O(MN^2).

II.9 Conclusion

Nous concluons en fin de ce chapitre en donnant l'avantage des AE. Elles permettent d'atteindre les solutions les plus performantes à partir d'une population initiale créée aléatoirement en passant par plusieurs populations engendrées grâce aux opérations de variation (mutation, croisement). Ces solutions expriment les optima de la fonction objective.

Dans le prochain chapitre, nous détaillerons un autre algorithme basé sur la dominance de Pareto qui est ϵ -MOEA, puis nous proposons un nouvel algorithme nommé ϵ-MOEA-LMO, qui peut être considéré comme une autre version de ϵ-MOEA.

Chapitre III

AEMO ET Problème test

III.1 Introduction

Ce chapitre est composé de deux parties, la première partie est consacrée à la présentation de l'algorithme évolutionnaire multi-objectif dit ϵ-MOEA, sur lequel est porté notre intérêt. Par la suite, une modification est apportée à ce dernier en faisant intervenir la dominance Lexico-Max-Ordering à la place d'un mécanisme combinant la dominance de Pareto et la distance euclidienne. Cet algorithme nous l'avons baptisé ϵ-MOEA-LMO par rapport à la technique utilisée. Afin de procéder à des expérimentations numériques (chapitre IV) permettant l'évaluation de notre algorithme, nous présentons dans la deuxième partie de ce chapitre les problèmes tests et les métriques nécessaires pour comparer les deux algorithmes.

III.2 ϵ-MOEA : ϵ-Domination based Multi-Objective Evolutionnary Algorithm [5][30]

Cet algorithme appartient à l'ensemble des algorithmes qui sont basés sur la dominance de Pareto. Son principe est de diviser l'espace de recherche en un certain nombre de boxes et chaque boxe doit contenir au plus une seule solution. L'algorithme commence par la création d'une population initiale P_t, $(o\grave{u}\ t = 0)$ aléatoirement, cette dernière (P_0 ou une partie de P_0) est utilisée pour remplir l'archive A_t selon le principe de dominance. A chaque itération $t+1, (o\grave{u}\ t \geq 0)$, nous choisissons deux individus aléatoirement, l'un à partir de la population P_t et l'autre à partir de l'archive A_t, pour générer de nouvelles solutions en appliquant les opérateurs de variations (le croisement binaire simulé (SBX) et la mutation polynomiale). Par la suite, nous comparons chaque solution c_i engendrée par rapport à la population P_t selon la dominance et par rapport à l'archive A_t selon la ϵ-dominance, si c_i domine un ou plusieurs solutions de P_t , c_i remplace une de ces solutions aléatoirement, si c_i est dominée par au moins une solution de P_t, elle sera éliminée, si les deux premiers tests échouent c_i remplace une solution choisie aléatoirement, de manière bien précise. **L'algorithme III.1** expose les différentes étapes de l'ϵ-MOEA:

Algorithme III.1 ϵ-MOEA [5]

1. Initialiser une population P_0 et initialiser un compteur d'itération t = 0.
2. copier les individus non-dominés de P_0 dans un ensemble d'archivageA_0.
3. une solution est choisie de P_t
$$p = pop\_selection(P_t)$$
4. Une solution est choisie deA_t.
$$e = archive\_selection(A_t).$$
5. générer un individu à partir de p et e
$$c = générer\ (p, e)$$
6. Inclure c dans P_t
$$pop\_acceptance(P_t, c)$$
7. Inclure c dans A_t
$$archive\_acceptance(A_t, c)$$
8. Si la condition d'arrêt est satisfaite, retourner A_t, sinon, aller à 3.

Par la suite, nous exposerons les procédures suivantes : Pop_sélection, Archive_sélection, Pop_acceptance et Archive_acceptance).

Algorithme III.1.1 Pop sélection [5]

1. Deux individus sont choisis aléatoirement de P_t
$$p_1 = random\_pick\_up(P_t)$$
$$p_2 = random\_pick\_up(P_t)$$
2. Comparer $p_1$et p_2 et retourner l'individu choisi :
$$if(p_1 \succ p_2) return\ p_1$$
$$else\ if\ (p_2 \succ p_1) return\ p_2$$
$$else\ return\ random\_pick\_up(p_1, p_2)$$

Archive_sélection : Pour choisir une solution de A_t, plusieurs stratégies peuvent être suivies. Dans ce qui suit, nous choisissons cette solution aléatoirement de manière uniforme [5].

Algorithme III.1.2 Pop_acceptance[5]

1. comparer c avec tous les individus de P_t. Soit D_t l'ensemble des individus de P_t dominés par c et $\widehat{D}_t$, l'ensemble des individus de P_t dominant c :

$$D_t = \{p \in P_t | c \succ p\}$$
$$\widehat{D}_t = \{p \in P_t | p \succ c\}$$

2. Si l'ensemble des individus dominés par c, D_t, n'est pas vide, remplacer l'un d'eux (choisi aléatoirement) par c.

$$replace(P_t, \text{random\_pick\_up } (D_t), c)$$

3. Si l'ensemble des individus dominant c, $\widehat{D}_t$, n'est pas vide, c est rejeté.
4. Si D_t et $\widehat{D}_t$ sont vides, c remplace un individu de P_t choisi aléatoirement.

$$replace(P_t, \text{random\_pick\_up}(P_t), c)$$

Archive_acceptance [5] [30] : Pour toute solution de l'archive A_t, nous associons un vecteur d'identification $\vec{B} = (B_1, B_2, \dots, B_M)^T$, où M est le nombre d'objectifs, comme suit:

$$B_j(f) = \left\lfloor \frac{f_j - f_j^{min}}{\epsilon_j} \right\rfloor$$

où $\lfloor . \rfloor$ désigne la valeur entière, f_j^{min} le minimum possible du $j^{ème}$ objectif et ϵ_j l'erreur tolérée pour le $j^{ème}$ objectif.

Le vecteur d'identification B divise l'espace des critères en cuboïdes (boxes), de volumes $\prod_{i=1}^{M} \epsilon_i$, comme le montre la **Figure III.1.** En effet dans cette figure, le vecteur d'identification de P n'est autre que les coordonnées du point A dans l'espace des critères. Nous dirons que P domine la région PECFP et ϵ -domine la région ABCDA.

Le vecteur d'identification permet de conserver la diversité de l'archive A_t: chaque cuboïde contient au plus, qu'un seul individu de A(t). Ainsi, la taille de l'archive A_t est bornée, sa valeur maximale dépend de la valeur de ϵ :

$$|A| \leq \min_{m \in \{1,..,M\}} \frac{f_m^{max} - f_m^{min}}{\epsilon_m}$$

où f_m^{max} et f_m^{min} désignent respectivement le maximum et le minimum possibles du $m^{ème}$ objectif.

Pour intégrer la solution c dans l'archive A_t, cette solution est comparée avec tous les individus de A_t au sens de ϵ-dominance.

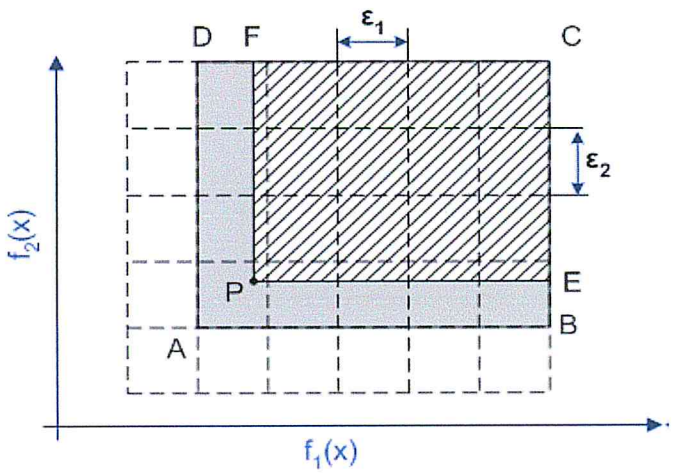


Figure III.1 - Concept de vecteur d'identification [5]

Algorithme III.1.3 Archive_acceptance [5]

1. calculer le vecteur d'identification de l'individu c et de toutes les solutions de l'archive A_t,

$$Calculer\ B_c\ et\ B_a \forall a \in A_t$$

2. S'il existe une solution a de A_t tel que $B_a \succ B_c$, alors, c n'est pas accepté.
3. S'il existe une solution a de A_t tel que $B_c \succ B_a$, alors, c remplace a dans A_t.
4. Si aucune des précédentes conditions n'est vérifiée, deux cas se présentent :

(a) S'il existe une solution a de A_t tel que $B_a = B_c$, c'est à dire les deux individus se trouvent dans le même hyper-box alors :

- Si $a \sim c$, retenir la solution ayant la plus petite distance par rapport au vecteur d'identification B.
- Sinon, retenir la solution qui domine l'autre.

(b) Sinon, intégrer l'élément c dans l'archive A(t).

La **Figure III.2** représente les différentes étapes de l'algorithme ϵ-MOEA

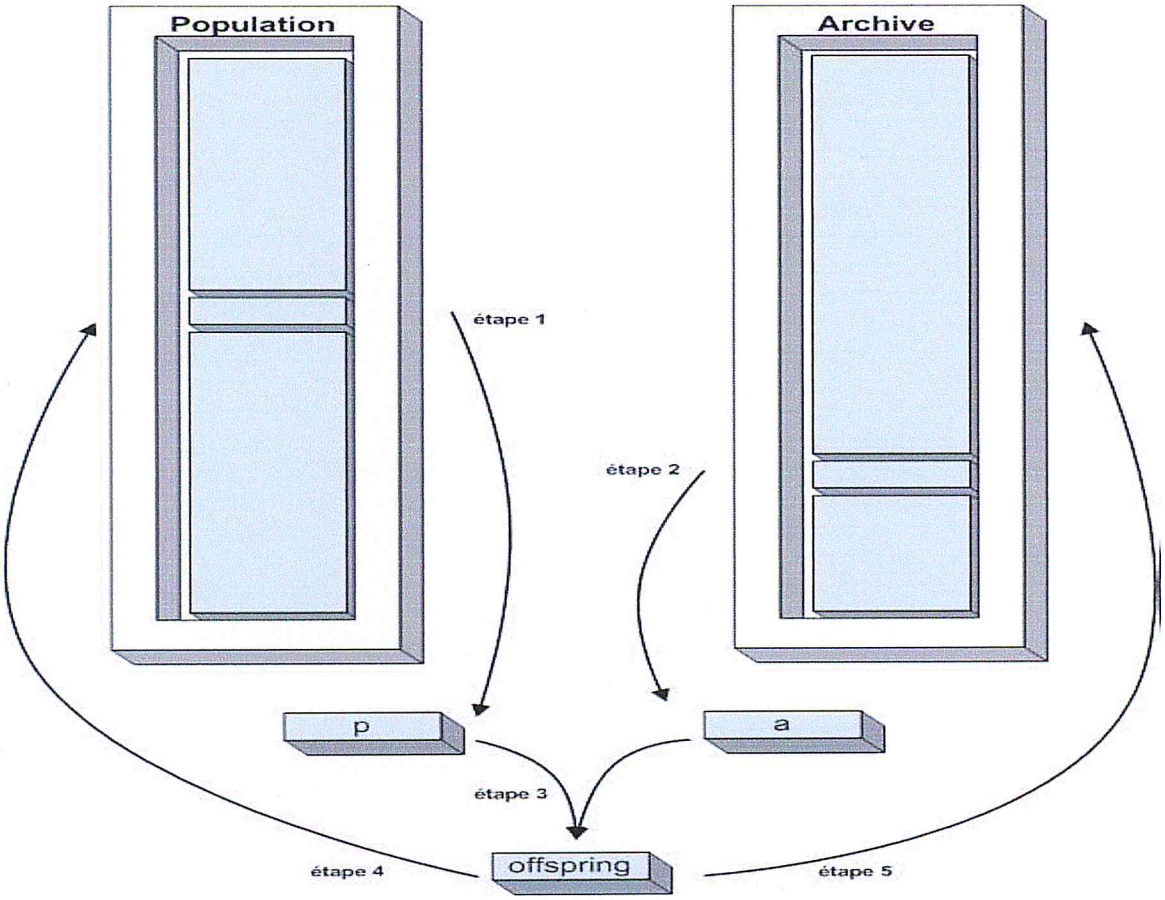


Figure III.2 - Principe de fonctionnement de ϵ-MOEA [5]

Avantage [5]

L'exécution de cet algorithme montre une efficacité remarquable par rapport à **NSGA-II** et relativement,

- au temps d'exécution,
- à la convergence vers le front Pareto,
- à la diversité des solutions sur le front de Pareto.

III.3 ϵ-MOEA-LMO: ϵ-Domination based Multi-Objective Evolutionnary Algorithm Lexicographique Max Ordering

Cet algorithme est une version modifiée de l'algorithme ϵ-MOEA. Les procédures pop sélection, archive sélection et pop acceptance sont pareilles à celles détaillées lors de la description de l'algorithme ϵ-MOEA (voir la section III.1). La différence se situe dans la procédure « Archive_acceptance », qui consiste à remplacer la dominance usuelle combinée à la distance euclidienne par la dominance lexico_Max_Ordering.

