Université Saad DAHLAB de Blida

Faculté des Sciences

Département d'informatique

Mémoire présenté par :

M^elle BOUYEKHF Karima.

et

M^elle BOUCHAMA Imene.

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et Informatique.

Filière : Informatique.

Spécialité : Informatique.

Option : ingénierie de logiciel.

Sujet :

Algorithmes évolutionnaires multiobjectifs : Application à la recherche de l'optimum Lexicographique Max-Ordering

Soutenue le :02/07/2012 devant le jury composé de:

M. HAMOUDA	Président.
Mr. AIT AKKACHE Mustapha	Promoteur.
M^ME. BOUMEHDI	Examinateur.
M^r. DJENOURI	Examinateur.

Année : 2011-2012

Remerciement

Notre cursus universitaire tend à sa fin, en cette heureuse occasion

Nous remercions d'abord

ALLAH le tout puissant de nous avoir donné la force, la santé et la volonté, pour arriver à accomplir ce modeste travail.

Et puis Nos parents qui ont été la cause et le support de notre vie, Que dieu les protège.

Nous tenons à exprimer notre profonde gratitude et nos remerciements et considération à notre promoteur Mr. Ait Akkache Mustapha pour sa compréhension, sa patience, sa compétence, ses remarques et pour sa disponibilité à l'accomplissement de cette noble mission

Aussi, les membres du jury pour nous avoir honorés en acceptant de juger notre travail.

Enfin toute personne qui à participé de près ou de loin à l'achèvement de ce travail, tous nos amies, nos collègues, à toute la promo 2012 merci de tout cœur.

Dédicace

Je dédie ce modeste travail à mes très chers parents que j'aime et que j'aimerai toujours, ils étaient toujours là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.

Que dieu le tout puissant les protège.

A mes chers grand parents Ben yahia et Fatma.

A mes très chers frères et sœur : Hayet, Amine et Farouk.

A mes cher oncles Mohamed et Ahmed et tantes Fatiha, Ghania, Awaweche, Lila, et Assia.

A mon beau frère Moufid et ma belle sœur Zineb.

A mes préférées cousines : Selma, Merième, Narimene, Sara, Inesse, et a chaque cousin et cousine.

A mon adorable neveu Aymen.

A la mémoire de ma tante Anissa et mon cousin Abd Rahmene.

A toute ma famille.

A ma binôme Karima sans qui ce travail sera démuni de toute son essence.

A mon promoteur M^R AIT AKKACHE Mustapha et

A mes meilleurs amies : Meriem, karima, Messaouda , Fatima Zohra, Khadîdja, Fatima, Nadjia et Amina.

A tous mes amis.

IMENE

Dédicaces

Je dédis ce modeste travail à :

Mes très chers parents "Khaled et Malika".

Mes sœurs Nabila, Fella et Dikra.

Mes frères Mohamed et Achraf Amine

Mes oncles, tantes et grande mère.

Mes très chers amis en citant : Omar, Imene Bouchama, Imene Mohamadi , Imene ben zahra Taher, Imene zernini et à tous que j'ai pas citer.

A mon promoteur Mᴿ AIT AKKACHE Mustapha

Au enseignants du centre universitaire de Khemis Miliana, en nommant : Mʳ Hachamma mohamed et mᵉˡˡᵉ mghetria Farida

Tous ce qui m'a aidé même si par un petit souhait pour que je termine ce travail.

Karima

Résumé

Dans ce mémoire nous avons étudié un type de dominance appelé l'optimalité de Lexicographique Max-Ordering .L'optimalité au sens LEX-MO est une combinaison de l'optimalité de Pareto, Max-Ordering et ordre lexicographique ainsi, les solutions de LEX-MO sont toujours des solutions optimales au sens de Pareto. En se basant sur ce principe nous avons développé au premier temps l'algorithme MO-NSGA qui a convergé plus rapidement vers l'optimum LEX-MO que l'algorithme NSGA et, le plus souvent le front résultant est plus concentré autour de l'optimum LEX-MO que l'algorithme NSGA. Après nous avons implémenté l'algorithme LMOGA, qui:

- Est Capable de trouver l'optimum au sens LEX-MO.
- Nous a permit d'avoir un test d'arrêt bien déterminé.
- Est plus rapide que MO-NSGA.

Mots clés : optimisation multi-objective, Algorithme génétique multi-objectif, la dominance Lexicographique Max-Ordering, la dominance de Pareto, problèmes test de Zitzler, Deb et Thiele.

Abstract

In this memory we have study a kind of dominance named Lexicographic Max-Ordering optimality. LEX-MO optimality is a combination of Pareto, Max-Ordering and lexicographic optimality in addition, LEX-MO solutions are always Pareto optimal. Through this principle we have developed at first MO-NSGA algorithm which converged faster than NSGA to the LEX-MO optimum and, most of the time the resulted front from MO-NSGA is more concentrate around the LEX-MO optimum than NSGA. After we have implemented LMOGA which:

- Is capable to find LEX-MO optimum.
- Allow us to have a definite stopping test.
- Is faster than MO-NSGA.

Key words : multi-objective optimisation , multi-objective genetic algorithm ,the Lexicographic Max-Ordering dominance, Pareto dominance , Zitzler, Deb and Thiele test problems .

Sommaire

Introduction Générale..... 1

I. CHAPITRE I : Optimisation multi-objective

I.1 Introduction 4

I.2 L'optimisation mono-objective..... 4

I.3 L'optimisation multi-objective..... 6

I.4 Dominance et optimalité De Pareto..... 7

I.4.1 Concept de dominance et solutions Pareto-optimales..... 7

I.5 Optimalité au sens lexicographique 10

I.6 Optimalité au sens Max ordering..... 11

I.7 optimalité lexicographique max-ordering..... 11

I.8 Vecteur idéal et vecteur de Nadir..... 14

I.9 La convexité 14

I.10 La surface de compromis..... 15

I.10.1 La représentation de la surface de compromis..... 16

I.11 Conclusion..... 16

II. CHAPITRE II : Evolution Artificielle

II.1 Introduction..... 17

II.2. Algorithmes évolutionnaires..... 17

II.3 Principe de fonctionnement d'un algorithme évolutionnaire..... 18

II.4 Représentation d'un individu..... 21

II.4.1. Le principe..... 21

II.4.2. La représentation binaire..... 21

II.4.3. La représentation réelle..... 21

II.5 Initialisation de la population..... 22

II.6 Darwinisme Artificiel..... 22

II.6.1Sélection..... 22

II.6.2 Remplacement 24

II.7 Operateurs de variation 24

II.7.1 Croisement..... 25

- Croisement binaire 25
- Croisement réel..... 27

II.7.2 La mutation 27

- La mutation binaire..... 28
- La mutation réelle..... 28

II.8 Conclusion..... 29

III.CHAPITRE III : *AEMO* ET problème Test

III.1 Introduction..... 30

III.2 Algorithmes évolutionnaires multiobjectifs..... 30

III.2.1 Algorithmes non-Pareto..... 30

- VEGA..... 30
- VOES..... 32

III.2.2Algorithmes basés sur la dominance de Pareto..... 33

- MOGA..... 33
- NSGA..... 34

III.3 Problèmes Tests..... 36

III.1 ZDT1..... 37

III.2 ZDT2..... 38

III.3 ZDT3..... 39

III.4 ZDT4..... 40

III.5 ZDT6..... 40

III.4. Conclusion..... 41

IV.CHAPITRE IV : MO-NSGA : Description et validation

IV.1 Introduction..... 42

IV.2 Max Ordering Non dominated Sorting Genetic Algorithm (MO-NSGA).. 42

IV.3 Résultats expérimentaux de MO-NSGA..... 44

IV.3.1 Conditions expérimentales..... 44

IV.3.1 .1 Problème ZDT1..... 45

IV.3.1.2. Problème ZDT2..... 46
IV.3.1 .3 Problème ZDT3..... 46
IV.3.1 .4 Problème ZDT4..... 47
IV.3.1 .5 Problème ZDT6..... 47
IV.4 LMOGA..... 48
IV.4.1 ZDT1 49
IV.4.2 ZDT2..... 49
IV.4.3 ZDT3..... 50
IV.4.4 ZDT4..... 50
IV.4.5 ZDT6..... 50
IV.5 Conclusion..... 51
Conclusion Générale 53

Liste des figures

I.1 Relation entre kilométrage et prix..... 6

I.2 Exemple de dominance..... 8

I.3 Représentation des solutions dans le plan f1,f2..... 8

I.4 Les solutions et leurs rangs de Pareto..... 9

I.5 L'optimalité locale au sens de Pareto..... 10

I.6 Les solutions de Pareto, Lex-MO, max-ordering..... 13

I.7 Vecteur idéal Z* et vecteur de Nadir Z^{nad}..... 14

I.8 Exemple d'ensemble convexe et d'ensemble non convexe..... 15

I.9 la surface de compromis..... 15

I.10 La représentation de la surface de compromis..... 16

II.1Principe générale de fonctionnement d'un algorithme évolutionnaire typique... 20

II.2 Selection par roulette..... 23

II.3 Croisement binaire a un point..... 25

II.4 Croisement binaire a trois points..... 26

II.5 Le Croisement binaire uniforme..... 26

II.6 Mutation binaire..... 28

III.1 Principe de l'algorithme V.E.G.A..... 31

III.2 Défauts de l'attribution de la performance dans les algorithmes non-Pareto.. 33

III.3 Le calcule de l'efficacité 35

III.4 La surface de compromis de la fonction ZDT1..... 38

III.5 La surface de compromis de la fonction ZDT2..... 39

III.6 La surface de compromis de la fonction ZDT3..... 39

III.7 La surface de compromis de la fonction ZDT4..... 40

III.8La surface de compromis de la fonction ZDT6..... 41

IV.1 Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT1..45

IV.2 Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT2..46

IV.3 Évolution de MO-NSGA et NSGA au cours de résolution du problème ZD..... 46

IV.4 Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT4..47

IV.5 Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT6..47

IV.6 Évolution de LMOGA au cours de résolution du problème ZDT1..... 49

IV.7 Évolution de LMOGA au cours de résolution du problème ZDT2..... 49

IV.8 Évolution de LMOGA au cours de résolution du problème ZDT3..... 50

IV.9 Évolution de LMOGA au cours de résolution du problème ZDT4..... 50

IV.10 Évolution de LMOGA au cours de résolution du problème ZDT6..... 51

Liste des tableaux

I.1 Les valeurs d'objectifs et de Sort..... 12

IV.1Comparaison de performance entre NSGA, MO-NSGA et LMOGA..... 51

Introduction générale

Durant les trois dernières décennies du 20ème siècle, l'entreprise a subi une mutation en profondeur. Des changements notables ont modifié l'ensemble des composantes des systèmes industriels. Les éléments à l'origine de cette évolution sont nombreux : évolution rapide et importante de la technologie, rapprochement des frontières et des cultures, ouverture d'un marché au niveau planétaire, mondialisation de l'économie...

Face à cette évolution en profondeur de l'entreprise, des techniques, des méthodes et des outils nouveaux ont émergé pour assister l'utilisateur potentiel dans des domaines très divers de la production.

L'entreprise se voit assigner plusieurs objectifs simultanément (physiques, monétaires ou sociaux) à optimiser tous à la fois, afin de prendre la décision la plus optimale possible, dans un temps opportuns. Ainsi cela peut être exprimé sous la forme générale d'un ***problème d'optimisation***.

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

Plus précisément l'entreprise doit résoudre un problème de la forme ***min F(x)=*** $(F_1(x), F_2(x),\ldots, F_n(x))$ tel que x soit une solution réalisable. Les composantes du vecteur F sont les différentes fonctions (objectifs) à optimiser, qui sont la plupart du temps conflictuels les uns avec les autres. Nous sommes face à un problème d'optimisation multi objectif. La solution d'un problème multiobjectif n'est pas une solution unique mais un ensemble de solutions, pour lesquelles l'amélioration d'un des objectifs entraine systématiquement la détérioration de la qualité d'au moins un autre objectif, ces solutions sont appelées ensemble non-dominé ou surface de Pareto.

L'optimisation multiobjective, qui s'intéresse à l'étude et la résolution des problèmes multiobjectifs, possède ses sources dans les travaux d'Edgeworth **[8]** et de Pareto **[19]** dans le cadre d'études d'économie au 19eme siècle.

Cependant, l'optimisation multiobjective connaît un intérêt croissant depuis le milieu des années 1990 avec l'apparition de méthodes d'optimisation multiobjectif évolutionnaires

[26]. Actuellement, L'optimisation multiobjective est appliquée dans de nombreux domaines académiques et industriels.

Les méthodes d'optimisation multiobjective évolutionnaires sont des méthodes approchées. Le but de telles méthodes est de produire des solutions de meilleure qualité possible avec un temps de calcul raisonnable **[5].**

Le but de ce rapport, est l'étude des méthodes d'optimisation multiobjective évolutionnaire, qui ont reçu un intérêt croissant ces dernières années.
Les Algorithmes Evolutionnaires Multiobjectif (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 unique point de l'espace de recherche. Cette particularité des Algorithmes Evolutionnaires rend possible l'obtention d'un ensemble de Pareto approché en un seul essai de l'algorithme, et sans obliger l'utilisateur de définir des paramètres subjectifs supplémentaires tels les poids relatifs des critères d'optimisation. En plus ce type d'algorithmes peut être appliqué aussi bien aux problèmes discrets que continus ou mixtes **[25]**.

Le présent travail a été réalisé grâce à une série d'expérimentations, nous nous sommes intéressés aux difficultés de résolution des problèmes d'optimisation multiobjective et à l'amélioration des performances de certaines approches existantes. Nous avons commencé par étudier et mettre en œuvre un Algorithme Evolutionnaire Multiobjectif NSGA proposée par N. Srinivas, K. Deb **[6]**. Ensuite, nous avons adapté cet algorithme pour trouver une solution Lexicographique Max Ordering **[9][10]** que nous avons appelé MO-NSGA. Afin d'améliorer la convergence de ce dernier (MO-NSGA) et de réduire sa complexité nous avons développé une nouvelle approche.

Cette nouvelle approche nous a aider à implémenter un algorithme appelé Lexicographique Max Ordering Genetic Algorithm (LMOGA) qui permet :

- De trouver cette solution sans passer par la recherche de tout le front.
- De réduire le temps d'exécution d'une manière considérable.
- Et d'avoir un test d'arrêt bien déterminer.

Ce rapport comporte quatre chapitres. Nous commençons dans le premier chapitre par présenter l'optimisation multi objective. Nous introduisons des concepts fondamentaux tels que la dominance de Pareto et la dominance lexicographique Max_Ordering (Lex-Mo) sur la quelle se porte notre intérêt. Le deuxième chapitre permet d'introduire les Algorithmes Evolutionnaires ainsi que leur principe de fonctionnement. Dans le troisième chapitre, nous présentons quelques Algorithmes Evolutionnaires Multiobjectif trouvés dans la littérature. Ensuite, en se basant sur le même principe de base de l'algorithme NSGA, nous proposons deux nouvelles méthodes qu'on désigne par : MO-NSGA, et LMOGA. Le dernier chapitre est consacré à l'illustration des résultats des différents tests expérimentaux. Ensuite, nous comparons les résultats de nos approches, MO-NSGA et LMOGA, avec ceux de NSGA.

