République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Abderrahmane MIRA de Bejaia

Faculté de Technologie

Département d'Automatique, Télécommunication et Electronique

MEMOIERE DE FIN DE CYCLE

En vue de l'obtention du diplôme Master en Automatique

Thème

Etude comparative des algorithmes génétiques multi-objectifs

Réalisé par :

BENHAMA Abdelouhab

BENKHELOUF Yacine

Encadré par :

Mr. O. GUENOUNOU

Année universitaire 2014/2015

TABLE DES MATIERES

Liste des Figures

Liste des Pseudocodes

Liste des Tableaux

Introduction

Historique

L'idée de l'imitation du processus biologique de l'évolution pour la résolution des problèmes d'optimisation a été inventée par John Holland et exposée dans son ouvrage "Adaptation in natural and artificial systems," en 1975 [19], connue aujourd'hui sous le nom "Algorithmes Génétiques" et devenue populaire dans le domaine d'optimisation grâce à Goldberg (1989 [16]). Les algorithmes génétiques sont basés sur l'évolution naturelle (Darwin, 1859 [8]), le principe de ce processus est la survie du plus adapté. Les éléments concernés par un processus évolutionnaire sont sujets d'une sélection qui ne gardera à la fin que ceux qui présentent les meilleures facultés d'adaptation. Les organismes biologiques sont un exemple d'adaptation et d'optimisation car ils ont réussi à s'adapter et à survivre l'environnement naturel pendant des millions d'années. Holland dans son ouvrage (1975 [19]) a voulu traduire ce processus d'évolution et d'adaptation mathématiquement afin de pouvoir optimiser des problèmes qu'on rencontre dans la majorité des domaines, l'économie, la médecine, l'intelligence artificielle et les réseaux neuronaux … etc.

Problématique

Contrairement aux problèmes mathématiques simples, les problèmes multi-objectifs consistent à optimiser plusieurs problèmes en même temps et trouver des compromis et non-pas une seule solution finale. Ces compromis représentent des choix qui dépendent du système étudié.

Les Algorithmes génétiques sont une méthode de résolution des problèmes multi-objectifs, la multitude des modèles des AGs offre un choix, mais parfois ce choix est difficile car il existe plusieurs types de problèmes multi-objectifs. Notre travail consiste à départager entre quelques types d'algorithmes génétiques en menant une étude comparative.

Le premier est l'AG le plus populaire dans le domaine, le NSGA II créé par un pionnier de l'optimisation multi-objectif qui est Kalaynmoy Deb, la deuxième est également très utilisée qu'est le SPEA II, tandis que pour les deux autres, nous avons choisi deux méthodes relativement récentes et intéressantes.

Le FastPGA ayant une nouvelle méthode de régulation favorise la convergence vers le front de Pareto réel. Ainsi que le MOCell, le modèle cellulaire de l'algorithme génétique qui possède une structure particulière de population favorise la diversité et une distribution uniforme des solutions sur le front de Pareto réel.

Organisation du mémoire

Le mémoire est organisé en quatre chapitres :

Le premier chapitre décrit le principe de fonctionnement d'un algorithme génétique simple qui est une méthode d'optimisation mono-objectif. Pour ne pas rester dans le cadre descriptif, des exemples d'applications sont donnés à la fin du chapitre.

Le **deuxième chapitre** est consacré à l'optimisation multi-objectif**.** On rappelle tout d'abord quelques notions de base relatives à l'optimisation multi-objectif, puis on donne un état de l'art sur les méthodes de résolution par AGs dans ce domaine.

Le **troisième chapitre** est une description détaillée des quatre algorithmes génétiques que nous avons choisis.

Le **quatrième chapitre** est une étude comparative entre les AGs choisis en testant leurs performances avec des fonctions de benchmark[1].

[1] Fonctions de test.

Chapitre I

Algorithmes Génétiques Mono-objectifs

I.1 - Introduction

Les algorithmes génétiques (AGs) sont une méthode d'optimisation numérique inspirée du processus naturel de l'évolution génétique (Darwin, 1859 [8]), c'est une méthode d'optimisation puissante, elle est appliquée à un large éventail de problèmes. Les AGs sont reconnus pour leur effectivité dans la majorité des cas et sont tout le temps utilisés pour résoudre des problèmes pratiques.

Dans ce premier chapitre, nous allons porter lumière sur les algorithmes génétiques mono-objectifs et expliquer le fonctionnement de base ainsi que les notions relatives à cette méthode, un exemple pratique d'un AG est donné à la fin de ce chapitre.

I.2 - Algorithmes génétiques

Les AGs (Holland, 1975 [19]) font partie des algorithmes méta-heuristiques, ils travaillent sur un espace de recherche sous forme de population de pseudo-solutions en y opérant d'une manière stochastique sans qu'il y ait de contraintes de continuité ni de différentiabilité de la fonction à optimiser, ce qui est d'ailleurs un des points qui différencie les AGs des autres méthodes classiques. Les caractéristiques qui font l'unicité de cette méthode sont :

1- Les AGs ne travaillent pas directement sur les paramètres, ces derniers sont codés avant l'opération.

2- Ils opèrent sur une population de solutions au lieu d'une seule solution.

3- Les AGs n'utilisent que les valeurs de la fonction à étudier, sa dérivée ou autres connaissances ne sont pas prises en considération.

4- Les AGs se basent sur des données aléatoires, donc ils sont probabilistes.

Les AGs utilisent les éléments suivant pour se définir :

- Individu/chromosome/séquence : une potentielle solution du problème abordé.
- Population : un ensemble de chromosomes de l'espace de l'exploration
- Environnement : l'espace de l'exploration et de la recherche.
- Fonction d'évaluation (fitness) : la fonction objectif qui sert à évaluer la précision de la solution du problème.

I.3 - Fonctionnement des AGs

Semblables au processus naturel de l'évolution, les algorithmes génétiques comportent des phases à travers lesquelles une solution potentielle est estimée :

1- **Initialisation :** Un ensemble de N chromosomes est aléatoirement créé formant une population de solutions potentielles.

2- Evaluation : Décodage puis évaluation de chaque individus/chromosomes.

3- Reproduction : Après l'évaluation, une nouvelle population de N individus est créée à l'aide d'une méthode de sélection adéquate.

4- Opérations génétiques : Croisement et mutation de certains individus au sein de la nouvelle population.

5- Retour : Tant que la condition d'arrêt n'est pas satisfaite, l'algorithme recommence à partir de la phase d'évaluation.

Un AG génère une population en utilisant une représentation chromosomique des individus puis à l'aide de la fonction objectif (fitness), il évalue chaque individu et sélectionne les mieux adaptés pour former une nouvelle population de solutions potentielles à travers les opérateurs génétiques (croisement, mutation) là ou se passe la reproduction afin de créer de nouveaux individus au sein de la nouvelle population.

À travers ces phases, le mécanisme de la sélection naturelle est simulé, afin d'assurer la survie, les AGs gardent à chaque fois les meilleurs individus en écartant les plus faibles.

Ce processus est illustré dans l'organigramme suivant :

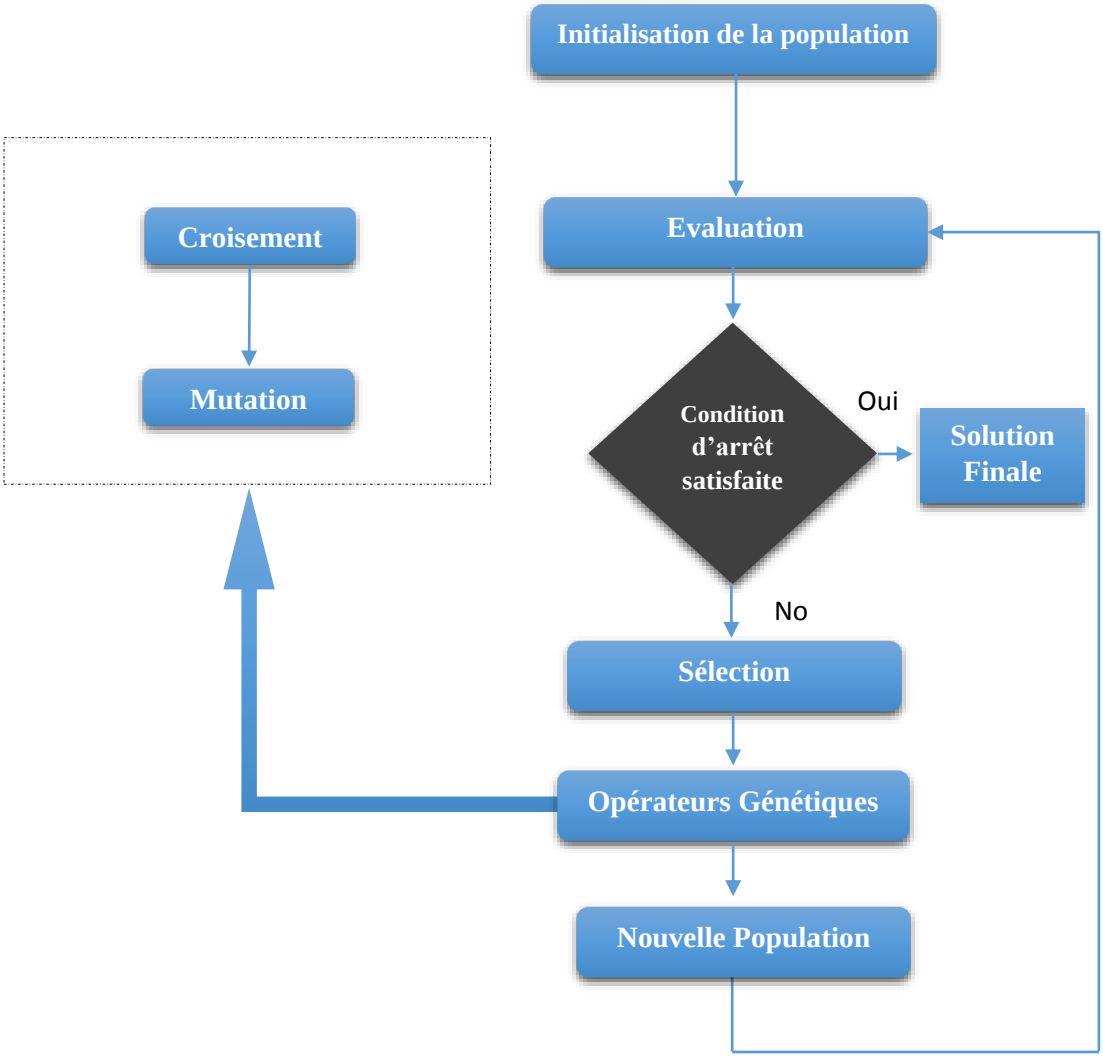

Figure I-1 – Fonctionnement de l'algorithme génétique

I.3.1. Représentation chromosomique (Codage)

Au début, chaque individu (chromosomique) de la population est codé soit en binaire/gray (Goldberg 1989 [16]), ou en réel (Goldberg 1990 [17]), représenté par un vecteur dont la longueur dépend de la précision requise, le codage donne à l'algorithme génétique une représentation génotype-phénotype.

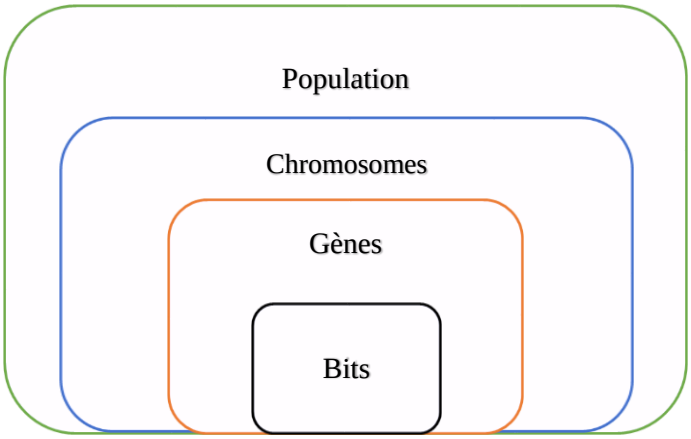

Figure I-2 – Représentation Chromosomique

La rapidité de la convergence d'un AG peut être affectée par le choix initial de la population, car si des renseignements à priori sur le problème sont disponibles, l'orientation du choix de la population peut considérablement améliorer la rapidité du processus de résolution.

Le codage des paramètres d'un AG dépend du problème à optimiser, le codage binaire, gray et réel sont les plus utilisé dans le domaine.

I.3.1.1. Codage binaire

Ce codage est caractérisé par sa simplicité car il ne comporte que deux caractères (0 et 1), chaque individu est représenté par un chromosome et chaque caractéristique de l'individu est un gène. Le chromosome est codé en une suite de bits appelée chaîne binaire, ceci est illustré à travers l'exemple suivant :

Trois variables (203, 13, 45) codées sur 8 bits formants une chaine binaire de taille 24 :

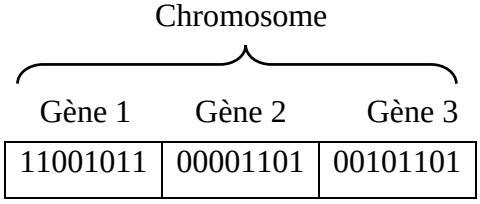

Figure I-3 – Codage Binaire de trois variables

I.3.1.2. Codage réel

L'ensemble des variables est représenté par un vecteur $\bar{x} = < x_1, x_2, \ldots \ldots x_n >$ où chaque x_i est un nombre réel. Les opérateurs de croisement et de mutation seront alors soigneusement définis afin de conserver chacune des variables dans son domaine, la plupart de ces opérateurs s'inspirent des opérateurs du codage binaire.

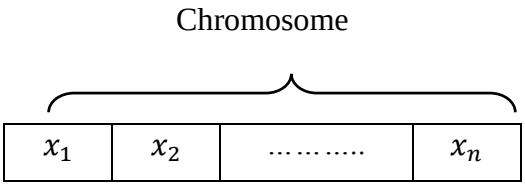

Figure I-4 – Codage Réel

I.3.1.3. Codage gray

Le codage gray a la propriété que deux points adjacents (n et n+1) dans l'espace étudié diffèrent seulement d'un seul bit, autrement dit, une incrémentation d'un pas dans la valeur du paramètre correspond à un changement d'un seul bit dans le code, ce qui est un avantage par rapport au codage binaire qui en terme de distance de Hamming ne code pas forcément deux éléments voisins dans l'espace d'exploration. Le tableau suivant illustre cette différence :

Valeur entière	Binaire	Gray
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

Tableau I-1 - Codage Binaire/Gray

I.3.2. Evaluation

A l'aide d'une fonction objectif (Fitness), l'AG sera en mesure de classer les individus d'une population selon leurs pertinences, elle est une partie inséparable de l'AG car elle représente (sous forme mathématique) le problème à résoudre.

En général, l'algorithme génétique essai de minimiser la fonction de fitness en fonction d'un vecteur ou d'une valeur issus de la population, la solution est donc jugée selon son image par la fonction de fitness. Il existe des fonctions de test qui permettent de mettre en épreuve l'efficacité de l'algorithme génétique utilisé, certaines de ces fonction seront utilisées ultérieurement dans ce chapitre.

I.3.3. Sélection

C'est dans cette étape où se décide quels sont les individus de la population qui vont être dupliqués dans la nouvelle population où ils deviendront des parents. Les individus ayants la meilleure fitness auront donc plus de chance d'être sélectionnés.

Il existe essentiellement quarte méthodes de sélection, leur utilisation dépend du problème étudié :

- La méthode de la "loterie biaisée" (roulette wheel) (Goldberg, 1989 [16]).
- La méthode de rang (Baker, 1985 [3]).
- La sélection par tournoi (Goldberg, Deb, 1991 [18])
- La sélection universelle stochastique (Baker, 1987 [4]).

I.3.3.1. Méthode de la loterie biaisée

C'est la méthode la plus utilisée, elle consiste à dupliquer des individus de la population dont les performances sont relativement bonnes.

Pour un chromosome dont la fitness est $f(ch_i)$, la probabilité $p(ch_i)$ avec laquelle il sera dupliqué dans la nouvelle population de taille N est :

$$p(ch_i) = \frac{f(ch_i)}{\sum_{j=1}^{N} f(ch_j)} \qquad \text{(I-1)}$$

La figure suivante illustre le principe de ce type de sélection, en utilisant une population de quatre individus, l'individu 1 avec la probabilité $p(ch_i) = 0.37$ a plus de chance d'être sélectionné que les autres.

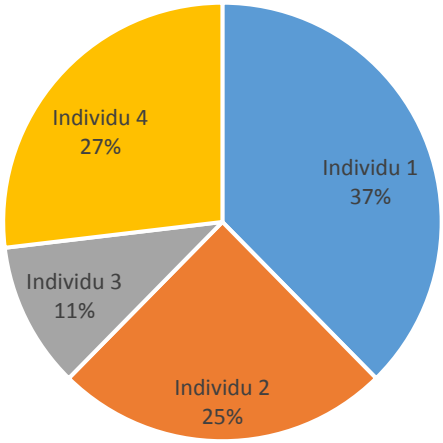

Figure I-5 - Méthode de sélection de la loterie biaisée

I.3.3.2. Méthode de sélection par rang

Le principe de cette méthode est de garder les meilleurs individus (selon leurs fitness) d'une population P_t pour ensuite les introduire dans la prochaine population P_{t+1}. C'est une façon de protéger des solutions potentielles de la disparition lors de la phase de sélection. C'est une méthode performante, mais dans certains cas, le manque de diversité dans les solutions provoque la convergence prématurée ce qui représente l'inconvénient majeur de cette méthode de sélection.

I.3.3.3. Sélection par tournoi

Cette méthode est une forme de duel entre deux individus, celui qui domine l'autre fonction de sa fitness est sélectionné pour être réintroduit dans la nouvelle population, le perdant est écarté. L'opération est répétée jusqu'à ce que la nouvelle population de n individus soit pleine.

I.3.3.4. Méthode « stochastique du reste »

Cette méthode repose principalement sur la fonction d'évaluation et sa moyenne numérique, chaque individu d'une population est dupliqué n_{a_i} fois avec n_{a_i} la valeur de la partie entière de $\left(\frac{f_i}{f_{moy}}\right)$. Chaque individu se fait associer une probabilité de sélection :

$$P_s(a_i) = \left(\frac{f_i}{f_{moy}}\right) - E(\left(\frac{f_i}{f_{moy}}\right)) \quad \text{(I-2)}$$

Avec $E(x)$ La partie entière de x.

I.3.4. Opérateurs génétiques

I.3.4.1. Croisement

Pendant cette opération, la population est divisée en deux sous-populations, ensuite, l'opération de reproduction est appliquée sur chaque individu de la première sous-population avec un autre de la deuxième sous-population, chaque couple de parents forme deux nouveaux chromosomes (enfants) dont les caractéristiques sont issues des deux parents. Le taux de croisement est défini en général entre 0.5 et 0.9, un taux excessivement élevé peut engendrer la perte de bonnes structures, mais aussi un taux faible provoque la stagnation puisque l'exploration est diminuée.

Deux méthodes classiques sont à mentionner, le croisement en un point et le croisement en deux points :

Chaque chromosome de longueur "l" est divisé en segments : deux segments pour le croisement en **un seul point** (Goldberg, 1989 [16]) et trois segments pour le croisement en **deux points** (Man et al. 1996 [23])**.** La position de chaque chromosome est tirée aléatoirement ensuite les gênes des individus parents sont échangés selon une probabilité de croisement P_c pour former deux nouveau chromosomes.

Les schémas ci-dessous illustrent les deux méthodes :

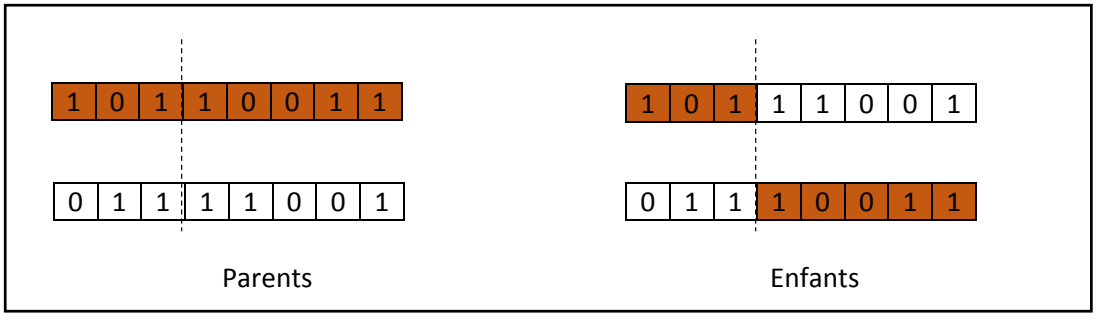

Figure I-6 - Croisement en un seul point

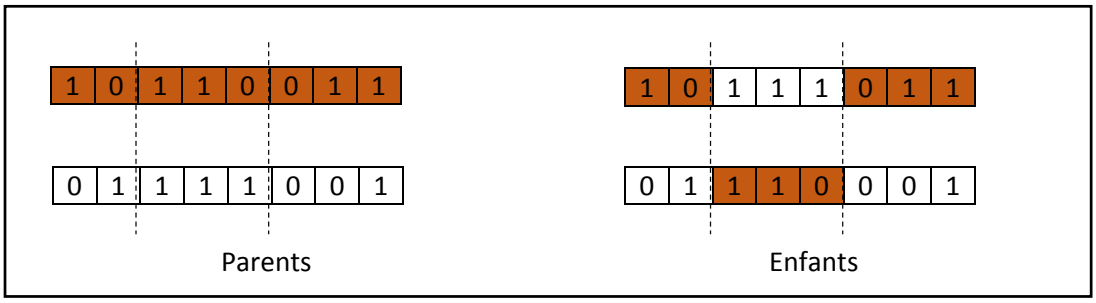

Figure I-7 - Croisement en deux points

I.3.4.2. Mutation

Cet opérateur tire au hasard un ou plusieurs gênes d'un individu, puis les modifie aléatoirement à condition que cet individu reste une solution possible au problème.