Algorithme III.2 LMO_archive_acceptance

1. calculer le vecteur d'identification de l'individu c et de toutes solutions de l'archive A_t.

$$Calculer\ B_c\ et\ B_a \forall a \in A_t$$

2. S'il existe une solution a de A_t tel que $B_a \succ B_c$, alors, c n'est pas accepté.
3. S'il existe une solution a de A_t tel que $B_c \succ B_a$, alors, c remplace a dans A_t.
4. Si aucune des précédentes conditions n'est vérifiée, deux cas se présentent :

(a) S'il existe une solution a de A_t tel que $B_a = B_c$, c'est à dire les deux individus se placent dans le même hyper-box alors :

garder la solution lexico_Max_Ordering dominante.

(b) Sinon, intégrer l'élément c dans l'archive A_t.

III.4 Problèmes tests [1][26][23]

Dans l'optimisation évolutionnaire multi-objectif et afin de pouvoir évaluer les différentes méthodes de recherche (algorithmes), les chercheurs de ce domaine ont introduit les problèmes tests. Chacun de ces problèmes permet de tester le comportement de l'algorithme face à une difficulté bien particulière comme : la convexité ou la non-convexité dans le front Pareto-optimal, les discontinuités dans la surface optimale de compromis et la multi-modalité des fonctions objectives.

Dans la littérature, plusieurs problèmes tests ont été proposés lors des études sur les AEMO.

Dans les premiers travaux, chaque apparition d'une nouvelle approche était l'occasion pour les auteurs d'introduire leurs propres problèmes tests pour la validation de leur technique.

Parmi ces travaux, nous pouvons noter, par exemple celui de Deb [13]. Avant de passer en revue les fonctions tests de l'optimisation multi-objectives, nous devons rappeler qu'une représentation de la surface de compromis est satisfaisante si la répartition des points sur celle-ci est uniforme.

Dans ce cas, si la solution n'est pas satisfaisante, une autre pourra être choisie dans son voisinage. Du fait de l'uniformité de la représentation des solutions, la variation des valeurs des fonctions objectives ne sera pas brusque lors du passage d'une solution à une voisine (voir **Figure I.9**).

Une fonction test est donc une fonction destinée à mettre en difficulté la méthode d'optimisation multi-objective, dans la recherche d'une répartition uniforme des points solutions sur la surface de compromis. Nous distinguons deux familles de difficultés :

- celles qui empêchent la méthode de converger vers la surface de compromis.
- celles qui empêchent d'aboutir à une répartition uniforme des solutions sur la surface de compromis.

Dans la première famille, nous trouvons la multi-frontalité de la surface de compromis.

Dans la seconde famille, nous trouvons les difficultés suivantes :

- la non convexité,
- la discontinuité,
- la non uniformité de la surface de compromis.

Deb [13] a proposé une série de fonctions de tests basées sur le problème bi-objectif "générique" suivant:

$$\left|\begin{array}{c} minimiser\ f_1(\vec{x}) = f(x_1, \dots, x_m) \\ minimiser\ f_2(\vec{x}) = g(x_{m+1}, \dots, x_N).h(f(x_1, \dots, x_m), g(x_{m+1}, \dots, x_N)) \\ avec\ \vec{g}(\vec{x}) \leq 0 \\ et \quad \vec{h}(\vec{x}) = 0 \end{array}\right.$$

Dans ce problème générique, chaque fonction joue un rôle particulier :

f: contrôle l'uniformité de la répartition des solutions le long de la surface de compromis.

g: contrôle la multi-frontalité de la surface de compromis. Elle permet aussi de créer un optimum isolé.

h: permet de contrôler l'aspect de la surface de compromis (convexe, non convexe, etc.).

Nous présentons à la suite les cinq tests de **ZDT1** à **ZDT6**. Nous n'avons pas cité le problème **ZDT5**, qui est formulé pour les variables binaires.

III.4.1 ZDT1

Le premier de cet ensemble de tests est le plus simple. Le front de Pareto correspondant est continu, convexe et avec la distribution uniforme des solutions le long du front.

$$\text{ZDT1}: \begin{cases} f_1(x) = x_1 \\ g(x_2) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(x) = 1 - \sqrt{f_1/g} \end{cases}$$

où $x_i \in [0,1]$ pour tout i $= 1,\ldots,$n et n $= 30$. Le front de Pareto de ce problème est présenté dans la **Figure III.3**.

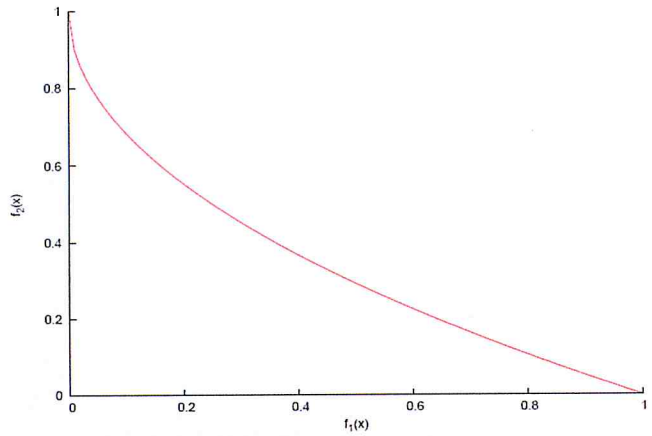


Figure III.3 - Solution de ZDT1 [5]

III.4.2ZDT2

Le ZDT2 pose une difficulté qui est la non-convexité du front de Pareto et elle est donnée par :

$$ZDT2 = \begin{cases} f_1(x) = x_1 \\ g(x_2) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(x) = 1 - (f_1/g)^2 \end{cases}$$

Où $x_i \in [0,1]$ pour tout i $= 1,\ldots,$n et n $= 30$. Le front de Pareto de ce problème est présenté dans la **Figure III.4.**

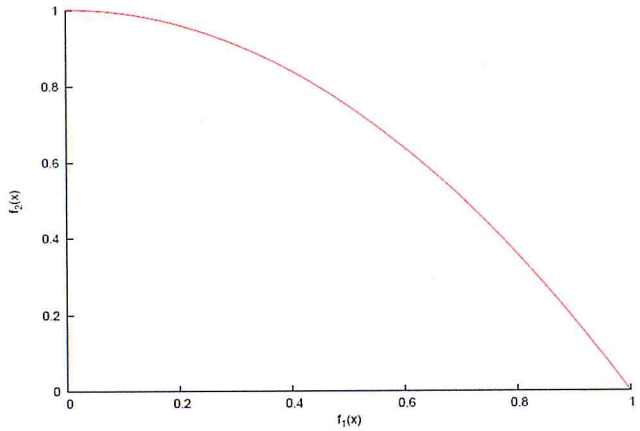


Figure III.4 - Solution de ZDT2 [5]

III.4.3 ZDT3 :

Le ZDT3 pose une difficulté qui est la discontinuité du front de Pareto et la fonction est comme suit :

$$ZDT3 = \begin{cases} f_1(x) = x_1 \\ g(x_2) = 1 + \dfrac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(x) = 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1) \end{cases}$$

Où $x_i \in [0,1]$ pour tout i $= 1,\ldots,$ n et n $= 30$. Le front de Pareto de ce problème est présenté dans la **Figure III.5.**

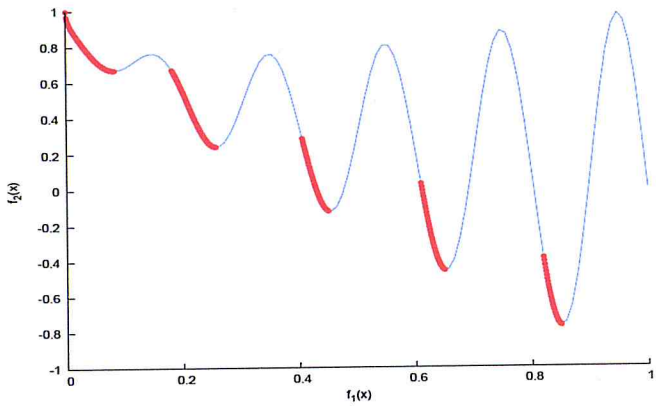


Figure III.5 - Solution de ZDT3 [5]

III.4.4ZDT4

Ce test modélise la difficulté de la convergence vers le front de Pareto global en présence de plusieurs fronts locaux causés par la fonction multimodale g. Cette fonction s'écrit comme suit :

$$ZDT4 = \begin{cases} f_1(x) = x_1 \\ g(x_2) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)) \\ h(x) = 1 - \sqrt{f_1/g} \end{cases}$$

Où $x_1 \in [0,1]$, $x_i \in [-5,5]$pour tout i = 2,…, n et n = 10. Ce test comporte $(21^9 - 1)$ fronts locaux. Quelques-uns de ces derniers ainsi que le front global sont présentés dans la **Figure III.6.**

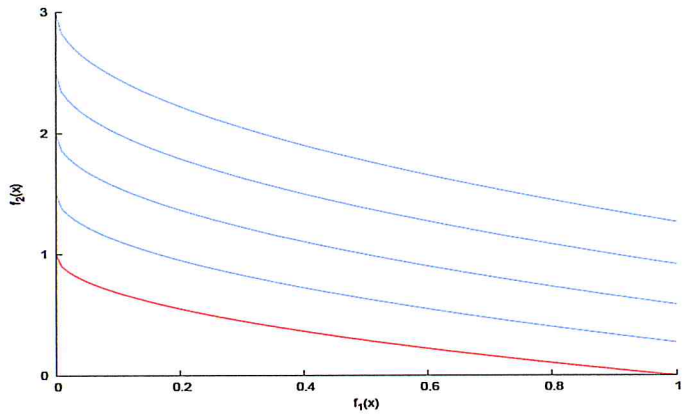


Figure III.6 - Solution de ZDT4 [5]

III.4.5 ZDT6

La particularité de ce problème est que les solutions optimales ne sont pas uniformément distribuées le long du front de Pareto. Cela est dû à la non-linéarité de la fonction f_1. La fonction s'écrit comme suit :

$$ZDT6 = \begin{cases} f_1(x) = 1 - \exp(-4x_1)\, sin^6(4\pi x_1) \\ g(x_2) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{1/4} \\ h(x) = 1 - (f_1/g)^2 \end{cases}$$

Où $x_i \in [0,1]$ pour tout i = 1,…, n et n = 10. Le front de Pareto de ce problème est présenté dans la Figure III.7 ainsi que les solutions correspondantes à 100 valeurs de x_1 uniformément distribuées sur le segment [0,1].

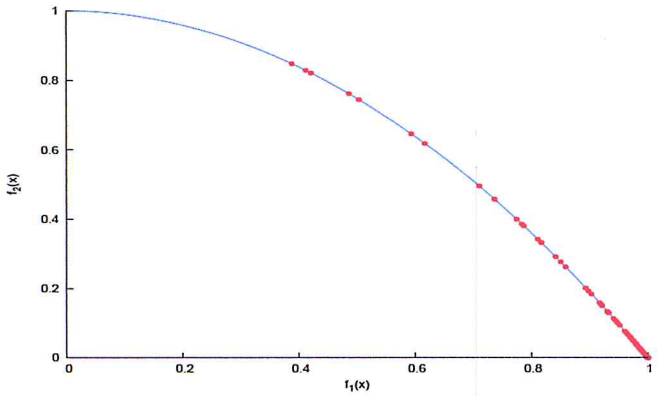


Figure III.7 - Solution de ZDT6 [5]

III.5 Mesure de performance [5][26]

De nombreuses mesures quantitatives ont été proposées pour évaluer la qualité de l'ensemble des compromis optimaux. Nous distinguons dans ce qui suit des métriques qui exigent la connaissance de la surface de Pareto exacte, dites exactes et celles qui n'exigent pas cette information, dites aveugles. Notons que seules ces dernières sont utilisables pour tester l'applicabilité de différents algorithmes à un problème réel.

III.5.1 Métriques exactes

Sous ce titre, nous réunissons des métriques qui nécessitent la connaissance de la surface de Pareto exacte. Dans ce qui suit, Q désignera l'ensemble des solutions non dominées trouvées par l'algorithme, n le cardinal de l'ensemble Q et P^* l'ensemble de Pareto exact.

Taux d'erreur (Error Ratio)

Cette métrique mesure la « proximité » des solutions non-dominées trouvées par un algorithme de la surface de Pareto exacte [20]. Elle est définie par :

$$ER(Q) = \frac{\sum_{i=1}^{n} e_i}{n}$$

Où $e_i = 0$ si l'individu $p_i \in P^*$ et $e_i = 1$ sinon. Plus la valeur de $ER(Q)$ est petite, meilleur est l'ensemble Q. L'inconvénient de cette métrique est qu'elle ne distingue pas la proximité relative entre P^* et les solutions de Q n'appartenant pas à P^*. Pour contourner ce problème, nous pouvons introduire un certain seuil δ et redéfinir e_i comme suit [26]:

$$e_i = \begin{cases} 0 & si\ d_i < \delta \\ 1 & sinon \end{cases}$$

Où d_i est la distance euclidienne (mesurée dans l'espace des critères) entre l'individu p_i et P^*.