I.1 Introduction

Dans la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels. Dans les problèmes de conception, par exemple, il faut le plus souvent trouver un compromis entre des besoins technologiques et des objectifs

L'optimisation multi-objective consiste donc à optimiser simultanément plusieurs fonctions. La notion de solution optimale unique dans l'optimisation mono-objective disparait pour les problèmes d'optimisation multi-objective au profit de la notion d'ensemble de solutions Pareto optimales.

I.2. L'optimisation mono-objective [25]

Un problème d'optimisation mono-objectif consiste à rechercher la valeur minimale ou maximale, appelé optimum global d'une fonction $f: D \rightarrow \Re$ donnée. Comme maximiser une fonction f est équivalente à minimiser la fonction –f .Nous considérons dans ce manuscrit que les fonctions doivent être minimisées .On peut trouver des problèmes d'optimisation pour lesquels les variables de la fonction à optimiser sont contraintes d'évoluer dans une certaine partie de l'espace de recherche. Dans ce cas, on a une forme particulière de ce que l'on appelle un problème d'optimisation sous contraintes.

Ce besoin d'optimisation vient de la nécessité de l'ingénieur de fournir à l'utilisateur un système qui réponde au mieux au cahier des charges. Ce système devra être calibré de manière à :

– Occuper le volume minimum nécessaire à son bon fonctionnement (coût des matières premières),

– Consommer le minimum d'énergie (coût de fonctionnement),

– Répondre à la demande de l'utilisateur (cahier des charges).

D est couramment appelé espace décisionnel et le plus souvent on utilise $D \subseteq \Re^n$.Un élément $x=(x_1,\ldots x_n)$ de D est appelé une solution du problèmes d'optimisation .Les composantes x_i de x sont les variables de décision .C'est en faisant varier les valeurs de ces variables que l'on modifie la valeur de la fonction objectif .Une solution $x \in D$ est dite réalisable si elle respecte l'ensemble de contraintes du problème. L'ensemble $S \subseteq D$ est l'ensemble des solutions réalisables que l'on appelle ensemble réalisable.

Mathématiquement parlant, un problème d'optimisation se présentera sous la forme suivante: [25]

Minimiser $f(\vec{x})$	(fonction à optimiser)
Avec $\vec{g}(\vec{x})>0$	(k contraintes d'inégalité)
Et $\vec{h}(\vec{x})=0$	(p contraintes d'égalité)

On a $\vec{x} \in \mathbf{R}^n$, $\vec{g}(\vec{x}) \in \mathbf{R}^k$ et $\vec{h}(\vec{x}) \in \mathbf{R}^p$.

Ici, les vecteurs $\vec{g}(\vec{x})$ et $\vec{h}(\vec{x})$ représentent respectivement k contraintes d'inégalité et p contraintes d'égalité. L'optimum global d'un problème d'optimisation peut alors se définir ainsi :

Définition [16] : Pour $S \neq \emptyset$ *on note* $f^* = f(x^*)$ avec $f^* > -\infty$ et $x^* \in S$ si :

$\forall x \in S$, $f^* \leq f(x)$, f^* est appelé l'optimum global de (S, f).

Il est à noter que x^* peut ne pas être unique : cependant il n'existe qu'un unique optimal global f^*.

La formulation précédente était relative à un problème dans lequel on recherchait un optimum pour une fonction objectif (f) unique.

La principale difficulté que l'on rencontre en optimisation mono-objectif vient du fait que modéliser un problème sous la forme d'une fonction unique peut être une tâche difficile et souvent cela peut biaiser la modélisation comme par exemple le cas d'une personne souhaitons acheter une voiture d'occasion. La voiture idéale est celle qui est peu chère avec peu de kilomètres, mais cette voiture idyllique n'existe pas. Notre acheteur va donc devoir identifier les meilleurs compromis possibles correspondant à son budget (**voire figure I.1**) Dans ce cas, on parle de problème d'optimisation multi-objectif (ou problème d'optimisation multicritère pour le cas discret).

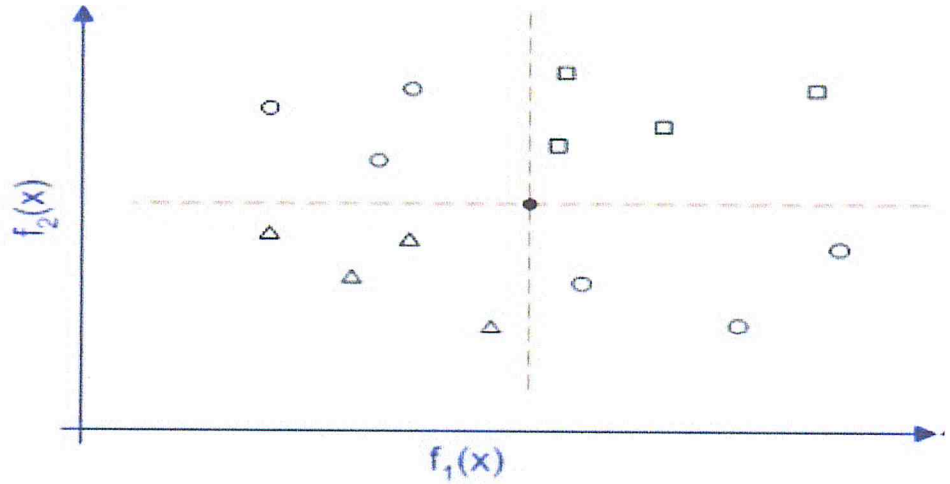


Fig. I.2 – Exemple de dominance [25]

Exemple

Pour mieux comprendre ce qu'est une relation de dominance, nous allons traiter un exemple. On considère un problème à deux objectifs : maximiser *f*1 et minimiser *f*2. Pour ce problème, on représente un ensemble de solution dans le plan *f*1, *f*2 (voir figure **I.3**).

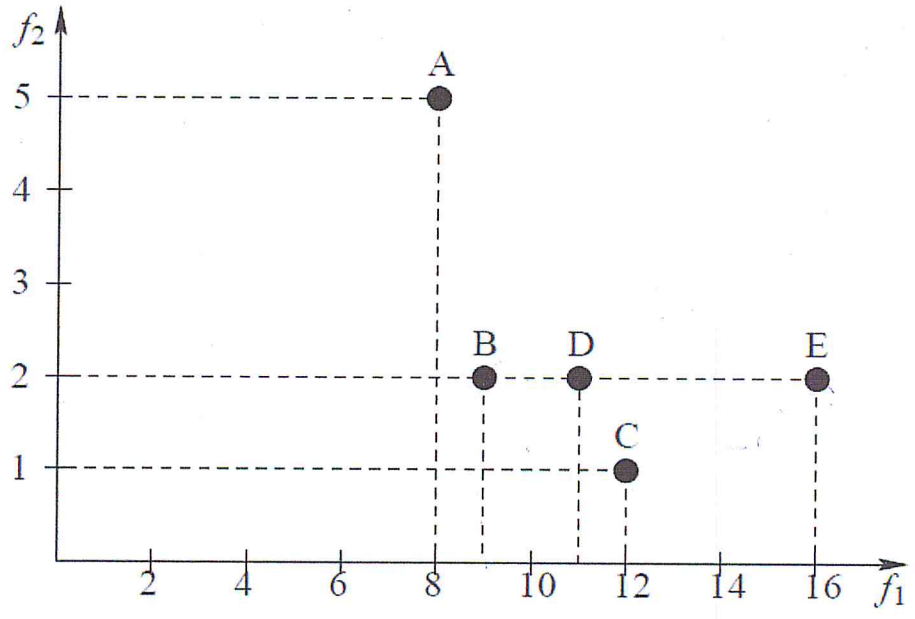


FIG. I.3– Représentation des solutions dans le plan *f*1, *f*2.[25]

Les deux points (*E* et *C*) de la figure **I.3** *forment* ***l'ensemble des points non dominés***. On peut 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 rang 1. La figure **I.4** représente les différents points et leur rang.

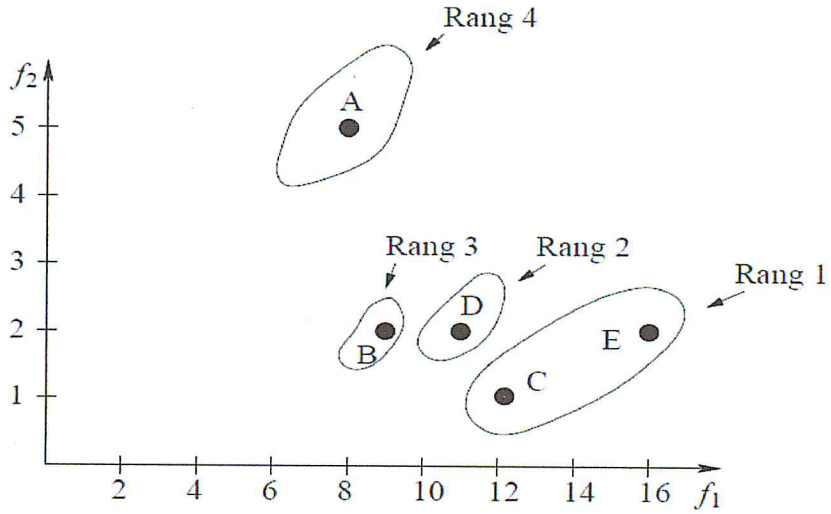


FIG. I.4– Les solutions et leurs rangs de Pareto[25].

Définition 2 (Ensemble de Pareto) [2][25][21]

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,

$$S^* = \{x \in S | \nexists x' \in S : x' < x\}$$

L'image de l'ensemble de Pareto dans l'espace des critères est appelée ***la surface de Pareto*** (ou ***le front de Pareto***)

Définition 3 (optimalité locale au sens de Pareto) [1] [20] [25]

Le vecteur $\vec{x} \in R^n$ est optimal localement au sens de Pareto s'il existe un réel $\delta > 0$ tel qu'il n'y ait pas de vecteur $\vec{x}\,'$ qui domine le vecteur $\vec{x}$ avec $\vec{x}\,' \in R^n \cap B(\vec{x}, \delta)$, où $B(\vec{x}, \delta)$ représente une boule de centre $\vec{x}$ et de rayon δ.

Un vecteur $\vec{x}$ est donc optimal localement au sens de Pareto s'il est optimal au sens de Pareto sur une restriction de l'ensemble R^n. Cette définition est illustrée par la figure **I.5**

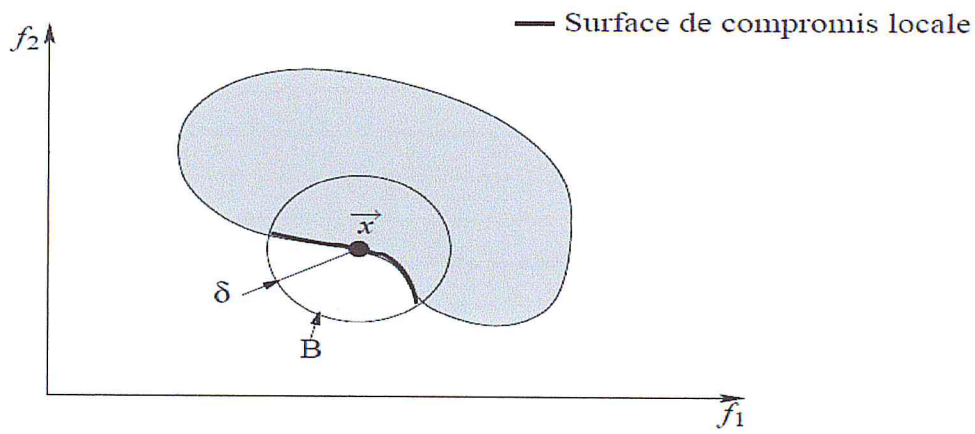


FIG. I.5– L'optimalité locale au sens de Pareto [1][25]

Définition 4 (optimalité globale au sens de Pareto)[1][20][25]

Un vecteur $\vec{x}$ est optimal globalement au sens de Pareto (ou optimal au sens de Pareto) s'il n'existe pas de vecteur $\vec{x}$ 'tel que $\vec{x}$ ' domine le vecteur$\vec{x}$.

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

I.5. Optimalité au sens lexicographique [9] [25]

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

$$\vec{x}^* \leq_{lex} \vec{x}. \forall \vec{x} \in \Re^n - \{\vec{x}^*\}$$

Si $\overrightarrow{x}, \vec{y} \in \Re^n$, on dit que $\overrightarrow{x} \leq_{lex} \vec{y}$ s'il existe une valeur d'index q^* telle que $x_q = y_q$ pour $q = 1, \cdots, (q^*-1)$ et $x_q^* < y_q^*$. Les relations entre x_q et y_q 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 $x_q < y_q$).

Cette définition implique que l'utilisateur ait rangé par ordre d'importance les différents objectifs. La comparaison entre les deux solutions se fera dans l'ordre de classement des objectifs. Illustrons l'utilisation de cette relation en prenant un exemple.

Soient deux points A et B :

$$A = (1, 2, 3, 4, 5, 6)$$
$$B = (1, 2, 3, 9, 4, 9)$$

Pour ces deux points, on a $A \leq_{lex} B$ car, jusqu'à la troisième position, on a $Ai = Bi$,

$i = 1, 2, 3$ et, pour la quatrième position, on a $4 < 9$. On conclusions la solution **A domine lexicographiquement la solution B.**

I.6 Optimalité au sens Max-ordering (optimalité maximale)[9][25]

Une solution $\vec{x}^* \in \mathfrak{R}^n$ est max-ordering optimale si la valeur du pire (ici le max) objectif est aussi petite que possible

$$\text{Max } x_q^* \leq \text{Max } x_q$$
$$q \in \{1, \cdots, n\} \qquad q \in \{1, \cdots, n\}$$
$$\vec{x} \in \mathfrak{R}^n - \{\vec{x}^*\}$$

I.7 optimalité lexicographique max-ordering [10]

Ce type d'optimalité combine les caractéristiques de Pareto, max-ordering et l'optimalité lexicographique qui (ont été définis dans le chapitre 1).

Définition 1: Pour tout élément $x \epsilon \mathfrak{R}^n$ on définit $Sort\,(x) = (Sort_1\,(x), ..., Sort_n\,(x))$ comme étant le vecteur contenant les composantes de x dans un ordre décroissant :

$$Sort_1\,(x) \geq .... \geq Sort_n\,(x).\{x_1, ..., x_n\} = \{Sort_1\,(x), ..., Sort_n\,(x)\}$$

Définition 2 une solution faisable $x \epsilon \mathfrak{R}^n$ est dite solution lexicographique max-ordering (Lex-MO) si le vecteur des objectifs associe à x est lexicographiquement minimal par rapport à $\text{Sort}(f(x))$ où $f = (f_1, f_2, \ldots\ldots, f_n)$ représente des fonctions objectives. Donc

$$Sort\,(f\,(x)) \leq_{lex} Sort\,(f\,(x))\; x' \epsilon \mathfrak{R}^n$$

Voici ci-dessous l'algorithme qui permet de tirer l'optimum au sens LEX-MO :

Algorithme I.1 [10] :

For j=1,…, N $\grave{F}_j$= Sort (F_j (X))

Min =1;

i:=1

While (i<N) do

Begin

if ($\grave{F}_i$<lex $\grave{F}_{min}$) then min:=i;

i:=i+1;

End

Opt={ aj: $\grave{F}_j$=$\grave{F}_{min}$}

- où N représente le nombre de la population

Exemple : Maintenant nous présentons un simple exemple tiré de [10].qui illustre la relation entre la solution de LEX-MO et de Pareto.