Chaque bit d'un individu a une probabilité P_m de subir une mutation, un réel $r \in \{0,1\}$ est généré, si r < P_m le bit sera donc remplacé par complémentaire (Goldberg 1989 [16]).

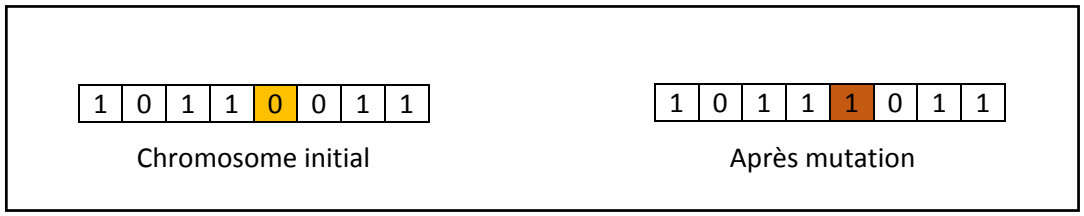

Figure I-8 - Mutation d'un chromosome

L'opérateur de mutation rapporte de la diversité à la population, en évitant la convergence prématurée et donc l'AG peut atteindre la propriété d'ergodicité[2], c'est-à-dire que l'AG peut atteindre tous les points de l'espace de recherche et donc l'optimum global.

I.4 - Application sur quelques fonctions de test

Dans cette partie, nous allons appliquer un algorithme génétique sur deux problèmes d'optimisation qui sont deux fonctions de test, la fonction de McCormick et la fonction Six-Hump Camel. (Adorio et Diliman, ,2005 [1])

Codage	Binaire
Taille de population	80
Nombre de générations	80
Longueur du chromosome	8 bits
Sélection	élitiste
Probabilité de mutation	0.02
Croisement	En un seul point avec $p_c = 0.5$

Tableau I-2 – Paramètres de l'AG mono-objectif

- **Optimisation de la fonction de McCormick :**

Cette Fonction est définie par :

$$f(\mathrm{x1},\mathrm{x2}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1 \qquad \text{(I-3)}$$

$$avec \quad [x_1, x_2] \in [-1.5,4]$$

Le minimum global est : $min\, f(x_1, x_2) = -1.9133$ à $[x_1, x_2] = [-0.54719, -1.54719]$

[2] Possibilité de parcourir l'intégralité l'espace de recherche.

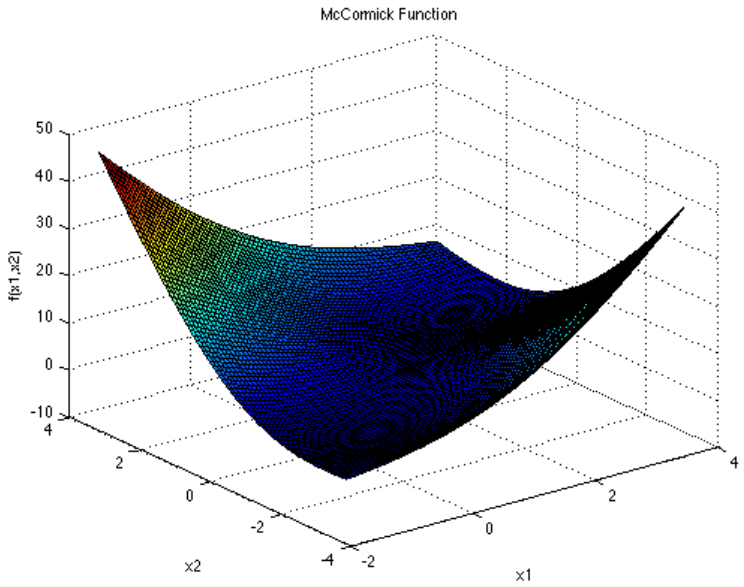

Figure I-9 – Fonction de McCormick

Après l'exécution de l'algorithme génétique, nous avons obtenu les résultats suivants :

$min\ f(x_1, x_2) = -1.9105$

Meilleure solution = $[x_1, x_2] = [-0.52941,\ -1.5]$

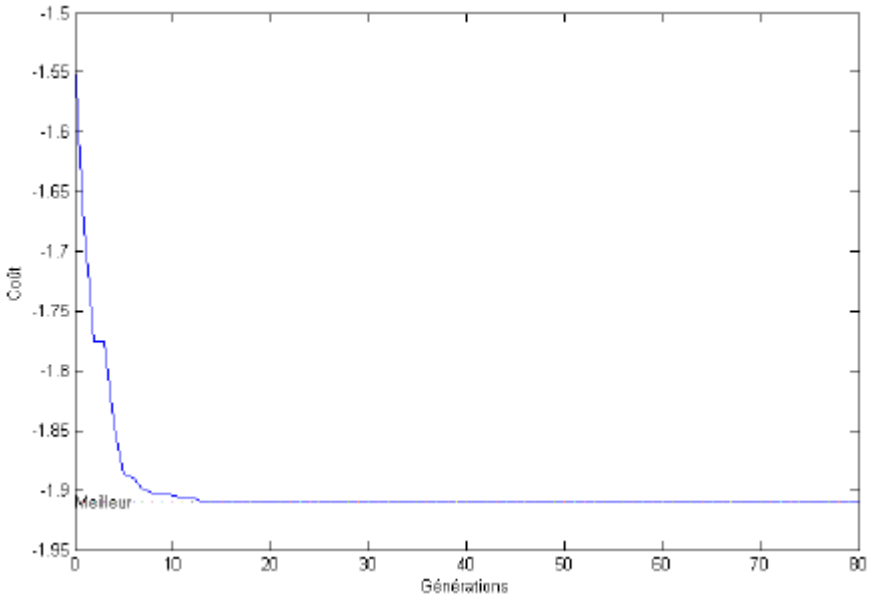

Figure I-10 – (McCormick) Evolution de l'optimum en fonction des générations

D'après la figure I-10, la meilleure solution a été trouvée à la douzième génération, le résultat obtenu est très proche du minimum global :

Minimum global $[x_1, x_2]$	$[-0.54719, -1.54719]$
Résultat obtenu $[x_1, x_2]$	$[-0.52941, \quad -1.5]$

Tableau I-3 - Comparaison des résultats (McCormick)

- **Optimisation de la fonction « *Six-Hump Camel* » :**

Elle est définie par:

$$f\,(x_1, x_2) = \left(4 - 2.1{x_1}^2 + \frac{{x_1}^4}{3}\right){x_1}^2 + x_1 x_2 + (-4 + 4{x_2}^2){x_2}^2 \qquad \text{(I-4)}$$

$$avec \quad [x_1, x_2] \in [-3, 3]$$

Les minimums globaux sont : $min\, f(x_1, x_2) = -1.0316$ à $[x_1, x_2] = [0.0898, -0.7126]$ et $[-0.0898, \ \ 0.7126]$

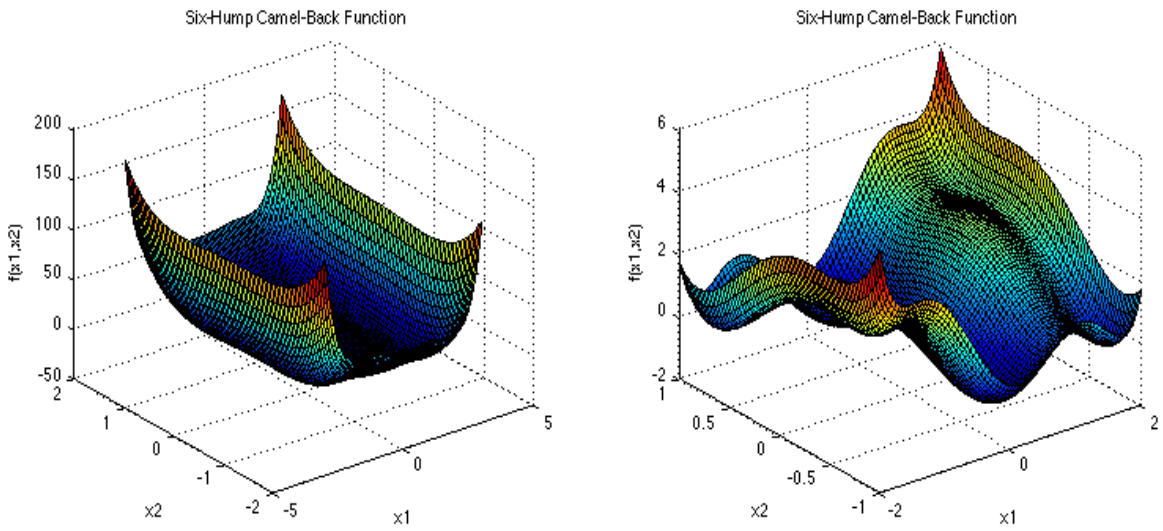

Figure I-11 - Fonction "Six-Hump Camel"

Après l'exécution de l'algorithme génétique, nous avons obtenu les résultats suivants :

$min\ f(x_1, x_2) = -1.0312$

Meilleure solution $= [x_1, x_2] = [-0.082353,\ \ 0.71765]$

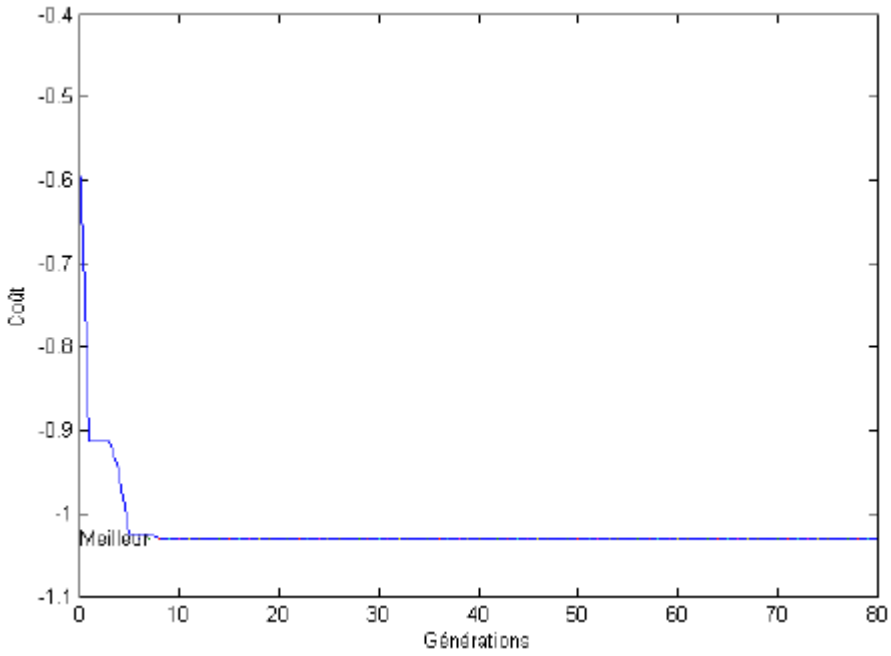

Figure I-12 – (Six-Hump Camel) Evolution du coût en fonction des générations

D'après la figure (I-12), la meilleure solution a été trouvée à la huitième génération, le résultat obtenu est très proche du minimum global :

Minimum global $[x_1, x_2]$	$[-0.0898,\ \ 0.7126]$
Résultat obtenu $[x_1, x_2]$	$[-0.082353,\ \ 0.71765]$

Tableau I-4 - Comparaison des résultats (Six-Hump Camel)

Les résultats des deux tests confirment que les des algorithms génétiques sont une méthode éfficace d'optimisation.

I.5 - Conclusion

Dans ce chapitre, nous avons présenté les notions de base et le principe de fonctionnement des algorithmes génétiques. L'évaluation des solutions par la fonction objectif permet de classer les individus selon leurs performances, la sélection, qui, à travers ses différentes méthodes, imite le processus de la reproduction naturel. Puis en s'appuyant sur les informations génétiques de chaque paire de chromosomes (individus) sélectionnés, l'AG forme deux nouveaux individus enfants qui ensuite seront modifiés pour former une nouvelle population différente et évoluée par rapport à la première. Nous avons également appliqué la méthode des algorithmes génétiques sur quelques fonctions de test mono-objectif, et les résultats obtenus étaient satisfesants, prouvant l'efficacité de cette méthode. Dans le prochain chapitre, nous allons aborder l'optimisation multi-objectif où les AGs se sont également montrés efficaces.

Chapitre II

Optimisation Multi-objectif

II.1 - Introduction

L'optimisation est une notion présente dans tous les domaines, en mathématique le terme optimal ne veut pas dire forcément trouver le maximum ou le minimum, optimiser veut dire trouver la meilleure solution selon plusieurs critères et objectifs, cette solution est appelée optimum, ce chapitre est consacré à l'optimisation multi-objectif avec la méthode des algorithmes génétiques.

II.2 - Optimisation multi-objectif

L'optimisation multi-objectif consiste à trouver un vecteur de décisions qui satisfait les contraintes et optimise le vecteur objectif dont les éléments représentent les fonctions objectif, ces dernières forment une description mathématique du critère de performance, ces fonctions sont généralement en conflit. Le terme « optimiser » veut dire trouver une solution de telle façon que toutes les fonctions objectifs renvoient des valeurs acceptables.

Un problème de minimisation est décrit comme ci-dessous (Zitzler et al. 2000 [38]) :

$$\begin{aligned} &minimiser \;\; y = \big(f_1(x), \ldots\ldots, f_n(x)\big) \\ &Avec \qquad x = (x_1, \ldots\ldots, x_m) \in X \\ &\qquad\qquad y = (y_1, \ldots\ldots\ldots, y_n) \in Y \end{aligned} \tag{II-1}$$

Où x est appelé vecteur de décision constitué de m variables de décision, dans un AGs, il représente un individu (solution potentielle). Et y vecteur objectif de n objectifs. X représente l'espace de décision, et Y l'espace des objectifs.

L'optimisation multi-objectif traite plusieurs fonctions objectifs en même temps, pour cela, elle utilise la notion de compromis[3] optimaux. Les solutions sont départagées en se basant sur la notion de dominance au sens de Pareto.

[3] Une solution qui satisfait plusieurs fonctions en même temps.

II.2.1. Dominance au sens de Pareto

Dans un problème de minimisation, un vecteur de décision $a \in X$ domine un autre vecteur de décision $b \in X$ et on écrit $a <_d b$, (Zitzler et al. 2000 [38]) si et seulement si

1- La solution a n'est pas pire que b dans tous les objectifs

2- a est meilleure que b dans au moins un seul objectif.

$$a <_{\boldsymbol{d}} b \; si \; et \; seulement \; si \begin{cases} f_i(a) \leq f_i(b), & \forall i \in \{1, \dots n\} \quad \text{(II-2)} \\ \exists j \in \{1, \dots . n\}, & f_j(a) < f_j(b) \quad \text{(II-3)} \end{cases} \wedge$$

Si la relation (II-2) est vérifiée alors que la relation (II-3) n'est pas vérifiée, dans ce cas, on parle de **dominance faible.**

II.2.2. Optimum de Pareto

Soit un ensemble de vecteurs de décision $X' \subseteq X$

- X' est dit un ensemble optimal de Pareto **local** si et seulement si

$$X' = \{\forall\, a' \in X' : \nexists a \in X : a <_d a' \wedge \|a - a'\| < \varepsilon\} \quad \text{(II-4)}$$

X' est dit un ensemble optimal de Pareto **local** s'il n'existe pas de solutions $a \in X$ qui dominent les solutions a' $(a' \in ')$ dans le voisinage $\|a - a'\| < \varepsilon$, sans le que reste de l'espace de décision ne soit vérifié. Rappelons que X représente l'espace de décision.

- X'' est dit un ensemble optimal de Pareto **global** si et seulement si

$$X'' = \{\forall\, a'' \in X'' : \nexists a \in X : a <_d a''\} \quad \text{(II-5)}$$

X' est dit un ensemble optimal de Pareto **global** s'il n'existe pas de solutions $a \in X$ qui dominent les solutions a'' $(a' \in '')$ sur tout l'espace de décision X.

L'image de l'ensemble de décision X^* par la fonction objectif représente **le front de Pareto** optimal (global ou local), ce dernier est fait de telle façon qu'on ne peut améliorer une performance (objectif) sans détériorer les autres performances.

La figure suivante illustre l'espace objectif contenant les images des solutions non-dominées, et les solutions dominées dans l'espace objectif, ainsi que le front de Pareto pour un problème de minimisation à deux objectifs :

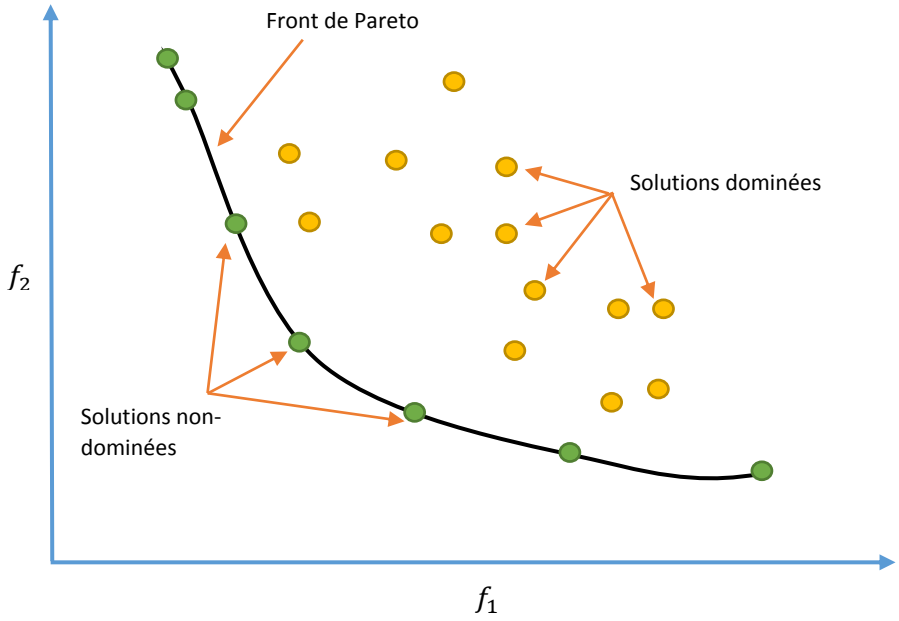

Figure II-1 - Front de Pareto

Dans la partie suivante, nous allons faire une description en l'état de l'art des méthodes d'optimisation par algorithmes génétiques et après on citera quelques exemples de quelques algorithmes génétiques.

II.3 - Optimisation multi-objectif par algorithmes génétiques

Les algorithmes génétiques sont une méthode efficace dans la résolution des problèmes d'optimisation multi-objectif, ils sont répartis en trois classes (Coello et Lamont 2004 [6]) :

- Ceux qui transforment le problème multi-objectif en un problème mono-objectif en utilisant une fonction objectif équivalente Les AGs de cette classe sont appelés : méthodes **agrégées**.
- Les méthodes **non-agrégées non-Pareto** utilisent des sous-populations pour optimiser chaque objectif indépendamment.
- Les méthodes **Pareto** consistent à déterminer un ensemble de solutions optimales, elles favorisent les solutions non-dominées dans leurs processus de sélection en utilisant la notion de dominance au sens de Pareto.

II.3.1. Méthodes agrégées

Elles consistent à transformer le vecteur objectif en un scalaire, il en existe plusieurs types :

II.3.1.1. La moyenne pondérée

Cette méthode additionne tous les objectifs après avoir multiplié chacun d'eux par un coefficient de poids, ce dernier dépend de l'importance de l'objectif, donnant un problème mono-objectif :

$$\begin{cases} min \sum_{i=1}^{n} w_i \, f_i(x) \quad avec \; w_i \geq 0 \\ \sum_{i=1}^{n} w_i = 1 \end{cases} \tag{II-6}$$

II.3.1.2. Goal Programming

Chaque objectif $f_i(x)$ a un but L_i à atteindre, les L_i sont des contraintes introduites dans le calcul des solutions :

$$min \sum_{i=1}^{n} |f_i(x) - L_i| \tag{II-7}$$

Un vecteur de buts *L* est imposé sur les objectifs $f_i(x)$. Le principe de cette méthode est de minimiser les écarts entre les objectifs $f_i(x)$ et leurs valeurs L_i à atteindre. La différence entre cette méthode et la méthode précédente est que dans cette méthode, les valeurs désirées de chaque objectif sont connues, tandis que dans la précédente, c'est l'importance relative des objectifs qui est requise.

II.3.1.3. Le min-max

Son principe est de minimiser le maximum de l'écart relatif entre chaque objectif est son but associé :

$$\min(max_i \left(\frac{f_i(x)-L_i}{L_i}\right) \; avec \; i = 1, \dots, n \tag{II-8}$$

II.3.1.4. Goal attainment

Cette méthode inclut les buts L_i à atteindre ainsi que les poids w_i associés à chaque objectif, la solution optimale est trouvée en résolvant le problème avec :

Minimiser α tel que

$$L_i + \alpha . w_i \geq f_i(x) \tag{II-9}$$

$$avec \sum_{i=1}^{n} w_i = 1$$

Avec α le paramètre à optimiser, en variant les valeurs de w_i, cette méthode peut générer des solutions sur le front de Pareto même dans le cas d'une surface concave, ce qui n'est pas réalisable avec la méthode de la somme pondérée.

II.3.1.5. La méthode ε-contrainte

Son principe est de minimiser f_i à condition que tous les autres objectifs f_j $(j \neq i)$ soient inférieurs ou égaux à une valeur ε_j :

$$\text{Minimiser } f_i(x) \text{ avec}$$
$$f_j(x) \leq \varepsilon_j \quad \forall j \neq i \tag{II-10}$$

Pour pouvoir utiliser cette méthode, des informations à priori sur les contraintes ε_j de chaque objectif doivent être disponibles.

II.3.2. Méthodes non-agrégées non-Pareto

Dans ce type d'approche, la population est utilisée pour diversifier la recherche, mais le principe de Pareto n'est pas directement incorporé dans le processus de sélection. Le modèle le plus représentatif de cette méthode est le Vector Evaluated Genetic Algorithm (VEGA).