Distance générationnelle (Generational Distance)

Comme l'error ratio, la présente métrique est une métrique de convergence. Elle est définie par:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

Où d_i est la distance euclidienne (mesurée dans l'espace des critères) entre l'individu p_i et P^*. Notons que les fonctions objectives doivent être normalisées avant de calculer les valeurs d_i.

III.5.2 Métriques aveugles

Sous ce titre, nous réunissons des métriques qui ne nécessitent pas la connaissance de la surface de Pareto exacte. Ces métriques peuvent être utilisées pour tester l'applicabilité de différents algorithmes à un problème réel. Notons que la dernière métrique qui sera citée dans cette partie, « Coverage metric », nécessite la disposition d'un ensemble référence de solutions.

Volume de l'espace dominé (Hypervolume)

La présente métrique tient compte des deux aspects à la fois, à savoir la convergence vers P^* et la distribution des solutions [21]. Elle mesure l'hyper-volume de la région construite par l'ensemble Q et un point de référence « Ref » dans l'espace des critères. Dans la pratique, nous pouvons prendre le vecteur de Nadir Z^{nad} (voir la section **I.7**), comme point de référence.

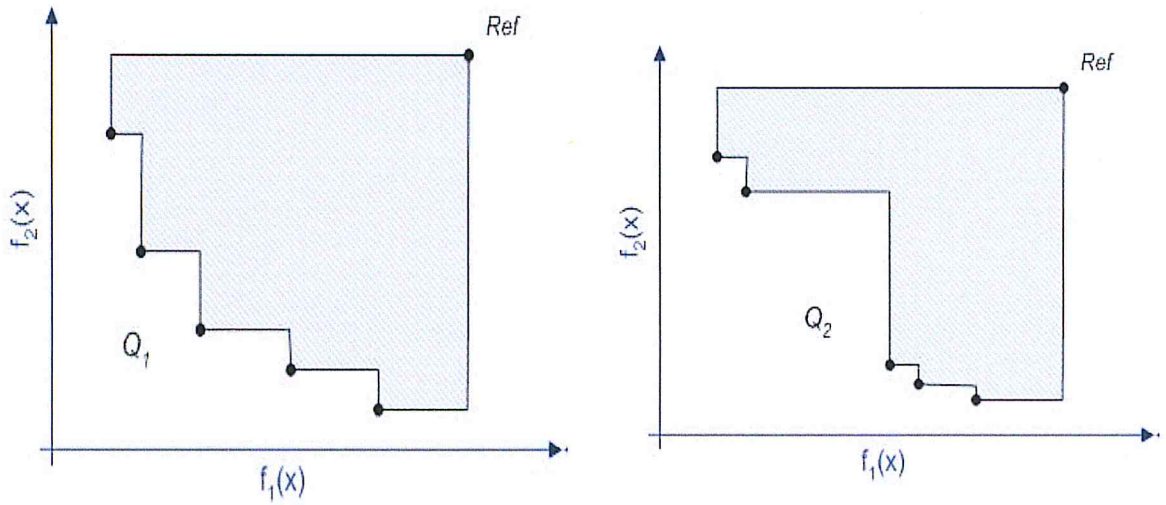


Figure III.8 -Exemple d'application de la métrique (Hypervolume)

Cette figure illustre une exemple d'application de cette métrique pour deux ensembles Q_1 et Q_2 pour un problème bi-objectif. L'ensemble Q1 présente une diversité meilleure que celle de Q_2, c'est pourquoi il présente un hyper-volume plus grand.

Espacement (Spacing)

La métrique suivante évalue l'uniformité de la distribution des solutions non-dominées [22].

$$SP = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \bar{d})^2}$$

Où d_i est défini comme suit:

$$d_i = \min \sum_{\substack{m=1 \\ k \in Q \\ k \neq i}}^{M} \left| f_m^i - f_m^k \right|$$

et $\bar{d}$ est la moyenne des distances d_i pour $i = 1, \ldots, n$.

SP représente donc la déviation standard des valeurs de distance entre deux solutions consécutives de l'ensemble Q. Plus SP est petit, meilleur sera la distribution des solutions.

D'ailleurs, une valeur nulle de SP indique que les solutions sont équidistantes. La présente métrique est une métrique de diversité.

Comparaison de deux ensembles (Set Coverage Metric)

Cette métrique est destinée à comparer deux ensembles non dominés au sens de Pareto [21]. Elle définit une semi-distance entre deux ensembles de solutions A et B par :

$$C(A,B) = \frac{\{b \in B | \exists\ a\ \in A : a \leq b\}}{|B|}$$

Cette quantité correspond à la proportion des solutions de l'ensemble B, qui sont faiblement dominées par au moins une solution de A. Notons que $C(A,B)$ n'est pas nécessairement égale à $1 - C(B,A)$. Pour comparer les ensembles A et B, toutes les deux valeurs $C(A,B)$ et $C(B,A)$ doivent être calculées. La **Figure III.9** représente un exemple d'évaluation de deux ensembles, avec $\mathrm{C(A,B)} = 2/8$, et $\mathrm{C(B,A)} = 4/8$ [26].

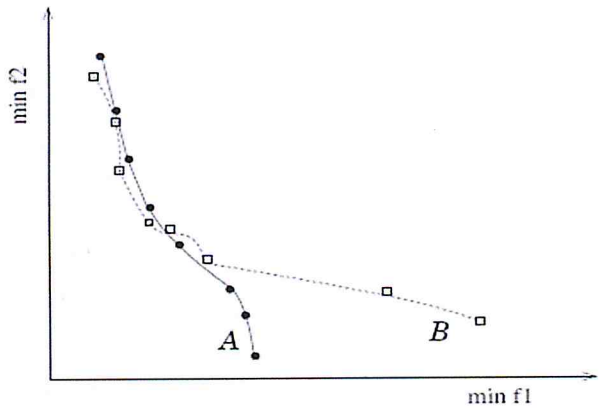


Figure III.9 – Métrique de recouvrement des deux ensembles [26]

III.6. Conclusion

Dans ce chapitre nous avons détaillé l'algorithme **ϵ-MOEA** ainsi que le nouvel algorithme nommé **ϵ-MOEA-LMO,** qui représente notre contribution. Dans le prochain chapitre, nous allons exposer nos expérimentations numériques pour la comparaison des deux versions de l'algorithme ainsi que leur comportement vis-à-vis des initialisations contenant des informations sur le vrai front de Pareto.

Chapitre IV

Réalisation et tests

IV.1Introduction

Ce chapitre est consacré aux expérimentations numériques ainsi qu'aux comparaisons que nous avons effectuées sur les deux algorithmes évolutionnaires multi-objectifs : l'algorithme **ϵ-MOEA** proposé par Deb ainsi que la version modifiée que nous avons appelé **ϵ-MOEA-LMO** (voir **III.3**). Ces résultats numériques sont obtenus grâce à l'application que nous avons réalisé avec le langage de programmation **Java** et L'ide **NetBeans** pour simuler ces algorithmes sur les fonctions tests considérées et calculer les différentes métriques permettant de mesurer les performances relatives.

IV.2 Résultats expérimentaux

Dans cette section, nous illustrons des échantillons des résultats représentatifs des deux algorithmes (**ϵ-MOEA** et **ϵ-MOEA-LMO)** sur les problèmes ZDT1, ZDT2, ZDT3, ZDT4 et ZDT6.

IV.2.1 Conditions expérimentales

Pour pouvoir comparer les résultats de l'algorithme **ϵ-MOEA-LMO** avec ceux de l'algorithme **ϵ-MOEA**, nous avons choisi d'utiliser dans un premier temps les mêmes paramètres que Deb afin de valider l'algorithme **ϵ-MOEA** [30], ensuite de faire varier certains paramètres tels que ϵ et le nombre d'itération. Ainsi, nous avons lancé les deux algorithmes avec les paramètres suivants :

* Taille de la population : 100.
* Valeur de $\epsilon = 0.075, 0.0075, 0.005 , 0.00261 \; et \; 0.001$.
* Nombre d'itération = 5000 et 10000.
* Probabilité de croisement = 1.
* Probabilité de mutation = 0.033.

IV.3 Implémentation de ϵ-MOEA

Pour les mêmes valeurs utilisées par Deb [30], c'est-à-dire $\epsilon = 0.0075$ et nombre d'itération 10000, nous avons retrouvé les fronts de Pareto des cinq fonctions en question. La **Figure IV.1** montre l'exécution de l'algorithme **ϵ-MOEA,** pour le problème ZDT1.

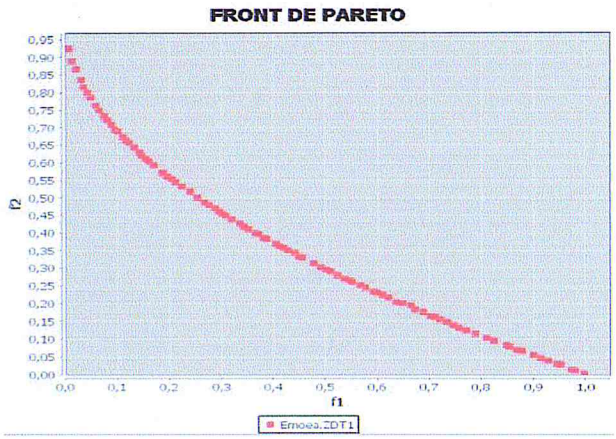


Figure IV.1 - Front de Pareto de ϵ-MOEA pour le problème ZDT1

IV.4 Comparaison entre ϵ-MOEA et ϵ-MOEA-LMO

Nous avons simulé les deux algorithmes **ϵ-MOEA** et **ϵ-MOEA-LMO** pour 50 initialisations différentes. Par la suite, nous avons calculé la valeur moyenne pour chaque mesure de performance (vu dans **III.5**). Ces valeurs moyennes sont calculées après 5000 et 10000 itérations pour chaque valeur de ϵ.

Parmi les métriques citées au chapitre précédant, nous avons choisi pour mesurer la diversité l'espacement (SP) et pour la comparaison de la convergence la métrique (SCM). Les autres mesures (ER) et (GD), sont considérées comme des mesures théoriques et sont difficiles à appliquer pour des raisons techniques.

Dans la suite, nous présenterons les résultats relatifs à chaque problème test avec toutes les valeurs de ϵ proposées.

Signification des valeurs de mesure de performance :

La mesure SP : plus la valeur de SP est petite, mieux distribuées sont les solutions.

La mesure SCM : plus la valeur de SCM est petite, meilleure est la convergence. Prenons un exemple pour calculer la mesure SCM pour les deux algorithmes **ϵ-MOEA** et **ϵ-MOEA-LMO**. Pour calculer la valeur de SCM de **ϵ-MOEA,** nous calculons C(**ϵ-MOEA-LMO, ϵ-MOEA**)=A et pour calculer la valeur de SCM de **ϵ-MOEA-LMO,** nous calculons C(**ϵ-MOEA, ϵ-MOEA-LMO**)=B.

A= nombre de solutions de **ϵ-MOEA** qui sont dominées par des solutions de **ϵ-MOEA-LMO** sur le cardinal de **ϵ-MOEA**.

B=nombre de solutions de **ϵ-MOEA-LMO** qui sont dominées par des solutions de **ϵ-MOEA** sur le cardinal de **ϵ-MOEA-LMO**.

L'algorithme qui a moins de solutions dominées est le meilleur, cela veut dire que l'algorithme qui correspond à la petite valeur entre A et B est le meilleur.

ZDT1

Ce test permet de tester la convexité, le front de Pareto correspondant est continu avec une distribution uniforme des solutions (les solutions ont la même chance d'apparition tout au long de front de Pareto). Le **tableau IV.1,** représente les mesures **SP** et **SCM** pour les deux algorithmes **ϵ-MOEA** et **ϵ-MOEA-LMO** après **5000** et **10000** itérations.

	5000 itérations				*10000 iterations*			
	ϵ-MOEA-LMO		ϵ-MOEA		ϵ-MOEA-LMO		ϵ-MOEA	
ϵ	SP	SCM	SP	SCM	SP	SCM	SP	SCM
0.075	0.0610	0.0	0.0327	0.0	0.0620	0.0	0.03531	0.0
0.0075	0.0056	0.1007	0.0065	0.1271	0.0045	0.0208	0.0057	0.0395
0.005	0.0048	0.1697	0.0049	0.17605	0.0035	0.044571	0.00415	0.0678
0.00261	0.0036	0.2339	0.0038	0.23451	0.0023	0.1306	0.0025	0.10254
0,001	0.0031	0.3119	0.0030	0.27587	0.0015	0.2082	0.00146	0.20699

Tableau IV.1 - Comparaison entre ϵ-MOEA et ϵ-MOEA-LMO pour le problème ZDT1

Discussion

Pour la valeur de $\epsilon = 0,075$, la comparaison entre les deux algorithmes montre qu'à **10000** itérations *l'*ϵ-**MOEA** à une diversité meilleure que celle de **ϵ-MOEA-LMO,** et rien d'important par rapport à la convergence. Quand nous réduisons la valeur de l'**ϵ=0.0075 et 0.005,** l'ϵ-**MOEA-LMO** est meilleur en terme de convergence et de diversité, indépendamment du nombre d'itérations. A la valeur de $\epsilon = 0,00261$ l'ϵ-**MOEA** remporte seulement la convergence à **10000** itérations et remporte les deux mesures quand $\epsilon = 0,001$, les **figure IV.2 et figure IV.3** représentent la variation des deux mesures par rapport à ϵ.