Considérons un problème dont l'ensemble réalisable de la solution est F ={a, b, c, d, e}. Supposons que les valeurs de la fonction objective et les vecteurs Sort sont présentés dans le tableau I.1.

F	F(x)	Sort(F(x))
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)

Table I.1 les valeurs des objectifs et de sort [9] [10]

Il est à noter que *a, b, c* et d sont des solutions Pareto. La solution optimale lexicographiquement 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.

D'après [9] : les solutions de LEX-MO sont toujours des solutions optimales au sens de Pareto et optimales au sens max ordering. La preuve de ces résultats on été fait dans [9].Nous les illustrons on utilisant les données (de l'exemple précédant).

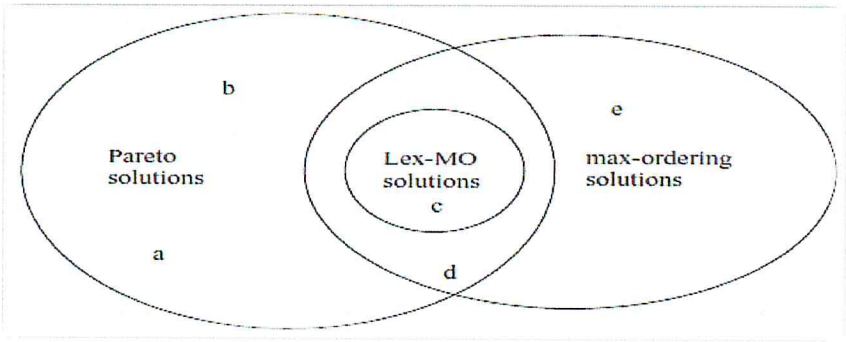


Fig. I.6 : les solutions de Pareto, LEX-MO, max-ordering [9]

Il est clairement remarquable que :

-L'ensemble des solutions optimales de Pareto peut être très grand pour un décideur. Nous renvoyons à [18] et [14], où des exemples sont donnés dans lesquelles tous les solutions réalisable sont un optimum de Pareto.

- Pour l'optimalité de max-ordering, la valeur de la solution optimale est définie d'une manière unique, mais pas forcement toutes les solutions max- ordering optimales sont optimale au sens de Pareto, il existe au moins une seule solution qui est Pareto optimale. Autre faiblesse est que, une fois que les objectifs sont évalués pour une solution réalisable. Seulement une (la plus mauvaise) détermine si la solution est considérée comme optimale ou non. Donc le décideur peut demander: pourquoi utiliser plusieurs objectifs pour évaluer une solution?

- Il est évident que la comparaison des solutions en fonction de l'ordre lexicographique implique un classement des objectifs (f1 est plus important que f2 et ainsi de suite). Un décideur peut ne pas être prêts, ou même être incapables de le faire, parce qu'il est indifférent envers les objectifs. Mais encore une fois la valeur de la solution optimale est unique.

Pour conclure on peut dire que le LEX-MO combine les caractéristiques de Pareto, Max-ordering et ordre lexicographique .Ce qui revient à considérer toutes les fonctions objectives, solution optimale unique, et ordre lexicographique. En combinant ces caractéristiques nous allons non seulement être en mesure de conserver l'avantage de ces trois définitions d'optimalité, mais nous pouvons en même temps se débarrasser des principaux inconvénients tels que la grande cardinalité de l'ensemble des solutions optimales et de la négligence des fonctions objectives.

I.8. Vecteur idéal et vecteur de Nadir [25]

Le vecteur idéal z* du problème d'optimisation multi-objectifs est le vecteur de l'espace des critères dont chaque composante z_m est la solution du problème de minimisation de la fonction f_m sous les contraintes du problème d'optimisation. Généralement, ce vecteur ne correspond pas à une solution de l'espace de décision mais il est quelques fois utile en tant qu'une référence, par exemple, lors de la normalisation des valeurs des objectifs. À la différence du vecteur idéal qui représenté les bornes inferieures 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 **1.7**). Ce vecteur est bien plus difficile de trouver que le vecteur idéal. Pour normaliser chaque objectif, le vecteur idéal et le vecteur de Nadir sont en particulier utilises de façon suivante

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

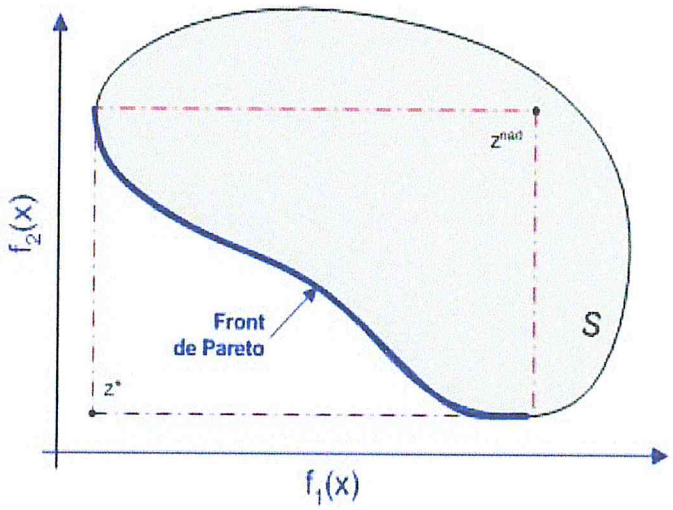
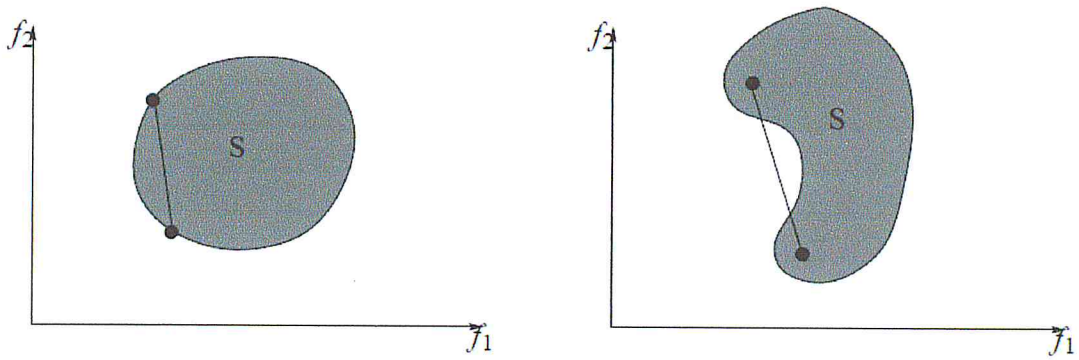


Fig. I.7– Vecteur idéal Z* et vecteur de Nadir z^{nad}[25]

I.9 La convexité [25]

Un ensemble *S* est convexe si, étant donnés deux points distincts quelconques de cet ensemble, le segment qui relie ces deux points est contenu dans l'ensemble *S*.

Un exemple d'ensemble convexe et un exemple d'ensemble non convexe sont représentés à la figure **I.8**.



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

FIG. I.8 – Exemples d'ensemble convexe et d'ensemble non convexe[25].

I.10 La surface de compromis [25]

Le petit nombre de solutions de rang 1 que l'on a sélectionnées en utilisant la règle de classement basée sur la définition de la dominance de Pareto forme ce que l'on appelle la ***surface de compromis*** (ou ***front de Pareto***).

Imaginons que nous ayons un problème à deux objectifs (minimiser f_1 et minimiser f_2 sous les contraintes $\vec{g}(\vec{x}) \leq 0$ et $\vec{h}(\vec{x}) = 0$) :

– On appelle S l'ensemble des valeurs du couple $(f1\ \vec{x}), f2\ (\vec{x}))$ quand $\vec{x}$ respecte les contraintes $\vec{g}(\vec{x})$ et $\vec{h}(\vec{x})$.

– On appelle P la surface de compromis.

On représente S et P sur la figure **I.9.**

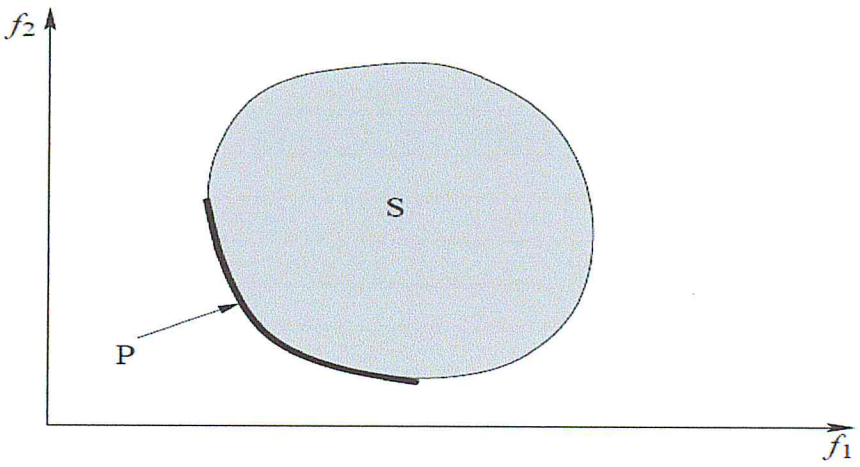
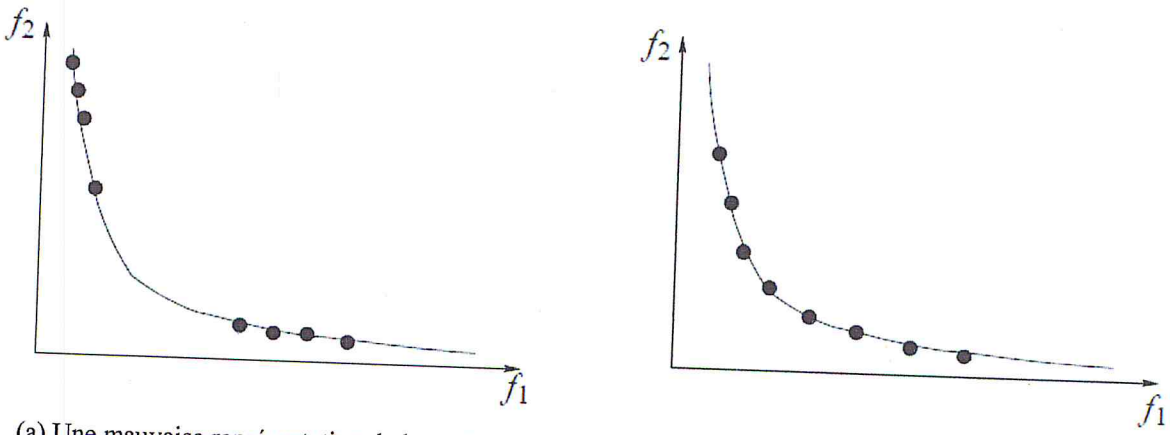


FIG. I.9 –la surface de compromis [25].

I.10.1 La représentation de la surface de compromis

Toutes les représentations de la surface de compromis, pour un même problème, ne sont pas équivalentes. En effet, la représentation idéale de la surface de compromis devra être constituée de points solution de notre problème répartis de manière uniforme sur la surface de compromis (voir figure **I.10**).

Dans le premier cas, les points représentant la surface de compromis ne sont pas répartis de manière uniforme. La détermination d'une bonne représentation de la surface de compromis sera un critère de choix d'une méthode d'optimisation multiobjectifs.



(a) Une mauvaise représentation de la surface de compromis

(b) Une bonne représentation de la surface de compromis

FIG. I.10– La représentation de la surface de compromis. [25]

I.11.Conclusion

Nous avons défini dans ce chapitre les problèmes d'optimisation multi-objectifs et les éléments de base relatifs. Comme nous avons pu le voir tout au long de ce chapitre, l'optimisation multi-objective n'est pas une tâche facile. Nous consacrons le prochain chapitre à la présentation des algorithmes génétique qui ont été proposées pour résoudre ces problèmes.

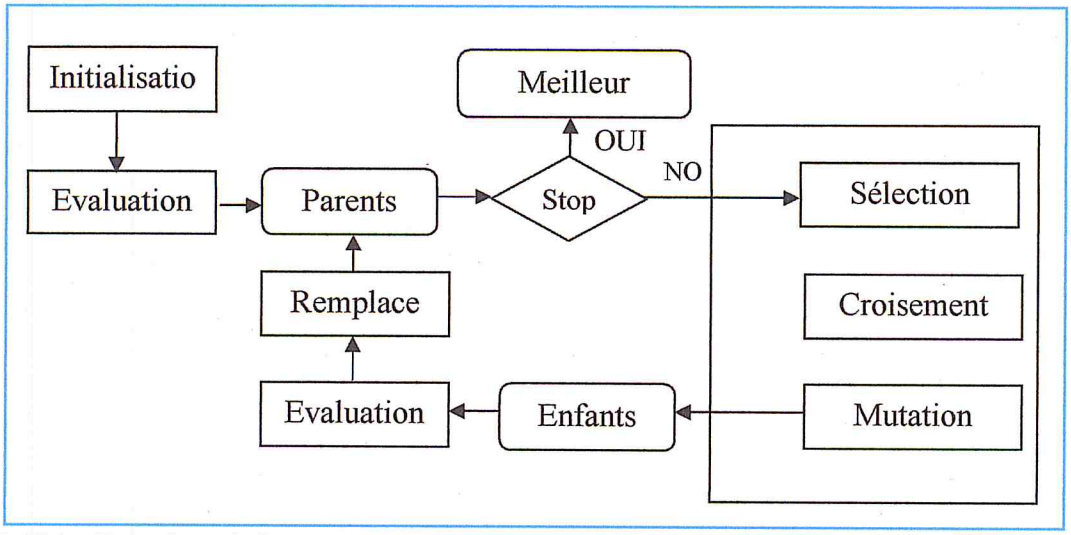


FIG. II.1 – Principe général de fonctionnement d'un algorithme évolutionnaire typique [2]

- **Exploitation et exploration**

A chacune des étapes de l'algorithme décrit ci-dessus, il est important de réaliser un compromis entre l'exploitation et l'exploration **[20,25]** .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 des 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 des informations innovatrices. 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 [2]

Le codage (ou représentation) d'un individu doit :

- il doit aussi être manipulable par des opérateurs de variation (transformations génétiques : mutation, croisement, ...).
- permettre une transition facile simple et efficace vers l'espace de recherche.

Il existe trois principaux type de codage : binaire, réel et gray.

II.4.2. La représentation binaire

Le codage binaire est le cadre général des AE traditionnels **[13]** .Chaque individu A est représenté par un vecteur binaire (ou chaîne de bits), ou chaque élément prend la valeur 0 ou 1 .$A= (a_1,\dots a_l) \in \{0,1\}^l$, où l est la taille du vecteur (nombre de bits).

II.4.3. La représentation réelle

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

Cette représentation a été introduite initialement pour les Stratégies d'Evolution **[20]**, mais son utilisation s'est étendue rapidement aux autres types d'Algorithmes Evolutionnaires.