II.3.2.1. Vector Evaluated Genetic Algorithm (VEGA)

Proposé par Schaffer en 1985 [28], le VEGA est un AGs multi-objectif avec un processus de sélection particulier. À chaque génération, une sous-population est générée à partir de la population initiale pour chaque objectif, donc pour n objectifs, n sous-populations de p/n individus sont générées (p le nombre d'individus de la population initiale). Ensuite, ces sous-populations sont rassemblées pour obtenir une nouvelle population sur laquelle les opérateurs génétiques seront appliqués (figure II-2).

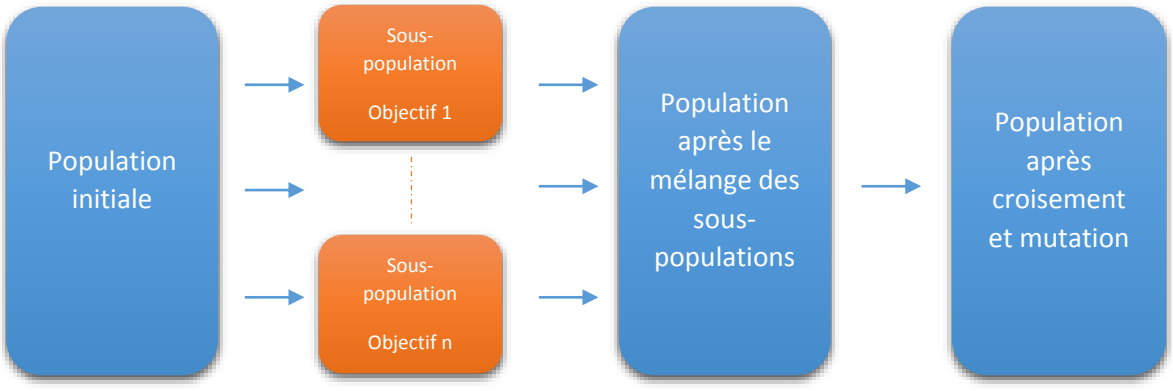

Figure II-2 - Vector Evaluated Genetic Algorithm (VEGA)

II.3.3. Méthodes Pareto

Cette catégorie de méthodes introduit et prend en charge le concept d'optimalité de Pareto dans leurs mécanismes de sélection, il existe plusieurs variétés d'algorithmes génétique qui sont répartis en deux types, on va citer quelques-unes :

II.3.3.1. Les Méthodes non-élitistes

II.3.3.1.1. Nondominated Sorting Genetic Algorithm (NSGA)

Proposée par (Srinivas et Deb 1994 [30]), dans cette méthode, avant que la sélection soit entamée, les solutions sont classifiées à base de non-dominance. Tous les individus non-dominés sont classés dans une catégorie avec une valeur de fitness factice proportionnelle à la taille de la population afin de fournir une possibilité de reproduction égale pour tous les individus.

II.3.3.1.2. Multi-Objective Genetic Algorithm (MOGA)

Fonseca et Fleming (1993 [14]) ont proposé cette méthode qui classifie les individus selon le nombre d'individus qui les dominent, tous les individus non-dominés dans la population obtiennent le même rang et la même valeur de fitness pour qu'ils aient tous la même possibilité d'être sélectionnés. Le MOGA utilise la méthode de formation de niches afin de diversifier la population.

II.3.3.1.3. Niched Pareto Genetic Algorithms (NPGA)

Proposée par Horn et Napfliotis en 1994 [20], cette méthode utilise la sélection par tournoi basée sur la dominance de Pareto. Deux individus de la population sont choisis aléatoirement et comparés à une sous-population, si un des deux est non-dominé et l'autre dominé, alors l'individu non-dominé est sélectionné, s'il y a un nœud (les deux individus sont non-dominée), la fonction de sharing est appliquée, elle permet de distinguer le meilleur entre plusieurs individus non-dominées.

II.3.3.2. Les méthodes élitistes

On appelle élitistes, les méthodes qui conservent les solutions Pareto-optimales pour une réintroduction dans des populations nouvelles.

II.3.3.2.1. Strength Pareto Evolutionary Algorithm (SPEA)

Proposée par Zitzler et Thiele en 1998 [36], cette méthode actualise l'archive (population externe) avec les solutions non-dominées, chaque individu de la population se fait attribuer une valeur de fitness proportionnelle au nombre de solutions qu'il domine. La diversité est préservée en utilisant la technique de clustring qui permet de réguler le nombre d'individus tout en gardant la diversité des solutions.

II.3.3.2.2. Pareto Archived Evolution Strategy (PAES)

Proposée par Knowles et Corne en 1999 [21], cette méthode utilise la stratégie d'évolution (1+1), chaque parent génère un enfant à travers la mutation. Un archive est utilisé afin de préserver les solutions non-dominées et de les comparer par la suite aux individus dans la population, cette méthode utilise une grille qui divise en hypercubes l'espace des objectifs. Un des inconvénients de cette méthode c'est sa performance limitée aux cas d'un front de Pareto discontinu.

II.3.3.2.3. Pareto Envelope Based Selection Algorithm (PESA)

Proposée par Cornes et al. en 2000 [7], le point de différence entre cette méthode et la précédente c'est que cette méthode n'utilise pas la stratégie d'évolution (1+1), elle utilise deux paramètres relatifs à la population : P_1 qui représente la taille de la population interne, P_E la taille de l'archive (population externe), si une solution dans P_1 est non-dominée, elle est systématiquement transférée à l'archive, en même temps, toutes les solutions dominées par cette nouvelle solution sont supprimées de la population.

II.3.3.2.4. Nondominated Sorting Genetic Algorithm (NSGA II)

Proposée par Deb et al. en 2002 [11], dans cette méthode, la population est classifiée selon la dominance de Pareto, elle attribue à chaque individu un rang de non-dominance et une distance de crowding qui permet d'évaluer de la densité de la région autour de cet individu. Cette méthode est bien moins complexe que sa devancière.

II.4 – Conclusion

Ce deuxième chapitre a été consacré à l'optimisation multi-objectif et aux algorithmes génétiques multi-objectifs. Il existe plusieurs AGs et on distingue ceux qui sont basés sur le principe de Pareto, ceux qui travaillent sur une population, et ceux qui n'opèrent qu'avec un seul individu à la fois, leur utilisation dépend du type du problème à résoudre. Afin de comprendre le fonctionnement des AGs multi-objectifs, une description approfondie de quelques algorithmes génétiques est entamée dans le troisième chapitre.

Chapitre III

Description Détaillée des AGMOs

III.1- Introduction

Nous allons procéder dans ce chapitre à une description détaillée de quelques algorithmes génétiques. Nous avons sélectionné quatre méthodes de références, le Nondominated Sorting Genetic Algorithm (NSGA II), le Strength Pareto Evolutionary Algorithm II (SPEA II), le Fast Pareto Genetic Algorithm (FastPGA) et le Multiobjective Cellular Genetic Algorithm (MoCell).

III.2- Nondominated Sorting Genetic Algorithm (NSGA II)

Cette nouvelle version est basée sur la méthode non-élitiste NSGA (Srinivas et Deb, 1994 [30]), le NSGA II (Deb et al. 2002 [11]), corrige les principales failles de la première version qui sont : la complexité, l'approche non-élitiste et la fonction de sharing qui nécessite l'intervention humaine pour fixer certains paramètres. Nous allons dans cette partie, expliquer d'une manière détaillée le fonctionnement, les avantages et les failles de cette méthode.

III.2.1. Définitions

- **Fitness sharing : (**Sareni et Krähenbühl, 1998 [27]) C'est une technique de partage qui modifie l'espace de recherche en réduisant le rendement dans les régions relativement peuplées, elle diminue les fitness des éléments semblables de la population afin d'assurer une diversité :

$$f_i' = {}^{f_i}/_{m_i} \qquad \text{(III-1)}$$

Avec f_i la fitness de l'élément i, et m_i la niche qui contient le nombre d'individus qui partagent la même fitness f_i.

$$m_i = \sum_{j=1}^{N} sh(d_{i,j}) \qquad \text{(III-2)}$$

Avec N la taille de la population et $d_{i,j}$ représente la distance entre l'individu i et l'individu j, cette distance peut être euclidienne dans le cas de codage réel ou de Hamming dans le cas de codage binaire.

La fonction de sharing (sh) mesure le degré de similarité entre les individus i fixé et j allant de 1 jusqu'à N, elle renvoie 1 comme résultat si les éléments comparés sont identiques, et 0 si la distance qui les sépare dépasse le seuil de similarité noté par σ_s. Ce paramètre doit être défini avant l'exécution de l'algorithme, ce qui représente un des inconvénients de cette technique de partage.

- **Crowding distance :** (Deb et al. 2002 [11]) Pour une solution i on appelle « crowding distance » la distance moyenne entre les deux points $(i-1)$ et $(i+1)$ encastrant le plus large cuboïde contenant uniquement la solution i. La figure suivante illustre la notion du crowding distance.

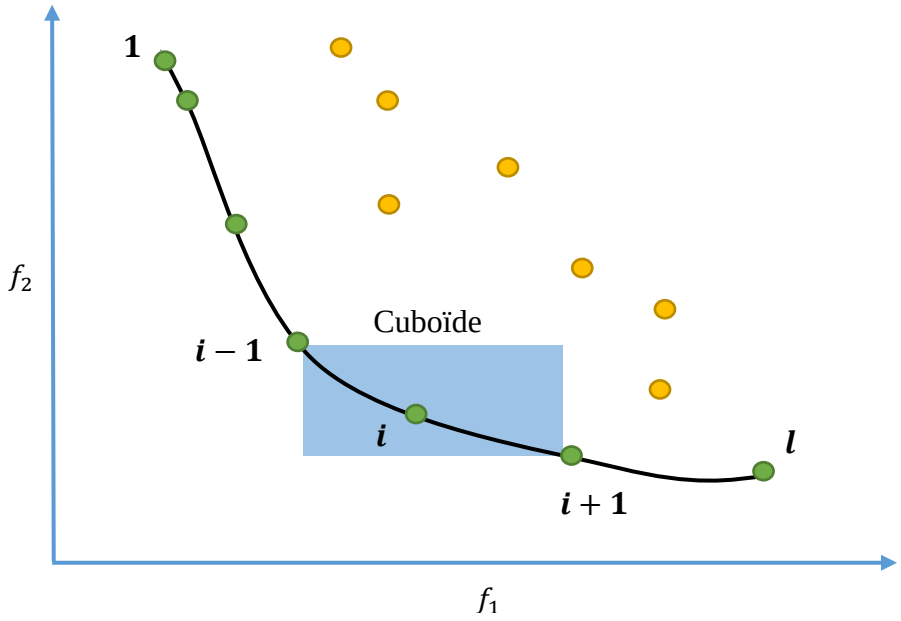

Figure III-1 - Crowding Distance

Le NSGA dans sa deuxième version passe de la technique de partage à la distance de crowding dans son mécanisme de diversité.

- **Simulated Binary Crossover (SBX) :** Introduit par Deb et Agrawal en 1995 [9], le croisement binaire simulé est une adaptation du croisement binaire en un seul point avec quelques amélioration (Goldberg 1989 [16]) pour une utilisation par des algorithmes évolutionnaires codés en réel. Notons que la majorité des AGs récents sont codés en réel, y compris les quatre AGs abordés dans notre travail, vu la complexité de cet opérateur nous recommandons de se référer à l'article original (Deb et Agrawal 1995, [9]).

- **Mutation polynomiale :** Introduite par Deb et Goyal en 1996 [10], destinée au codage réel, ce type de mutation change la valeur d'une variable à une valeur voisine en ajoutant une valeur $\Delta_{\max}$ qui représente la valeur de la perturbation maximale, pour plus de détail, il est préférable de se référer à l'article original (Deb et Goyal en 1996 [10]) .

III.2.2. Les Nouvelles Caractéristiques du NSGA II

Le NSGA I (Srinivas et Deb, (1994) [30]) à son époque était l'un des plus performants de sa catégorie, mais après l'importante avancée dans le domaine de l'optimisation multi-objectif par algorithmes évolutionnaires, cette méthode a montré ses limites dont les principales critiques sont :

- La complexité de calcul, le tri des solutions non-dominées est d'ordre $O(mN^3)$ avec m le nombre d'objectifs et N la taille de la population, dans le cas d'une population assez large, le temps d'exécution devient un sérieux problème sachant que la population est triée à chaque génération.

- L'approche non-élitiste de cette méthode augmente le risque de perdre des solutions potentiellement bonnes et par conséquent la convergence prématurée vers un optimum local se produit fréquemment

- La technique du sharing (que nous avant détaillé antérieurement) qui nécessite la spécification du paramètre du sharing σ_s, et donc une intervention supplémentaire afin d'assurer la diversité.

Deb et al. (2002 [11]) ont alors travaillé sur une version (NSGA II) qui intègre de nouvelles fonctionnalités afin de corriger les défauts qui faisaient sujets de critiques. Les nouveautés apportées sont les suivantes :

- **Une méthode de tri rapide** : Le degré de complexité est réduit à $O(mN^2)$, pour cela :

D'abord, pour chaque solution i, on calcule n_i (indice de dominance) qui représente le nombre de solutions qui dominent i, et S_i, l'ensemble de solutions dominées par la solution i. Le calcul de ces deux entités nécessite une comparaison de complexité $\boldsymbol{O(mN^2)}$. Toutes les solutions dont $n_i = 0$ (non-dominées) sont isolées dans une liste F_1, appelée le front. Puis pour chaque solution i dans F_1, on modifie les indices de dominance de solutions j contenues dans son ensemble S_i en soustrayant 1 de n_j, et chaque j dont n_j renvoie 0 (donc non-dominée) est mis dans une autre liste H. Après avoir balayé toutes les solutions du front courant, F_1 est considérée comme le premier front, et H devient le front actuel F_2 et on refait la même opération jusqu'à ce que toutes les solutions sont diagnostiquées et que tous les fronts soient identifiés. Chaque itération de ce genre est de complexité $O(N)$, et puisque le nombre maximal de front est de N, le plus haut degré de complexité de cette boucle est de $O(N^2)$, la complexité global de l'algorithme donc est égale à $O(N^2)$ + $\boldsymbol{O(mN^2)}$ donc $O(mN^2)$. (O(.) symbolise la complexité (Black, P. E. (2007) [5])

- **La distance du crowding :** Afin de décider quelles sont les meilleures solutions dans un front, le NSGA II les distingue à base de distances de crowding (que nous avons défini en haut).

- **Elitisme :** Le NSGA II est fait de telle façon à intégrer les fronts obtenus dans les populations qui suivent, la distance du crowding et la sélection par tournoi contribuent à l'obtention des populations diverses sans pour autant perdre des bonnes solutions, cette caractéristique est illustrée dans la figure (III-2).

III.2.3. Fonctionnement du NSGA II

Comme son nom l'indique, le Non-dominated Sorting Genetic Algorithm II trie (rapidement) la population à base de non-dominance, voici le pseudocode[4] de cette procédure :

```
tri − non − dominance(P)
S_p = ∅
n_p = 0
pour chaque p ∈ P
    pour chaque q ∈ P
        si (p < q) alors                 Si p domine q alors
            S_p = S_p ∪ {q}              inclure q dans S_p
        sinon si (q < p) alors           Si q domine p alors
            n_p = n_p + 1                incrémenter n_p de 1
    si n_p = 0 alors                     Si aucune solution ne domine p, alors
        p_rank = 1                       Attribuer rang non-dominance
        F_1 = F_1 ∪ {p}                  p devient membre du premier front
i = 1
tant que F_i ≠ ∅                         Pour chaque membre p de F_i
    H ≠ ∅                                initialiser le front H
    pour chaque p ∈ F_i
        Pour chaque q ∈ S_p              modifier chaque membre de l'ensemble S_p
            n_q = n_q − 1                décrémenter n_q de 1
            si n_q = 0 alors             Si n_q = 0, alors
                q_rank = i + 1           attribuer rang non-dominance
                H = H ∪ {q}              q devient membre de l'ensemble H
    i = i + 1
Fin tant que
F_i = H                                  H devient le front courant
```

Pseudocode 1 – Tri à base de non dominance (NSGA II)

[4] Une description d'un programme destinée uniquement à la lecture humaine.

L'algorithme suivant permet de calculer et attribuer une distance de crowding à chaque solution dans un ensemble I

$assignement - distance - crowding(I)$
$l = |I|$ Le nombre de solutions dans I
$pour\ chaque\ i, mettre\ I[i]_{distance} = 0$ Initialisation de la distance
$pour\ chaque\ objectif\ m$
$I = tirer(I, m)$ Tirer les solutions en fonction de leurs coûts
$I[1]_{distance} = I[l]_{distance} = \infty$ Les solutions extrêmes toujours sélectionnées
$pour\ i = 2\ \ jusqu'$à $(l - 2)$ Pour toutes les autres solutions
$a = I[i]_{distance} + (\ I[i+1].m - I[i-1].m)$
$I[i]_{distance} = a/(f_m^{max} - f_m^{min})$ calculer les distances

Pseudocode 2 - Attribution des distances de crowding

Le terme "$I[i].m$" signifie la m-ième image du i-ème individu par la fonction objectif et f_m^{max}, f_m^{min}, les valeurs objectifs maximales et minimales de l'ensemble, la complexité de cette procédure peut atteindre (au cas de toutes les solutions contenues dans un seul front) $O(mN\ logN)$.

L'opérateur de comparaison de crowding ($<_n$) comme nous l'avons déjà spécifié contribue au processus de sélection afin d'assurer une uniformité de dispersion des solutions sur le front:

$$i <_n j \ \ si\ (i_{rank} < j_{rank})\ ou\ \ ((i_{rank} = j_{rank})\ et\ (i_{distance} > j_{distance}))$$

Pseudocode 3 - Opérateur de comparaison de crowding

Avec i_{rank} le classement de l'individu i et $i_{distance}$ la distance de crowding de ce dernier, si i est mieux classé que j selon la non-dominance, i est choisi, si deux individus sont du même rang, l'individu avec la distance de crowding la plus grande est sélectionné.

A l'aide des opérations décrites antérieurement, le NSGA II pourra donc effecteur une recherche rapide, élitiste et moins couteuse, voici le pseudocode de la méthode :

$NSGA - II(N, g)$
$P_0 = initialisation - de - la - population\ (N)$
$Evaluation\ des\ objectifs$
$tri - non - dominance(P_0)$
$générer\ une\ population\ d'enfants\ Q_0$
    $sélection\ par\ tournoi\ binaire$
    $mutation\ et\ recombination$
$pour\ t = 1\ jusqu'à\ g$
    $R_t = P_t \cup Q_t$ — Combiner les 2 populations
    $\{F_1, F_1..F_k\} = tri - non - dominance(R_t)$ — Trier R_t
    $P_{t+1} = \emptyset$ — Initialiser P_{t+1}
    $i = 1$ — Initialiser le compteur
  $tant\ que\ N - |P_{t+1}| \geq |F_i|$ — Tant que P_{t+1} est incomplète
        $P_{t+1} = P_{t+1} \cup F_i$ — inclure les fronts F_i dans P_{t+1}
        $i = i + 1$ — incrémenter *i* de 1
  $Fin\ tant\ que$
  $si\ \ N - |P_{t+1}| < |F_i|\ \&\ N - |P_{t+1}| \neq 0$ — Si dernier front F_i est trop large
    $assignement - distance - crowding(F_i)$ — attribuer les distances
    $tirer\ (F_i, <_n)$ — opérateur de crowding
    $P_{t+1} = P_{t+1} \cup F_i[1: N - |P_{t+1}|]$ — inclure une partie de $F_i \cup P_{t+1}$
  $sinon$ — Si P_{t+1} est complète
    $Q_{t+1} = créer - nouv - pop\ (P_{t+1})$ — créer une nouvelle population
$t + 1$ — Prochaine génération

Pseudocode 4 - NSGA II

Explication : D'abord, une population P_0 aléatoire de taille N est générée, elle est tirée à base de non dominance, chaque solution se fait attribuer une fitness factice égale à son niveau de non-dominance, (1) veut dire non-dominée donc meilleure fitness. Après l'application des opérations génétiques sur P_0, une population enfant Q_0 de taille N est créée. Après, cette première procédure, une boucle commence. Tant que le critère d'arrêt n'est pas vérifié, les populations, parents et enfants sont mélangées ($R_t = P_t \cup Q_t$), puis tirées à base de non dominance. Ensuite, une nouvelle population de parents de taille N est créée à bases de fronts non-dominés.

À l'aide de l'opérateur de comparaison de distance de crowding $<_n$, le dernier front admis dans P_{t+1} est trié à base de crowding ce qui assure une sélection élitiste. Après avoir créé P_{t+1} de taille N, une population d'enfants Q_{t+1} est générée en utilisant les opérateurs génétiques (sélection, recombinaison et mutation).