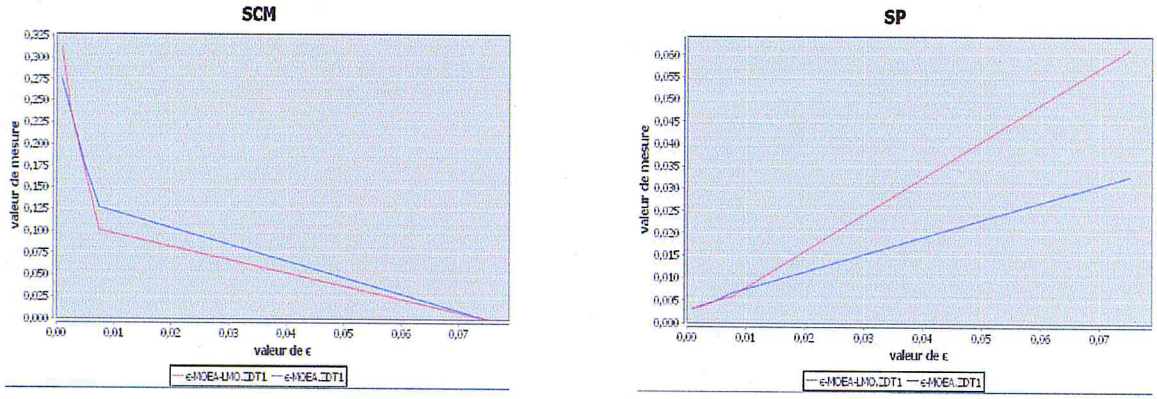


Figure IV.2 – Valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT1

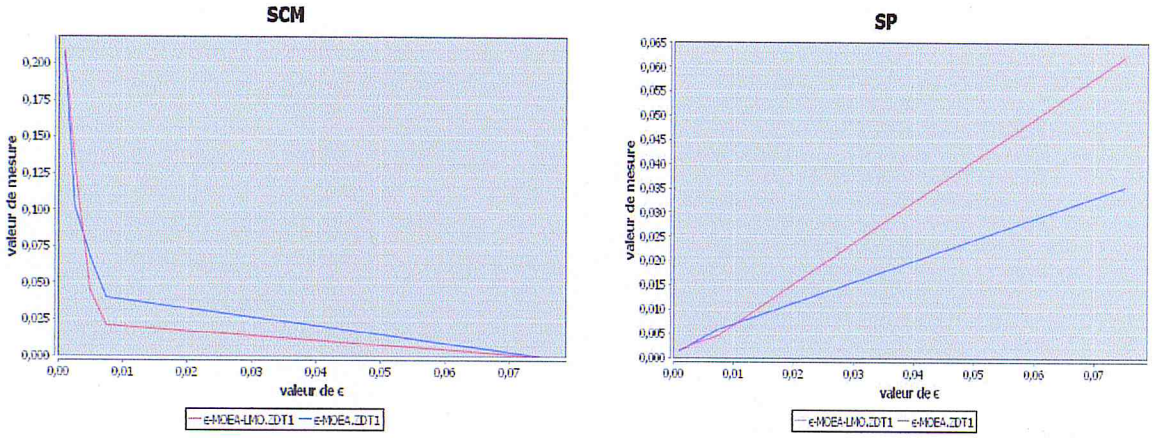


Figure IV.3 – Valeurs de SCM et SP en fonction de ϵ pour 10 000 itérations pour ZDT1

La remarque la plus importante concerne le front de Pareto généré par **ϵ-MOEA-LMO.** Sur ce front, nous trouvons toujours deux points qui sont très proches l'un de l'autre relativement aux autres points du front. Ces deux points sont voisins à la solution Lexico-Max-Ordering (la solution relative à cette dernière dominance et elle est unique), l'un se trouve à gauche et l'autre à droite de cette solution (voir figure IV.4). Cela est dû principalement à la dominance utilisée pour sélectionner les solutions dans les boxes pour l'algorithme **ϵ-MOEA-LMO,** car cette dominance tire les points vers elle.

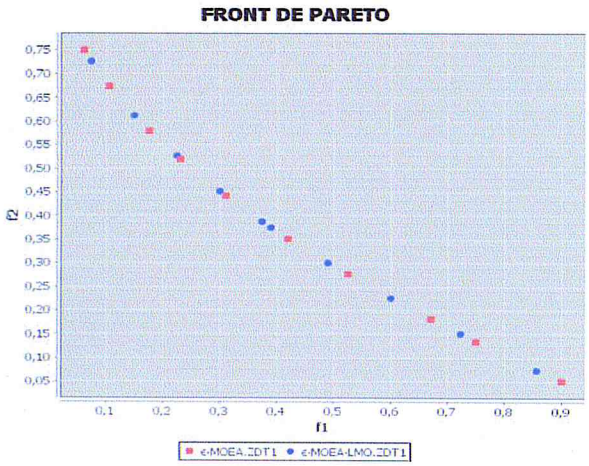


Figure IV.4 - Front ZDT1 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations [sur le front généré par ϵ-MOEA-LMO en bleu, les deux points situés dans la région f_1 entre (0,35 et 0,4) et f_2 entre (0,35 et 0,4) sont très proches et ils sont dans le voisinage de la solution lexico-max-ordering]

ZDT2

La difficulté de ce problème est la non-convexité. Nous avons calculé **SP** et **SCM** pour **5000** et **10000** itérations pour chacun des deux algorithmes:

	5000 itérations				*10000 iterations*			
	ϵ-MOEA-LMO		ϵ-MOEA		ϵ-MOEA-LMO		ϵ-MOEA	
ϵ	SP	SCM	SP	SCM	SP	SCM	SP	SCM
0.075	0.0530	0.0	0.0283	0.0018	0.0529	0.0	0.03158	0.00181
0.0075	0.0101	0.0395	0.0084	0.245444	0.0075	0.010317	0.00759	0.10222
0.005	0.0069	0.1141	0.0056	0.265562	0.0050	0.024058	0.00495	0.15615
0.00261	0.0045	0.2688	0.0044	0.241250	0.0031	0.121405	0.00302	0.19839
0,001	0.0030	0.2482	0.0034	0.300243	0.0021	0.251171	0.00175	0.27369

Tableau IV.2 - Comparaison entre ϵ-MOEA et ϵ-MOEA-LMO pour le problème ZDT2

Discussion

Pour la valeur de $\epsilon = 0{,}075$, nous remarquons que la diversité de **ϵ-MOEA** est meilleure que celle de **ϵ-MOEA-LMO,** à l'inverse de la convergence.
- Pour la valeur de $\epsilon = 0,0075$, nous remarquons que **ϵ-MOEA-LMO** est meilleur que

ϵ-MOEA par rapport aux deux mesures et cela indépendamment du nombre d'itération.
- Pour une valeur de $\epsilon = 0,005$ et à **10000** itérations, les deux algorithmes partagent les deux mesures, **ϵ-MOEA-LMO** prend la convergence et ϵ-MOEA la diversité.
- Pour une valeur de $\epsilon = 0,00261$, nous remarquons que **ϵ-MOEA** est meilleur dans les deux mesures à **5000** itérations mais à 10000 itérations, il garde uniquement la diversité.
-La dernière valeur de $\epsilon = 0,001$ montre que l'**ϵ-MOEA-LMO** est meilleur que **ϵ-MOEA** après **5000** itérations par rapport aux deux mesures, mais après 10000 itérations l'**ϵ-MOEA** reprend la diversité.

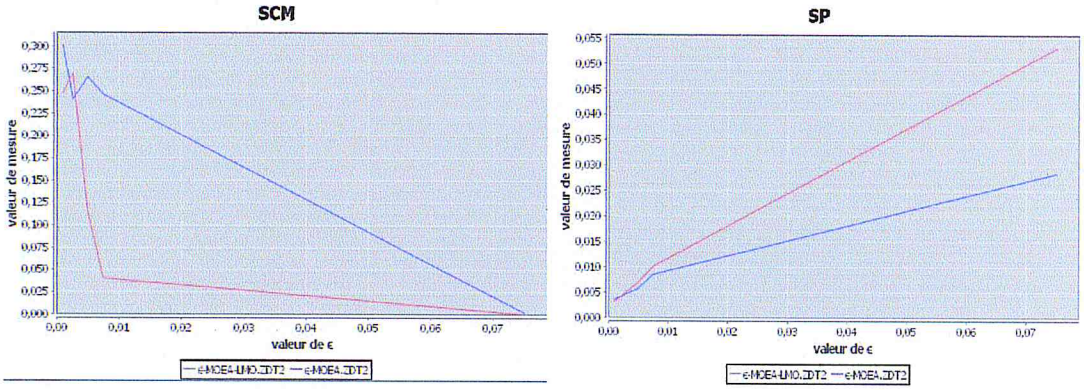


Figure IV.5 – Valeurs de SCM et SP en fonction de ϵ pour 5 000 itérations pour ZDT2

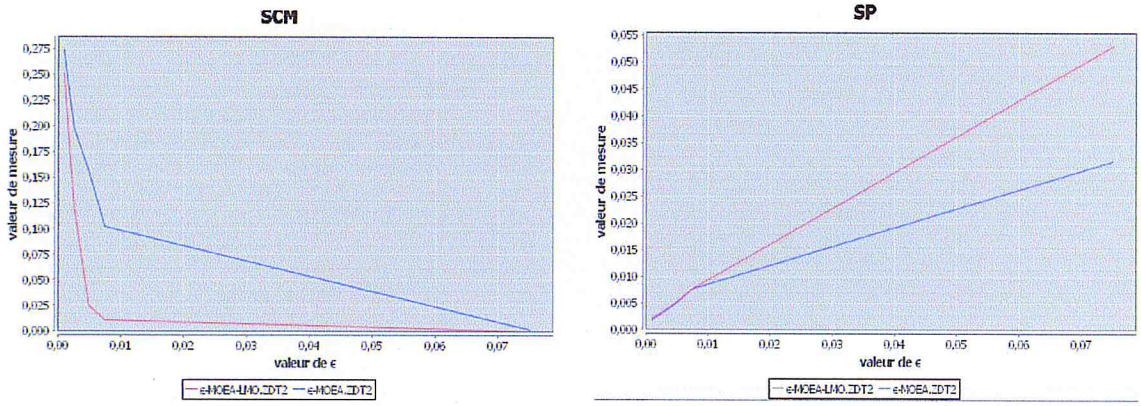


Figure IV.6 – Valeurs de SCM et SP en fonction de ϵ pour 10 000 itérations pour ZDT2

La même remarque pour le front de ZDT2, le front expose un rapprochement entre deux points dans la région définie par f_1 entre (0,60 et 0,65) et f_2 entre (0,60 et 0,65). Cette région est au voisinage de la solution lexico-max-ordering. **La figure IV.7** représente le Front de Pareto pour les deux algorithmes.

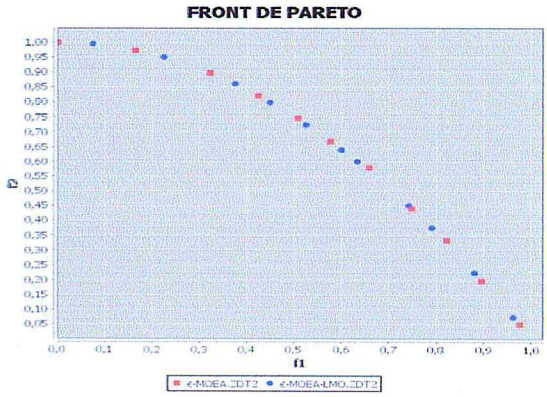


Figure IV.7-Front ZDT2 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations

ZDT3

La difficulté de ce problème est la discontinuité du front de Pareto, l'évaluation des deux algorithmes est exprimée en détail dans le **tableau IV.3.**

	5000 itérations				*10000 iterations*			
	ϵ-MOEA-LMO		ϵ-MOEA		ϵ-MOEA-LMO		ϵ-MOEA	
ϵ	SP	SCM	SP	SCM	SP	SCM	SP	SCM
0.075	0.1806	0.0170	0.2038	0.007333	0.1725	0.0	0.1989	0.03
0.0075	0.0286	0.0994	0.0303	0.096521	0.0280	0.014431	0.02743	0.01389
0.005	0.0191	0.1710	0.0193	0.109357	0.0196	0.042395	0.01958	0.02851
0.00261	0.0107	0.1641	0.0100	0.195073	0.0123	0.098680	0.01047	0.07986
0,001	0.0064	0.2550	0.0045	0.280100	0.0044	0.164661	0.00397	0.17021

Tableau IV.3 - Comparaison entre ϵ-MOEA et ϵ-MOEA-LMO pour le problème ZDT3

Discussion

Nous remarquons que pour une valeur donnée de $\epsilon = 0,075, 0,0075$ *et* $0,005$ et à **5000** itérations les deux algorithmes partagent les deux mesures. L'**ϵ-MOEA-LMO** assure la diversité et **ϵ-MOEA** assure la convergence. A **10000** itérations, l'**ϵ-MOEA-LMO** remporte les deux mesures pour une valeur de $\epsilon = 0,075$ et **ϵ-MOEA** remporte les deux mesures pour la valeur de $\epsilon = 0.0075, 0.005$ *et* 0.00261, pour $\epsilon = 0.001$ et après 10000 itérations les deux algorithmes partagent les deux mesures, **ϵ-MOEA** garde la diversité et **ϵ-MOEA-LMO** reprend la convergence.