Pour des problèmes d'optimisation dans l'espace réel, l'espace des génotype s'identifie à l'espace des phénotypes : $A=\vec{x}= (x_1,\dots,x_n) \in \Re^n$.

Nous pouvons facilement passer d'un codage à l'autre**[20]** .Certains auteurs n'hésitent pas à faire le parallèle avec la biologie et parlent de génotype en ce qui concerne la représentation binaire d'un individu, et de phénotype pour ce qui est de sa valeur réelle correspondante dans l'espace de recherche. Rappelons que la transformation la plus simple d'une chaîne binaire A en nombre entier x s'opère par la règle suivante :

$$X=d(A)= \sum_{i=1}^{l} a_i \, 2^{l-i-1}.$$

Où l est la taille du vecteur (nombre de bits), et $a_i \in \{0,1\}$.

Ainsi, admettons que nous cherchons à maximiser $\mathcal{F}$ en fonction d'une variable réelle x.

Soit $S = [x_{min} , x_{max}]$ avec $S \subseteq \Re^n$, est l'espace de recherche permis avec x_{min} et x_{max} les bornes inférieures et supérieures. Soit nb le nombre de bits du chiffre binaire A .

Soit $ld = x_{max} - x_{min}$ la longueur de l'intervalle S.

Et $n=d(A)= \sum_{i=1}^{l} ai \; 2^{l-i-1}.$

L'idée, c'est de partager en tranches l'intervalle S, qui se compose de ld unité. Ces ld unités on va les trancher en $2^{nb} - 1$ part égales.

La valeur réelle correspondante de A :

$$X = n.\frac{1}{2^{nb}-1} \ .ld + x_{min}$$

II.5 Initialisation de la population

Pour ce qui est de la phase d'initialisation, la procédure est assez simple. Elle consiste en un tirage aléatoire de *N* individus dans l'espace de recherche S en veillant éventuellement à ce que les individus produits respectent les contraintes. En codage binaire, selon la taille *l* de la chaîne, nous effectuons pour un chromosome *l* tirage dans *{0,1}* avec équiprobabilité.

II.6 Darwinisme artificiel

La partie darwinienne de l'algorithme évolutionnaire comprend deux étapes : l'étape de reproduction afin de sélectionner les parents qui vont se reproduire et l'étape de remplacement.

II.6.1Sélection [2, 4, 20, 25]

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 convergence).

On trouve dans la littérature un nombre important de stratégies de sélection plus ou moins adaptées aux problèmes qu'elles traitent. Nous présentons ici les procédures de sélection plus fréquemment rencontrées.

- **Sélection par roulette (*roulette wheel Selection*) :**

La sélection des individus par le système de la roulette de Goldberg (1989)**[13]**. S'inspire des roues de loterie. A chacun des individus de la population est associé un secteur d'une roue. L'angle du secteur étant proportionnel à la qualité de l'individu qu'il représente. Vous tournez la roue et vous obtenez un individu. Les tirages des individus sont ainsi pondérés par leur qualité. Et presque logiquement, les meilleurs individus ont plus de chance d'être croisés et de participer à l'amélioration de notre population.

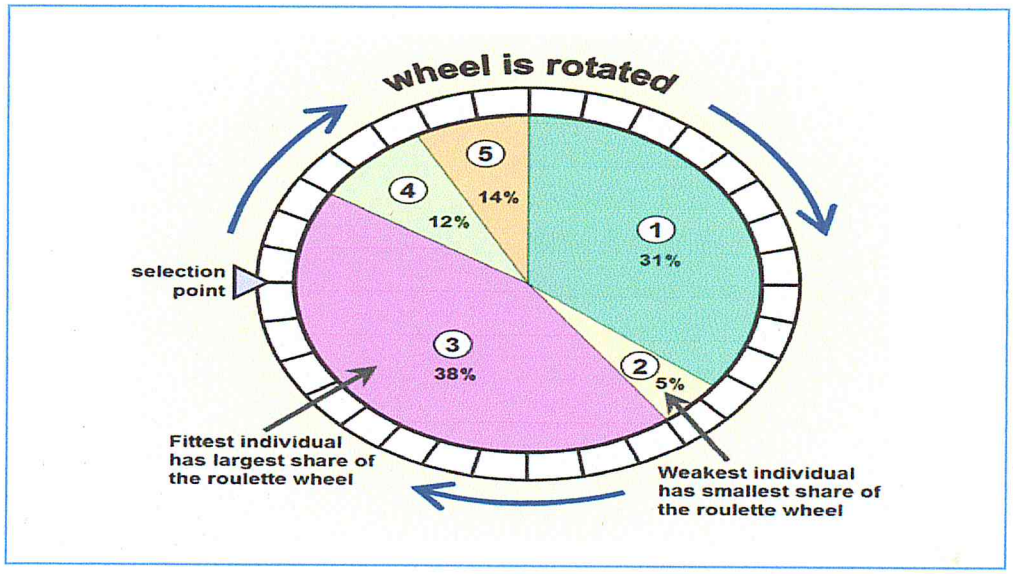


FIG II.2 - Sélection par roulette [4]

- **Sélection par rang**

La sélection par rang est une variante du système de roulette. Il s'agit également d'implémenter une roulette, mais cette fois ci les secteurs de la roue ne sont plus proportionnels à la qualité des individus, mais à leur rang dans la population triée en fonction de la qualité des individus.

D'une manière plus parlante, il faut trier la population en fonction de la qualité des individus puis leur attribuer à chacun un rang. Les individus de moins bonne qualité obtiennent un rang faible (à partir de 1). Et ainsi en itérant sur chaque individu on finit par attribuer le rang N au meilleur individu (où N est la taille de la population). La suite de la méthode consiste uniquement en l'implémentation d'une roulette basée sur les rangs des individus. L'angle de chaque secteur de la roue sera proportionnel au rang de l'individu qu'il représente.

- **Sélection par tournoi**

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 du tournoi. Pour sélectionner un individu, on en tire T uniformément dans la population, et on sélectionne 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 bon que lui). De plus, elle est la moins chère en termes de coût d'exécution, facilement paramétrable par la valeur de T **[7].**

II.6.2 Remplacement

Diverses stratégies de remplacement peuvent être utilisées, le principe étant de 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 : **[2,20]**

- **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 maintenir une stratégie de recherche cohérente).

Une autre technique de remplacement est **le remplacement déterministe [20]:**

Le remplacement déterministe est utilisé typiquement dans les Stratégies d'évolution .Son caractère purement déterministe lui donne un rôle clef dans l'évolution vu qu'il guide la recherche vers les zones des meilleurs individus. Il opère en sélectionnant les $1 < \mu \leq \lambda$ meilleures solutions parmi :

-l'union de μ parents et λ enfants : schéma appelé $(\mu + \lambda)$-ES.

-l'ensemble de λ enfants : schéma appelé (μ, λ)-ES.

Le remplacement $(\mu + \lambda)$ est élitiste, et 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, λ), on peut perdre les meilleurs individus, mais l'algorithme est plus flexible avec les changements d'environnement.

II.7 Operateurs de variation

Les opérateurs de variation ont pour but de produire de nouveaux individus à partir de ceux préalablement sélectionnés. On distingue les opérateurs de croisement et les opérateurs de mutation.

II.7.1 Croisement [2]

Il 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

$c : E \times E \rightarrow E \times E$, qui croise, avec une certaine probabilité p_c $(0 \leq p_c \leq 1)$, deux parents $(P_1, P_2) \in E \times E$. 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 :

- **Croisement binaire**

C'est un opérateur sur $E \times E \rightarrow E \times E$, avec $E = \{0, 1\}^l$. Il correspond à un échange de gènes (bits) entre les parents. Il en existe plusieurs variantes :

- **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 P1 et P2, et construire deux nouveaux individus (enfants) E1 et E2 en échangeant leurs gènes de part et d'autre de ce point.

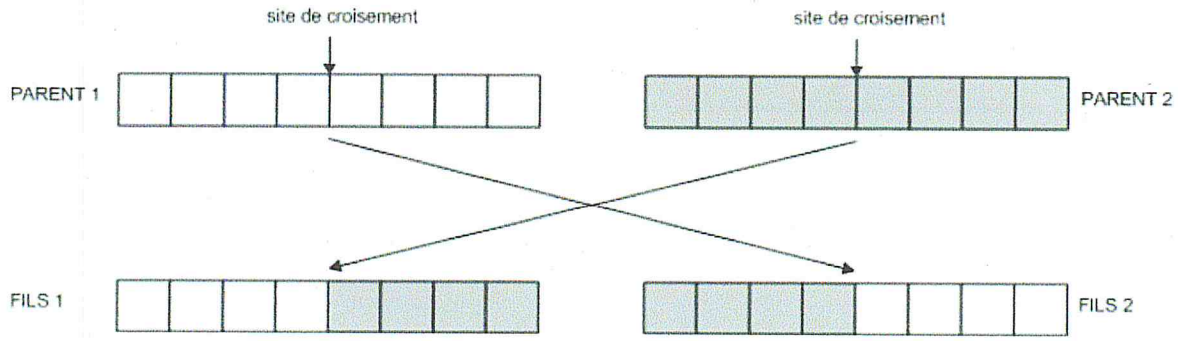


FIG II.3 - Croisement binaire à un point [2]

Le fait de choisir un seul point de croisement biaise l'effet du crossover : 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**

Le croisement multi-points évite ce problème en considérant les chromosomes comme circulaires plutôt que linéaires et en les choisissant k points de coupure. La figure 2.3 présente un exemple de croisement multi-points avec k = 3.

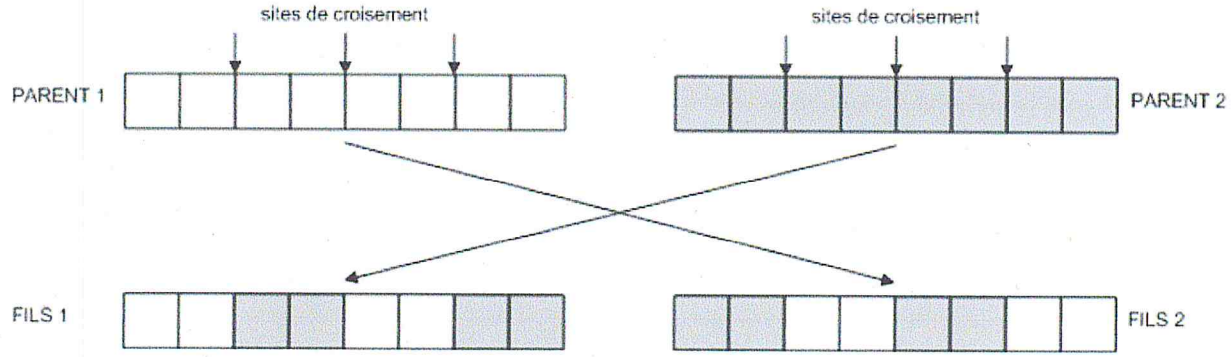


Fig. II.4 – Croisement binaire à trois points [2]

- **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 2.4).

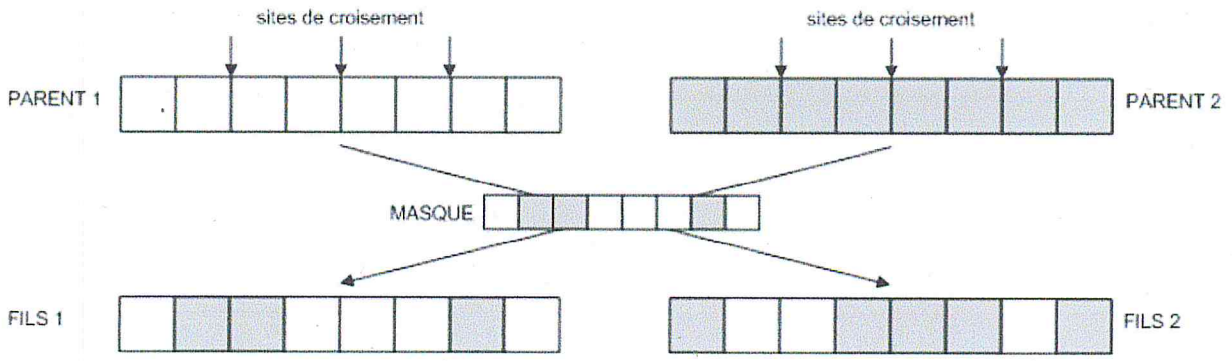


Fig. II.5 – Le croisement binaire uniforme [2]

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 obéissent néanmoins à un principe commun, l'échange d'information entre individus.

- **croisement réel**

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 arithmétique (barycentrique)

La représentation réelle permet cependant de développer toute une série de nouveaux types de croisement, principalement à base de combinaison linéaire de deux individus (Vecteurs réels). On peut trouver dans une étude du croisement arithmétique.

Il consiste à choisir deux gènes P1(i) et P2(i) dans chacun des parents à la même position i, et à définir les gènes correspondants E1(i) et E1(i) chez les enfants par combinaison linéaire :

$$E_1(i) = \alpha P1(i) + (1-\alpha)P2(i)$$
$$E_2(i) = (1-\alpha) P1(i) + \alpha P2(i)$$

Où α est la réalisation d'une variable aléatoire uniforme appartenant à l'intervalle [0, 1].

Quoi qu'il en soit, il se peut que l'action conjointe de la reproduction et du croisement soit insuffisante pour assurer la réussite de l'AG. Ainsi, dans le cas du codage binaire certaines informations peuvent disparaître de la population. Ainsi aucun individu de la population initiale ne contient de 1 en dernière position de la chaîne, et que ce 1 fasse partie de la chaîne optimale à trouver, tous les croisements possibles ne permettront pas de faire apparaître ce 1 initialement inconnu. En codage réel, une telle situation peut arriver si on utilisant un opérateur simple de croisement, il se trouvait qu'initialement toute la population soit comprise entre 0 et 40 et que la valeur optimale était de 50. Toutes les combinaisons convexes possibles de chiffres appartenant à l'intervalle [0,40] ne permettront jamais d'aboutir à un chiffre de 50. C'est pour remédier entre autre à ce problème que l'opération de mutation est utilisée.

II.7.2 La mutation [2]

L'idée générale de la mutation est la modification (avec une certaine probabilité pm, $0 \leq pm \leq 1$) d'un ou plusieurs gènes de l'individu sélectionné, afin d'introduire de la variabilité dans la population. Cet opérateur agit de S dans S. 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 réintroduction de diversité perdue et la lutte contre les minima locaux (c'est-à-dire les au minima non absolu du problème).

- **La mutation binaire [2]**

La mutation binaire est une modification aléatoire de la valeur d'un gène qui se produit avec une probabilité fixée pm 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.

- $\frac{c}{l}$ ***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.