L'organigramme ci-dessous illustre le fonctionnement du NSGA II :

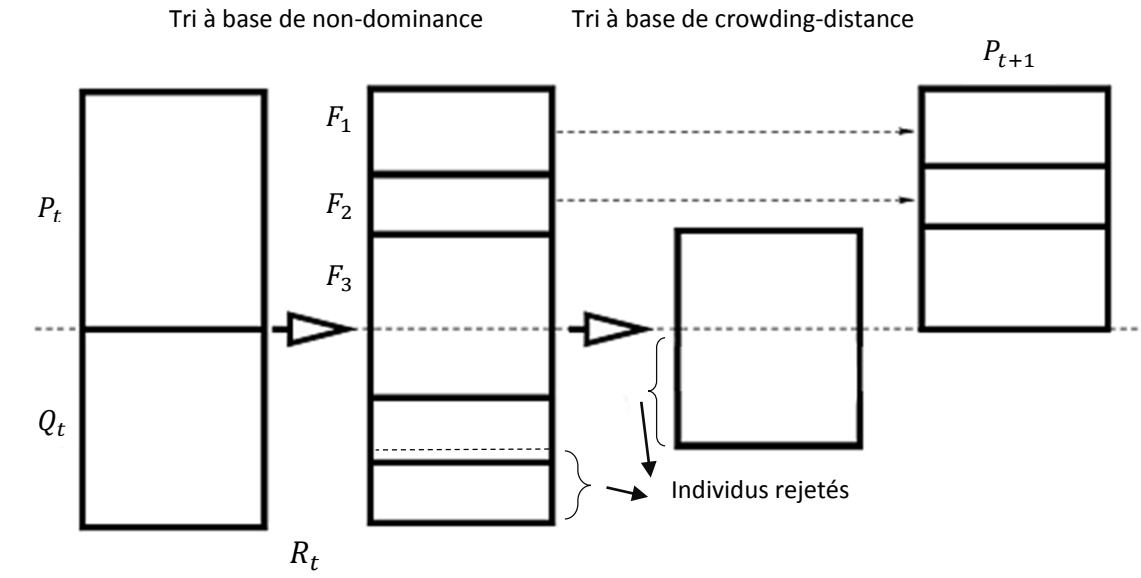

Figure III-2 - Fonctionnement du NSGA II (exemple 3 fronts admis)

III.3 - Strength Pareto Evolutionary Algorithm II (SPEA II)

Le SPEA II (Zitzler et al. 2001 [39]) inclut des améliorations par rapport au (SPEA 1998) telles que l'attribution de fitness. Les principales différences entre les deux versions sont :

- Le processus d'attribution de fitness amélioré de telle façon que pour chaque individu, le nombre d'individus qui le dominent et ceux dominés par ce dernier soit comptabilisé.
- La technique d'estimation de densité « k plus proches voisins », qui accentue la précision durant l'exploration.
- Une nouvelle méthode de troncature de l'archive qui garantit la préservation des solutions extrêmes.

III.3.1. Définitions

- **La sélection environnementale :** En l'addition à la population principale, un archive contenant les solutions non-dominées. Un membre d'archive est supprimé seulement si une solution qui le domine vient d'être repérée dans la population où bien au cas de surcharge de l'archive.

- **La sélection parentale :** À chaque génération, l'ensemble des individus est évalué en deux étapes, d'abord, tous les individus sont comparés à base de dominance de Pareto, ce qui définit un ordre partiel en attribuant un rang de dominance à chaque individu, puis ce classement est raffiné en incluant les informations de densité pour chaque individu. Il existe plusieurs techniques d'estimation de densité qui permettent de garder certaines portions du front non-dominé au cours des générations.

- **La technique du clustering :** Un front non-dominé peut être extrêmement large, par conséquent la distribution des solutions sur le front est non-uniforme ce qui engendre une favorisation de certaines régions dans l'espace de recherche. La technique du clustering permet de résoudre cette problématique, elle consiste à groupe p éléments en q groupes (clusters) avec $q < p$. Puis dans chaque groupe, une solution représentative est déterminée, le reste du groupe est supprimé. le SPEA II utilise une méthode de clustring différente que celle de la première version, ce sujet est abordé dans ce qui suit.

III.3.2. Principales améliorations

Cette nouvelle version inclut une technique d'attribution de fitness qui prend en compte les informations de densité. La taille de l'archive est fixée, à chaque fois que la taille de l'archive diminue, ce dernier est systématiquement rempli alors qu'avec le SPEA, la taille de l'archive varie avec le temps, et comme nous l'avons déjà mentionné, la technique du clustring est légèrement modifiée afin d'intégrer les solutions limites du front, ajoutons à cela, avec le SPEA II, La sélection des parents n'est appliquée que sur l'archive.

- **Attribution de fitness :** Afin d'éviter que des individus dominés par les membres d'archive aient une fitness identique (SPEA), avec SPEA II, pour chaque individu, les solutions qui le dominent et celles dominées par ce dernier sont prises en considération. Pour être plus clair, chaque individu $\boldsymbol{i}$ de l'archive $\overline{\boldsymbol{P_t}}$ et de la population $\boldsymbol{P_t}$ se fait attribuer une valeur de puissance $\boldsymbol{S_i}$ représentant le nombre de solutions qu'il domine :

$$\boldsymbol{S_i} = |\{\boldsymbol{j} | \boldsymbol{j} \in \boldsymbol{P_t} \cup \overline{\boldsymbol{P_t}} \wedge \boldsymbol{i} < \boldsymbol{j}\}| \qquad \text{(III-3)}$$

$|\,.\,|$ représente la cardinalité de l'ensemble et $(<)$ représente la relation de dominance.
Basant sur la valeur $\boldsymbol{S_i}$, la valeur de fitness brute de $\boldsymbol{i}$ est donnée par :

$$\boldsymbol{R_i} = \sum_{\boldsymbol{j} \in \boldsymbol{P_t} \cup \overline{\boldsymbol{P_t}}, \boldsymbol{j} < \boldsymbol{i}} \boldsymbol{S_j} \qquad \text{(III-4)}$$

R_i représente la fitness brute de la solution i déterminée par les valeurs des puissances des solutions qui la domine dans l'archive mais aussi dans la population, contrairement à SPEA où seulement les membres de l'archive sont comptabilisés. Sachant que dans un problème de minimisation, $R_i = 0$ correspond à un membre non-dominé.

Bien que l'attribution des fitness brutes fournit un mécanisme de niches basé sur le concept de dominance de Pareto, cette technique montre ses limites dans le cas où la plupart des solutions ne dominent pas les unes les autres. L'incorporation des informations de densité permet à l'AG de discriminer les individus ayants des fitness brutes identiques.

La technique d'estimation de densité utilisée dans SPEA II est une adaptation de la méthode "*k-th* nearest neighbor" (Silverman 1986 [29]). Pour chaque individu i dans l'espace objectif, les distances vers tous les individus j dans l'archive et de la population sont calculées et rangées dans une liste, après avoir tiré la liste selon un ordre croissant. L'élément k donne les distance recherchées, notées σ_i^k, k est égale à la racine de la taille de l'espace étudié $k = \sqrt{N + \overline{N}}$, après cela, la densité D_i de l'élément i est définie comme suit : (III-5)

$$\boldsymbol{D_i = \frac{1}{\sigma_i^k + 2}}$$

Au dénominateur, 2 est ajouté pour assurer que la valeur ne sera jamais nulle et que $\boldsymbol{D_i < 1}$.

Finnalement l'addition de la densité $\boldsymbol{D_i}$ avec la fitness brute $\boldsymbol{R_i}$ donne la valeur de la fitness $\boldsymbol{F_i}$:

$$\boldsymbol{F_i = R_i + D_i} \quad \text{(III-6)}$$

- **La sélection environnementale :** L'actualisation de l'archive de SPEA II diffère de celle de la première version en deux points, premièrement, la taille de l'archive toujours la même, deuxièmement, la nouvelle méthode de troncation préserve les solutions extrêmes.

La première étape de la sélection environnementale consiste à copier tous les individus non-dominés (qui ont une fitness inférieure à 1) de l'archive et de la population vers l'archive de la prochaine génération.

$$\boldsymbol{\overline{P}_{t+1} = \{i | i \in P_t \cup \overline{P}_t \wedge F_i < 1\}} \quad \text{(III-7)}$$

Si le nombre des individus non-dominés est identique à la taille de l'archive $\boldsymbol{|\overline{P}_{t+1}| = \overline{N}}$, alors la première étape de la sélection environnementale est terminée, sinon, il pourrait y avoir deux possibilités : soit l'archive est trop court pour contenir toutes les solutions non-dominées $\boldsymbol{|\overline{P}_{t+1}| < \overline{N}}$, ou bien trop large $\boldsymbol{|\overline{P}_{t+1}| > \overline{N}}$. Concernant le premier cas, l'archive $\boldsymbol{\overline{P}_{t+1}}$ est rempli par les $\boldsymbol{\overline{N} - |\overline{P}_{t+1}|}$ meilleures solutions dominées à partir de $\boldsymbol{P_t \cup \overline{P}_t}$.

Ceci se fait en tirant les éléments de $\boldsymbol{P_t} \cup \overline{\boldsymbol{P}}_{\boldsymbol{t}}$ selon leurs fitness, puis à partir de la liste ordonnée, copier les premiers $\overline{\boldsymbol{N}} - |\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}|$ éléments $\boldsymbol{i}$ ayants $\boldsymbol{F_i} \geq \mathbf{1}$ vers $\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}$.

Concernant le deuxième cas où l'archive est trop large, le nombre des éléments dans $\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}$ dépasse $\overline{\boldsymbol{N}}$, l'algorithme fait recours à la technique de troncation de l'archive qui supprimera itérativement des individus de l'archive $\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}$ jusqu'à ce que $|\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}| = \overline{\boldsymbol{N}}$:

A chaque itération, l'individu $\boldsymbol{i}$ est choisi pour la suppression pour qui $\boldsymbol{i} \leq_{\boldsymbol{d}} \boldsymbol{j}$, pour tout $\boldsymbol{j} \in \overline{\boldsymbol{P}}_{\boldsymbol{t+1}}$ avec :

$$\boldsymbol{i} \leq_{\boldsymbol{d}} \boldsymbol{j} : \Leftrightarrow \begin{cases} \forall\, \mathbf{0} < \boldsymbol{k} < |\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}|\colon\ \boldsymbol{\sigma_i^k} = \boldsymbol{\sigma_j^k} \\ \exists\, \mathbf{0} < \boldsymbol{k} < |\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}|\colon\ \left[\left(\forall\, \mathbf{0} < \boldsymbol{l} < \boldsymbol{k} :\ \boldsymbol{\sigma_i^l} = \boldsymbol{\sigma_j^l}\right) \wedge\ \boldsymbol{\sigma_i^k} < \boldsymbol{\sigma_j^k}\right] \end{cases}^{\vee} \qquad \text{(III-8)}$$

Avec $\boldsymbol{\sigma_i^k}$ la distance de $\boldsymbol{i}$ envers son *k* plus proches voisins dans $\overline{\boldsymbol{P}}_{\boldsymbol{t+1}}$. D'une manière générale, l'individu avec la plus courte distance envers un autre est supprimé de l'archive, et s'il existe plusieurs individus avec la même distance minimale, ce nœud est dénoué en considérant les deuxièmes plus courtes distances et ainsi de suite, cette procédure est illustrée dans la figure ci-dessous :

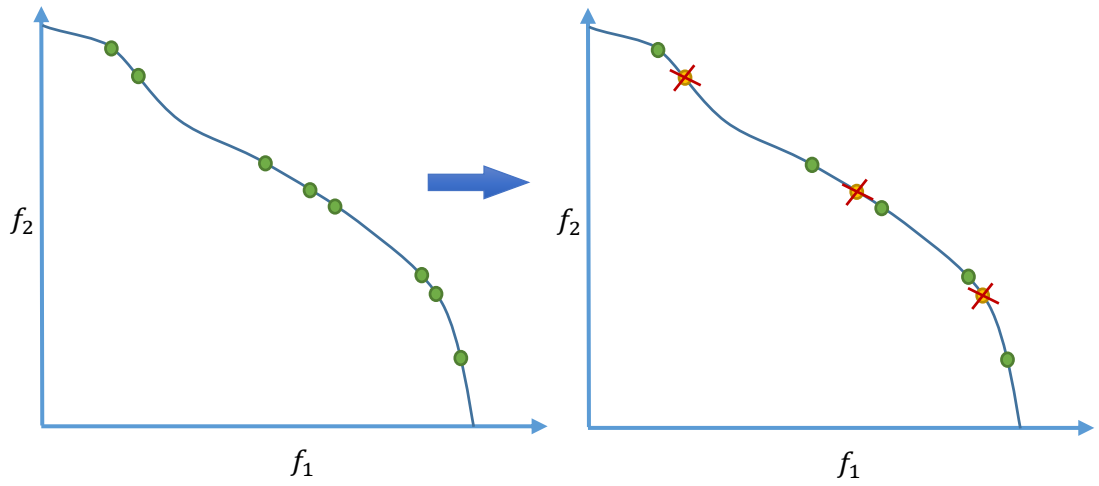

Figure III-3 - Troncation de l'archive (exemple $\overline{\mathbf{N}} = \mathbf{5}$)

III.3.3. Le Fonctionnement du SPEA II

Le SPEA II a été conçu de telle sorte à corriger et à mettre à jours certaines techniques par rapport à la première version (Zitzler et Thiele 1998) afin d'améliorer ses performances, voici le pseudocode de la méthode :

$SPEA - II(N, g)$	
$P_0 = générer - population - initiale(N)$	
$\overline{P_0} = \emptyset$	Initialiser l'archive
$t = 0$	Initialiser le compteur de générations
$Pour\ t = 1\ jusqu'$à g	
$Attribuer - fitness(P_t \cup \overline{P_t})$	Attribuer les fitness archive + population
$Sélection - environnementale(\overline{P_t})$	Actualiser l'archive
$S = Sélection - Parentale((\overline{P_{t+1}}, N)$	Sélectionner les parents (tournoi)
$P_{t+1} = Opérateurs - génétiques(S)$	Croisement, mutation pour donner P_{t+1}
$t = t + 1$	Prochaine génération

Pseudocode 5 - SPEA II

Explication : Tout d'abord, une population P_0 est aléatoirement générée, l'archive $\overline{P_0}$ est initialisé, la première étape consiste à attribuer des valeurs de fitness à tous les individus de l'archive et de la population ($P_t \cup \overline{P_t}$). Puis, à l'aide de la sélection environnementale l'archive est actualisé afin de contenir les meilleures solutions possibles, puis une sélection avec tournoi est effectuée sur l'archive $\overline{P_t}$. Enfin, les opérateurs génétiques (croisement, mutation) sont appliqués sur la population résultante donnant une nouvelle population P_{t+1}, ces opérations sont répétées tant que le critère d'arrêt[5] n'est pas satisfait. Le diagramme suivant illustre le fonctionnement de l'algorithme SPEA II.

[5] Il peut être le nombre de générations, d'évaluations ou une fitness précise.

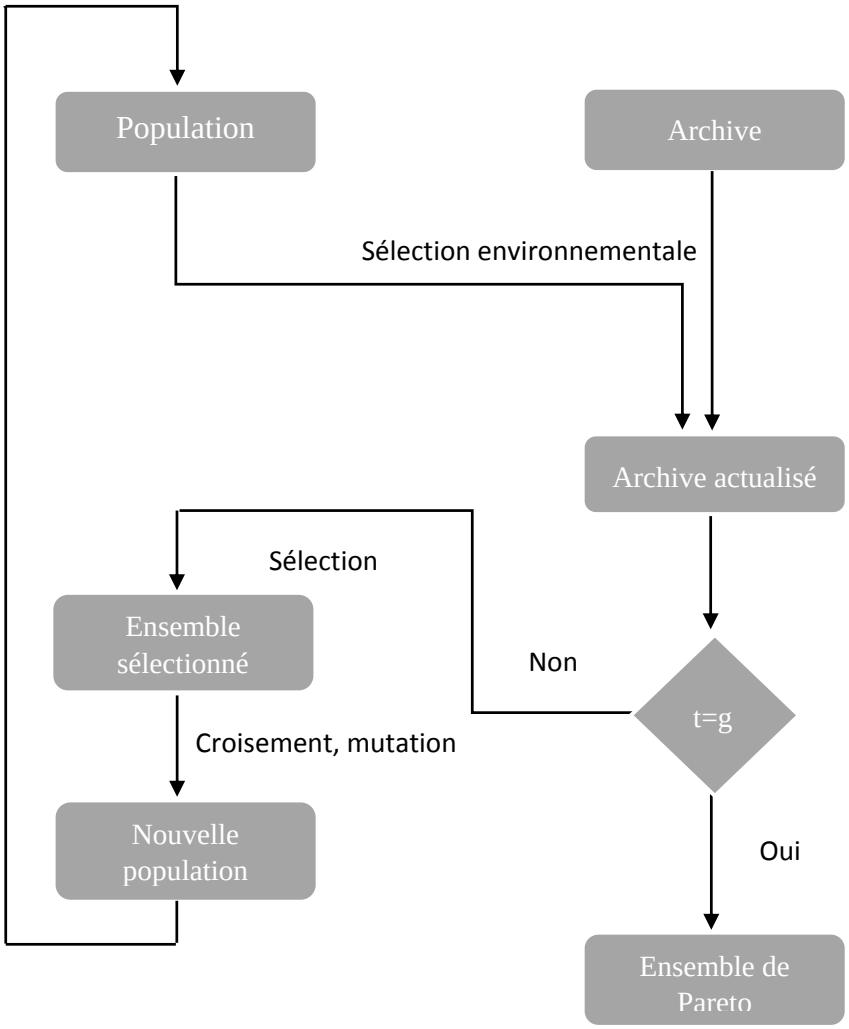

Figure III-4 - Fonctionnement du SPEA II

III.4- Fast Pareto Genetic Algorithm (FastPGA)

Cet algorithme a été mis au point par Eskandari et al. en 2007 [13], cette méthode introduit une nouvelle stratégie d'attribution de fitness et de classification des individus. Un opérateur de régulation de la population a été incorporé afin de modifier dynamiquement la taille de la population selon la nécessité, dans la limite de la taille maximale.

III.4.1. Caractéristique du FastPGA

Le FastPGA est un algorithme à base de population, il utilise un codage réel afin d'éviter les difficultés du codage binaire, essentiellement dans les opérations à espaces de recherche continus de larges dimensions, les caractéristiques de cette méthode sont détaillées dans ce qui suit :

- **Initialisation et évaluation :** Après l'initialisation de la population, les éléments de cette dernière sont évalués, et à chaque génération, les solutions obtenues ainsi que leurs valeurs objectifs correspondantes sont sauvegardées. Dans le FastPGA, avant que la classification et l'attribution des fitness soient effectuées, la nouvelle population O_t créée à l'aide du croisement et de la mutation est combiné avec la population précédente P_{t-1} afin de former une population composée $CP_t = \{P_{t-1} \cup Q_t\}$.

- **Classification et attribution de fitness :** Les solutions candidates de la population composée CP_t sont classifiées sous deux catégories selon leurs statuts de dominance, toutes les solutions non-dominées sont mises au premier rang alors que toutes les solutions dominées sont mises dans le deuxième rang. Ces rangs sont utilisés pour l'évaluation des fitness dans le processus de reproduction. Les fitness des solutions non-dominées (du premier rang) sont calculées en comparant ces dernières les unes avec les autres, ces valeurs sont calculées en utilisant la méthode de « crowding distance » proposée par Deb et al. (NSGA II).

Chaque solution dans le second rang est comparée à toutes les autres et se fait attribuer une fitness selon le nombre des solutions qui la dominent, le concept est similaire à celui de la méthode SPEA II, en revanche, dans le FastPGA, ce concept est généralisé, en d'autres mots, l'attribution de fitness prend en compte les solutions dominées et les non-dominées pour toute solution dominée i.

Chaque solution i de la population composée CP_t se fait attribuer une valeur de puissance S_i qui indique le nombre de solutions qu'elle domine :

$$S_i = |\{j | \forall\, j \in CP_t \wedge i < j \wedge j \neq i\}| \qquad \text{(III-9)}$$

Ensuite, la fitness de la solution dominée *i* est calculée en utilisant la relation suivante

$$F_i = \sum_{i<j} S_i - \sum_{k<i} S_k \;\; ; \;\; \forall\, j,k \in CP_t \wedge i \neq j \neq k \qquad \text{(III-10)}$$

La fitness attribuée à chaque solution dominée i est égale à la somme des valeurs de puissance des solutions qu'elle domine moins l'intégralité des valeurs de puissance des solutions qui la dominent. Contrairement au SPEA/II, qui ne prend en considération que les valeurs de puissance des solutions qui dominent i, cette stratégie fournis davantage plus d'informations sur la dominance de Pareto et réduit le risque que deux solutions aient la même valeur de fitness. Le FastPGA n'a pas besoin alors de mécanisme de préservation de diversité additionnel pour les solutions dominées (second rang). Si la plupart des solutions sont au premier rang, l'opérateur de distance de crowding est utilisé afin de maintenir la diversité.

Après le calcul de toutes les fitness des solutions candidates de la population CP_t, les solutions sont comparées, là où un des trois scénarios se produit.

- Dans le premier scénario, deux solutions de rangs différents sont sélectionnées, dans ce cas, la solution avec le rang supérieur (premier) est préférée.

- Deuxième scénario, les deux solutions ont le même rang mais différentes fitness, celle avec la plus grande fitness est préférée.

- Dernier scénario, les deux solutions ont le même rang et la même fitness, dans ce cas, une solution est choisie aléatoirement.

- **Elitisme et régulation de la population :** Un opérateur d'élitisme est introduit afin d'assurer la propagation des solutions non-dominées à travers les générations, ceci est accompli en copiant toutes les solutions de la population des générations précédentes P_{t-1} vers la population composée CP_t. La combinaison de P_{t-1} avec la population générée O_t donne l'opportunité de garder les meilleures solutions dans

la prochaine génération et d'abandonner les mauvaises solutions selon le nombre total de solutions non-dominées obtenues dans la population composée.

Le nombre des solutions non-dominées augmente à travers les générations impliquant une basse intensité d'élitisme pendant les premières générations si la taille de la population est significativement grande et invariante. De plus, la fluctuation du nombre de solutions non-dominées à travers les générations nécessite une stratégie adaptative de dimensionnement de la population afin de mettre en place la bonne intensité d'élitisme sur les solutions non-dominées. Si l'intensité d'élitisme est forte, elle risque une convergence prématurée, si elle est faible, la convergence risque d'être très lente et compliquée, pour cela le FastPGA utilise un opérateur de régulation afin d'ajuster dynamiquement la taille de la population jusqu'à ce qu'elle atteigne la taille maximale prédéfinie, elle est calculée comme suite :

$$|P_t| = min\{a_t + \lceil b_t * |\{i|\ i \in CP_t \wedge i\ est\ non - dominé\}|\rceil, taillepopmax\} \quad \text{(III-11)}$$

avec $|P_t|$ la taille de la population à la génération t, a_t est un entier positif qui peut changer au cours des générations, b_t est une variable réelle positive qui peut changer au cours des générations, $\lceil x \rceil$ est l'entier $\geq x$.