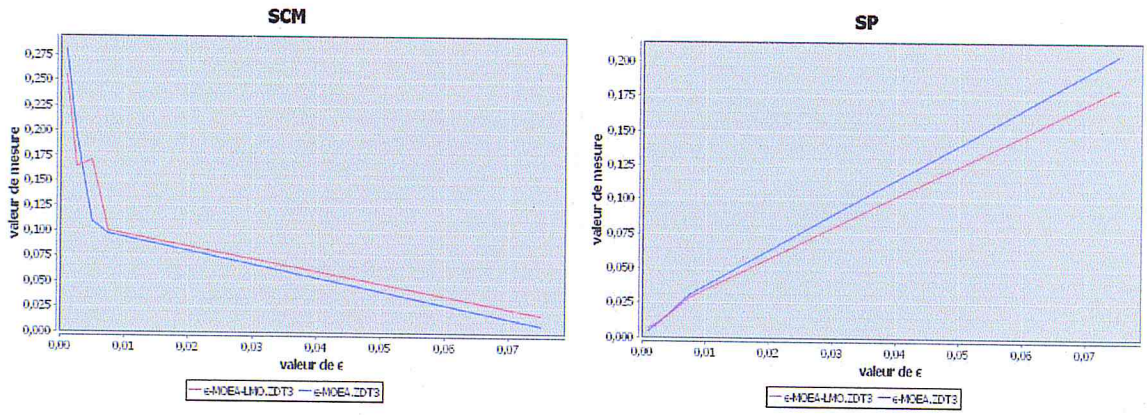


Figure IV.8 – Valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT3

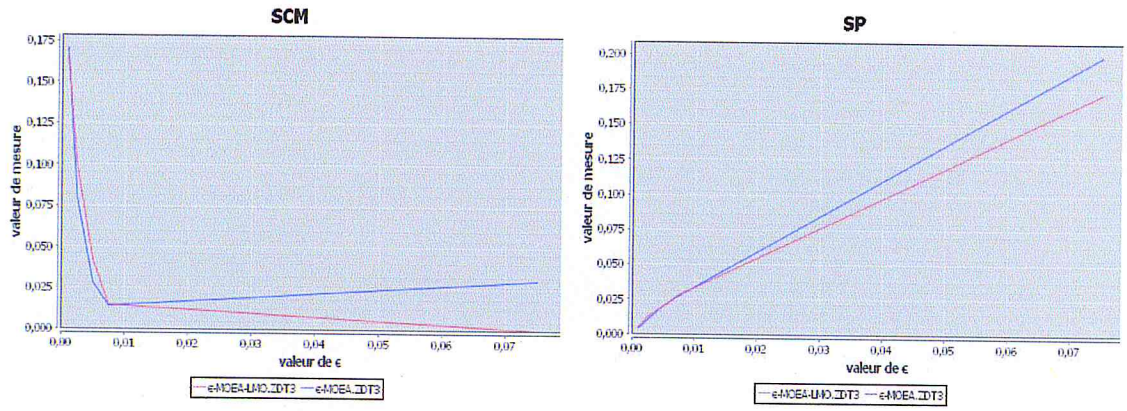


Figure IV.9 –Les valeurs de SCM et SP en fonction de ϵ pour 10000 itérations pour ZDT3

Le front de ZDT3 montre que la solution lexico-max-ordering se trouve dans la région f_1 entre (0,2 et 0,25) et f_2 entre (0,2 et 0,4) (Voir la **figure IV.10**).

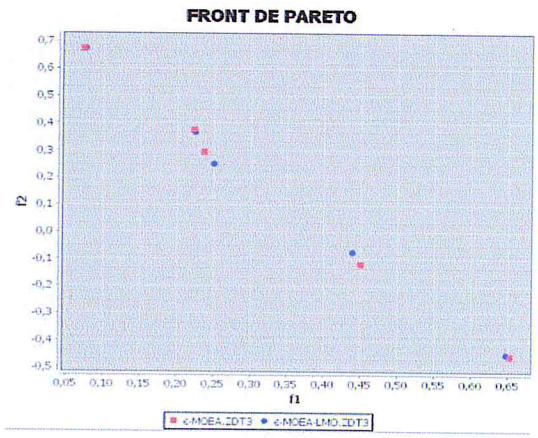


Figure IV.10 – Front ZDT3 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations

ZDT4

Ce test modélise la difficulté de la convergence vers le front de Pareto global en présence de plusieurs fronts locaux. Le **tableau IV.4** expose les différents tests effectués:

	5000 itérations				*10000 iterations*			
	ϵ-MOEA-LMO		ϵ-MOEA		ϵ-MOEA-LMO		ϵ-MOEA	
ϵ	SP	SCM	SP	SCM	SP	SCM	SP	SCM
0.075	0.0463	0.3628	0.0365	0.267547	0.0543	0.040444	0.03283	0.002
0.0075	0.0132	0.3713	0.0134	0.476555	0.0055	0.243417	0.00674	0.25740
0.005	0.0122	0.5370	0.0096	0.381019	0.0044	0.357233	0.00477	0.24245
0.00261	0.0110	0.4354	0.0080	0.464731	0.0034	0.290013	0.00340	0.42285
0,001	0.0082	0.3938	0.0099	0.462529	0.0024	0.367945	0.00281	0.42153

Tableau IV.4 - Comparaison entre ϵ-MOEA et ϵ-MOEA-LMO pour le problème ZDT4

Discussion

Pour La valeur de $\epsilon = 0,075$ nous remarquons qu'à 10000 itérations, l'**ϵ-MOEA** présente une meilleure convergence et diversité de ces solutions par rapport à l'**ϵ-MOEA-LMO.** Pour $\epsilon = 0,0075$, nous remarquons que la convergence et la diversité est meilleure dans l'**ϵ-MOEA-LMO.** Pour $\epsilon = 0,005$, nous remarquons que l'**ϵ-MOEA** remporte les deux mesures après 5000 itérations et après 10000 itérations les deux algorithmes partagent les deux mesures, l'**ϵ-MOEA** garde la convergence et l'**ϵ-MOEA-LMO** reprend la diversité.

Pour la valeur $\epsilon = 0,00261$ et après 5000 itérations, l'**ϵ-MOEA-LMO** gagne la convergence et l'**ϵ-MOEA** gagne la diversité. Après 10000 itérations, ***l'*ϵ-MOEA-LMO** remporte les deux mesures et présente une meilleure convergence et diversité. Il en est de même pour $\epsilon = 0,001$ et après les 10000 itérations.

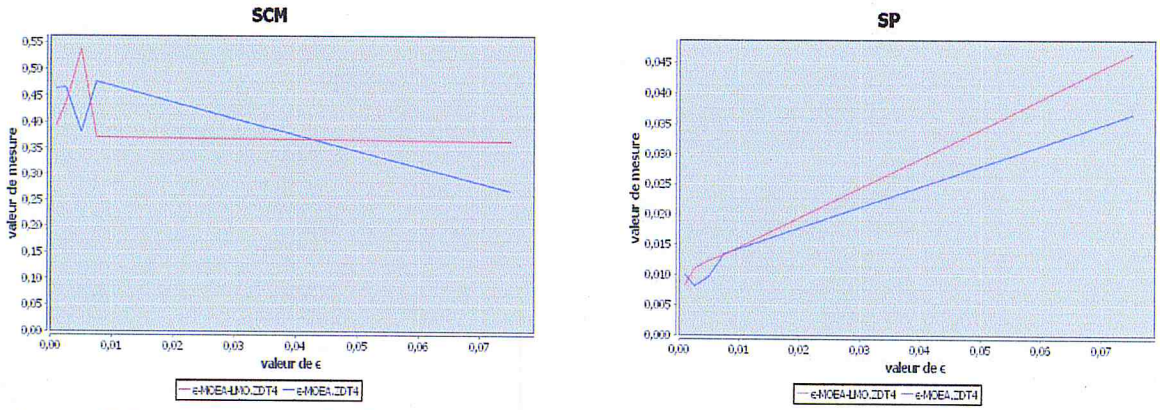


Figure IV.11 – Valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT4

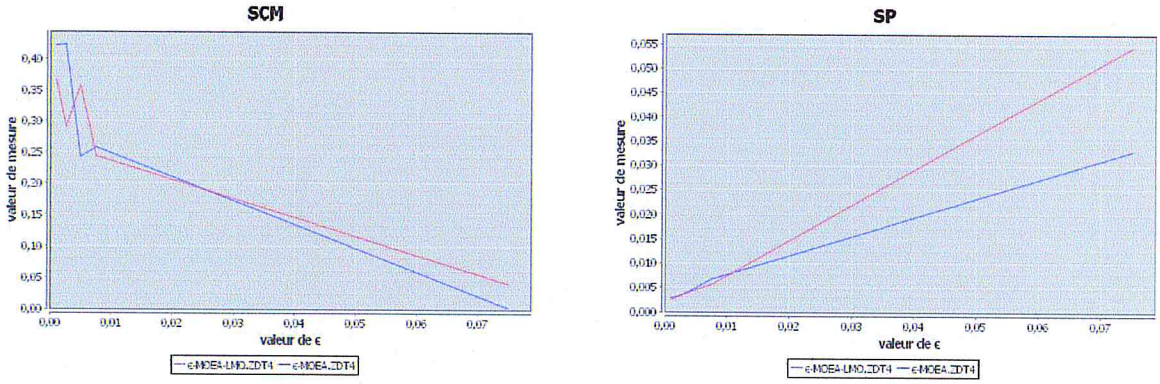


Figure IV.12 – Valeurs de SCM et SP en fonction de ϵ pour 10000 itérations pour ZDT4

Le front de ZDT4 de l'algorithme l'**ϵ-MOEA–LMO** montre que la solution lexico-max-ordering se trouve dans la région f_1 entre (0,35 et 0,45) et f_2 entre (0,35 et 0,45). C'est la région où il y a les deux points les plus proches comme le montre la **figure IV.13.**

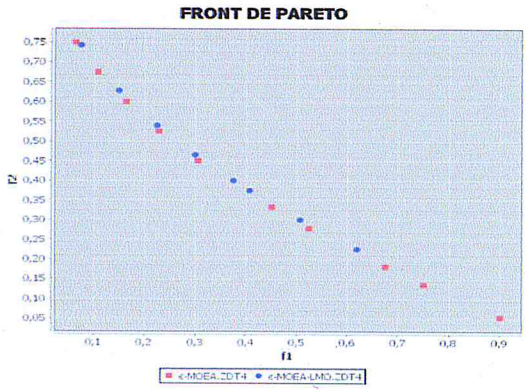


Figure IV.13 – Front ZDT4 de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations

ZDT6

La particularité de ce problème est que les solutions optimales ne sont pas uniformément distribuées le long du front de Pareto. Le **tableau IV.5** présente les différentes valeurs de SP et SCM:

	5000 itérations				*10000 iterations*			
	ϵ-MOEA-LMO		ϵ-MOEA		ϵ-MOEA-LMO		ϵ-MOEA	
ϵ	SP	SCM	SP	SCM	SP	SCM	SP	SCM
0.075	0.0397	0.0088	0.0435	0.200833	0.0553	0.0	0.03019	0.0475
0.0075	0.0060	0.2737	0.0061	0.423150	0.0043	0.084581	0.00401	0.39637
0.005	0.0054	0.4072	0.0053	0.362158	0.0034	0.270347	0.00317	0.28997
0.00261	0.0044	0.4914	0.0045	0.392863	0.0024	0.330972	0.00246	0.41170
0,001	0.0042	0.4969	0.0041	0.392114	0.0021	0.452987	0.00205	0.38137

Tableau IV.5 - Comparaison entre ϵ-MOEA et ϵ-MOEA-LMO pour le problème ZDT6

Discussion

avec $\epsilon = 0{,}075$ et $0{,}0075$ et après 5000, itérations nous remarquons que **ϵ-MOEA-LMO** présente une bonne convergence et diversité par rapport à **ϵ-MOEA**. Par contre après 10000 itérations, les deux algorithmes partagent les deux mesures **ϵ-MOEA** assure la diversité et **ϵ-MOEA-LMO** présente la convergence. Pour la valeur de $\epsilon = 0{,}005$ et après **5000** itérations, l'**ϵ-MOEA** est meilleur, mais les deux algorithmes partagent les mesures après 10000 itérations. Il en est de même pour la valeur de $\epsilon = \mathbf{0,00261}$ après **5000** itérations, mais **ϵ-MOEA-LMO** remporte les deux mesures après **10000** itérations.

La dernière valeur de $\epsilon = 0{,}001$ montre que l'**ϵ-MOEA** est meilleur que l'**ϵ-MOEA-LMO** par rapport aux deux mesures après 10000 itérations.