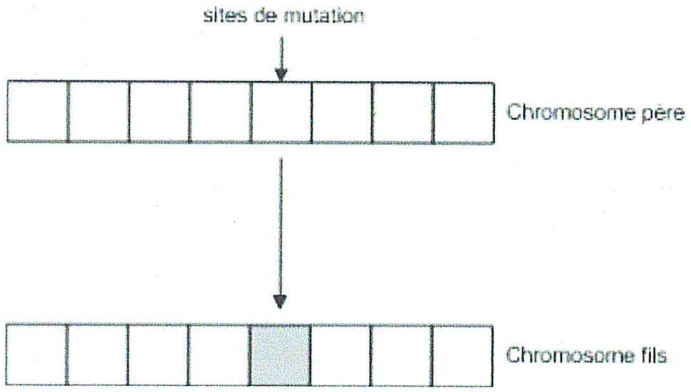


Fig. II.6- Mutation binaire[2]

- **La mutation réelle**

Le principe de l'opérateur de mutation réelle consiste généralement à ajouter une perturbation aléatoire tirée selon une distribution de probabilité Gaussienne aux différentes composantes de l'individu X.

$$xi := xi + \sigma N(0, 1)$$

Où σ et N (0, 1) sont respectivement l'écart type (déviation standard) de la mutation et la loi gaussienne centrée réduite.

xi peut être calculé de la façon suivante

$$xi=xi+\sqrt{R}(\cos(\Theta), \sin(\Theta))\ \sigma$$

Où $\Theta = 2\pi u_2$ et $\sigma=2 \ln(u_1)$ u_1 et u_2 sont des variables aléatoires

La difficulté de cette approche est l'ajustement des déviations standards des variables Gaussiennes utilisées. En effet, si la déviation standard est trop petite, les déplacements dans l'espace de recherche sont insuffisants au début de l'algorithme, et

l'algorithme peut rester au voisinage d'un optimum local et ne permet pas de visiter des nouvelles régions de l'espace de recherche. Par contre, si l'écart est élevé, l'algorithme pourra accéder à une région contenant l'optimum mais la qualité de convergence ne sera pas bonne. Ainsi au début de l'évolution, la déviation standard σ doit être assez élevée pour explorer rapidement l'espace de recherche, et en fin la convergence peut devenir plus faible pour permettre une meilleure exploration des solutions.

II.8 Conclusion

Sous la pression du milieu, les individus se reproduisent, se croisent (échangent des informations) et mutent (explorent de nouvelles régions de l'espace de recherche). Ainsi, au bout d'un certain nombre de générations, on espère que les individus les plus performants vont apparaitre dans la population (les optima de F).

Dans ce chapitre nous avons présenté les algorithmes évolutionnaires en toute généralité. Il faut en retenir leur souplesse d'utilisation, que nous allons détailler dans le chapitre suivant.

Chapitre IIII

AEMO ET PROBLEMES TESTS

III.1 Introduction

Comme nous l'avons mentionné dans le chapitre précédent, l'idée maîtresse des Algorithmes Evolutionnaires l'ors de la recherche de la zone de Pareto, à la différence des méthodes de résolution "classiques", consiste à obtenir toute une population de solutions incomparables deux à deux, au lieu de chercher une seule solution. En laissant le choix à l'utilisateur ou le décideur. Cette différence donne aux AE un avantage crucial quand il s'agit de résoudre un problème d'optimisation multi-objectif ou, plus exactement, de trouver de multiples solutions Pareto-optimales.

Dans ce chapitre, nous exposons, dans un premier temps un certain nombre d'Algorithmes Evolutionnaires Multi-objectif décrits dans la littérature. Dans un second temps, nous listons certains problèmes tests utilisés pour valider ces algorithmes.

III.2 Algorithmes évolutionnaires multi-objectif

L'optimisation multicritère a déjà été abordée par plusieurs auteurs dans la littérature. Parmi les méthodes les plus répandues il convient de mentionner VEGA (Vector *Evaluated Genetic Algorithm*) **[23],** Vector-Optimized Evolution Strategy (VOES) **[2],** MOGA (*Multiple Objective Genetic Algorithm*) **[12],** NSGA (*Nondominated Sorting in Genetic Algorithm*) **[27]**.

Ces méthodes sont classées soit dans la catégorie des algorithmes non Pareto ou dans la catégorie basés sur la dominance de Pareto. Suivant est ce qu'elles tiennent en compte ou pas le concept de dominance de Pareto lors de la comparaison des différents individus.

III.2.1 Algorithms non-Pareto

- ✓ **VEGA (Vector Evaluated Genetic Algorithm)**

Cette méthode permet de traiter un problème d'optimisation multi-objectif sans avoir à agréger les fonctions objectives en une seule fonction.

L'algorithme V.E.G.A. considère une population de N individus . Ces N individus sont répartis en k groupes (k étant le nombre de fonctions objectif de notre problème) de $\frac{n}{k}$ individus (avec N multiple de k). A chaque groupe, on associe une fonction objective. Cette fonction objective permet de déterminer l'efficacité d'un individu au sein du groupe. Ensuite, les individus sont mélangés et les croisements sont opérés en tenant compte de l'efficacité de chaque individu. (La figure **III.1)** représente les différentes séquences de

fonctionnement de la méthode. Sur cette figure, on a représenté les différents types de populations que l'on traite au cours du déroulement de la méthode V.E.G.A. (soit un ensemble d'individus, soit des groupes d'individus).

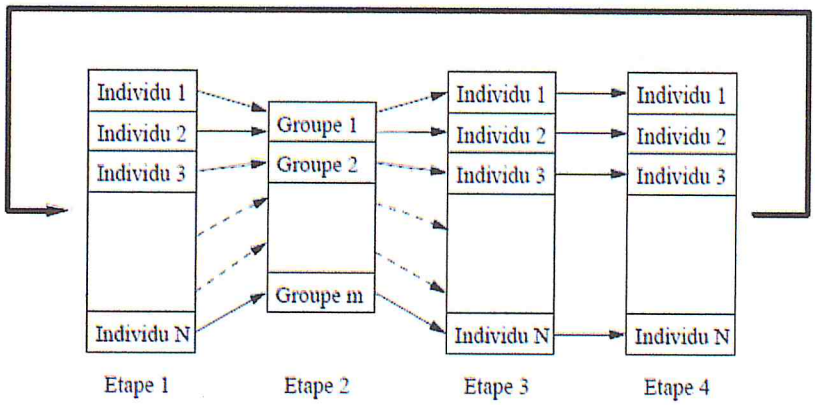


FIG. III.1 – Principe de l'algorithme V.E.G.A [25].

Voici la description des différentes étapes

AlgorithmeIII.1: l'algorithme VEGA [2, 20, 25]

Etape 1 : Itération *i*. Initialisation d'une population de taille *N*.

Etape 2 : Création des *k* groupes (sous-populations).

Etape 3 : Calcul des efficacités.

Mélange des individus.

Etape 4 : On applique l'algorithme génétique classique (croisement-mutation- sélection).

Puis on passe à l'itération suivante (i+1)

-Discussion

Le danger de cette méthode est d'obtenir, en fin d'optimisation, une population constituée d'individus moyens dans tous les objectifs. Une telle population ne permet pas d'obtenir une surface de compromis bien dessinée. En effet, la population va se concentrer autour d'un "point" moyen. De plus, il a été montré que cette méthode est équivalente à la méthode de pondération des fonctions objectives. Donc, elle ne permet pas de trouver des solutions qui se trouveraient dans une concavité.

✓ **Vector-Optimized Evolution Strategy (VOES)**

Dans cette approche **[2,20,25]**, un individu est composé de deux solutions dont une est marquée comme "dominante" et l'autre comme "récessive". Ainsi, deux évaluations correspondent à chacun des individus : f_d et f_r. L'évaluation finale et la sélection sont effectuées de la façon suivante :

La sélection est faite en M étapes (où M est le nombre des objectifs). A chaque étape, le vecteur des probabilités défini par l'utilisateur est utilisé pour choisir un des objectifs. Ce vecteur peut être fixé ou varier au cours de générations. Si la $m^{ême}$ fonction objectif est choisie, la performance d'un individu est calculée comme la moyenne pondérée des valeurs $(f_d)^i_m$ et $(f_r)^i_m$ en proportion 2/1 :

$$F^{(i)} = \frac{2}{3}(f_d)^i_m + \frac{1}{3}(f_r)^i_m$$

Comme dans une Stratégie d'évolution standard, λ solutions mutées sont créées à partir de μ parents. De plus, un opérateur swap qui échange des variables de décision des part "dominante" et "récessive" d'un individu est appliqué avec la probabilité 1/3 pour chacune des variables.

A chaque étape de la sélection, la population est ordonnée selon un des objectifs et $(M-1)/M$ meilleurs sont choisis en tant que parents. Cette procédure est répétée M fois et utilise à chaque fois la population qui a survécu lors de l'ordonnancement précédent. Le rapport entre λ et μ est alors le suivant :

$$\mu = ((M-1)/M)^{M} \lambda$$

Toutes les μ nouvelles solutions sont copiées dans un ensemble externe, dans lequel toutes les solutions non-dominées trouvées depuis le début de l'évolution sont stockées. Ensuite, l'ensemble ainsi obtenu est mis à jour par élimination de toutes les solutions dominées. Si la taille de cette archive est trop grande, un mécanisme de nichage est utilisé pour éliminer des solutions trop voisines.

-Discussion

Les algorithmes non-Pareto sont souvent simples et faciles à mettre en œuvre. Mais, ces méthodes répartissent la population sur les extrémaux du front de Pareto (**figure III.2**).

En effet, l'évaluation mono-objectif des sous-populations ne permet pas la comparaison des diverses solutions multiobjective disponibles à chaque itération en vue d'en exploiter les caractéristiques.

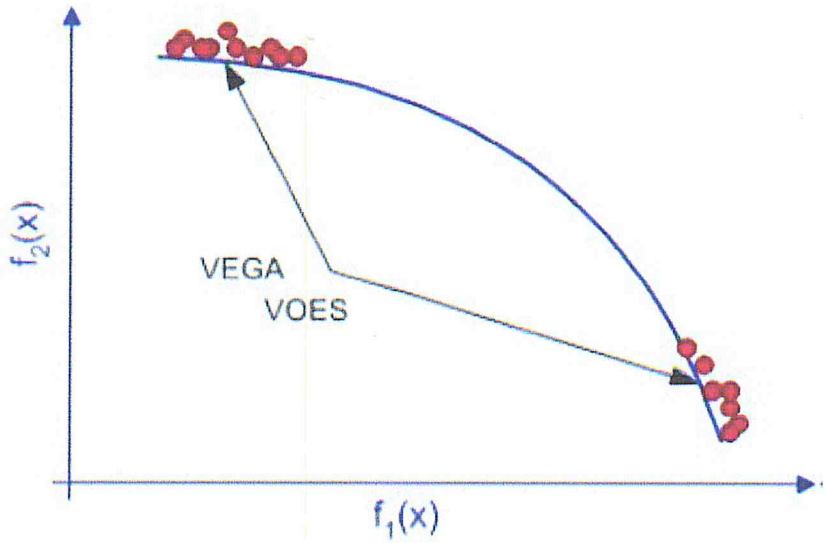


Fig. III.2 – Défauts de l'attribution de la performance dans les algorithmes non-Pareto[2,20,25]

III.2.2Algorithmes basés sur la dominance de Pareto

✓ **La méthode Multiple Objective Genetic Algorithm (M.O.G.A.)**

Cette méthode est présentée dans **[25]** et **[16]**. Elle utilise la relation de dominance pour déterminer l'efficacité d'un individu.

Présentation de la méthode

Cette méthode est basée sur la dominance au sens de Pareto. 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 on affecte le rang 1. Les individus dominés se voient donc affectés à un rang important (donc une forte pénalité). Pour le calcul de l'efficacité, on peut 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.

Le pseudo-code de cette méthode est représenté dans l'algorithme ci-dessous :

Algorithme III.2 Algorithme MOGA [16,25].

```
Initialisation de la population
Evaluation des fonctions objectives
Assignation d'un rang basé sur la dominance
Assignation d'une efficacité à partir du rang
For i=1 to 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
End For
```

-Discussion

La méthode MOGA ne permet pas, dans certains cas, d'obtenir une diversité dans la représentation des solutions. La méthode N.S.G.A, qui va être présenté maintenant, permet d'éviter cette difficulté.

✓ **La méthode *Non dominated Sorting Genetic Algorithm* (N.S.G.A.)**

Cette méthode reprend les grandes lignes de la méthode M.O.G.A précédemment décrite. La différence principale intervient lors du calcul de l'efficacité d'un individu [26]. Cette méthode est basée sur une classification en plusieurs niveaux des individus. Dans un premier temps, avant de procéder à la sélection, on affecte à chaque individu de la population un rang (en utilisant le rang de Pareto). Tous les individus non dominés de même rang sont classés dans une catégorie. A cette catégorie, on affecte une efficacité factice, qui est inversement proportionnelle au rang de Pareto de la catégorie considérée. On veut maintenant que les individus de la catégorie considérée se répartissent de manière uniforme au sein de celle-ci. On veut donc une bonne représentation, ou encore on veut qu'il y ait une diversité de solutions. Pour maintenir la diversité de la population, ces

individus classés se voient affectés d'une nouvelle valeur d'efficacité. Pour cela, on utilise la formule suivante :

$$m_i = \sum_{j=1}^{k} SH(d(i,j))$$

$$SH(d(i,j)) = \begin{cases} 1 - \left((d(i,j))/\sigma_{Share} \right)^2 & \text{si } d(i,j) < \sigma_{Share} \\ 0 & \text{sinon} \end{cases}$$

Ici, K désigne le nombre d'individus dans la catégorie considérée et $d(i,j)$ est la distance entre l'individu i et l'individu j la distance que l'on considère est la distance Euclidienne (si l'on considère un problème d'optimisation continu).$\sigma share$ est la distance d'influence. Comme on peut le voir dans l'expression ci-dessus, tous les individus qui sont suffisamment proches (dont la distance $d(i,j)$ est inférieure à σ_{share}) seront pris en compte dans le calcul de m_i.

Les autres seront ignorés. La valeur d'efficacité de l'individu i au sein de la catégorie considérée sera alors :

$$f_i = \frac{F}{mi}$$

Où F est la valeur d'efficacité affectée à la catégorie à laquelle appartient l'individu.

La figure **III.4** montre les différents paramètres qui interviennent dans le calcul de l'efficacité d'un individu.

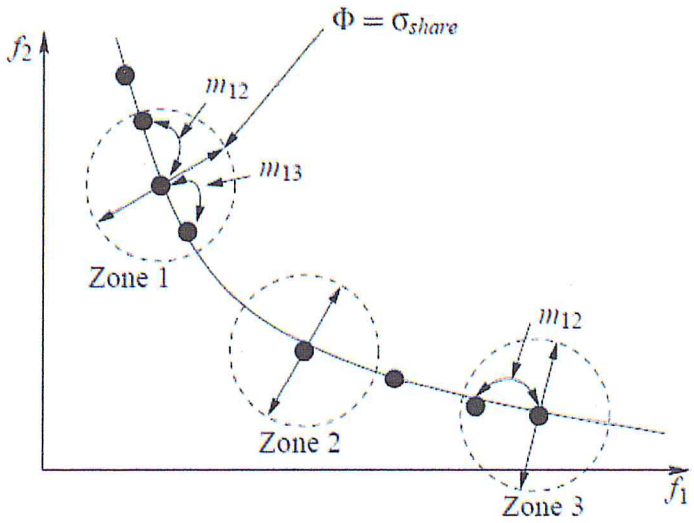


FIG. III.3 – Le calcul de l'efficacité [25].