Dans notre étude on fixe $a_t = 20,\ b_t = 1\ \ et\ taillepopmax = 100$, on aura donc :

$$|P_t| = min\{20 + |\{i|\ i \in CP_t \wedge i\ est\ non - dominé\}|, 100\} \quad \text{(III-12)}$$

Pour être plus explicite, la taille de la population à la génération t est égale à 20 plus le nombre de solutions non-dominées dans la population composée CP_t, si elle n'est pas plus grande ou égale à la taille maximale prédéfinie de la population (100).

A chaque génération, le FastPGA génère un petit nombre de solutions à travers le croisement et la mutation :

$$|O_t| = min\{c_t + \lceil d_t * |\{i|\ i \in CP_t \wedge i\ est\ non - dominé\}|\rceil, taillepopmax\} \quad \text{(III-13)}$$

avec $|O_t|$ le nombre d'enfants créés à la génération t, c_t est un entier positif, d_t est un réel positif, ces derniers peuvent varier au cours des générations.

Dans notre étude, on fixe $c_t = 20$, $d_t = 0$, ce qui donne $|O_t| = 20$. Sachant que $maxsoleval$ représente le nombre maximum d'évaluations effectuées par l'algorithme, cet opérateur de régulation contribue à la réduction d'évaluations de solutions surtout dans les problèmes ayants de larges populations, par conséquent, $maxsoleval$ n'est pas donc un obstacle pour le FastPGA, ce qui le rend approprié pour les problèmes couteux en matière de calculs. Dans les problèmes où $maxsoleval$ est petit, plus d'importance pour l'exploitation et moins pour l'exploration peut être bénéfique, pendant une exécution, le FastPGA nécessite un petit nombre d'évaluations de solutions, ce qui permet un nombre plus important de générations.

- **Critère d'arrêt :** Le FastPGA utilise un nouveau critère d'arrêt qui prend en compte la vitesse de convergence vers le vrai front de Pareto. Quand le nombre de solutions non-dominées atteint la taille maximale de la population et par conséquent, pas de changement concernant le nombre de solutions non-dominées à travers les générations, la recherche s'arrête, le critère est défini comme suit :

Le rapport de production de Pareto (*RPR*) est le rapport entre le nombre de solutions non-dominées et la taille de la population en cours à n'importe quelle génération t, il est calculé comme suit :

$$RPR = \frac{|NP_t|}{|P_t|} \tag{III-14}$$

avec $|P_t|$ la taille de la population à la génération t et $|NP_t|$ le nombre de solutions non-dominées appartenant à la population P_t. Quand *RPR* atteint 1, cela veut dire que toutes les solutions de la population sont non-dominées et ne change pas au cours des générations. Cela implique qu'aucune solution non-dominée supplémentaire ne peut être trouvée, la recherche donc s'arrête, et si RPR n'atteint pas 1 après un certain nombre prédéfini de générations ou d'évaluations, le critère d'arrêt peut être soit le nombre de générations, soit le nombre d'évaluations.

III.4.2. Le Fonctionnement du FastPGA

Le pseudo code suivant illustre le fonctionnement général de l'algorithme :

$FastPGA(N, g)$
$t = 0$
$P_t = créer - population - initiale(N)$
$Evaluer(P_t)$
$Tant\ que\ le\ critère\ d'arrêt\ n'est\ pas\ atteint$
    $t = t + 1$
    $P'_t = sélection(P_{t-1})$
    $O_t = croisement, mutation(P'_t)$
    $Evaluer(O_t)$
    $CP_t = P_{t-1} \cup O_t$
    $Classifier\ (CP_t)$
    $Réguler(CP_t)$
    $P_t = générer(CP_t)$
$Fin\ tant\ que$

Pseudocode 6 - FastPGA

Explication : D'abord, l'algorithme génère une population initiale aléatoirement. S'il s'agit de la première génération, alors la population est directement évaluée, sinon, incrémenter le compteur de générations. Ensuite, il sélectionne des paires de solutions P'_t à partir de la population précédente P_{t-1} à l'aide de la sélection par tournoi. Après, il effectue le croisement et la mutation sur P'_t pour produire une population enfant O_t puis il évalue les solutions de cette dernière, et combine O_t et P_{t-1} afin de former une population composée CP_t. Puis il classifie les solutions de cette dernière selon la nouvelle stratégie de classification (expliquée antérieurement), enfin, il régule CP_t et génère une nouvelle population P_t. La recherche est arrêtée si le critère d'arrêt est atteint.

III.5- Multiobjective Cellular Genetic Algorithm (MOCell)

Le MOCell (proposé par Nebroest et al. en 2009 [25]) est une adaptation du modèle canonique[6] de l'algorithme génétique. Comme plusieurs AGMOs (FastPGA, SPEA II), Le MOCell utilise un archive externe pour sauvegarder les solutions non-dominées, en revanche, ce qui caractérise le MOCell, c'est sa structure de population qui est sous forme de graphe divisé en cellules, ce qui lui donne la caractéristique cellulaire.

III.5.1. Définitions

- **Graphe :** Un graphe G est composé d'un ensemble d'éléments V(G) appelés "sommets" et d'un ensemble E(G) d'éléments appelés "arêtes", G est dit un graphe connexe s'il existe un chemin de n'importe quel point de G vers n'importe quel autre point du même graphe (Tutte, (1966) [33]). La figure suivante illustre la structure de la population d'un AG cellulaire sous forme d'un graphe connexe :

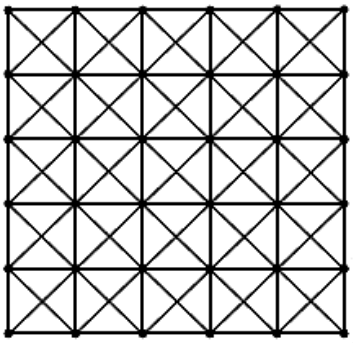

Figure III-5 - Structure de population sous forme de graphe

- **Le modèle cellulaire : (**Alba et Dorronsoro, 2009 [2]) l'idée essentielle de ce modèle est la structure particulière de la population. Elle est représentée sous forme d'un graphe connexe dans lequel chaque sommet représente un individu qui communique avec ses plus proches voisins. Les individus situés sur les bords de la grille sont connectés avec les individus sur les bords opposés ce qui donne une grille toroïdale où les individus ont tous le même nombre de voisins. Les individus ne sont autorisés à se recombiner (croiser) qu'avec leurs proches voisins, ce qu'on appelle *isolation par distance* (Wright, Sewall

[6] Simple, mono-objectif.

(1946) [35]). L'ensemble des partenaires potentiels d'un individu est appelé son « voisinage ».

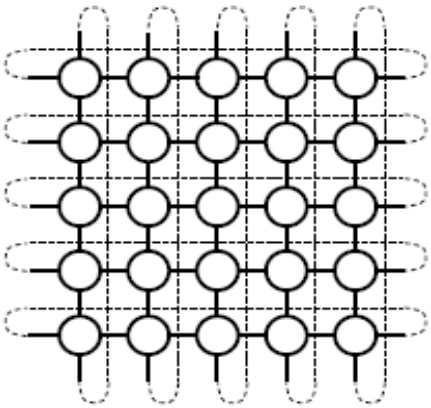

Figure III-6 - Population toroïdale

- **Voisinage :** Le voisinage d'un individu de la grille est défini en termes de *distance de Manhattan (*Krause, 1973 [22]*)* entre l'individu considéré et les autres individus de la population. Chaque point (individu) de la grille possède un voisinage qui chevauche les voisinages des individus à proximité, tous les voisinages ont la même taille et sont tous du même type, les types des voisinages les plus récurrents sont le L_5 (linaire) appelé *Von Neumann*, il désigne le voisinage selon les directions axiales (nord, sud, est, ouest) et le C_9 (compact) appelé voisinage de *Moore* (horizontal, vertical, diagonal).

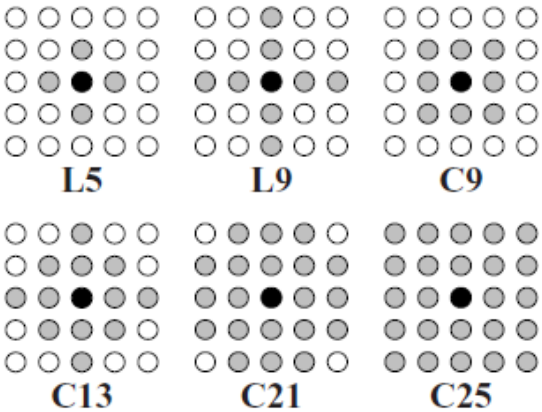

Figure III-7 - Types de voisinages

III.5.2. Caractéristique du MOCell

Cette méthode est basée sur le modèle canonique des algorithmes génétiques, le pseudocode suivant illustre le fonctionnement de ce dernier :

1: $cGA(N)$
2: $P = Céer\ une\ population\ initiale\ et\ l'évaluer$
3: $R = Population\ auxiliaire$
4: $tant\ que\ condition\ d'arrêt\ n'est\ pas\ satisfaite$
5: $Pour\ chaque\ individu\ i\ jusqu'à\ N$
6: $l = acquérir_{voisiage(position(individu))}$
7: $Parents = sélection(l)$
8: $Enfant = Recombination, mutation\ (Parents)$
9: $Evaluer(Enfant)$
10: $R = remplacer\ (Enfant)$
11: $P = R$
12: $Fin\ tant\ que$

Pseudocode 7 - AG cellulaire Mono-objectif (canonique)

Les individus ne peuvent interagir qu'avec leurs voisins (ligne 6) pendant le cycle de reproduction où les opérateurs de variation sont appliqués (croisement/recombinaison, mutation) (ligne 8). Ce cycle de reproduction est exécuté pour chaque individu en son voisinage (ligne 5 – ligne 10). Ça consiste à sélectionner deux parents parmi ses voisins (ligne 7), puis appliquer les opérateurs de variation, puis remplacer l'individu par l'enfant récemment créé si l'enfant représente une meilleure solution que l'individu choisi, La figure suivante illustre le cycle de reproduction d'un AG cellulaire canonique :

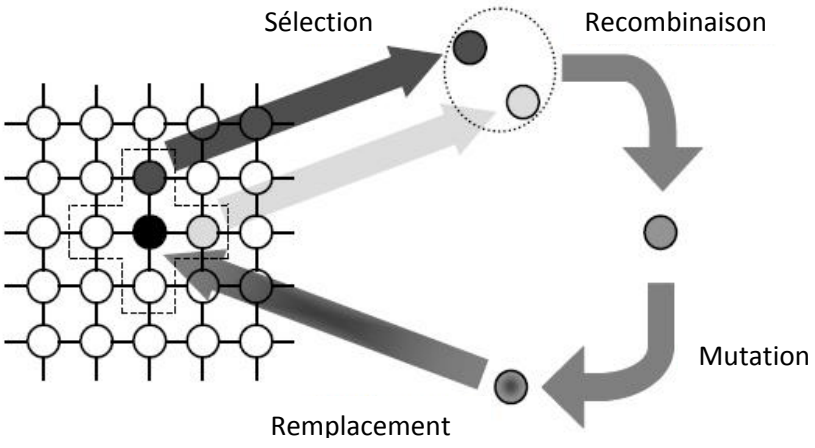

Figure III-8 - Cycle de reproduction (AG cellulaire canonique)

Il existe deux types d'AG cellulaires, ils diffèrent selon les techniques de mise à jour de leurs populations :

- **Les AG cellulaires synchrones :** Dans ce type, les nouveaux individus issus de la reproduction sont mis dans une population temporaire, puis à la fin de la génération, la nouvelle population remplace l'ancienne, le *pseudocode 7* est celui d'un AG cellulaire synchrone.

- **Les AG cellulaires asynchrones :** La différence avec le premier type est que cette technique permet de mettre à jour la population à chaque fois qu'un individu enfant est créé, ce dernier est inséré dans la population pendant la génération pour remplacer la pire solution du voisinage dont les parents de l'enfant sont issus. Le *pseudocode 7* peut facilement être transformé en un AG asynchrone en supprimant la population temporaire et en modifiant la ligne 10 de telle sorte que l'enfant est inséré dans la population en cours : $P = remplacer\ (Enfant)$.

III.5.3. Fonctionnement du MOCell

Le MOCell n'est pas très différent du modèle canonique (standard mono-objectif) de l'AG cellulaire, la différence principale c'est que le MOCell utilise un archive qui contient les solutions non-dominées :

1: $cGA(N)$
2: $P = Céer\ population\ initiale\ et\ évaluer$
3: $R = Population\ auxiliaire$
4: $H = Initialiser\ l'archive$
5: $tant\ que\ condition\ d'arrêt\ n'est\ pas\ satisfaite$
6: $Pour\ chaque\ individu\ i\ jusqu'à\ N$
7: $l = acquérir\_voisiage(position(individu))$
8: $Parents = sélection(l)$
9: $Enfant = Recombination, mutation\ (Parents)$
10: $Evaluer(Enfant)$
11: $R = remplacer\ (Enfant)$
12: $H = Insertion\ (Enfants)$
13: $P = R$
14: $P <- Feedback(H)$
15: $Fin\ tant\ que$

Pseudocode 8 - MOCell

Explication : D'abord, une population initiale sous forme de grille toroïdale bidimensionnelle est créée. Un archive vide est initialisé (même structure que la population), représentant le front de Pareto. Ensuite, la boucle principale est lancée : tant que le critère d'arrêt (nombre de générations, évaluations … etc.) n'est pas satisfait, les opérateurs : sélection, recombinaison, mutation, sont appliqués sur les voisinages de chaque individu de la population P, donc pour chaque individu i de la population P, un enfant est créé, puis évalué. Si ce dernier représente une solution non-dominée il remplacera l'individu i ou un autre individu du voisinage de i de la population auxiliaire R puis il sera inséré dans l'archive H, si l'enfant et l'individu à remplacer sont tous les deux non-dominés, l'opérateur de crowding (NSGA II) est appliqué. Enfin, la population auxiliaire R remplace la population courante P, puis un nombre prédéfini de meilleures solutions de l'archive sont sélectionnées pour remplacer le même nombre de solutions aléatoirement choisies de la population P en utilisant l'opérateur de crowding. Cette opération s'appelle *Feedback* (ligne 14), elle améliore la diversité.

III.6- Conclusion

Dans ce chapitre, quatre algorithmes multi-objectifs sont décrits :

- Le NSGA II avec sa technique de crowding et de tri rapide qui permettent une exécution rapide, élitiste et efficace.

- Le SPEA II ayant une technique de clustring plus avancée que la première version, il représente une méthode très performante dans l'exploitation des solutions.

- Le FastPGA est sa technique de régulation de la population assure un élitisme relativement supérieur à ses prédécesseurs.

- Le MOCell dont la particularité est sa structure de population qui offre un meilleur environnement pour l'exploitation et l'exploration des solutions.

Ces quatre algorithmes seront mis en examen à l'aide de quelques fonctions de test dans le chapitre suivant.

Chapitre IV

Etude Comparative des AGMOs

IV.1 - Introduction

Ce chapitre présente une étude comparative entre les quatre algorithmes génétiques étudiés dans le chapitre précédant, et ce, à l'aide des fonctions de test (benchmark). La comparaison entre les méthodes étudiées se fera selon certains critères que nous allons détailler dans ce chapitre.

IV.2 – Les fonctions benchmark (test) utilisées

Dans notre étude, on utilisera deux types de fonctions de test, des fonctions avec contraintes et des fonctions sans contraintes. Pour le premier type, on utilisera trois fonctions : Srinivas, Tanaka et Osyczka2 (2objectifs) tandis que pour les fonctions sans contraintes, on a choisi deux familles de problèmes : ZDT(1-6) (2 objectifs) et DTLZ(1-7) (3 objectifs).

IV.2.1. Les fonctions avec contraintes

L'expression générale des problèmes d'optimisation est sous la forme suivante :

$$
\begin{aligned}
&Trouver\ l'optimum\ x\ pour : \\
&\min_{x\in\mathbb{R}^n} f_i(x),\ \ i = 1,2,\dots p \\
&Tel\ que \\
&Contraintes\ d'inégalité :\ g_k(x) \geq 0,\ \ k = 1,\dots.K \\
&Contraintes\ d'égalité :\ \ h_m(x) = 0,\ \ m = 1,\dots.M
\end{aligned}
\qquad \text{(IV-1)}
$$

Avec $x \in \mathbb{R}^n$ le vecteur de décision $x = [x_1, \dots. x_n]^T$, où chaque $x_j,\ j = 1, \dots n$ est borné par deux limites $Inf_i \leq x_j \leq Sup_i$. L'ensemble de Pareto est composé par des solutions admissibles, c'est-à-dire les solutions qui satisfont les contraintes.

- **Problème de Srinivas :** Cette fonction de test a été proposée par Srinivas et Deb en 1994 [30] dont voici l'énoncé :

$$
\begin{aligned}
&Minimiser \\
&\quad f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \\
&\quad f_2(x) = 9x_1 - (x_2 - 1)^2 \\
&tel\ que \\
&\quad g_1(x) \equiv x_1^2 + x_2^2 \leq 225 \\
&\quad g_2(x) \equiv x_1 - 3x_2 \leq -10 \\
&avec \\
&\quad -20 \leq x_1, x_2 \leq 20
\end{aligned}
\qquad \text{(IV-2)}
$$

Le front de Pareto réel de cette fonction est donné par la figure ci-dessous :

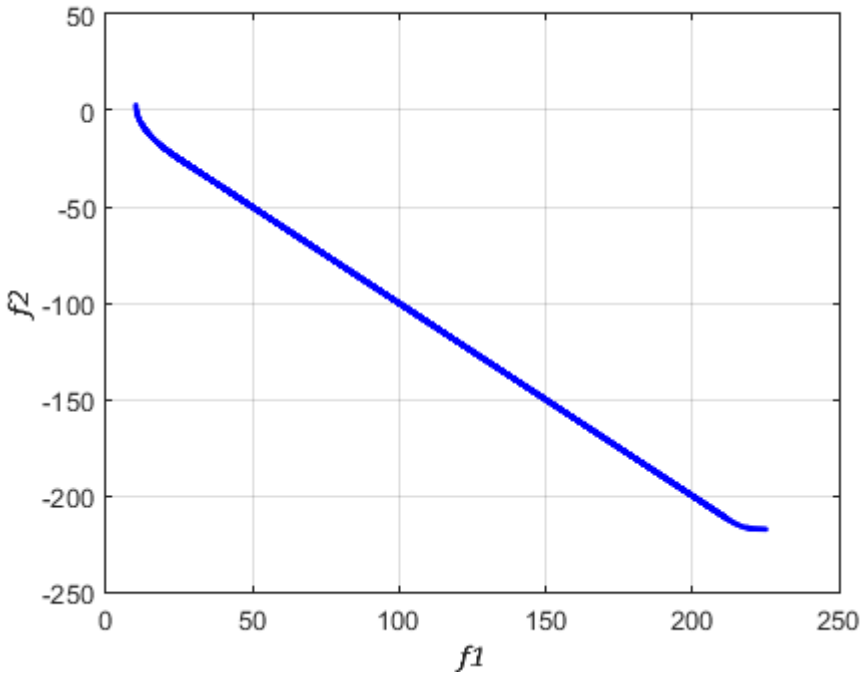

Figure IV-1- Front de Pareto réel (Srinivas)

Ce problème est plutôt simple, ne contient pas beaucoup de contraintes, son front de Pareto est uni-modal, cette fonction de test évalue la qualité d'un AG en terme de distribution de solutions sur le front de Pareto pour un problème simple avec contraintes.

- **Problème de Tanaka :** Proposé par Tanaka en 1995 [31] :

Ce problème est exprimé comme suit :

$$
\begin{aligned}
&Minimiser && \\
&\quad f_1(x) = x_1 && \\
&\quad f_2(x) = x_2 && \\
&tel\ que && \\
&\quad g_1(x) \equiv x_1^2 + x_2^2 - 1 - 0.1 * \cos(16 * \arctan\left(\frac{x_1}{x_2}\right)) \leq 0 && \\
&\quad g_2(x) \equiv (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 && \\
&avec && \\
&\quad -\pi \leq x_1, x_2 \leq \pi &&
\end{aligned}
\qquad \text{(IV-3)}
$$

Le front de Pareto réel (discontinu) de cette fonction est donné par la figure ci-dessous :

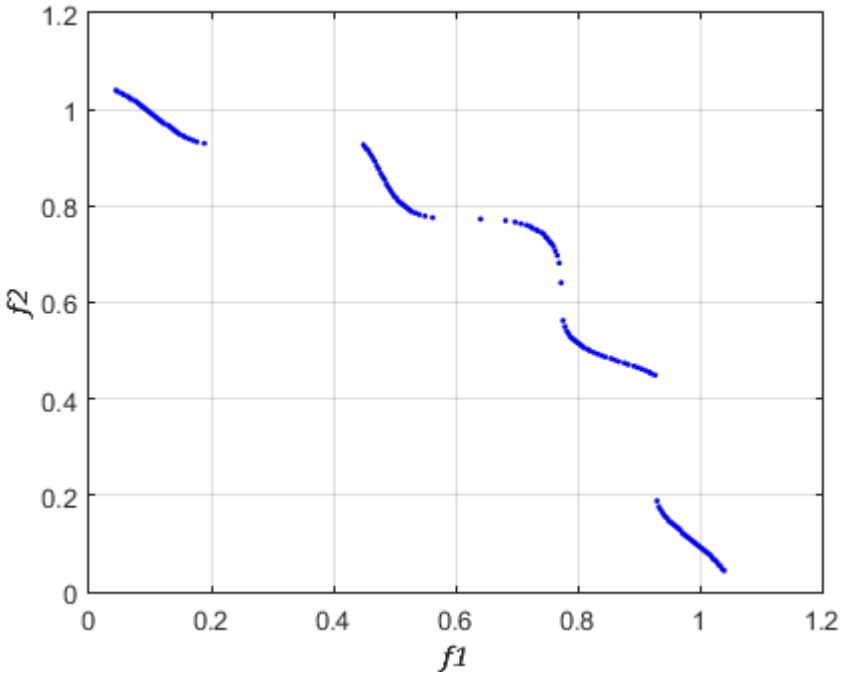

Figure IV-2 - Front de Pareto réel (Tanaka**)**

Ce problème est plus compliqué que le précédent puisque il représente un front de Pareto déconnecté.