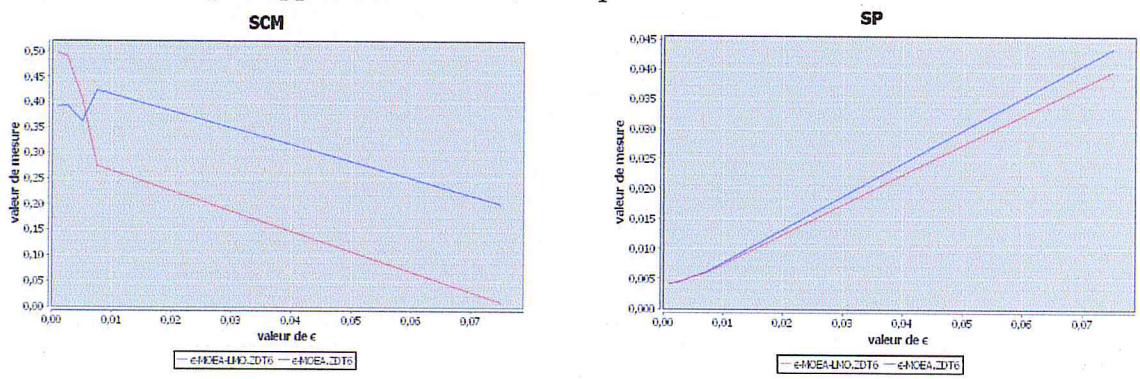


Figure IV.14 – Valeurs de SCM et SP en fonction de ϵ pour 5000 itérations pour ZDT6

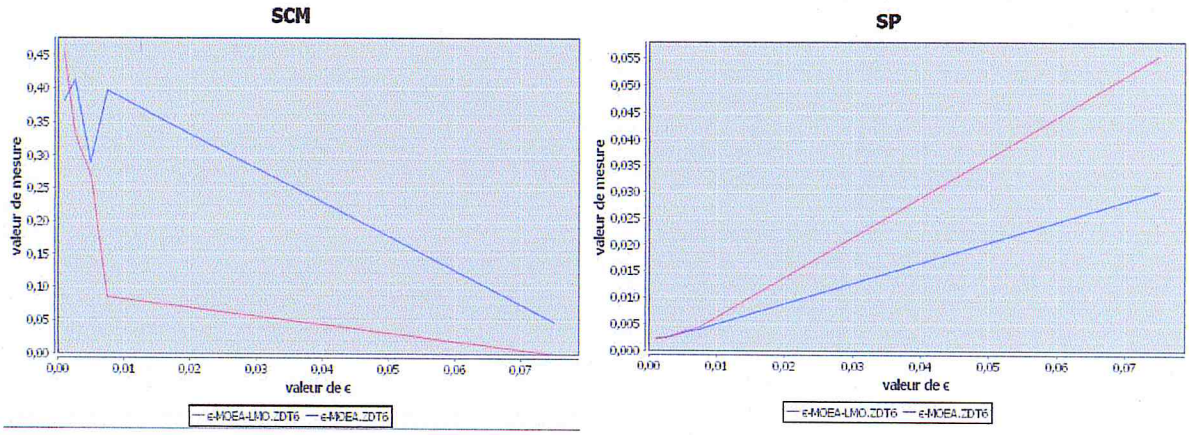


Figure IV.15 – Valeurs de SCM et SP en fonction de ϵ pour 10000 itérations pour ZDT6

La solution lexico-max-ordering du problème ZDT6 pour **ϵ-MOEA-LMO** se trouve dans la région f_1 entre (0,60 et 0,65) et f_2 entre (0,60 et 0,65) comme le montre la **figure IV.16.**

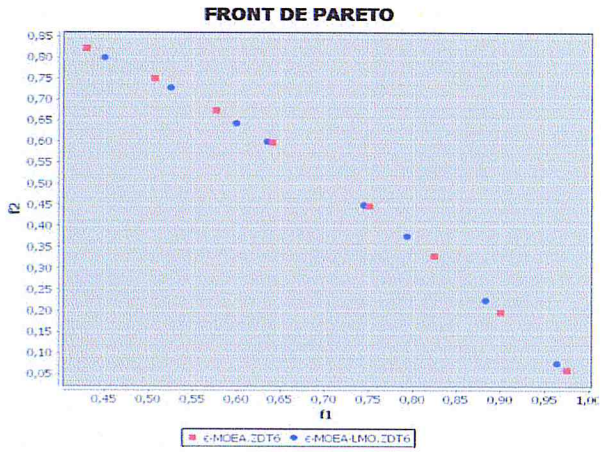


Figure IV.16 – Front ZDT4de ϵ-MOEA-LMO et ϵ-MOEA avec ϵ=0,075 après 10000 itérations

Conclusion

Pour le problème ZDT1 qui est un problème relativement simple, nous avons remarqué que pour les valeurs de ϵ=0.0075 et ϵ=0.005 que l'algorithme **ϵ-MOEA-LMO** est meilleur que l'algorithme **ϵ-MOEA** par rapport à la convergence et la diversité. Par contre, pour des valeurs de ϵ qui sont dans le voisinage de 0.075 et 0.001, l'algorithme **ϵ-MOEA** est meilleur que **ϵ-MOEA-LMO**.

Pour le problème ZDT2, l'algorithme **ϵ-MOEA-LMO** a souvent une convergence meilleure que l'algorithme **ϵ-MOEA,** alors que l' **ϵ-MOEA** a souvent une meilleure diversité, ce qui est probablement dû à la non-convexité de problème ZDT2.

Pour le problème ZDT3, l'algorithme **ϵ-MOEA** est souvent meilleur que l'algorithme **ϵ-MOEA-LMO** pour des valeurs petites de ϵ. Par contre, pour des valeurs de ϵ supérieures à 0.0075, l'algorithme **ϵ-MOEA-LMO** est meilleur.

Pour le problème ZDT4 pour des valeurs petites de ϵ, l' **ϵ-MOEA-LMO** est souvent meilleur. Par contre, pour des valeurs supérieures à 0.0075, l' **ϵ-MOEA** est meilleur.

Pour le problème ZDT6 c'est souvent proportionnel, c'est-à-dire à chaque fois la valeur de ϵ augmente l'algorithme **ϵ-MOEA-LMO** est meilleure, ce qui est probablement dû à la taille du boxe de la grille générée dans l'espace des objectifs : « quand le boxe est grand nous avons plus de chance d'avoir plus de points dans le même boxe et cela permet à la dominance lexico-max-Ordering de rentrer en activité ».

IV.5 Comportement de ϵ-MOEA-LMO par rapport à l'incorporation de connaissance en initialisation

Cette partie est consacrée à une comparaison et une évaluation du comportement de l'algorithme **ϵ-MOEA-LMO** vis-à-vis d'informations (ou connaissances) introduites lors de son initialisation. Pour **ϵ=0,0075, nous avons choisi la solution** Lexico-Max-Ordering comme connaissance à introduire dans la population initiale.

ZDT1

ZDT1	ϵ-MOEA-LMO		ϵ-MOEA-LMOI	
Itérations	SP	SCM	SP	SCM
5000	0.00635	0.4028	0.00466	0,0002127
10 000	0.00461	0.161	0.00438	0.0

Tableau IV.6 - Comparaison entre ϵ-MOEA-LMO et ϵ-MOEA-LMOI pour ZDT1

ZDT2

ZDT2	ϵ-MOEA-LMO		ϵ-MOEA-LMOI	
Itérations	SP	SCM	SP	SCM
5000	0.00887	0.282	0.00699	0.0100
10 000	0.00752	0.094	0.00686	0.0063

Tableau IV.7 - Comparaison entre ϵ-MOEA-LMO et ϵ-MOEA-LMOI pour ZDT2

ZDT3

ZDT3	ϵ-MOEA-LMO		ϵ-MOEA-LMOI	
Itérations	SP	SCM	SP	SCM
5000	0.0296	0.16560	0.0231	0.0016
10 000	0.030422	0.11407	0.0228	0.0

Tableau IV.8 - Comparaison entre ϵ-MOEA-LMO et ϵ-MOEA-LMOI pour le problème ZDT3

ZDT4

ZDT4	ϵ-MOEA-LMO		ϵ-MOEA-LMOI	
Itérations	SP	SCM	SP	SCM
5000	0.011066	0.93819	0.00455	0.0
10 000	0.005438	0.46536	0.00438	0.0

Tableau IV.9 - Comparaison entre ϵ-MOEA-LMO et ϵ-MOEA-LMOI pour ZDT4

ZDT6

ZDT6	ϵ-MOEA-LMO		ϵ-MOEA-LMOI	
Itérations	SP	SCM	SP	SCM
5000	0.0062464	0.985282	0.004018	0.0
10 000	0.0054385	0.465364	0.004387	0.0

Tableau IV.10 - Comparaison entre ϵ-MOEA-LMO et ϵ-MOEA-LMOI pour ZDT6

Discussion

Pour tous les problèmes traités ZDT1, ZDT2, ZDT3, ZDT4 et ZDT6 à **10000** itérations, l'**ϵ-MOEA-LMO** réagit favorablement vis-à-vis d'une initialisation lexico-max-Ordering. Comme l'atteste les résultats numériques obtenus, l'algorithme présente une meilleure convergence vers le front et une meilleure répartition des solutions. Nous pouvons conclure que faire incorporer des connaissances sur le vrai front de Pareto lors d'une initialisation rend l'algorithme plus efficace avec une « convergence rapide et une très bonne diversité ».

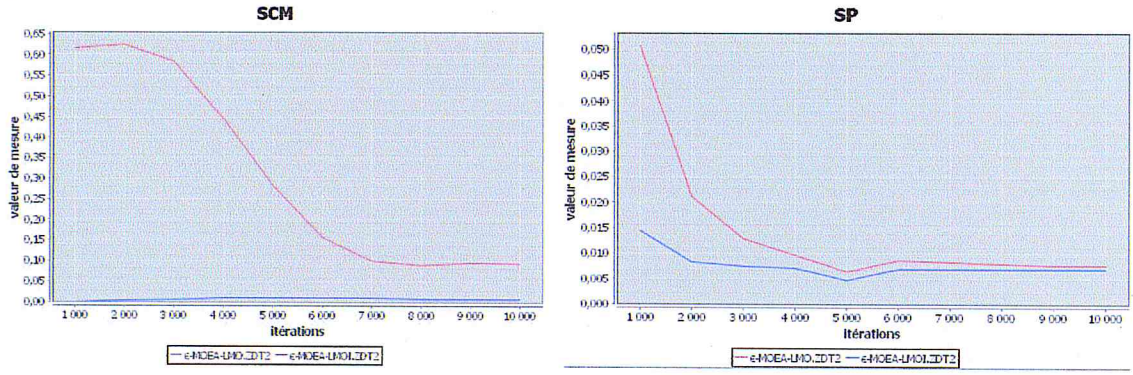


Figure IV.17 – SCM et SP entre ϵ-MOEA-LMO et ϵ-MOEA pour ZDT2

IV.6 Comportement de ϵ-MOEA par rapport à l'incorporation de connaissance en initialisation

Toutes les mesures sont calculées par rapport à la solution lexico-max-Ordering et pour $\epsilon = 0,0075$

ZDT1

ZDT1	ϵ-MOEA		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.00641	0.585038	0.00592	2.15E-4
10 000	0.00586	0.193920	0.00559	0.006000000000000001

Tableau IV.11 - Comparaison entre ϵ-MOEA et ϵ-MOEAI pour ZDT1

ZDT2

ZDT2	ϵ-MOEA		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.00848	0.32745	0.0070	0.010900
10 000	0.00752	0.16276	0.00687	0.010495

Tableau IV.12 - Comparaison entre ϵ-MOEA et ϵ-MOEAI pour ZDT2

ZDT3

ZDT3	ϵ-MOEA		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.029009	0.31051	0.0209075	0.0
10 000	0.027836	0.08243	0.0223348	0.00374

Tableau IV.13 - Comparaison entre ϵ-MOEA et ϵ-MOEAI pour ZDT3

ZDT4

ZDT4	ϵ-MOEA		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.010881	0.917200	0.005356	0.0
10 000	0.006270	0.583055	0.005305	0.0

Tableau IV.14 - Comparaison entre ϵ-MOEA et ϵ-MOEAI pour ZDT4

ZDT6

ZDT6	ϵ-MOEA		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.005871	1.0	0.0037042	0.0
10 000	0.004056	0.9984	0.0037382	0.0

Tableau IV.15 - Comparaison entre ϵ-MOEA et ϵ-MOEAI pour ZDT6

Discussion

D'après les tableaux **tableau IV.11, tableau IV.12, tableau IV.13, tableau IV.14 et tableau IV.15,** nous remarquons que **ϵ-MOEAI** montre sa puissance par rapport aux deux mesures dans tous les problèmes tests.

Dans ce cas, nous pouvons attester que **ϵ-MOEA** réagit lui aussi, favorablement à une initialisation lexico-max-Ordering, car la solution Lexico-Max-Ordering introduite lors de l'initialisation a rendu l'algorithme plus puissant.