Comme on voit à la figure **III.4,** le paramètre σ_{share} permet de définir une zone d'influence pour le calcul de l'efficacité d'un individu. Ensuite, on ignore les individus de ce groupe. Le processus reprend alors avec les individus restants, où l'on opère une nouvelle classification, une nouvelle catégorisation et une nouvelle opération de partage. Ce processus est répété jusqu'à ce que tous les individus aient été traités.

Comme les individus qui ont un rang de Pareto de valeur 1 ont une meilleure efficacité, ils sont reproduits en plus grand nombre que les autres. Cette propriété permet d'obtenir une convergence plus rapide vers la surface de compromis. Le partage, lui, permet de maintenir une répartition "uniforme" sur la surface de compromis.

-Discussion

L'efficacité de cette méthode tient dans le fait que les objectifs sont réduits à une valeur d'efficacité factice obtenue en utilisant le classement en fonction du rang de Pareto. Cette méthode présente l'inconvénient d'être sensible au choix de la valeur de σ_{share}.

Pour cette méthode, le nombre de comparaisons effectuées sur une population pour une génération est le même que pour la méthode MOGA. Cependant, un surplus de calculs dû au partage apparaît. Ce surplus est proportionnel à $N \cdot (N-1)$.

III.3 Problèmes tests [2, 5, 6, 20, 25]

Nous avons vu que le domaine de l'optimisation multi-objectif est foisonnant. Les méthodes que l'on a à notre disposition sont nombreuses.

Ainsi, dans cette section, nous allons présenter les tests bi-objectifs de Zitzler, Deb et Thiele , qui ont servi dernièrement comme base commune pour la comparaison des AEMO existants et pour l'évaluation de nouvelles techniques. Chaque problème permet de tester une difficulté bien particulière (surface de compromis discontinue, non convexe, etc.).

Avant de passer en revue des fonctions tests de l'optimisation multiobjective, il faut se 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 l'on n'est pas satisfait par une solution, on pourra en choisir une autre, située dans son voisinage. Du fait de l'uniformité de la représentation des solutions, la variation de valeur des fonctions objectives ne sera pas brusque lorsque l'on passe d'une solution à sa voisine (voir la figure I.7).

Une fonction test est donc une fonction destinée à mettre en difficulté la méthode d'optimisation multiobjective, dans la recherche d'une répartition uniforme des points solutions sur la surface de compromis.

On distingue 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, on trouve la multifrontalité de la surface de compromis.

Dans la seconde famille, on trouve les difficultés suivantes :

– la non convexité.

– la discontinuité.

Deb [5] A présenté une méthode pour construire des problèmes tests on donnant des expressions génériques. Ces expressions génériques permettent de construire tous les types de problèmes tests : convexe, non convexe, avec optima locaux, etc.

Cette série de fonctions de test est basée sur le problème biobjectif "générique" suivant :

$$\begin{cases} \text{minimiser f1 } (\vec{x}\ ) = f(x1, \ldots\ldots, xm) \\ \text{minimiser f2 } (\vec{x}\ ) = g(x_{m+1}, \cdots, x_N) \cdot h(f(x_1, \cdots, x_m)\ .g(xm_{+1}, \cdots, x_N) \\ \text{avec } g\vec{}(\vec{x}\ ) \leq 0 \\ \text{et } (h)\vec{}(\vec{x}\ ) = 0 \end{cases}$$

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 "multifrontalité" 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, discontinuité, etc.).

nous n'avons pas cité ici le problème ZDT5, qu'est formulé pour les variables binaires.

III.1 ZDT1

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

Ce problème peut être considéré comme l'étape de qualification lors du test d'une méthode : si une bonne approximation de l'ensemble des optimaux n'a pas été trouvé, ca ne vaut pas la peine d'essayer d'appliquer l'approche étudiée aux tests suivants.

$$\begin{cases} f(x1) = x_1 \\ g(x_2) = 1+9/n\text{-}1 \sum_{i=2}^{n} xi \\ h(f,g) = 1- \sqrt{(f_1/g)} \end{cases}$$

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

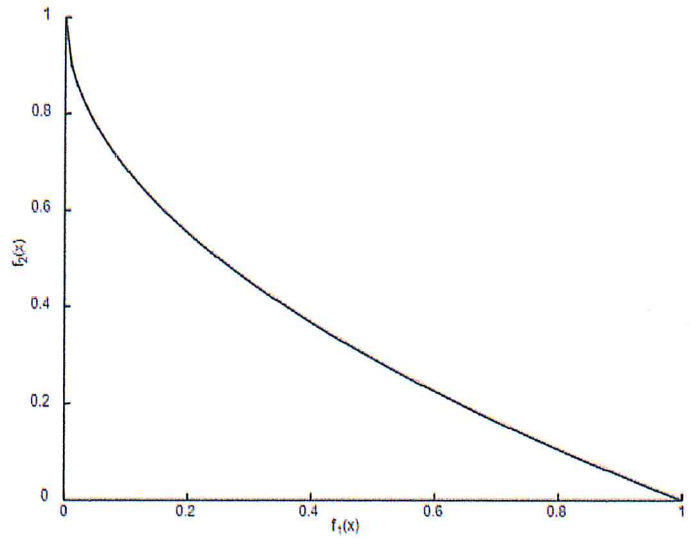


FIG. III.4 – La surface de compromis de la fonction ZDT1.

III.2. ZDT2

Certaines méthodes réagissent très mal face à la non-convexité. La plupart du temps, ces méthodes n'arrivent pas à couvrir une zone non convexe. La difficulté de ce problème se présente dans la non-convexité du front de Pareto.

$$\begin{cases} f1(x) = x1 \\ g(x2) = 1+\frac{9}{n-1} \sum_{i=2}^{n} xi \\ h(x) = 1\text{-}(f1/g)^2 \end{cases}$$

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

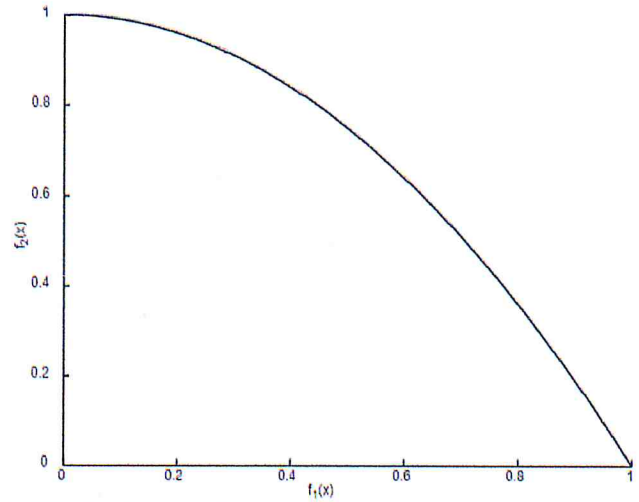


FIG. III.5 – La surface de compromis de la fonction ZDT2.

III.3. ZDT3

La grande majorité des méthodes d'optimisation multiobjective ne parviennent pas à assurer l'uniformité de la répartition des solutions sur la surface de compromis lorsqu'elles sont confrontées à une discontinuité.

La fonction discontinue de ZDT3 est définie comme suit :

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

où xi ∈ [0, 1] pour tout i = 1, . . . , n et n = 30. Le front de Pareto de ce problème est présenté dans la figure III.6.

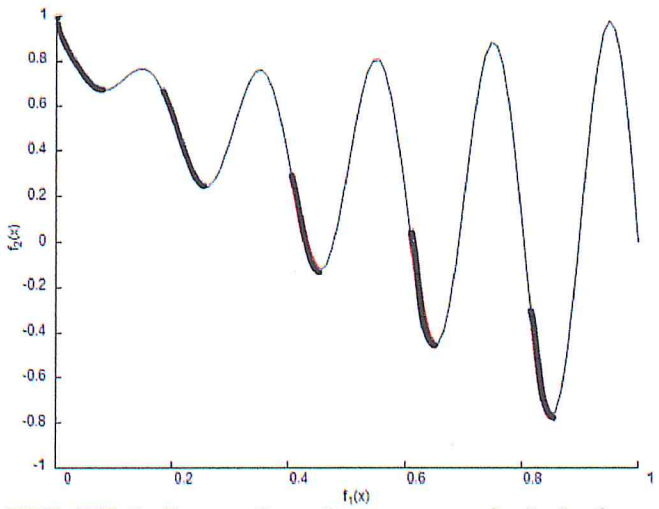


FIG. III.6– La surface de compromis de la fonction ZDT3.

III.4. ZDT4

Une autre grande difficulté que l'on peut rencontrer est la multifrontalité. On la rencontre lorsqu'un problème d'optimisation multi-objectif possède une ou plusieurs surfaces de compromis ''fantômes''.

Une surface de compromis fantôme est, le plus souvent, due à l'existence d'un optimum global isolé et d'un optimum local. Lors du processus d'optimisation, les solutions convergent vers l'optimum local et forment ainsi une surface de compromis, qui est différente de la surface de compromis "optimale" et cela a cause du fait que la fonction g est multimodale.

Ce test modélise cette difficulté

$$\begin{cases} f\ (x_1) = x1 \\ g\ (x_2) = 1+10(n-1)+\sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi\ x_i)) \\ h\ (f,g) = 1 - \sqrt{\frac{f_1}{g}} \end{cases}$$

où xi ∈ [0, 1] pour tout i = 1, . . . , n et n = 30. Quelques fronts locaux et le front global sont présentés dans la figure III.7.

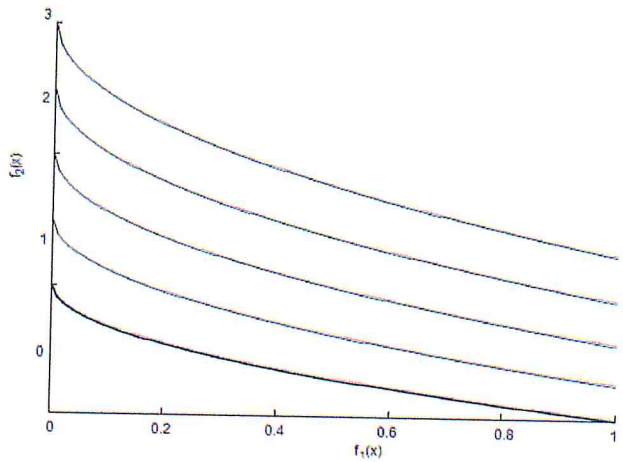


FIG. III.7 – La surface de compromis de ZDT4.

III.4. ZDT6

La particularité de ce problème c'est que les solutions optimales ne sont pas uniformément distribuées le long du front de Pareto. Cette effet est du à la non-linéarité de la fonction *f1*.

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

où xi ∈ [0, 1] pour tout i = 1, . . . , n et n = 30. Le front de Pareto de ce problème est présenté dans la figure III.6.

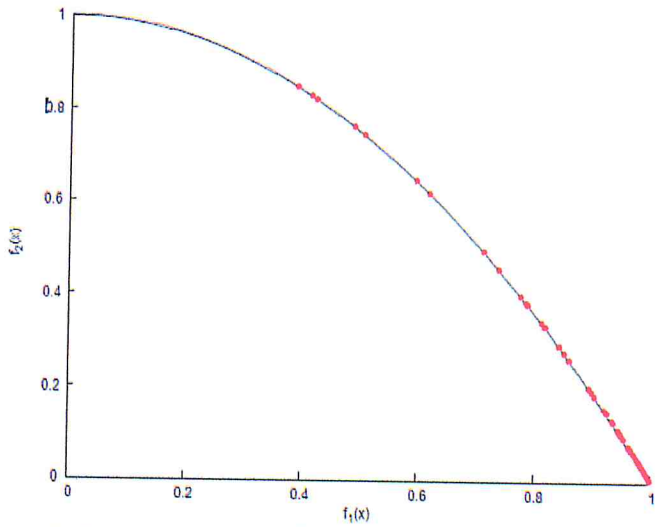


FIG. III.8 – La surface de compromis de ZDT6.

III.7. Conclusion

Nous venons d'achever ce chapitre. A travers cette étude nous avons obtenu une vue globale sur les différentes méthodes de résolution de problèmes d'optimisation multiobjectif, dont nous constatons que le principe consiste à obtenir toute une population de solutions appelé le front de Pareto au lieu de chercher une seule solution. En donnant la main à l'utilisateur ou le décideur pour choisir un des points du front du Pareto.

Dans le chapitre suivant nous allons proposer un algorithme qui repose sur le concept de la dominance Lexicographique Max-Ordering qui donne comme résultat une seule solution, appelé optimum au sens Lexicographique Max-Ordering, qui représente le meilleurs compromis possible entre les fonctions objective d'un problème donné et ainsi permettra le décideur d'avoir le meilleure choix.

Chapitre IV

MO-NSGA LMOGA Description et Validation

IV.1 Introduction

Dans ce chapitre, nous proposons deux nouveaux Algorithmes Evolutionnaire Multiobjectifs pour trouver la solution LEX-MO que nous avons nommés : *Max Ordering Non dominated Sorting Genetic Algorithm (MO-NSGA)* et *Lexicographique Max Ordering Genetic Algorithm* (*LMOGA*). Nous validons ensuite ces approches en se basant sur une série d'expériences indépendantes et nous les comparons avec les solutions trouvées par NSGA.

IV.2 Max Ordering Non dominated Sorting Genetic Algorithm (*MO-NSGA)*

Nous présentons dans cette section un Algorithme Evolutionnaire Multi-Objectif qui hérite le même principe de base que l'algorithme NSGA. L'idée clé consiste à introduire une procédure qui se base sur la dominance lexicographique max ordering qui nous a permis de classer les individus qui ont été classé dans des rangs suivant le principe de dominance de Pareto, dont l'algorithme NSGA est basé. Ainsi cette notion permet de s'approcher plus vers l'optimum au sens LEX-MO que l'on a nommé la solution LEX-MO.

Voici l'algorithme (MO-NSGA) :

Algorithme IV.1 MO-NSGA

- Initialiser une population aléatoire P(0) faisable et initialiser un compteur d'itération t = 0.
- Evaluation des fonctions objectives
- Assignation d'un rang basée sur le rang de dominance sur chaque surface de compromis
- Trie des éléments de chaque surface de compromis en se basant sur la dominance lexicographique Max ordering.
- Assignation d'une efficacité

For i=1 to G

- Sélection aléatoire proportionnelle à l'efficacité
- Croisement
- Mutation
- Evaluation des fonctions objectives.

- Assignation d'un rang basée sur le rang de dominance sur chaque surface de compromis.
- Trie des éléments de chaque surface de compromis en se basant sur la dominance lexicographique Max ordering.
- Assignation d'une efficacité.

End For

Nous avons exploité l'algorithme qui se trouve dans [24], afin d'arriver à mettre en œuvre cet algorithme qui permet de trier les individus suivant le principe de dominance LEX-MO.

Algorithme IV.2:

For j=1,…, N $\grave{F}_j$= Sort (F_j (X)) End

For p=1…N

Begin

Min =1;

i:=1

While (i<N) do

Begin

if ($\grave{F}_i$<lex $\grave{F}_{min}$) then min:=i;

i:=i+1;

End

Opt_p={ aj: $\grave{F}_j$=$\grave{F}_{min}$}

End

Opt= Opt UOpt_p .

End

- où N représente le nombre de la population.

La complexité de cet algorithme est limité O (NQlog Q) où Q est le nombre de fonction objectives.