- **Problème d'Osyczka :** Proposé par Osyczka et Kundu en 1995 [26] :

Ce problème s'agit de minimiser :

$$f_1(x) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2(x_4 - 4)^2 + (x_5 - 1)^2]$$
$$f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \qquad \text{(IV-4)}$$

$tel\ que$

$$g_1(x) \equiv x_1 + x_2 - 2 \geq 0$$
$$g_2(x) \equiv 6 - x_1 - x_2 \geq 0$$
$$g_3(x) \equiv 2 - x_2 + x_1 \geq 0$$
$$g_4(x) \equiv 2 - x_1 + 3x_2 \geq 0$$
$$g_5(x) \equiv 4 - (x_3 - 3)^2 - x_4 \geq 0$$
$$g_6(x) \equiv (x_5 + 3) + x_6 - 4 \geq 0$$

$avec$

$$0 \leq x_1, x_2, x_6 \leq 10\,; \qquad 1 \leq x_3, x_5 \leq 5\,; \qquad 0 \leq x_4 \leq 6\,;$$

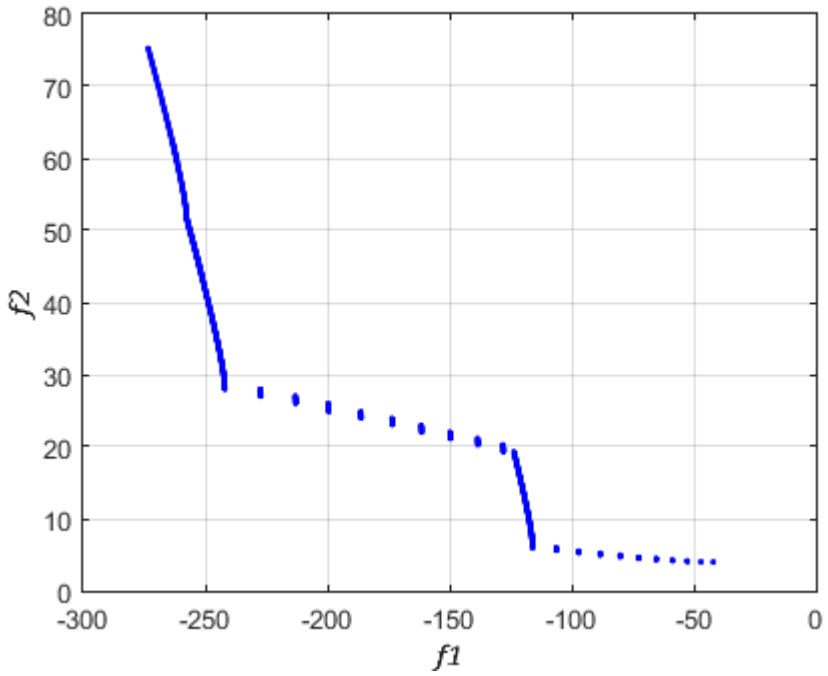

Figure IV-3 - Front de Pareto réel (Osyczka)

C'est le problème avec contraintes le plus compliqué des trois proposés, avec un front de Pareto déconnecté est un espace de 6 contraintes.

IV.2.2. Les fonctions sans contraintes :

Ce type de fonctions ne comporte pas de contraintes sur l'espace de recherche, les ZDT, DTLZ seront nos fonctions de test sans contraintes dans notre étude.

- **Les fonctions ZDT :** Cette famille de fonctions a été proposée par Zitzler, Deb et Thiele en 2000 [38], elle comporte 6 fonctions (on omettra la 5ème car elle ne fonctionne qu'avec le codage binaire), les ZDT font partie des plus populaires fonctions de test dans le domaine d'optimisation.

Tous les problèmes ZDT sont structurés de cette façon :

$$\begin{aligned} &minimiser\ \mathcal{T}(x) = \big(f_1(x_1), f_2(x)\big) \\ &tel\ que \quad f_2(x) = g(x_2, \dots, x_n) * h\big(f_1(x_1), g(x_2, \dots, x_n)\big) \\ &avec \quad x = (x_1, \dots\dots, x_n) \end{aligned} \tag{IV-5}$$

f_1 est la fonction de la première variable de décision et g est la fonction des $n-1$ variables restantes, les paramètres de h sont les valeurs des fonctions f_1 et g.

- **ZDT 1 :** Elle a un front de Pareto convexe :

$$\begin{aligned} &f_1(x_1) = x_1, \\ &g(x_2, \dots, x_n) = 1 + \frac{9}{n-1} * \sum_{i=2}^{n} x_i \\ &h(f_1, g) = 1 - \sqrt{f_1/g} \end{aligned} \tag{IV-6}$$

avec $n = 30$ et $x_i \in [0,1]$. Le front de Pareto réel est obtenu avec $g(x_2 \dots x_n) = 1$

- **ZDT 2 :** Cette fonction a un front non-convexe, elle est identique à ZDT1 sauf pour h :

$$h(f_1, g) = 1 - (f_1/g)^2 \tag{IV-7}$$

Les Fronts de Pareto réels des fonctions ZDT1 et ZDT2 sont illustrés dans les figures suivantes :

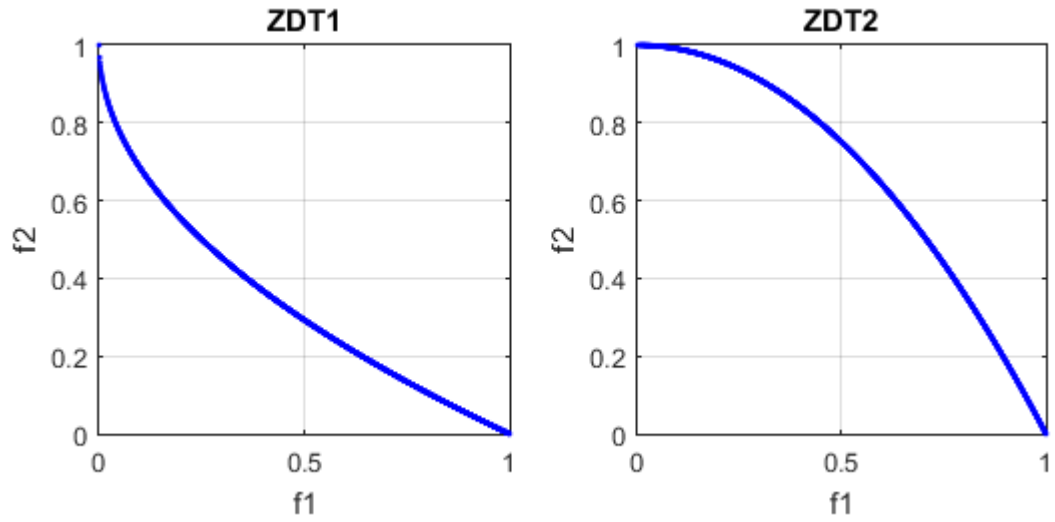

Figure IV-4 - Fronts de Pareto réels de ZDT1

- **ZDT3 :** Le Front de Pareto de cette fonction est caractérisé par sa discontinuité, il est composé de plusieurs parties discontinues convexes, elle est identique à ZDT1 sauf pour h :

$$h(f_1, g) = 1 - (f_1/g)^2 - (f_1/g) * \sin(10\pi f_1) \quad \text{(IV-8)}$$

L'introduction de la fonction sin dans h est la cause de la discontinuité du front.

- **ZDT4 :** Cette fonction comporte 21^9 fronts de Pareto optimaux locaux, elle donc met en épreuve la capacité de l'AG de faire face à la multimodalité, elle est identique à ZDT1 sauf pour g :

$$g(x_2, \dots, x_n) = 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)) \quad \text{(IV-9)}$$

Avec $n = 10$ et $x_1 \in [0,1]$ et $x_2 \dots x_n \in [-5,5]$,

Le front de Pareto réel global est obtenu avec $g(x_2, \dots, x_n) = 1$, le meilleur front de Pareto local est obtenu avec $g(x_2, \dots, x_n) = 1.25$

Les Front de Pareto réels de ZDT3 et ZDT4 sont illustrés par la figure suivante :

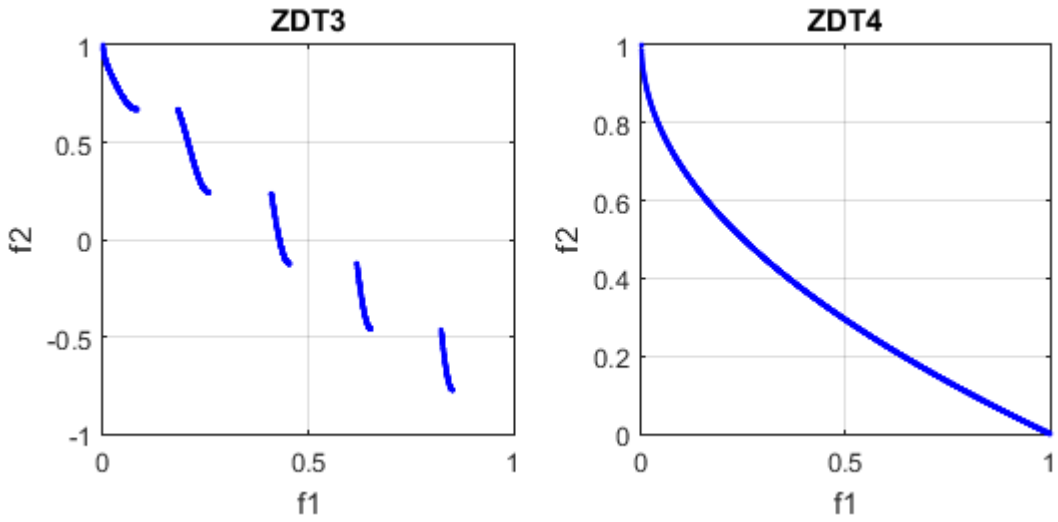

Figure IV-5 - Fronts de Pareto réels de ZDT3 et ZDT4

- **ZDT6 :** Le Front de Pareto de cette fonction est non-convexe, cette fonction de test inclut deux difficultés causées par la non-uniformité de l'espace de recherche :

- la première, les solutions optimales sont distribuées d'une manière non-uniforme le long du front de Pareto global (le front est déformé pour les solutions pour qui $f_1(x_1)$ est proche de 1.

- La deuxième difficulté est que la densité des solutions est moindre près du front de Pareto optimal, et plus importante loin du front.

$$\begin{aligned} f_1(x_1) &= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ g(x_2,\dots,x_n) &= 1 + 9 * \left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25} \\ h(f_1,g) &= 1 - (f_1/g)^2 \end{aligned} \qquad \text{(IV-10)}$$

Avec $n = 10$, $x_i \in [0,1]$, le front de Pareto optimal est formé avec $g(x_2,\dots,x_n) = 1$.

La prochaine figure représente le front de Pareto optimal réel de ZDT6.

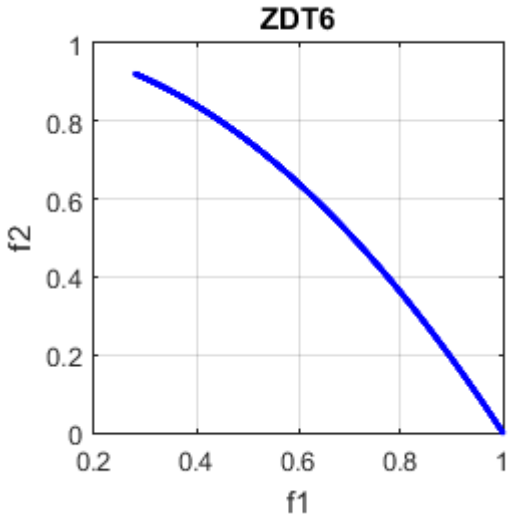

Figure IV-6 - Front de Pareto réel de ZDT6

- **Les fonctions DTLZ :** Proposées par Deb, Thiele, Laumanns et Zitzler en 2002 [12], la particularité de ces fonctions est qu'elles sont extensibles à plusieurs objectifs (2+), il existe 7 fonctions dont voici les descriptions :

- **DTLZ1 :** Le front de Pareto réel de cette fonction est linéaire, multimodale :

$$
\begin{aligned}
&minimiser\ \mathcal{T}(x) = \left(f_1(x_1), \ldots\ldots, f_M(x)\right)\ \ tel\ que \\
&f_1(x) = \left(1 + g(x_M)\right) * \frac{1}{2} \prod_{j=1}^{M-1} x_j \\
&f_{m=2:M-1}(x) = (1 + g(x_M)) * \left(\frac{1}{2} \prod_{j=1}^{M-m} x_j\right)(1 - x_{M-m+1}) \\
&f_M(x) = \left(1 + g(x_M)\right) * \frac{1}{2} * (1 - x_1) \\
&g(x_M) = 100(|x_M| + \textstyle\sum_{x_i \in x_M}(x_i - 0.5)^2 - \cos\left(20\pi(x_i - 0.5)\right)) \\
&Avec \quad x_i \in [0,1]\ \ i = 1, \ldots\ldots, n
\end{aligned}
\tag{IV-11}
$$

Le nombre d'objectifs $M = 3$ et $|x_M| = k = 5$, et le nombre total des variables est $n = M + k - 1$. Les solutions optimales de Pareto correspondent à ${x_i}^* = 0.5\ \ ({x_i}^* \in x_M)$ et les valeurs objectifs sont distribuée sur l'hyperplan $\sum_{m=1}^{M} {f_m}^* = 0.5$, l'espace de

recherche de ce problème contient $(11^k - 1)$ fronts de Pareto optimaux locaux qui peuvent attirer un AGMO.

- **DTLZ2 :** Cette fonction a un Front de Pareto sphérique :

$$minimiser\ \mathcal{T}(x) = \left(f_1(x_1), \ldots\ldots, f_M(x)\right)\ \ tel\ que \qquad \text{(IV-12)}$$

$$f_1(x) = \left(1 + g(x_M)\right) * \prod_{j=1}^{M-1} \cos\left(\frac{x_j \pi}{2}\right)$$

$$f_{m=2:M-1}(x) = (1 + g(x_M)) * \left(\prod_{i=j}^{M-m} \cos\left(\frac{x_j \pi}{2}\right)\right) \sin(x_{M-m+1} * \pi/2)$$

$$f_M(x) = \left(1 + g(x_M)\right) * sin\left(\frac{x_j \pi}{2}\right)$$

$$g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$$

$$Avec \quad x_i \in [0,1]\ \ i = 1, \ldots\ldots, n$$

Le nombre d'objectifs $M = 3$ et $|x_M| = k = 10$, et le nombre total de variables est $n = M + k - 1$. Les solutions optimales de Pareto correspondent à $x_i^* = 0.5\ \ (x_i^* \in x_M)$ et toutes les valeurs objectifs doivent satisfaire $\sum_{m=1}^{M} (f_m^*)^2 = 1$, cette fonction peut être utilisé afin de tester les performances d'un AG pour un grand nombre d'objectifs.

Les fronts de Pareto optimaux réels des fonctions DTLZ1 et DTLZ2 sont représentés dans la prochaine figure :

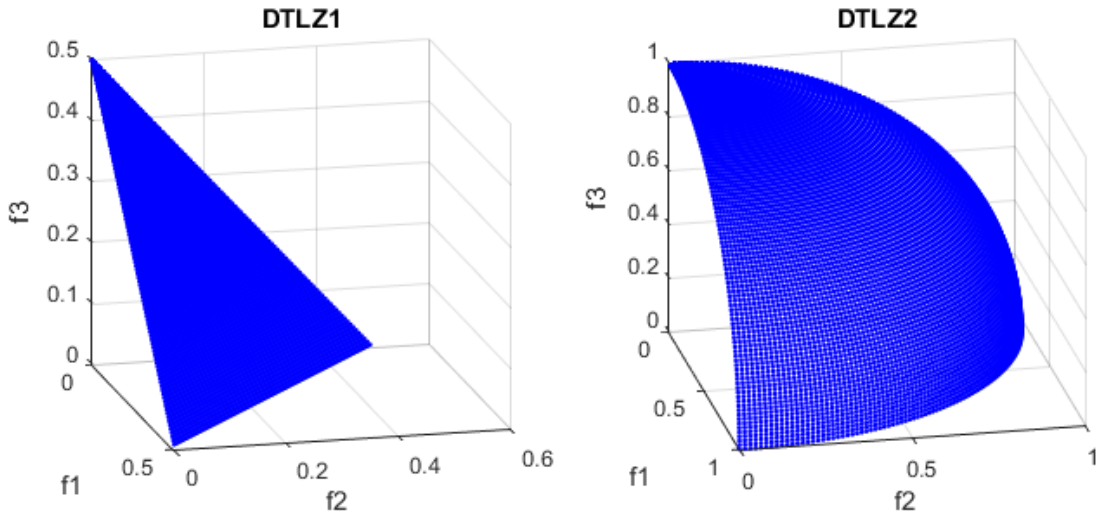

Figure IV-7 - Fronts de Pareto réels de DTLZ1, DTLZ2

- **DTLZ3 :** Elle est identique à ZDT2 sauf pour $g(x_M)$ qui est celui de ZDT1

- **DTLZ4 :** Identique à DTLZ2 sauf pour les x_j qui sont remplacés par ${x_j}^{\alpha}$ avec $\alpha > 0$ $(\alpha = 100)$ en général.

Les fronts de Pareto optimaux réels des fonctions DTLZ1 et DTLZ1 sont représentés dans la figure suivante :

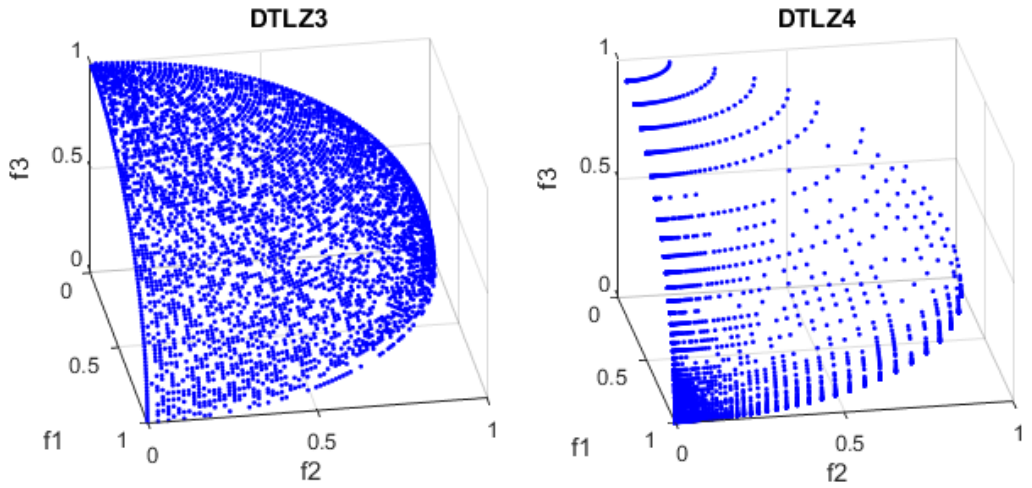

Figure IV-8 - Fronts de Pareto réels de DTLZ3 et DTLZ 4

- **DTLZ5 :** Son Front de Pareto est uni-modal, elle est identique à DTLZ2 sauf pour les x_j qui sont remplacés par :

$$\theta_j = \frac{\pi}{4(1+g(x_M))}(1+2g(x_M)x_j) \qquad \text{(IV-13)}$$

Ce problème teste la capacité d'un AGMO de converger vers une courbe dégénérée.

- **DTLZ6 :** Identique à DTLZ5 sauf pour $g(x_M)$ qui est remplacé par :

$$g(x_M) = \sum_{x_i \in x_M} (x_i)^{0.1} \qquad \text{(IV-14)}$$

Avec ce changement, le problème devient plus difficile pour l'AGMO, les solutions optimales de Pareto correspondent à $x_i^* = 0 \ \ (x_i^* \in x_M)$ c.-à-d. $x_i = 0$ pour tout $x_i \in x_M$.

Voici les fronts de Pareto réels de DTLZ5 et DTLZ6 :

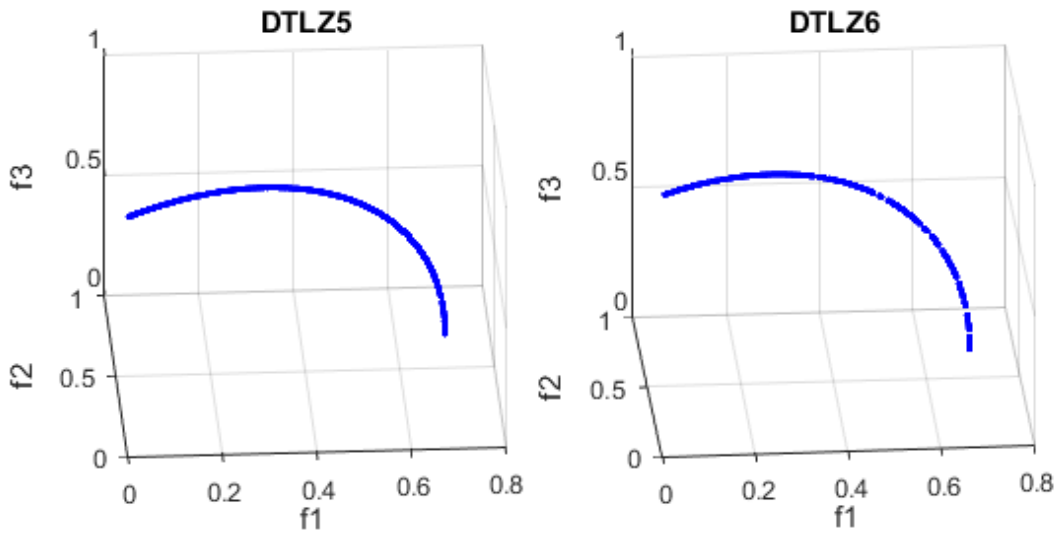

Figure IV-9 - Fronts de Pareto réels de DTLZ5, DTLZ6

- **DTLZ7 :** Son front de Pareto déconnecté :

$$
\begin{aligned}
&minimiser\ \mathcal{T}(x) = \left(f_1(x_1), \ldots\ldots, f_M(x)\right)\ \ tel\ que \\
&f_{m=1:M-1}(x) = x_m \\
&f_M(x) = \left(1 + g(x_M)\right) * h(f_1, \ldots, f_{M-1}, g(x_M)) \\
&g(x_M) = 1 + \frac{9}{|x_M|}\sum_{x_i \in x_M} x_i \\
&h(f_1, \ldots, f_{M-1}, g(x_M)) = M - \sum_{j=1}^{M-1}\left(\frac{f_i(x)}{1+g(x_M)}(1 + sin3\pi * f_i(x))\right) \\
&Avec \quad x_i \in [0,1]\ \ i = 1, \ldots\ldots, n
\end{aligned}
\tag{IV-15}
$$

Cette fonction de test possède $2 * (M - 1)$ régions optimales de Pareto dans l'espace de recherche, avec $k = 20$, les solutions optimales de Pareto correspondent à ${x_i}^* = 0\ \ ({x_i}^* \in x_M)$, ce problème met en épreuve la capacité de l'AGMO de maintenir les individus dans des différentes régions optimales de Pareto, cette fonction peut être rendue plus difficile en utilisant une fonction *sinus* d'une fréquence plus importante ou en utilisant une fonction g multimodale.