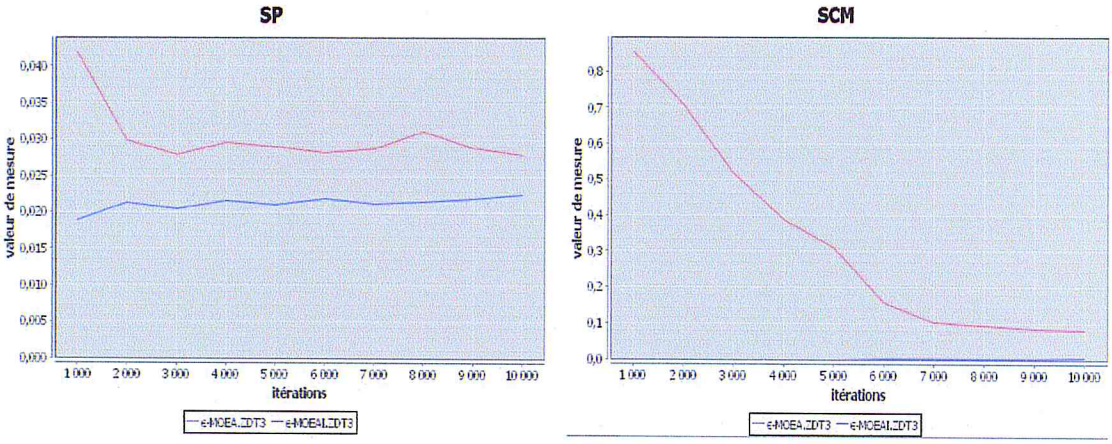


Figure IV.18 – SCM et SP entre ϵ-MOEA et ϵ-MOEAI pour ZDT3

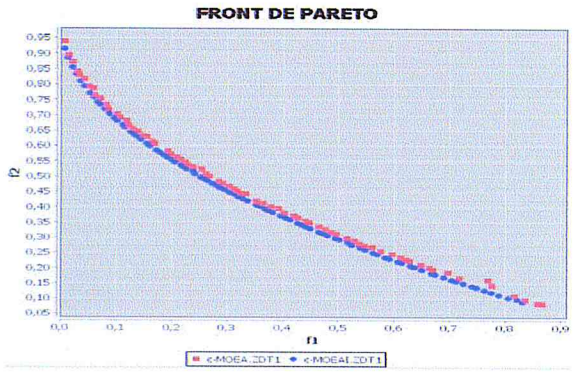


Figure IV.19- Front ZDT1 de ϵ-MOEA et ϵ-MOEAI après 3000 itérations

IV.7 Comparaison par rapport à l'initialisation entre l' ϵ-MOEA-LMO et ϵ-MOEA

Toutes les mesures sont calculées pour la même initialisation et pour $\epsilon = 0,0075$

ZDT1

ZDT1	ϵ-MOEA-LMOI		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.00490605	0.0068727	0.0057008	0.0104001
10 000	0.00439307	8.0E-4	0.0055214	0.0066000

Tableau IV.16 - Comparaison entre ϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT1

Discussion

Ce problème montre que **ϵ-MOEA-LMOI** est meilleur que **ϵ-MOEAI** par rapport aux deux mesures après les **10000** itérations.

ZDT2

ZDT2	ϵ-MOEA-LMOI		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.0070808002	0.003364395	0.00699957	0.02297437
10 000	0.0069037836	0.001386138	0.00686845	0.00891089

Tableau IV.17 - Comparaison entre ϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT2

Discussion

Ce problème montre que **ϵ-MOEAI** est meilleur que **ϵ-MOEA-LMOI** par rapport à la diversité et l'inverse par rapport à la convergence.

ZDT3

ZDT3	ϵ-MOEA-LMOI		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.0248904093	0.002684553	0.0255089	0.01028232
10 000	0.0235266012	0.001290322	0.0255295	0.00359801

Tableau IV.18 - Comparaison entre ϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT3

Discussion

Ce problème montre la force de *l'***ϵ-MOEA-LMOI** par rapport aux deux mesures après **10000** itérations.

ZDT4

ZDT4	ϵ-MOEA-LMOI		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.0044618107	0.00140202	0.0052974	0.0
10 000	0.0043877571	0.002	0.00531262	0.0

Tableau IV.19 - Comparaison entre ϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT4

Discussion

Après **10000** itérations, nous remarquons que *l'*ϵ**-MOEA-LMOI** assure la diversité des points et *l'*ϵ**-MOEAI** assure la convergence vers le Front.

ZDT6

ZDT6	ϵ-MOEA-LMOI		ϵ-MOEAI	
Itérations	SP	SCM	SP	SCM
5000	0.0040326955	0.0	0.00371346	0.0
10 000	0.0040639689	0.0	0.00373965	0.0

Tableau IV.20 - Comparaison entre ϵ-MOEA-LMOI et ϵ-MOEAI pour ZDT6

Discussion

Ce problème expose une équivalence de dominance par rapport aux deux algorithmes, ce qui exprime qu'il n'existe pas des points appartenant à **ϵ-MOEAI** qui dominent d'autres dans **ϵ-MOEA-LMOI** après **10000** itérations.

Par rapport à la diversité, nous remarquons que **ϵ-MOEAI** possède une diversité meilleure que celle de **ϵ-MOEA-LMOI** après **10000** itérations.

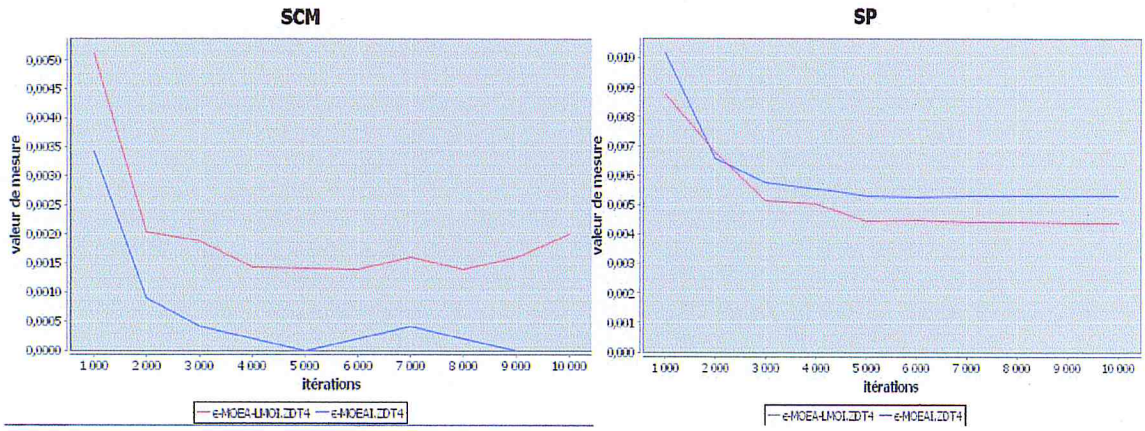


Figure IV.20- SCM et SP de ϵ-MOEA-LMOI et de ϵ-MOEAI pour ZDT4

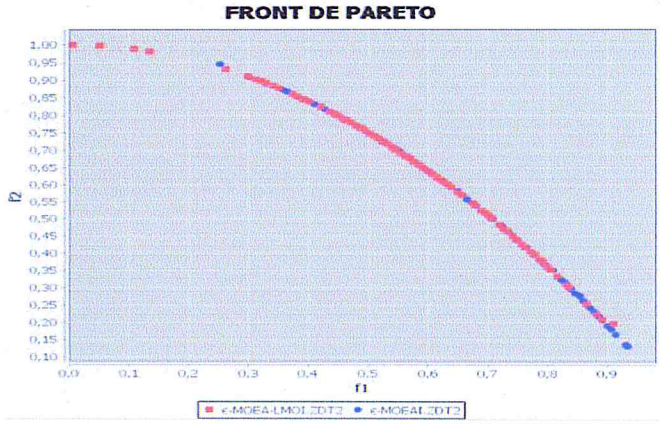


Figure IV.21 – Front ZDT2 de ϵ-MOEA-LMOI et ϵ-MOEAI après 1500 itérations

IV.8 Temps d'exécution

Ce tableau contient la valeur moyenne de temps d'exécution pour 50 initialisations différentes.

Itérations	Algorithme	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
5000	ϵ-MOEA-LMO	203.84	204.54	239.78	152.28	191.4
	ϵ-MOEA	209.46	207.36	245.06	145.92	186.64
	ϵ-MOEA-LMOI	165.4	165.76	212.34	146.96	182.22
	ϵ-MOEAI	172.78	175.54	215.26	151.18	183.02
10000	ϵ-MOEA-LMO	428.16	424.66	484.24	311.86	384.48
	ϵ-MOEA	427.44	431.76	490.6	299.24	378.46
	ϵ-MOEA-LMOI	321.76	323.58	407.6	289.72	360.14
	ϵ-MOEAI	341.92	347.6	412.46	292.32	362.8

Tableau IV.21 – Temps d'exécution des différents algorithmes

Discussion

Les résultats obtenus après les différentes comparaisons effectuées, montre une concurrence entre les différents algorithmes qui réside dans : **la valeur de ϵ** .Un bon choix de ϵ peut influencer favorablement ϵ-MOEA-LMO.

L'utilisation de solution Lexico-Max-Ordering : l'introduction des points Lexico-Max-Ordering dans la population initiale, accélère la convergence vers le front de Pareto et permet aussi d'avoir une bonne diversité tout au long du front de Pareto.

IV.9 Conclusion

En se basant sur les différentes comparaisons effectuées et à partir du **Tableau IV.16** qui expose le temps d'exécution de chaque algorithme, nous pouvons dire que l'ϵ-MOEA-LMO a une convergence et une diversité meilleure que celles de ϵ-MOEA dans certains problèmes tests et pour certaines valeurs de ϵ, et qu'il est moins bon que l' ϵ-MOEA sur d'autres problèmes tests pour d'autres valeurs de ϵ.

Dans le cadre de ce travail, nous nous sommes intéressés aux algorithmes évolutionnaires multi objectifs et en particulier à l'algorithme ϵ-MOEA. Cet algorithme procède de façon à équilibrer la convergence et la diversité en utilisant d'une part la ϵ-dominance à travers la grille créée dans l'espace des objectifs et d'autre part, une procédure qui combine la dominance et la distance euclidienne dans les cases (boxes) de la grille. Notre objectif premier était de voir comment cet algorithme réagit si nous remplaçons la procédure utilisée dans les boxes, par la dominance lexico-max-Ordering.

Dans la première partie de notre étude, nous avons présenté et implémenté l'algorithme évolutionnaire multi-objectif **ϵ-MOEA,** qui est proposé par **Deb** [30]. Par la suite, nous avons modifié cet algorithme. Cette modification nous a donné un nouvel algorithme que nous avons baptisé **ϵ-MOEA-LMO**.

Après la réalisation des deux algorithmes, nous avons effectué une série de simulations et en calculant les différentes mesures de performances nous sommes arrivés à la conclusion suivante :
Les résultats sont dispersés selon la valeur de ϵ et selon les caractéristiques du problème test.
Une deuxième comparaison a été effectuée pour voir le comportement de ces deux algorithmes par rapport à une incorporation de connaissance, sur le front à travers l'initialisation. Cette fois, une solution **Lexico-Max-Ordering** a été introduite dans les populations initiales pour les deux algorithmes.
Après comparaison, nous avons remarqué que les deux algorithmes se portent mieux en termes de convergence et de diversité, ce qui les rend disposés à des éventuelles hybridations avec d'autres méthodes.

Enfin, nous pouvons dire que notre approche comporte des limites et que nous ne pouvons pas l'achever sans proposer des perspectives :
* Construire des algorithmes hybrides combinant chacun de ces algorithmes avec d'autres méthodes.
* Appliquer le principe de **ϵ-MOEA-LMO** sur des problèmes de dimension supérieure ou égale à 3.

Annexe A

Dans cette annexe nous présentons l'application et les outils de développement adoptés et utilisés. Nous avons choisi de programmer avec le langage JAVA et d'utiliser l'environnement de développement intégré (EDI) NetBeans et l'API JFreeChart qui permet de créer des graphiques.

La figure A.1 représente l'interface de notre application.