IV.3 Résultats expérimentaux de MO-NSGA

Dans cette section, nous illustrons des échantillons de résultats représentatifs des deux approches (NSGA et MO-NSGA) lors de la résolution des problèmes test suivants :

- ZDT1
- ZDT2
- ZDT3
- ZDT4
- ZDT6

IV.3.1 Conditions expérimentales

Pour pouvoir comparer les résultats de l'algorithme **MO-NSGA** avec ceux de l'algorithme **NSGA**, nous avons choisi d'utiliser les mêmes valeurs qui ont été utilisées par Deb [5] [6] pour valider l'algorithme NSGA .Ainsi, nous avons lancé les deux algorithmes avec les paramètres suivants :

- Taille de la population : 100
- Nombre d''evaluations : 250
- Probabilité de croisement : 1
- Probabilité de mutation : 0.01

Pour la valeur de σ_{share} pour l'algorithme NSGA nous l'avons calculé de la façon suivante :

On applique l'algorithme suivant afin d'avoir une valeur unique de σ_{share} Pour chaque rang r. tel que N_r représente le nombre des rangs.

Algorithme IV.3[20]

Pour i=1 à N_r

Si nbr_r=1 alors σshare=1

Sinon

Calculer le min de f1 et f2

Calculer le max de f1 et de f2

A =maxf1-minf1

B=maxf2-minf2

$$\sigma \text{ share}_r = \frac{A+B}{Nr}$$

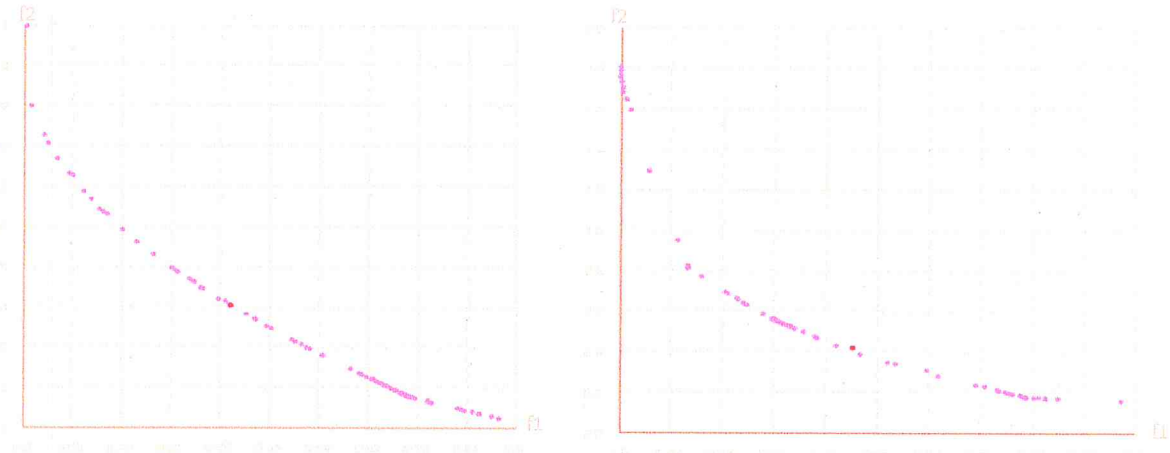
Fin.

-nbr_r est le nombre des éléments de un rang r.

Une centaine de tests indépendants de NSGA et MO-NSGA sont lancés pour la même population initiale pour chaque problème test. Dont voici un échantillon :

IV.3.1 .1 Problème ZDT1

La figure (**IV.1**) illustre l'évolution de **NSGA** et **MO-NSGA** au cours de résolution du problème ZDT1. Elle présente la surface de compromis de NSGA ainsi que celles de MO-NSGA après 250 itération ou générations.

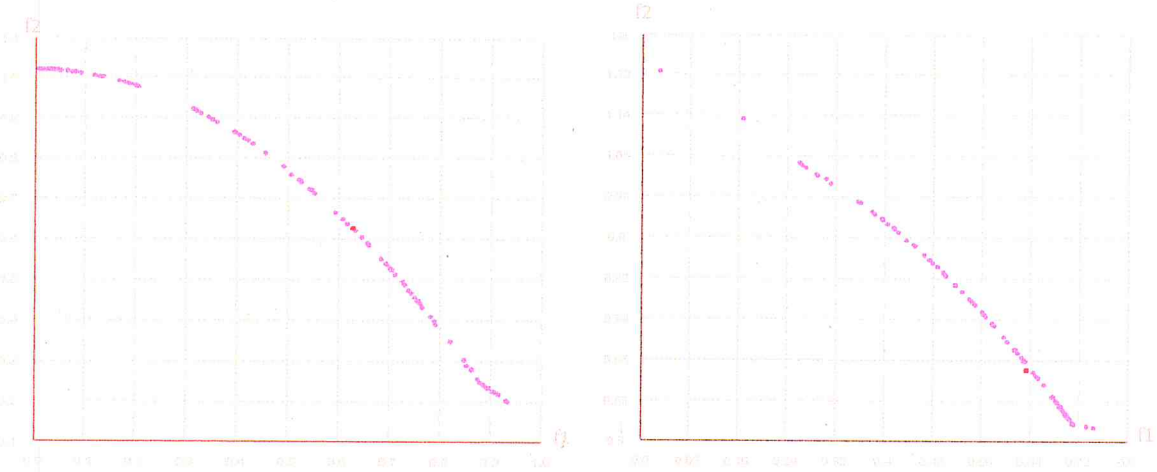


MO-NSGA	NSGA
Solution LEX-MO (0.3787 ,0. 4021)	Solution LEX-MO (0.4072 0.4157)
Temps d'exécution : 20secondes	Temps d'exécution:170secondes

Fig. IV.1 – Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT1

IV.3.1.2. Problème ZDT2

La figure IV.2 illustre l'évolution de NSGA et MO-NSGA au cours de résolution du problème ZDT2.

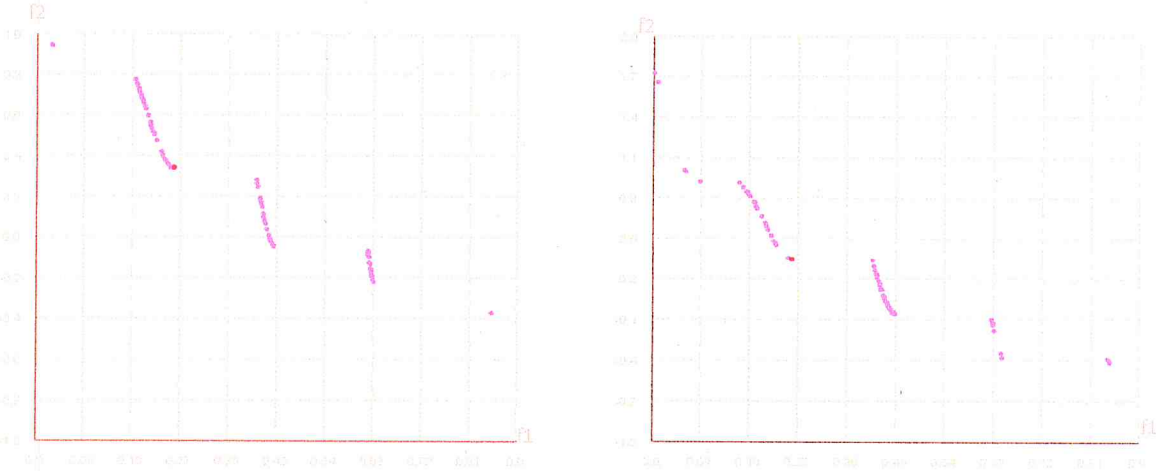


MO-NSGA	NSGA
Solution LEX-MO (0.6289, 0.6257)	Solution LEX-MO (0.6344 0.6388)
Temps d'exécution : 19secondes	Temps d'exécution : 115

Fig. IV.2 – Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT2

IV.3.1 .3 Problème ZDT3

Nous illustrons dans la figure IV.3 l'évolution de NSGA et MO-NSGA au cours de résolution du problème ZDT3.

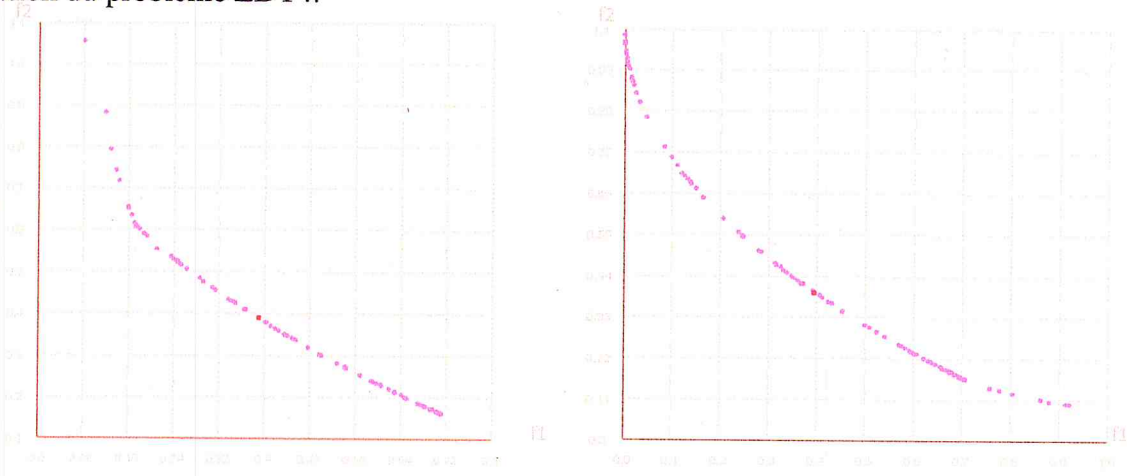


MO-NSGA	NSGA
Solution LEX-MO (0.257,0.3376)	Solution LEX-MO (0.2589, 0.3382)
Temps d'exécution : 31secondes	Temps d'exécution : 121 secondes

Fig. IV.3 – Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT3

IV.3.1 .4 ProblèmeZDT4:

Nous illustrons dans la figure IV.4 l'évolution de NSGA et MO-NSGA au cours de résolution du problème ZDT4.

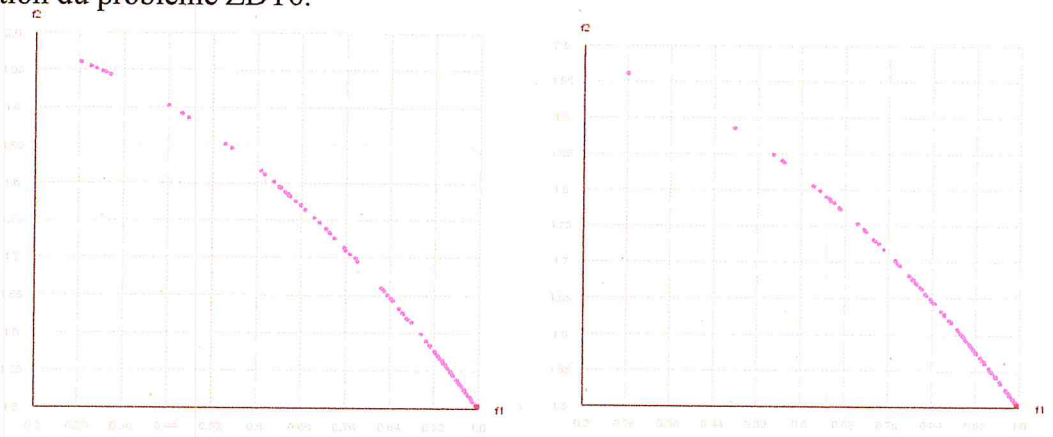


MO-NSGA	NSGA
Solution LEX-MO (0.391 0.3872)	Solution LEX-MO (0.3959 0.3952)
Temps d'exécution : 20secondes	Temps d'exécution :149 secondes

Fig. IV.4 – Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT4

IV.3.1 .5 Problème ZDT6 :

Nous illustrons dans la figure IV.5 l'évolution de NSGA et MO-NSGA au cours de résolution du problème ZDT6.

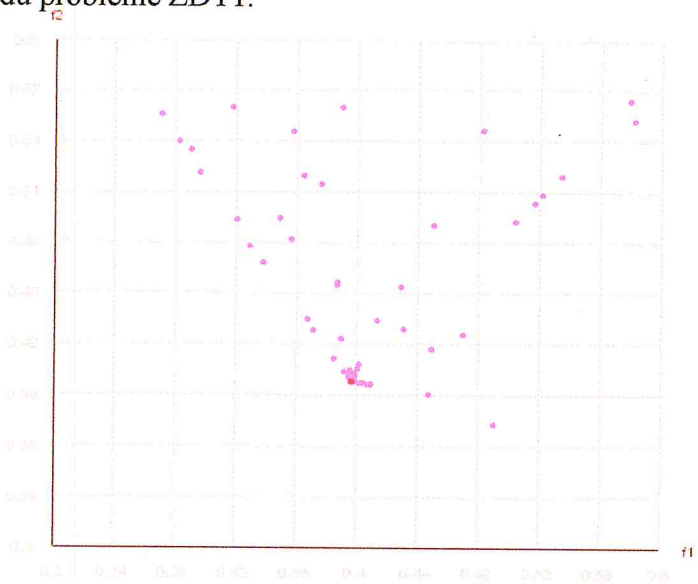


MO-NSGA	NSGA
Solution LEX-MO (0.9984, 1.5016)	Solution LEX-MO (0.9997, 1,5020)
Temps d'exécution : 10 secondes	Temps d'exécution : 33 secondes

Fig. IV.5 – Évolution de MO-NSGA et NSGA au cours de résolution du problème ZDT6

IV.4.1. ZDT1

Nous illustrons dans la figure IV.7l'évolution de LMOGA au cours de résolution du problème ZDT1.



LMOGA

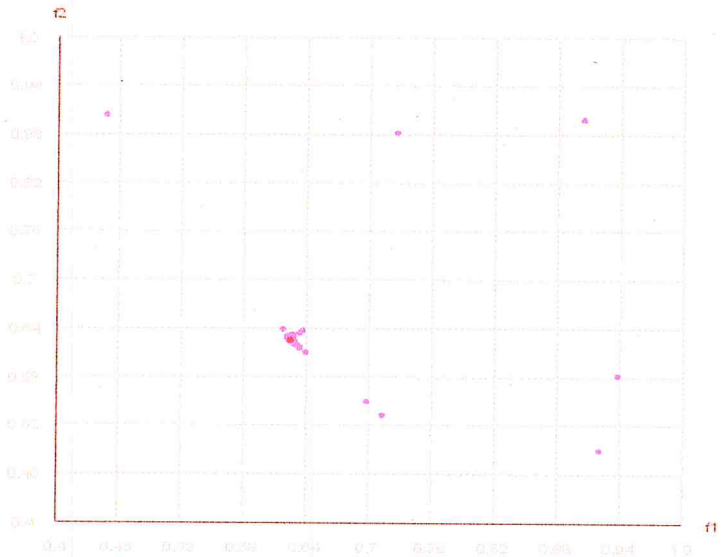
Solution LEX-MO (0.3975, 0.3974)

Temps d'exécution : 04 secondes

Fig. IV.6 – Évolution de LMOGA au cours de résolution du problème ZDT1

IV.4.2. ZDT2

Nous illustrons dans la figure IV.8l'évolution de LMOGA au cours de résolution du problème ZDT2.