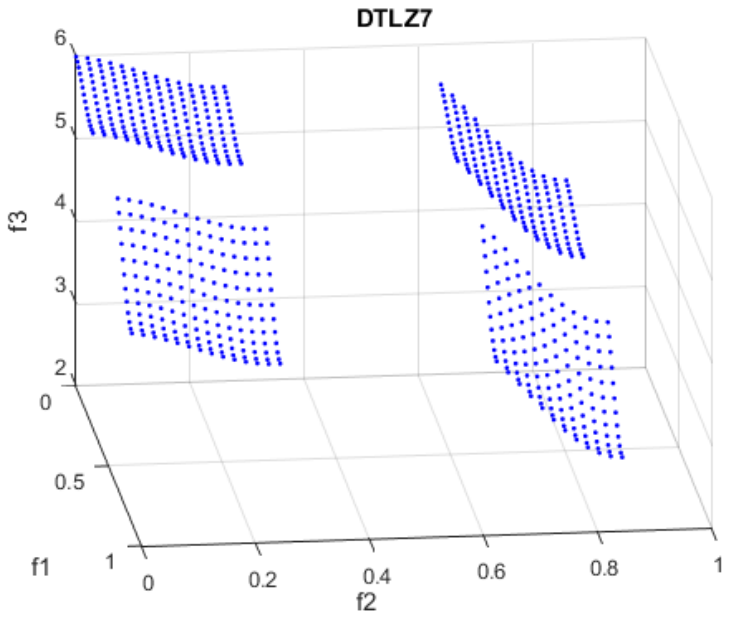

Figure IV-10 - Front de Pareto réel de DTLZ7

IV.3 - Les critères de comparaison

Les critères de comparaison dans notre étude sont les indicateurs de qualité des algorithmes utilisés. L'optimisation multi-objectif opte pour deux buts, la convergence vers le vrai ensemble optimal de Pareto et le maintien de diversité dans l'ensemble des solutions optimales de Pareto. Afin de mesurer la performance d'un algorithme, des méthodes de mesure de performances sont utiliséees, nous allons utiliser trois indicateurs de qualité : "Hypervolume", "Spread" et "Generational Distance" dont voici de brèves descriptions :

IV.3.1. Hypervolume (HV)

Utilisé pour la première fois par E. Zitzler et L.Thiele (1999) [37], c'est l'indicateur le plus utilisé dans le domaine car il prend en compte la convergence du front étudié vers le front réel ainsi que la distribution des solutions sur l'espace objectif, aussi le HV est le seul indicateur qui reflète la dominance faible dans le résultat donné. Dans un cas de comparaison, le meilleur résultat correspond à la valeur la plus grande de HV. À cause de la complexité de cet indicateur, nous nous contenteront de cette brève description.

IV.3.2. Spread

On utilisera le même que Deb et al. (2002 [11]) ont utilisé, cet indicateur mesure la distribution des solutions le long de la région non-dominée et il est exprimé comme suit :

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \qquad \text{(IV-16)}$$

Avec d_f et d_l les distances euclidiennes entre les solutions extrêmes obtenues et les solutions extrêmes réelles, le paramètre $\bar{d}$ est la moyenne de toutes les distances $d_i,\ i = 1, \ldots, (N-1)$ avec N le nombre de solutions de l'ensemble optimal de Pareto obtenu donc $(N-1)$ distances consécutives, une parfaite distribution des solutions implique que toutes les distances $d_i = \bar{d}$, ce qui fera que $d_f = d_l = 0$ et donc $\Delta = 0$, le meilleur résultat correspond à la plus petite valeur de Δ.

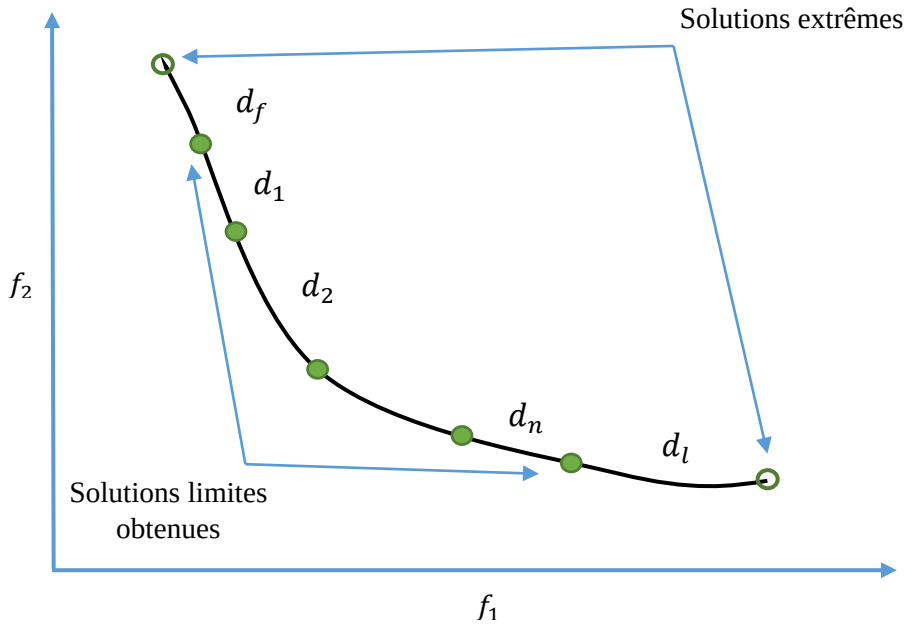

Figure IV-11 – Indicateur de diversité : Spread

IV.3.3. Generational Distance (GD)

Le concept de "Generational Distance" a été introduit par Veldhuizen et Lamont (1998) [34] en tant que façon d'estimer la distance entre l'ensemble optimal de Pareto obtenu et l'ensemble optimal réel global de Pareto, il est défini comme suit :

$$GD = \frac{(\sum_{i=1}^{n} {d_i}^p)^{1/p}}{|FP_{Obtenu}|} \qquad \text{(IV-17)}$$

Avec $|FP_{Obtenu}|$ le nombre de vecteurs dans FP_{Obtenu} (le front de Pareto obtenu), $p = 2$ et d_i la distance euclidienne entre chaque individu i dans l'espace objectif (distance phénotypique) et l'individu du Front optimal de Pareto global réel qui lui est le plus proche, quand $GD = 0$, cela implique que $d_i = 0$ donc $FP_{Obtenu} = FP_{Réel}$, le meilleur résultat correspond à la plus petite valeur de GD.

IV.4 – Méthodes de statistiques

Afin d'obtenir une interprétation concrète des résultats, nous avons utilisé deux méthodes de mesure statistiques, le diagramme de Box-Whiskers (Tukey 1977 [32]) ou le **Boxplot** et le Wilcoxon rank-sum-test (Mann et Whitney 1947 [24]) et une méthode de classement qui est la méthode de test de Friedman (Friedman 1937 [15])

IV.4.1. Le Box-Whiskers (Boxplot)

Cette méthode utilise l'écart interquartile afin de présenter la distribution des données (HV, GD et Δ), voici la description brève :

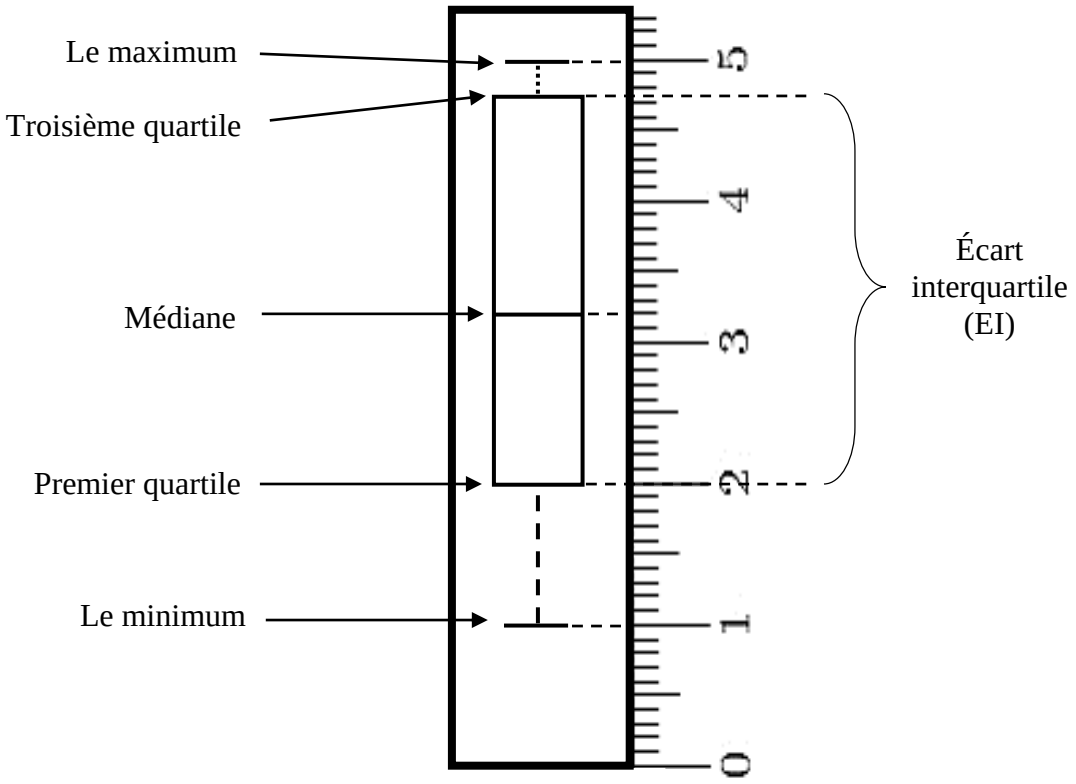

Figure IV-12 - Diagramme de Box-Whiskers

Après plusieurs exécutions indépendantes d'un AGMO, les valeurs des indicateurs de qualités de chaque exécution sont sauvegardées, par exemple, pour 50 exécutions indépendantes, pour chaque algorithme, chaque fonction de teste a un vecteur de 50 valeurs de chaque indicateur de qualité, puis à l'aide de la médiane et d'écart interquartile on pourra observer la variation des résultats le long des 50 exécutions, une variance faible correspond à une bonne crédibilité d'un AGMO.

IV.4.2. Wilcoxon rank-sum-test

Dans cette étude, nous utiliseront le Wilconxon rank-sum-test pour comparer les indicateurs de qualité entre les algorithmes, ce test se base sur l'hypotèse nulle, dans notre cas, cette dernière est que les vecteurs des indicateurs comparés aient la même médiane. Avec un test d'hypotèse de valeur 0,05, si la propabilité de l'hypotèse null est inférieure à 5%, alors cette dernière est rejetée, l'interface utilisée est un simple tableau :

	AGMO2			AGMO3			AGMO4		
	Fun_1	Fun_2	Fun_n	Fun_1	Fun_2	Fun_n	Fun_1	Fun_2	Fun_n
AGMO1	Δ	∇	∇	–	Δ	Δ	∇	–	Δ
AGMO2				∇	–	Δ	∇	∇	∇
AGMO3							∇	–	–

Tableau IV-1- Exemple de test de Wilcoxon

Dans cet exemple de quatre AGMOs et n fonctions de test, le symbole Δ signifie que l'algorithme sur la ligne a un meilleur résultat que celui sur la colonne, et ∇ veut dire le contraire et le signe "–" signifie que les deux algorithmes ont la même médiane dans l'indicateur choisi.

IV.4.3. Le test de Friedman

Cette méthode consiste à attribuer un classement à un algorithme en se basant sur plusieurs échantillons d'un même indicateur (50 pour 50 exécutions indépendantes).

Après avoir couvert tous les outils utilisés pour cette étude, nous allons, dans la partie suivante, entamer la comparaison entre les quatre AGMOs : Le NSGA II, SPEA II, FastPGA et MOCell.

IV.5 – Etude comparative

Pour tous les AGMOs, nous avons utilisé des populations et archives de 100 individus et 3000 évaluations et 50 exécutions indépendantes, une probabilité de croisement de 0.9 et une probabilité de mutation égale à $(1/le\ nombre\ de\ variables)$

Pour les trois types de fonctions de test (2 objectifs, 3 objectifs et avec contraintes) nous réservons des parties pour chacune d'elles.

IV.5.1. Etude sur les fonctions avec contraintes :

Les tableaux suivants représentent les résultats (Médiane et écart interquartile) pour chaque indicateur de qualité pour les fonctions Srinivas, Tanaka et Osyczka2 (2 objectifs avec contraintes).

Spread :

	NSGAII	SPEA2	FastPGA	MOCell
Srinivas	$4.03e-01_{4.1e-02}$	$1.75e-01_{1.7e-02}$	$3.56e-01_{3.5e-02}$	$6.81e-02_{1.6e-02}$
Tanaka	$7.93e-01_{3.7e-02}$	$7.41e-01_{4.3e-02}$	$7.73e-01_{3.6e-02}$	$7.08e-01_{2.6e-02}$
Osyczka2	$5.80e-01_{1.1e-01}$	$7.28e-01_{1.7e-01}$	$6.08e-01_{1.8e-01}$	$5.29e-01_{2.6e-01}$

Tableau IV-2 - Indicateur "Spread" pour les fonctions avec contraintes

Hypervolume :

	NSGAII	SPEA2	FastPGA	MOCell
Srinivas	$5.38e-01_{4.6e-04}$	$5.40e-01_{1.8e-04}$	$5.38e-01_{2.8e-04}$	$5.41e-01_{6.9e-05}$
Tanaka	$3.08e-01_{3.5e-04}$	$3.09e-01_{4.5e-04}$	$3.08e-01_{2.9e-04}$	$3.09e-01_{2.8e-04}$
Osyczka2	$7.45e-01_{7.6e-03}$	$7.34e-01_{3.6e-02}$	$7.45e-01_{2.8e-02}$	$7.36e-01_{2.5e-01}$

Tableau IV-3 - Indicateur "HV" pour les fonctions avec contraintes

Generational distance:

	NSGAII	SPEA2	FastPGA	MOCell
Srinivas	$1.88e-04_{3.7e-05}$	$1.20e-04_{2.5e-05}$	$1.93e-04_{4.2e-05}$	$4.44e-05_{1.7e-05}$
Tanaka	$7.78e-04_{1.2e-04}$	$6.46e-04_{1.3e-04}$	$7.60e-04_{9.0e-05}$	$7.13e-04_{1.2e-04}$
Osyczka2	$1.03e-03_{8.8e-05}$	$1.50e-03_{2.3e-04}$	$1.09e-03_{3.6e-04}$	$1.25e-03_{1.5e-02}$

Tableau IV-4 - Indicateur "GD" pour les fonctions avec contraintes

Les valeurs en gris foncé et en gris clair sont respectivement les meilleures valeurs et les deuxièmes meilleures valeurs.

Pour le test de Wilcoxon, on a choisi l'indicateur hypervolume :

	SPEAII			FastPGA			MOCell		
	SRVN	*TNK*	*OSZK2*	*SRVN*	*TNK*	*OSZK2*	*SRVN*	*TNK*	*OSZK2*
NSGAII	∇	∇	Δ	∇	∇	Δ	∇	∇	Δ
SPEAII				Δ	Δ	–	∇	∇	–
FastPGA							∇	∇	–

Tableau IV-5 - Test de Wilcoxon Pour les fonctions avec contraintes

Le schéma suivant est la représentation Boxplot des résultats de l'indicateur Spread :

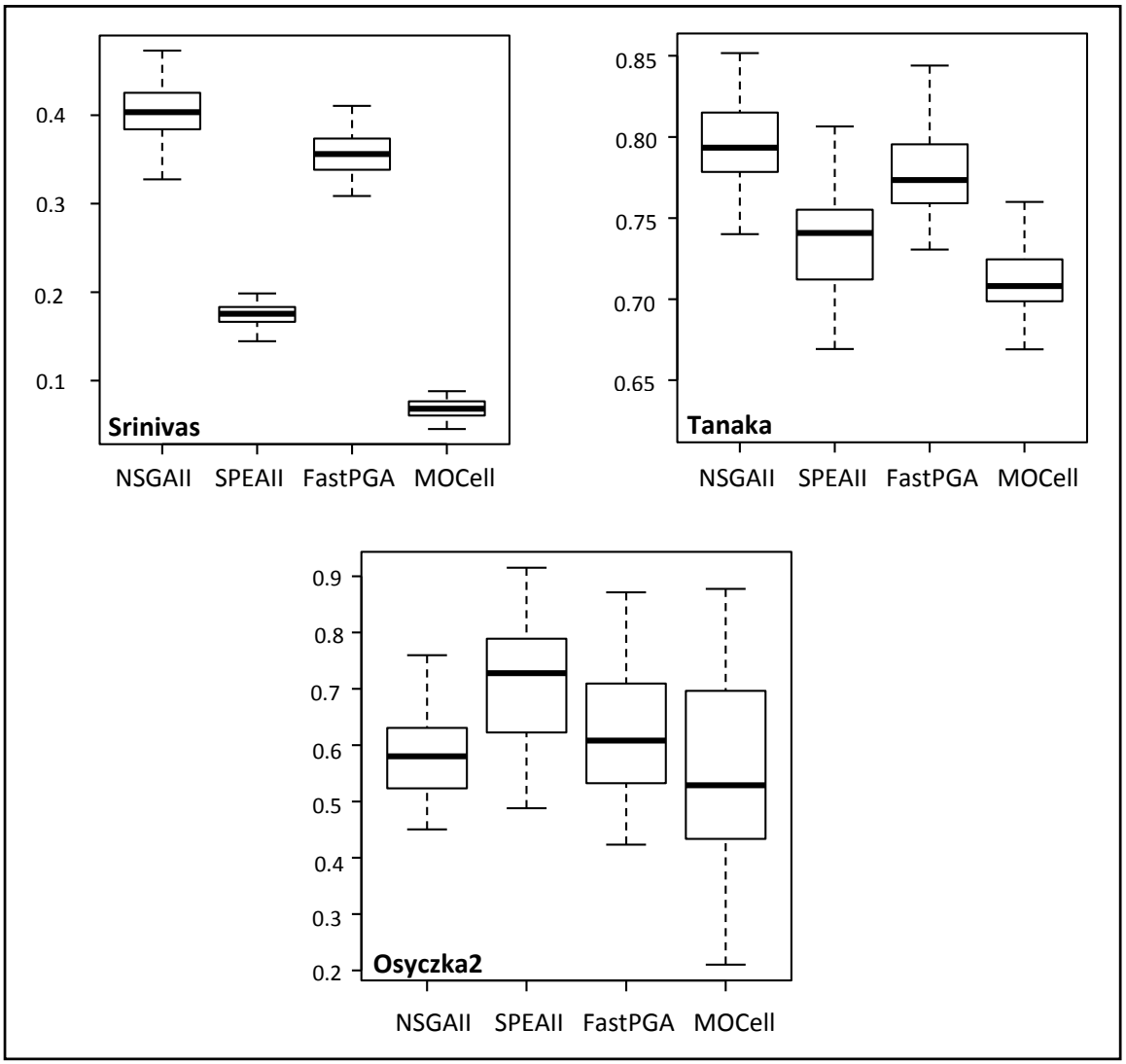

Figure IV-13 - Boxplot de l'indicateur Spread pour les fonctions avec contraintes

D'après les résultats obtenus, le MOCell est clairement le plus performant concernant la distribution des solutions sur l'espace objectif et pour le maintien de la diversité selon le test de Wilcoxon de HV. Suivi du SPEA II. Le NSGAII a donné les meilleurs résultats pour la fonction Osyczka2, le Boxplot nous montre que pour la fonction Srinivas, la variation des résultats le long des 50 exécutions est petite tandis que pour les autres fonctions, la variation est relativement grande, en particulier pour le MOCell avec la fonction Osysczka2, le FastPGA est sans doute le moins performant des quare AGMO concernant les fonctions étudiées.