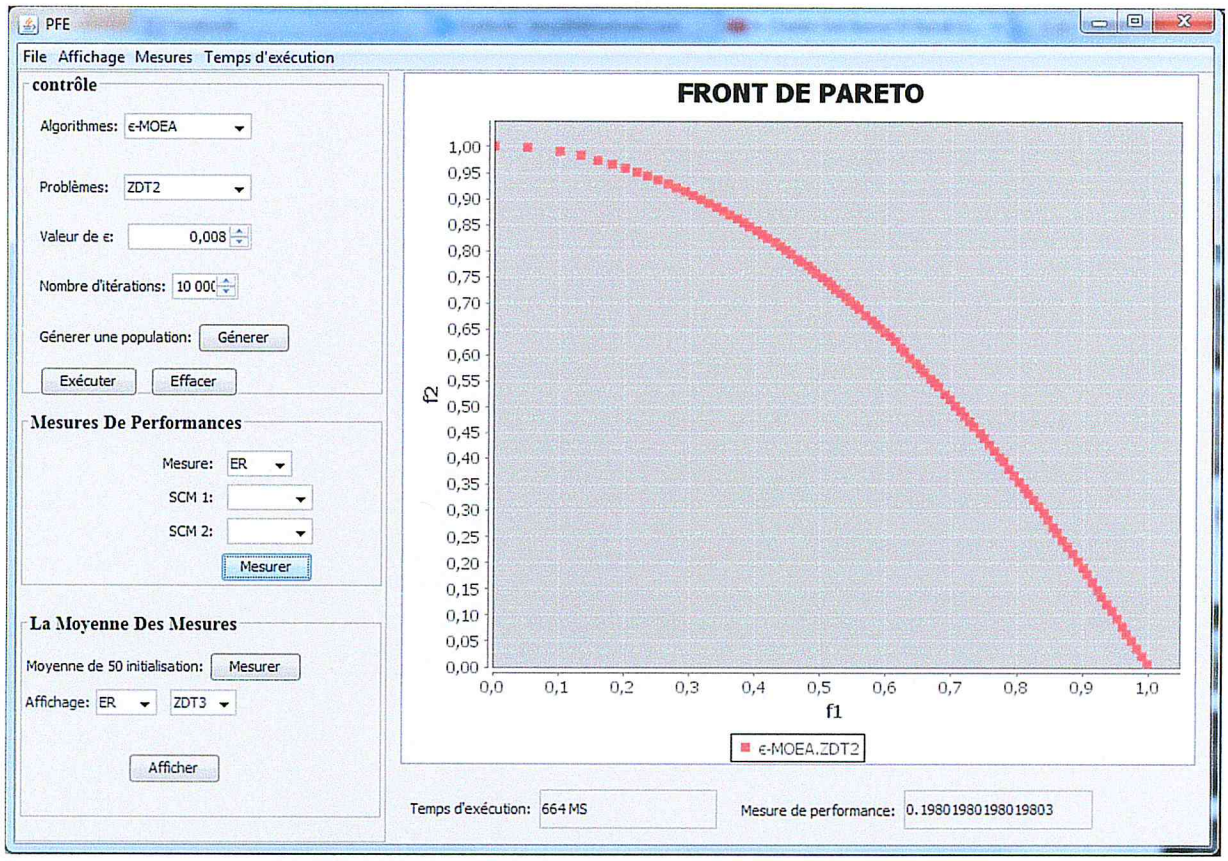


Figure A.1 - Interface principale de l'application

Description de l'Interfaces de l'application

La partie « contrôle » : permet de choisir quel algorithme nous voulons exécuter, pour quel problème test nous pouvons changer la valeur de ϵ et aussi le nombre d'itérations désiré. Cette partie contient trois boutons :

- Le bouton «générer » permet de créer une nouvelle population initiale.
- Le bouton « exécuter » permet de lancer l'exécution de l'algorithme choisi.
- Le bouton « effacer » permet de vider la zone d'affichage.

La partie « mesure de performance » : cette partie permet de mesurer la performance d'un algorithme en choisissant une des quatre mesures (ER, GD, SP, SCM). Pour la mesure ER, GD et SP, nous choisissons dans la partie contrôle, l'algorithme et le problème test sur lesquels la mesure doit être appliquée.

Pour la mesure SCM, nous choisissons dans la partie contrôle, le problème test sur lequel la mesure doit être appliquée et dans la partie mesure de performance nous choisissons les deux algorithmes (scm1 et scm2).

La partie « la moyenne des mesures» : cette partie contient deux boutons :

- Le bouton « mesurer » permet de calculer la moyenne des mesures de performance de 50 initialisations différentes.

- Le bouton « afficher » permet d'afficher les graphiques des mesures de performance (voir **figure A.2**).

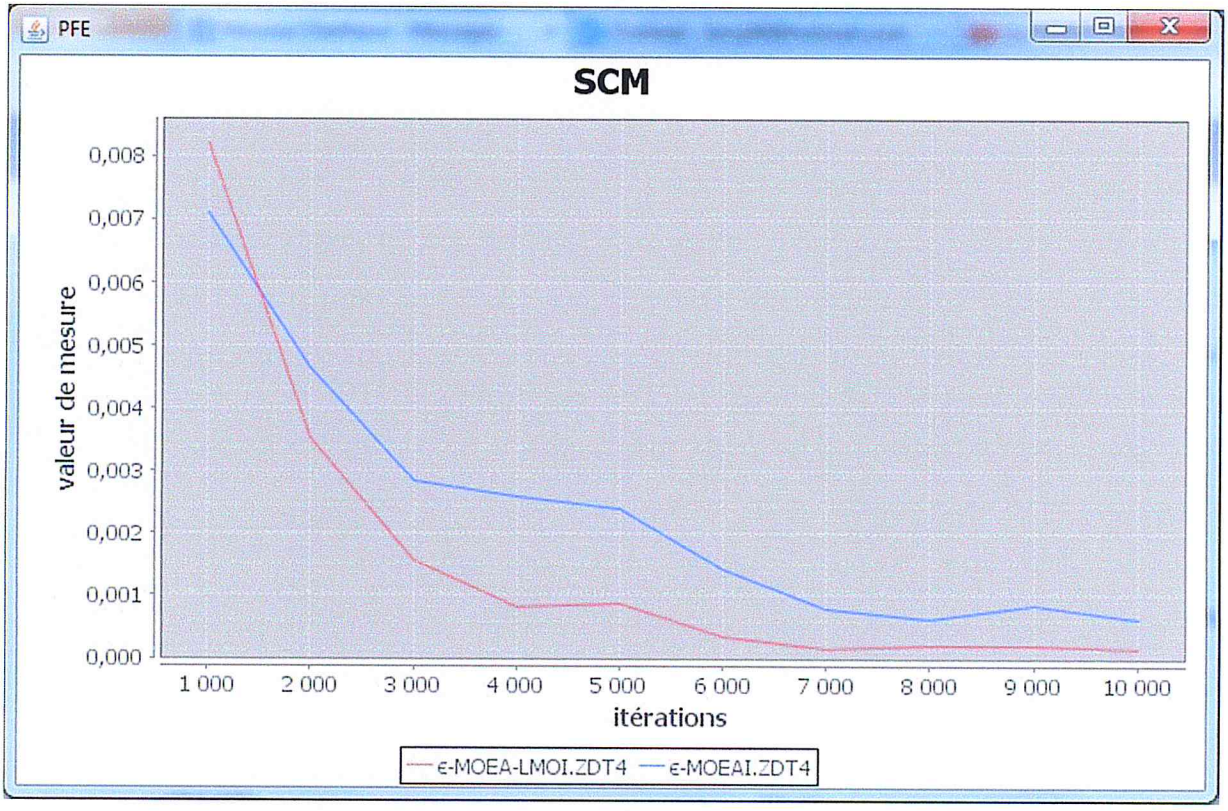


Figure A.2 – Graphe de la mesure de performance SCM

Menu affichage : quand la zone d'affichage contient deux graphes, nous pouvons les afficher séparément en sélectionnant l'item « affichage séparé ».

Menu mesure : permet de choisir quelle comparaison nous voulons effectuer en sélectionnant l'un des quatre items suivants :

- Comparer l'algorithme ϵ-MOEA-LMO avec ϵ-MOEA.

- Comparer l'algorithme ϵ-MOEA-LMO avec ϵ-MOEA-LMOI.

- Comparer l'algorithme ϵ-MOEA avec ϵ-MOEAI.

- Comparer l'algorithme ϵ-MOEA-LMOI avec ϵ-MOEAI.

Menu temps d'exécution : permet de calculer le temps d'exécution moyen de 50 exécutions

Pour chaque algorithme.

Bibliographie

[1] Yann Collecte-Patrick Siarry, optimisation multi-objectif Edition eyrolles, ÉDITIONS EYROLLES 61, Bld Saint-Germain, 75240 Paris Cedex 05, 2002.

[2] A. G. Hernández-Díaz, L. V. Santana-Quintero, C. A. Coello Coello, et J. Molina, Pareto-adaptive ε-dominance. *Evolutionary Computation,15*(4), 493-517, 2007.

[3] M. Laumanns, L. Thiele, K. Deb et E. Zitzler. Combining Convergence and Diversity in EvolutionaryMulti-Objective Optimization, Laumanns et al, 2002.

[4]V. Barichar, Approches hybrides pour les problèmes multi-objectifs, Thèse de Doctorat, Ecole doctorale d'Angers, novembre 2003.

[5]A. C. Ben Abdallah, Optimisation multi-objectif évolutionnaire, Rapport de mémoire de Master d'ingénierie Mathématique école polytechnique de Tunisie (2004/2005).

[6] M. EJDAY, Optimisation Multi-Objectifs à base de Méta-modèle pour les Procédés de Mise en Forme, Thèse de Doctorat, École doctorale n° 364 : Sciences Fondamentales et Appliquées (ParisTech), 17 mars 2011.

[7] DE Goldberg. Genetic algorithme in search, optimisation and machine learning, Addison Wesly, 1989.

[8] M. Samir, Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Méta/Exacte, Université Mentouri de Constantine, 2007.

[9] J. REGNIER, Conception de systèmes hétérogènes en Génie Électrique par optimisation évolutionnaire multicritère, LEEI (Toulouse, France), 2003.

[10] K. Deb, and D. Deb, Analyzing Mutation Schemes for Real-Parameter Genetic Algorithms, KanGAL Report Number 2012016, Aout 2012.

[11] K. Mahdi, L'optimisation multi-objectif et l'informatique quantique, Université Mentouri – Constantine, 2012.

[12] S. Amédée, & R. François-Gérard, Algorithme génétiques. Master's thesis, Université de Nice, 2004.

[13] K. Deb, Multi-objective genetic algorithms : problem difficulties and construction of test problems , technical Report CI-49/98,Dortmund, Department of computing Science/LS11, University of Dortmund, Allemagne, 8.2, 8.4, 1998.

[14] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette, editor, Proceedings of the 1st International Confe-rence on Genetic Algorithms, pages 93–100. Laurence Erlbaum Associates, 1985.

[15] K. Deb, S. Agrawal, A. Pratap, and T.Meyarivan. A fast and elitist non-dominated sorting genetic algorithm for multi-objective optimization : NSGA-II. In Proceedings of the 6th Conference on Parallel Problem Solving from Nature, pages 849–858. Springer Verlag, 2000.

[16] M. Ehrgott, A characterization of lexicographic max-ordering solutions. In: Methods of multi-criteria decision theory, proceedings of the 6th Workshop of the DGOR-Working Group Multi-criteria Optimization and Decision Theory Alexisbad. 1996.

[17] M. Ehrgott, Discrete Decision Problems, Multiple Criteria optimization classes and lexicographic Max-ordering Technical report, University of Kaiserslautern, Department of mathematics. Report in Wirtschaftsmathematik No .13, 1996.

[18] H. O. W. Hamacher and G.Ruhe. On spanning tree problems with multiple objectives. Annals of Operations Research, 52:209-230,1994.

[19] K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of wellspread paret-optimal solutions. In Proceedings of the Second Evolutionnary Multi-Criterion Optimization (EMO-03) Conference (LNCS 2632), pages 222–236, 2003.

[20]D. V. Veldhuizen. Multiobjective Evolutionnary Algorithms : Classifications, Analysis and Innovations. PhD thesis, Air Force Institute of Technology, Dayton,1999.

[21] E. Zitzler. Evolutionary Algorithms for Multi-objective Optimization : Methods and Applications. PhD thesis, Swiss Federal Institute of Technology, Suisse, Novembre1999.

[22] J. R. Schott. Fault tolerant design using single and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics, Boston, 1995.

[23]E. Zitzler, K. Deb, and L. Thiele. Comparison of multi-objective evolutionary algorithms: Empirical results. Evolutionary Computation Journal, 8(2):125-148, 2000.

[24]M. Ait Akkache, Optimisation multi-objectif, Rapport interne, Département de Mathématique, Université de Blida, 2011.

[25] N. JOZEFOWTEZ, Modélisation et résolution approchée de problème de tournée multiobjectif. PhD these, Université des Sciences et technologies de, Lille, 2004.

[26]O. Roudenko, Application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectifs avec contraintes, Ecole polytechnique, Paris 2004.

[27] G. Rudolf, Convergence analysis of canonical genetic algorithm, IEEE transaction on neural Network, 5(1):96-101, 1994.

[28] Y. Cooren, Algorithme adaptatif d'optimisation, thèse de Doctorat de l'université paris12 (Val de Marne), 2008.

[29] M.Yagoubi, Optimisation évolutionnaire multi-objectif parallèle : application à la combustion Diesel, Thése de Doctorat, Université de Paris Sud XI, 2012.

[30]K. Deb, M. Mohan et S. Mishra. A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions,Kanpur Genetic Algorithms Laboratory (KanGAL), India Institute of Technology Kanpur, PIN 208016, India 2003.

[31]H. P. Schwefel. Numerische Optimierung von computer-modellen mittels der evolutions strategie. Interdisiplinary systems research, 26, 1977.

[32] F. BOITHIAS, M. EL MANKIBI, P. MICHEL, Optimisation multi-objectif du paramétrage de régulateurs du climat intérieur, Université de Lyon – Ecole Nationale des Travaux Publics de l'Etat (ENTPE)/DGCB, 2011.