LMOGA

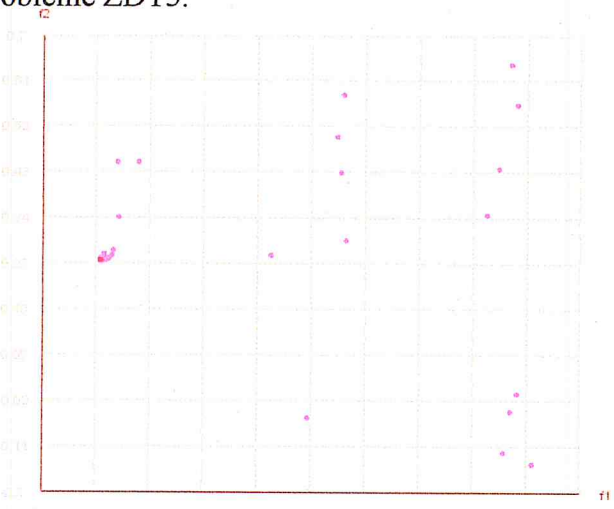
Solution LEX-MO (0.6247, 0.6247)

Temps d'exécution : 04 secondes

Fig. IV.7 – Évolution LMOGA au cours de résolution du problème ZDT2.

IV.4.3. ZDT3

Nous illustrons dans la figure IV.9 l'évolution de LMOGA au cours de résolution du problème ZDT3.



MO-NSGA sans rang

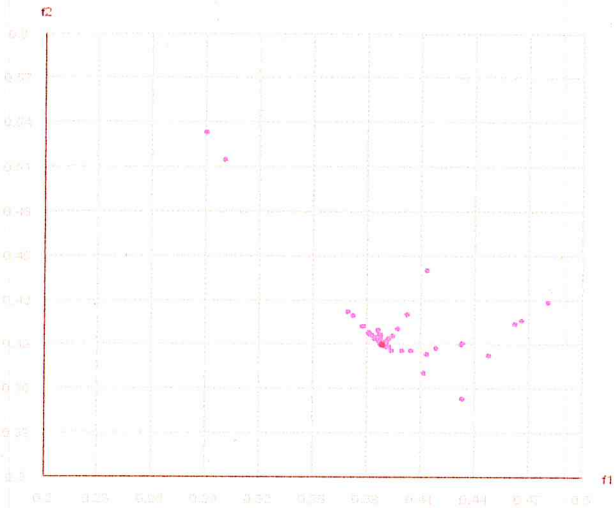
Solution LEX-MO (0.2544, 0.2543)

Temps d'exécution : 04 secondes

Fig. IV.8– Évolution de LMOGA au cours de résolution du problème ZDT3.

IV.4.4. ZDT4

Nous illustrons dans la figure IV.10 l'évolution de LMOGA au cours de résolution du problème ZDT4.



MO-NSGA sans rang

Solution LEX-MO (0.3889, 0.3889)

Temps d'exécution : 05 secondes

Fig. IV.9– Évolution de LMOGA au cours de résolution du problème ZDT4.

IV.4.5. ZDT6 non uniforme

Nous illustrons dans la figure IV.11 l'évolution de LMOGA au cours de résolution du problème ZDT6.

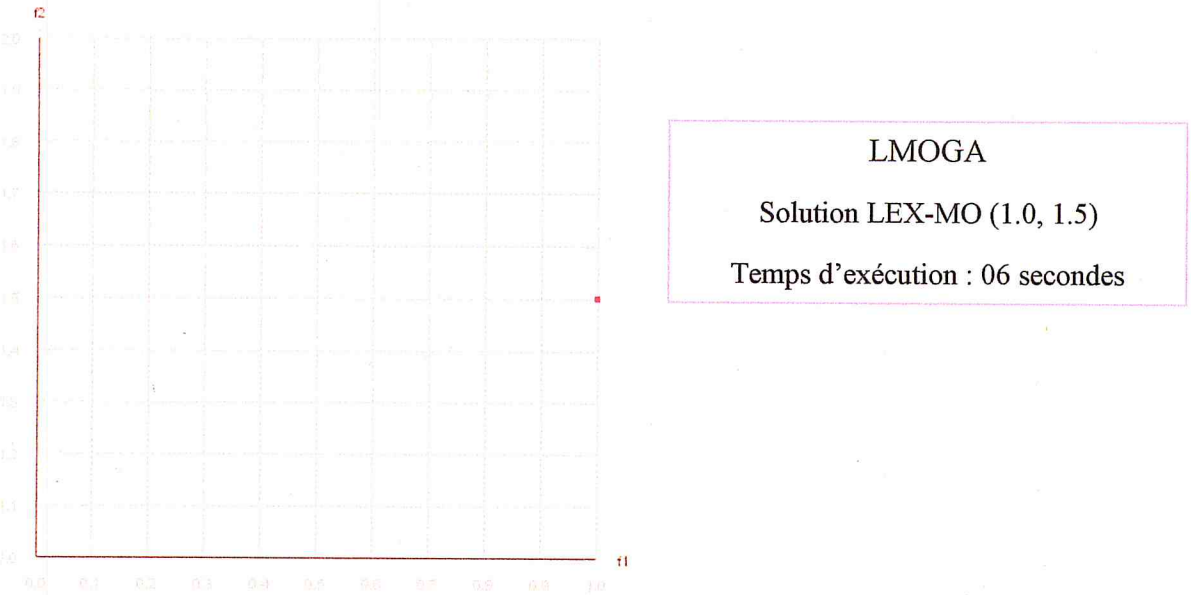


Fig. IV.10– Évolution de LMOGA au cours de résolution du problème ZDT6.

IV.5 Conclusion

GA / Test	NSGA		MO-NSGA		LMOGA	
	Temps D'exécution	**Solution LEX-MO**	**Temps D'exécution**	**Solution LEX-MO**	**Temps D'exécution**	**Solution LEX-MO**
ZDT1	170 Secondes	(0.4072 0.4157)	20 Secondes	(0.3787,0.4021)	04 Secondes	(0.3975, 0.3974)
ZDT2	115 Secondes	(0.6344 ,0.6388)	19 Secondes	(0.6289,0.6257)	04 Secondes	(0.6247, 0.6247)
ZDT3	121 Secondes	(0.2589, 0.3382)	31Secondes	(0.257,0.3376)	04 Secondes	(0.2544, 0.2543)
ZDT4	149 Secondes	(0.3959 0.3952)	20 Secondes	(0.391 0.3872)	05 Secondes	(0.3889, 0.3889)
ZDT6	33 Secondes	(0.9997, 1,5020)	10 Secondes	(0.9984,1.5016)	06 Secondes	(1.0, 1.5)

Tab IV.1Comparaison de performance entre NSGA, MO-NSGA et LMOGA

Les résultats prouve que l'algorithme MO-NSGA présente une vitesse et une convergence meilleure vers la Solution LEX-MO que celle de NSGA c'est-à-dire que le front de Pareto résultant de MO-NSGA est souvent plus concentré au tour du point solution LEX-MO pour tous nos problèmes tests.

Les solutions LEX-MO trouvé par les deux algorithmes sont presque identiques pour les 5 problèmes tests.

D'ailleurs l'algorithme (LMOGA) nous a permis d'avoir la Solution LEX-MO qui représente l'optimum au sens LEX-MO, dans un temps opportun par rapport MO-NSGA, sans avoir besoin de rechercher le front comme c'était le cas pour les deux autres algorithmes. En plus nous avons remarqué que la valeur de l'optimum de LEX-MO résultant de LMOGA s'est stabilisé au bout d'un certain nombre d'exécution pour chaque problème test .Ainsi nous avons gagné un test d'arrêt bien déterminé qui est bien la stabilisation de l'optimum LEX-MO. Donc cet algorithme est très utile pour un décideur où le temps et l'efficacité sont ces premiers soucis. Ainsi lorsque qu'il choisit ce point comme solution à son problème il sera sûr que :

- ✓ Il a choisi une solution optimale de Pareto, c'est à dire qu'il ne peut pas améliorer un critère sans aggraver un autre.
- ✓ Il a choisi Une solution qui a la plus petite valeur du pire objectif.
- ✓ Et que toutes les alternatives LEX-MO optimales sont équivalentes dans le sens que leurs vecteurs Sort sont les mêmes.

Conclusion Générale

L'objet de notre travail été l'étude et la mise en œuvre d'un algorithme génétique qui nous permet de trouver la solution LEX-MO et nous somme parvenue à l'algorithme (MO-NSGA) qui hérite le même principe de base que l'algorithme NSGA. Et après une série d'expériences indépendantes et tâtonnement, nous somme arrivé à proposer un autre algorithme qu'est meilleur en termes de temps d'exécution et de convergence selon les tests que nous avons présentés dans ce mémoire.

La première étape de ce projet nous a permis en premier lieu d'élargir nos connaissances dans le domaine de l'optimisation et en particulier l'optimisation Multi objective. Ensuite, nous avons eu à faire aux algorithmes évolutionnaires dont nous avons étudié une partie de l'existant au niveau de la littérature. Au départ nous avons procédé par implémenter NSGA pour retrouver le front Pareto des fonctions tests Zitzler, Deb et Thiele (ZDT) **[2] [20] [5]** de façon à s'assurer de notre démarche afin de déterminer la solution LEX-MO **[9] [10]**. Dans une seconde phase nous l'avons adapté (MO-NSGA) pour trouver notre solution et après comparaison avec NSGA nous concluons que :

- les solutions LEX-MO trouvé par les deux algorithmes sont presque identiques pour les 5 problèmes tests.
- le front de Pareto résultant de MO-NSGA est souvent plus concentré au tour du point solution LEX-MO pour tous nos problèmes tests.
- Le MO-NSGA se rapproche plus rapidement vers notre solution que NSGA.

Afin de faire mieux encore nous avons développé un nouvel algorithme que nous avons nommé Lexicographique Max Ordering Genetic Algorithm (LMOGA) et qui permet :

- De trouver cette solution sans passer par la recherche de tout le front comme dans le cas de NSGA et MO-NSGA.
- De réduire le temps d'exécution d'une manière considérable.
- Et d'avoir un test d'arrêt bien déterminé contrairement aux deux autres algorithmes.

Ce travail, comme toute œuvre humaine, comporte sans doute des limites et demeure perfectible,

Pour cela, nous ne pouvons pas l'achevez sans proposer des perspectives:

- tester notre approche sur des problèmes de dimension plus importante.
- l'appliquer sur des problèmes de la vie réelle.
- Comparer les résultats de LMOGA avec des algorithmes évolutionnaires élitistes au sens de la dominance.

Bibliographie

[1] V.BARICHAR, Approches hybrides pour les problèmes Multiobjectifs. Thèse de doctorat, Spécialité Informatique, Ecole Doctorale d'Angers ,novembre 2003.

[2] A. C. BEN ABDALLAH, Optimisation multiobjectif évolutionnaire, E Rapport de Mémoire de Master d'Ingénierie Mathématique école polytechnique de Tunisie , 2004/2005.

[3] R. Cerf. An asymptotic theory of genetic algorithms. In J. M. Alliot, E. Lutton,

E. Ronald, and M. Schoenauer, editors, Artificial Evolution, volume 1063 of LNCS, pages 37–53. Springer Verlag, 1996.

[4] J. Dalton www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php/ Newcastle Engineering Design Centre, Merz Court, Newcastle University April 2012 .

[5] K. Deb, Multiobjective genetic algorithms : problem difficulties and construction of test problems, Technical Report CI-49/98, Dortmund, Department of Computer Science/LS11, University of Dortmund, Allemagne, 1998. 8.2, 8.4

[6] K. Deb, E. Zitzler, L. Thiele, Comparison of Multiobjective Evolutionary Algorithms : Empirical Results, Technical Report TIK, 2000. 11.4.2.5

[7] K.A. DeJong and J. Sarma. On decentralizing selection algorithms. In L.J. Eshelman,

editor, Proceedings of the 6th International Conference on Genetic Algorithms, pages 17–23. Morgan Kaufmann, 1995.

[8] F.Y.Edgeworth .Mathematical physic .P.Keagan,Londres,Angleterre,1881.

[9] M. Ehrgott, *A characterization of Lexicographic Max-ordering Solutions*, Methods of Multicriteria Decision Theory: Proceedings of the 6thWorkshop of the DGORWorking GroupMulticriteria and Decision Theory, Egelsbach, Häsel-Hohenhausen, 1996.

[10] M.Ehrgott .Discrete Decision Problems,Multiple Criteria Optimization Classes and Lexicographic Max-Ordering. Technical report ,University of Kaiserslautern,Departement of

Mathematics ,1996.Report in Wirtschaftsmathematik No .13 .Accepted for Prceedings of the 6th Workshop of the DGOR-Working Group Multicriteria Optimization and Decicion theory.

[11] L.J.Fogel,A.J.Owens,and M.J.Walsh.Artificial Intelligentce through Simulated Evolution. John Wiley ,New York ,1966.

[12] Fonseca C., Fleming P., Genetic algorithms for multiobjective optimization: formulation, discussion and generalisation. Proc. Of the 5th Int. Conf. on genetics algorithms, 416-423.

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

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

[15] J. Holland. Outline for a logical theory of adaptive systems. Journel of The Association of Computing Machinery, 3, 1962.

[16] N.JOZEFOWIEZ, Modélisation et résolution approchée de problème de tournée multiobjectif. PhD thesis, Université des science et technologies de ,LILLE ,2004

[17] J.R.Koza .Genetic Programming :On the Programming of Computers by means of Natural Evolution .Mit Press ,Massachussetts,1999.

[18] E.Marchi and J.A.Oviedo. Lexicographic optimality in the multiple objective linear programming:The nucleolar solution.European Journal of Operational Research, 57:355-359,1992

[19] V.Pareto.Cours d'économie politique.Rouge,Lausanne,Suisse,1896.

[20] O. Roudenko. Application des Algorithmes Evolutionnaires aux Problèmes d'optimisation Multi-Objectif avec Contraintes. PhD thesis, Ecole Polytechnique, Paris, Mars 2004.

[21] G. Rudolph. Convergence analysis of canonical genetic algorithm. IEEE Transac-

tions on Neural Networks, 5(1) :96–101, 1994.

[22] G. Rudolph. Convergence Properties of Evolutionnary Algorithms. Kovac, Hamburg, 1997.

[23] J. D. Schaffer. Multiple objective optimization with vector evaluatedgenetic algorithms. In J. J. Grefenstette, editor, Proceedings of the 1st International Conference on Genetic

[24] H.-P.Schwefel .Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik .Technical university of Berlin ,1995.Dipl.-Ing.Thesis.

[25] Yann Collecte –Patrick Siarry, Optimisation multiobjectif ÉDITIONS EYROLLES , Paris Cedex 05, 2002, ww.editions-eyrolles.com.

[26] Srinivas et al. 93] N. Srinivas, K. Deb, Multiobjective optimization using non-dominated sorting in genetic algorithms, Rapport technique, Department of Technical Engineering, Indian Institute of Technology,Kanput, India, 1993.

[27] Srinivas N., Ded K., multionbjective optimisation using nondominated sorting in genetic algorithm. Evolutionary computation, 2, 3, 221-248.