On a choisis les fronts de Pareto suivants :

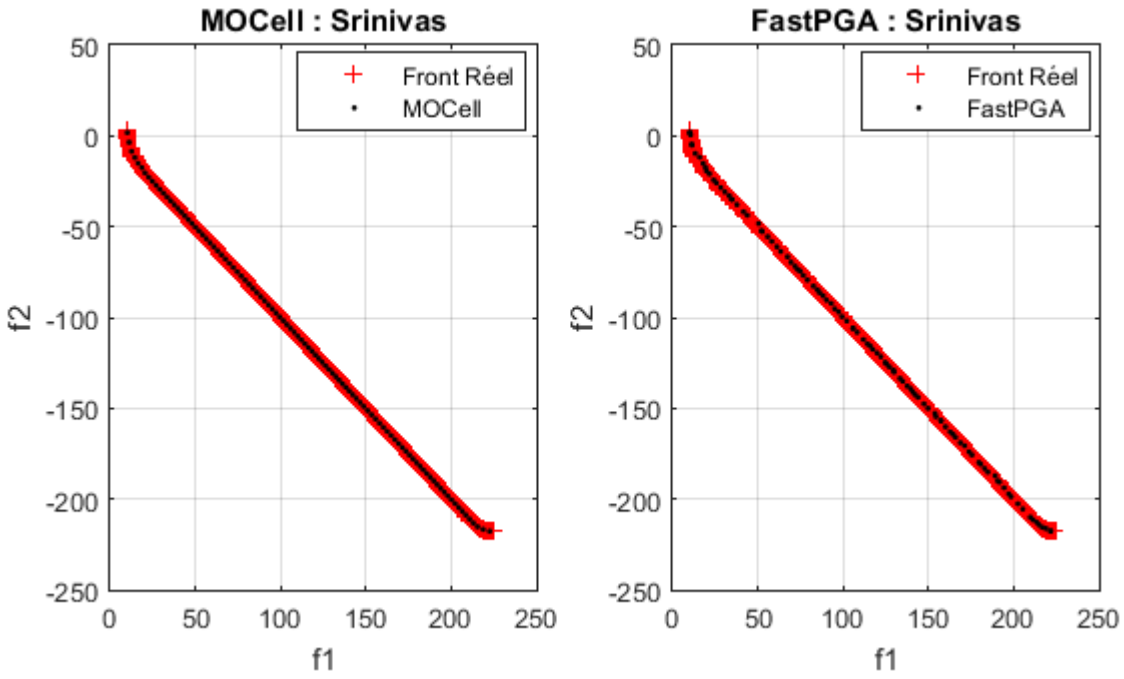

Figure IV-14 - Fronts de Pareto du MOCell et FastPGA : Srinivas

Pour le MOCell, la distribution des solutions sur le front est uniforme, cette figure confirme les résultats obtenus antérieurement elle montre aussi que le MOCell a une convergence optimale vers le font de Pareto optimal réel global confirmant les résultats du tableau IV-4 (indicateur GD).

Comme les résultats l'ont indiqué, le FastPGA montre une pauvre uniformité de distribution des solutions et une diversité limitée pour cette fonction.

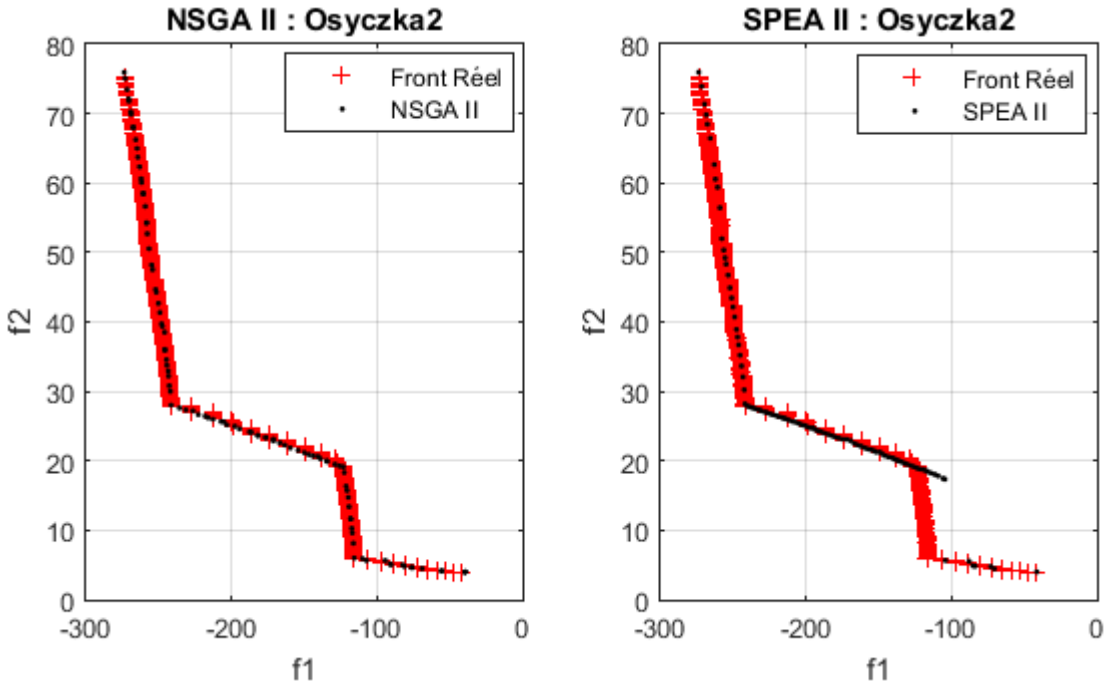

Figure IV-15 - Fronts de Pareto du NSGAII et SPEA II : Osyczka2

Le NSGA dépasse les autres AGMOs pour la fonction Oscyzka2 en matière de convergence vers le front réel, la distribution des solutions sur l'espace objectif ainsi que le maintien de diversité (tableaux IV-2, IV-3 et IV-4).

Les fonctions suivantes ont le même nombre d'objectifs mais sans contraintes, les performances des AGMOs étudiés seront examinées dans la partie suivante.

IV.5.2. Etude sur les fonctions sans contraintes ZDT (2 Objectifs)

Les résultats sont présentés dans les tableaux suivants :

Spread :

	NSGAII	SPEA2	FastPGA	MOCell
ZDT1	$5.98e-01_{5.4e-02}$	$1.43e-01_{1.6e-02}$	$6.35e-01_{7.8e-02}$	$6.77e-02_{2.0e-02}$
ZDT2	$3.77e-01_{3.1e-02}$	$1.36e-01_{1.9e-02}$	$3.53e-01_{4.0e-02}$	$7.71e-02_{1.3e-02}$
ZDT3	$7.44e-01_{1.7e-02}$	$7.07e-01_{4.7e-03}$	$7.35e-01_{1.7e-02}$	$7.05e-01_{6.0e-03}$
ZDT4	$4.06e-01_{4.2e-02}$	$1.74e-01_{8.0e-02}$	$3.64e-01_{4.8e-02}$	$1.07e-01_{3.3e-02}$
ZDT6	$3.82e-01_{4.0e-02}$	$1.76e-01_{1.9e-02}$	$3.67e-01_{5.2e-02}$	$8.16e-02_{1.2e-02}$

Tableau IV-6 - Indicateur "Spread" pour les fonctions sans contraintes (ZDT)

Hypervolume :

	NSGAII	SPEA2	FastPGA	MOCell
ZDT1	$6.60e-01_{6.0e-04}$	$6.62e-01_{6.1e-05}$	$6.59e-01_{5.6e-04}$	$6.62e-01_{2.3e-05}$
ZDT2	$3.27e-01_{4.2e-04}$	$3.28e-01_{3.7e-04}$	$3.27e-01_{2.2e-04}$	$3.29e-01_{2.1e-04}$
ZDT3	$5.15e-01_{1.3e-04}$	$5.15e-01_{1.4e-04}$	$5.15e-01_{1.1e-04}$	$5.15e-01_{2.6e-04}$
ZDT4	$6.57e-01_{3.1e-03}$	$6.57e-01_{2.8e-03}$	$6.57e-01_{3.3e-03}$	$6.59e-01_{2.6e-03}$
ZDT6	$3.95e-01_{1.1e-03}$	$3.90e-01_{2.3e-03}$	$3.99e-01_{5.0e-04}$	$3.97e-01_{7.6e-04}$

Tableau IV-7 -Indicateur "Hypervolume" pour les fonctions sans contraintes (ZDT)

Generational Distance :

	NSGAII	SPEA2	FastPGA	MOCell
ZDT1	$1.01e-04_{7.3e-05}$	$1.62e-04_{1.2e-05}$	$1.16e-04_{4.8e-05}$	$1.15e-04_{3.3e-05}$
ZDT2	$1.44e-04_{6.5e-05}$	$1.02e-04_{2.4e-05}$	$1.21e-04_{4.1e-05}$	$4.83e-05_{9.4e-06}$
ZDT3	$1.21e-04_{1.3e-05}$	$1.15e-04_{1.3e-05}$	$1.18e-04_{1.9e-05}$	$1.00e-04_{1.5e-05}$
ZDT4	$2.92e-04_{2.0e-04}$	$3.65e-04_{2.3e-04}$	$2.64e-04_{2.2e-04}$	$2.54e-04_{2.0e-04}$
ZDT6	$4.15e-04_{7.3e-05}$	$8.13e-04_{1.6e-04}$	$1.30e-04_{3.0e-05}$	$2.97e-04_{5.4e-05}$

Tableau IV-8 - Indicateur "GD" pour les fonctions sans contraintes (ZDT)

Le test Wilcoxon indique les résultats de l'indicateur Spread (Distribution des solutions) et le test de Friedmann donne un classement à chaque AGMO selon chaque indicateur.

Voici les résultats du test de Wilcoxon pour l'indicateur Spread :

	SPEAII					FastPGA					MOCell				
	Z1	*Z2*	Z3	Z4	*Z6*	*Z1*	*Z2*	Z3	Z4	*Z6*	*Z1*	*Z2*	Z3	Z4	*Z6*
NSGAII	∇	∇	∇	∇	∇	Δ	∇	∇	∇	–	∇	∇	∇	∇	∇
SPEAII						Δ	Δ	Δ	Δ	Δ	∇	∇	∇	∇	∇
FastPGA											∇	∇	∇	∇	∇

Tableau IV-9 - Test de Wilcoxon Pour les fonctions sans contraintes ZDT

Les résultats du test de Friedmann :

Spread :

AGMO	**Classement de Friedmann**
NSGAII	3.8
SPEAII	2.0
FastPGA	3.2
MOCell	1.0

Tableau IV-10 - Test de Friedman: Spread

Hypervolume :

AGMO	**Classement de Friedman**
NSGAII	2.2
SPEAII	2.0
FastPGA	2.6
MOCell	3.2

Tableau IV-11 - Test de Friedman: HV

Generational Distance :

AGMO	Classement de Friedmann
NSGAII	2.8
SPEAII	3.2
FastPGA	2.4
MOCell	1.6

Tableau IV-12 - Test de Friedman: GD

Le MOCell a la plus uniforme distribution de solutions sur le front, cela est soutenu par le test de Wilcoxon de l'indicateur Spread qui montre que le MOCell dépasse les autres AGMOs sur toutes les fonctions. Le test de Friedmann classifie le MOCell en première position, suivi du SPEAII qui montre aussi de bons résultats puisque il dépasse en qualité de distribution et diversité le FastPGA et le NSGAII. Pour au FastPGA, il se montre performant pour les fonctions ZDT3, 4 et 6 prouvant ses performances en matière de convergence au front Pareto réel. Le NSGAII montre des performances relativement faibles par rapports aux autres AGMOs mais il en reste concurrent ayant une bonne convergence vers le front réel. On a choisi quelques fronts de Pareto de quelques fonctions :

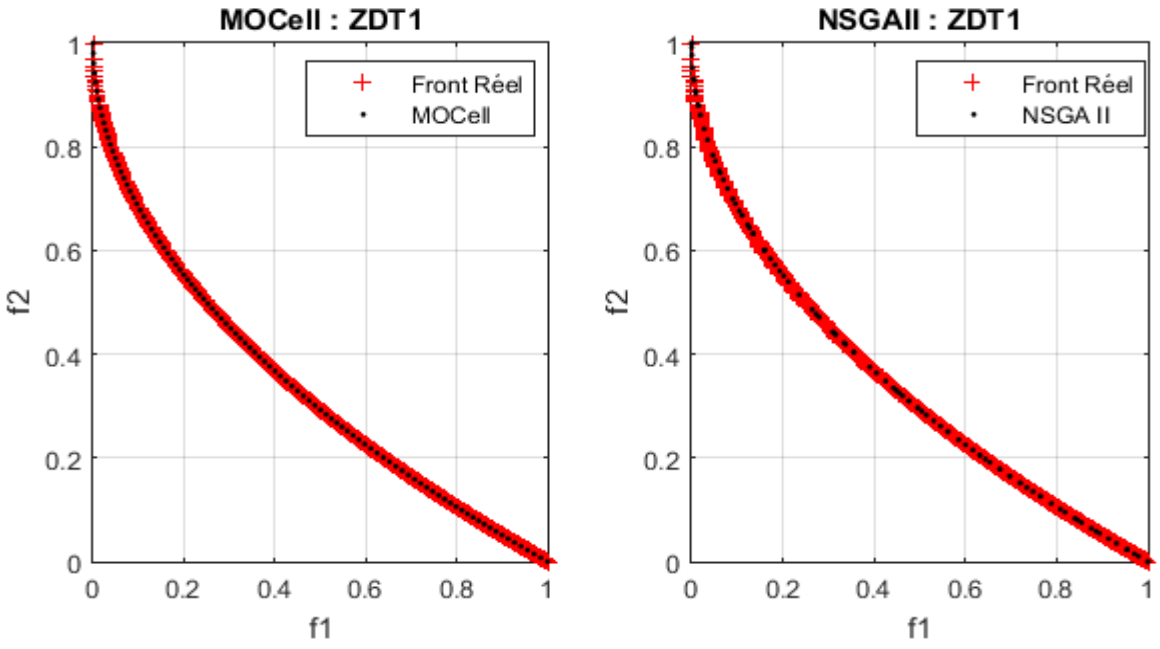

Figure IV-16 - Fronts de Pareto du MOCell et NSGAII : ZDT1

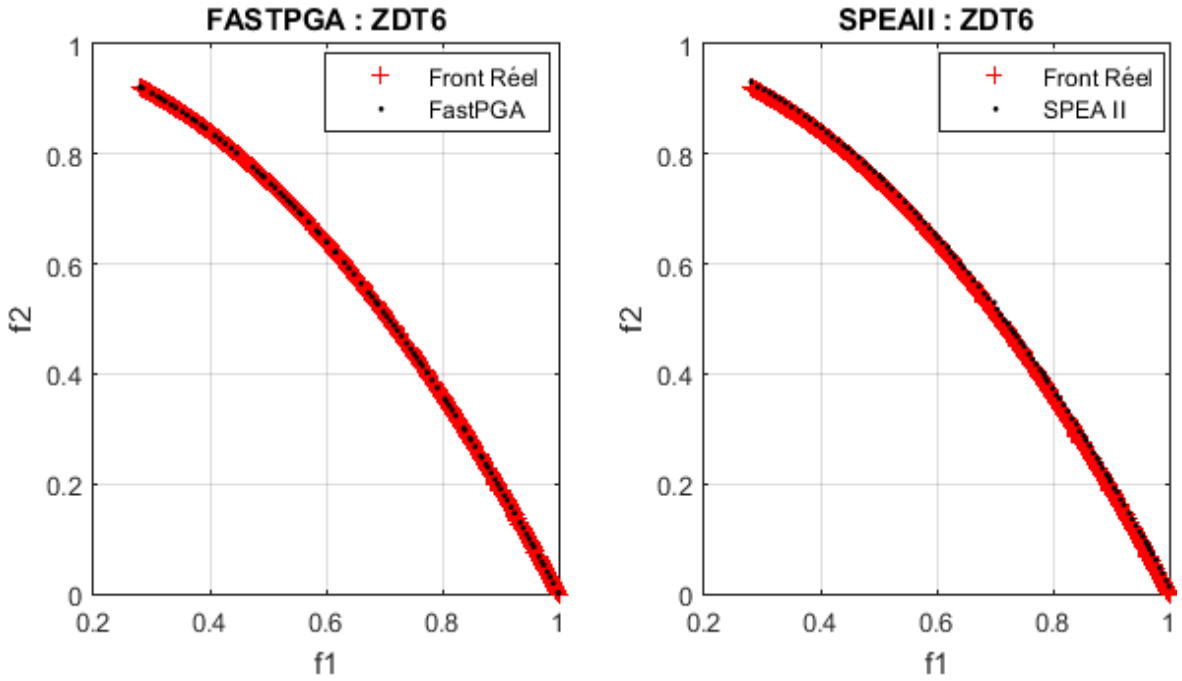

Figure IV-17- Fronts de Pareto du FastPGA et SPEAII : ZDT6

Le FastPGA a une meilleur convergence vers le front de Pareto réel que le SPEAII pour la fonction ZDT6, ce qui confirme le Test de Friedmann pour l'indicateur "Generational Distance" ainsi que le tableau IV-8 qui montre des résultats meilleurs pour la ZDT4 et 6 pour le FastPGA, on remarque que le front de SPEAII de ZDT6 est loin du front réel ce qui montre que le SPEAII a des difficultés de convergence pour les problèmes dont l'espace de recherche est non-uniforme.

Les fonctions précédentes sont des problèmes de minimisations à deux objectifs, dans l'étude suivante, un autre type de problème sera étudié, celui de 3 objectifs.

IV.5.3. Etude sur les fonctions sans contraintes DTLZ (3 objectifs)

Ce fonctions vont prmettre de determiner les performances des AGMOs face aux problèmes ayants plus de deux objectifs, voici les résultats de la simulation :

Spread :

	NSGAII	SPEA2	FastPGA	MOCell
DTLZ1	$8.45e-01_{9.1e-02}$	$7.33e-01_{3.8e-01}$	$8.63e-01_{2.1e-01}$	$1.02e+00_{3.8e-01}$
DTLZ2	$7.00e-01_{4.2e-02}$	$5.37e-01_{3.7e-02}$	$7.08e-01_{7.9e-02}$	$6.88e-01_{4.9e-02}$
DTLZ3	$7.27e-01_{5.9e-02}$	$5.72e-01_{2.1e-01}$	$7.25e-01_{9.3e-02}$	$8.45e-01_{5.2e-01}$
DTLZ4	$6.72e-01_{4.3e-02}$	$5.21e-01_{5.9e-02}$	$6.80e-01_{6.2e-02}$	$6.64e-01_{9.8e-02}$
DTLZ5	$4.39e-01_{7.8e-02}$	$2.25e-01_{3.9e-02}$	$4.01e-01_{4.4e-02}$	$1.31e-01_{6.0e-02}$
DTLZ7	$7.43e-01_{6.1e-02}$	$5.93e-01_{5.4e-02}$	$7.57e-01_{5.8e-02}$	$7.07e-01_{9.3e-02}$

Tableau IV-13 - Indicateur "Spread" pour les fonctions sans contraintes (DTLZ)

Hypervolume :

	NSGAII	SPEA2	FastPGA	MOCell
DTLZ1	$7.49e-01_{2.0e-02}$	$7.74e-01_{8.1e-03}$	$7.50e-01_{2.6e-02}$	$6.90e-01_{4.1e-01}$
DTLZ2	$3.74e-01_{9.0e-03}$	$4.04e-01_{2.4e-03}$	$3.76e-01_{5.7e-03}$	$3.76e-01_{6.1e-03}$
DTLZ3	$3.77e-01_{7.2e-03}$	$4.10e-01_{4.4e-03}$	$3.80e-01_{8.8e-03}$	$3.68e-01_{2.2e-02}$
DTLZ4	$3.76e-01_{6.2e-03}$	$3.97e-01_{1.9e-01}$	$3.78e-01_{7.0e-03}$	$3.78e-01_{1.4e-02}$
DTLZ5	$9.28e-02_{3.4e-04}$	$9.32e-02_{2.1e-04}$	$9.30e-02_{2.0e-04}$	$9.40e-02_{4.0e-05}$
DTLZ7	$2.81e-01_{5.6e-03}$	$2.94e-01_{3.0e-03}$	$2.85e-01_{5.5e-03}$	$2.71e-01_{2.8e-02}$

Tableau IV-14 - Indicateur "HV" pour les fonctions sans contraintes (DTLZ)

Generational Distance :

	NSGAII	SPEA2	FastPGA	MOCell
DTLZ1	$2.87e-03_{1.6e-02}$	$8.23e-02_{2.4e-01}$	$7.57e-03_{8.2e-02}$	$3.24e-01_{9.6e-01}$
DTLZ2	$1.41e-03_{1.9e-04}$	$1.29e-03_{2.0e-04}$	$1.18e-03_{2.4e-04}$	$1.62e-03_{6.4e-04}$
DTLZ3	$1.20e-03_{2.1e-04}$	$2.35e-02_{1.2e-01}$	$1.27e-03_{1.5e-02}$	$8.98e-02_{3.6e-01}$
DTLZ4	$5.12e-03_{3.8e-04}$	$4.75e-03_{1.4e-03}$	$5.09e-03_{3.3e-04}$	$5.07e-03_{5.4e-04}$
DTLZ5	$3.81e-04_{7.6e-05}$	$3.78e-04_{6.9e-05}$	$3.69e-04_{9.2e-05}$	$2.67e-04_{3.6e-05}$
DTLZ7	$2.84e-03_{5.9e-04}$	$2.56e-03_{4.1e-04}$	$2.63e-03_{6.0e-04}$	$3.18e-03_{1.3e-03}$

Tableau IV-15 - Indicateur "GD" pour les fonctions sans contraintes (DTLZ)

D'après les résultats, pour les fonctions à trois objectifs, le SPEAII détient les meilleurs résultats concernant la distribution des solutions, suivi par le FastPGA qui a une bonne convergence, le test suivant donne une idée plus claire sur les performances relatives des AGMOs étudiés selon les indicateurs de qualité.

Les résultats du test de Wilcoxon pour l'indicateur GD qui indique la convergence au front de Pareto réel sont exprimés dans le tableau suivant :

	SPEAII						FastPGA						MOCell					
	*D*1	*D*2	D3	Z4	*Z*5	D7	*D*1	*D*2	D3	Z4	*Z*5	D7	*D*1	*D*2	D3	Z4	*Z*5	D7
NSGAII	Δ	∇	Δ	∇	–	∇	–	∇	–	–	–	∇	Δ	Δ	Δ	–	∇	–
SPEAII							∇	–	∇	Δ	–	–	Δ	Δ	Δ	–	∇	Δ
FastPGA													Δ	Δ	Δ	–	∇	Δ

Tableau IV-16 - Test de Wilcoxon Pour les fonctions sans contraintes DTLZ

Le Boxplot de l'indicateur Hypervolume :

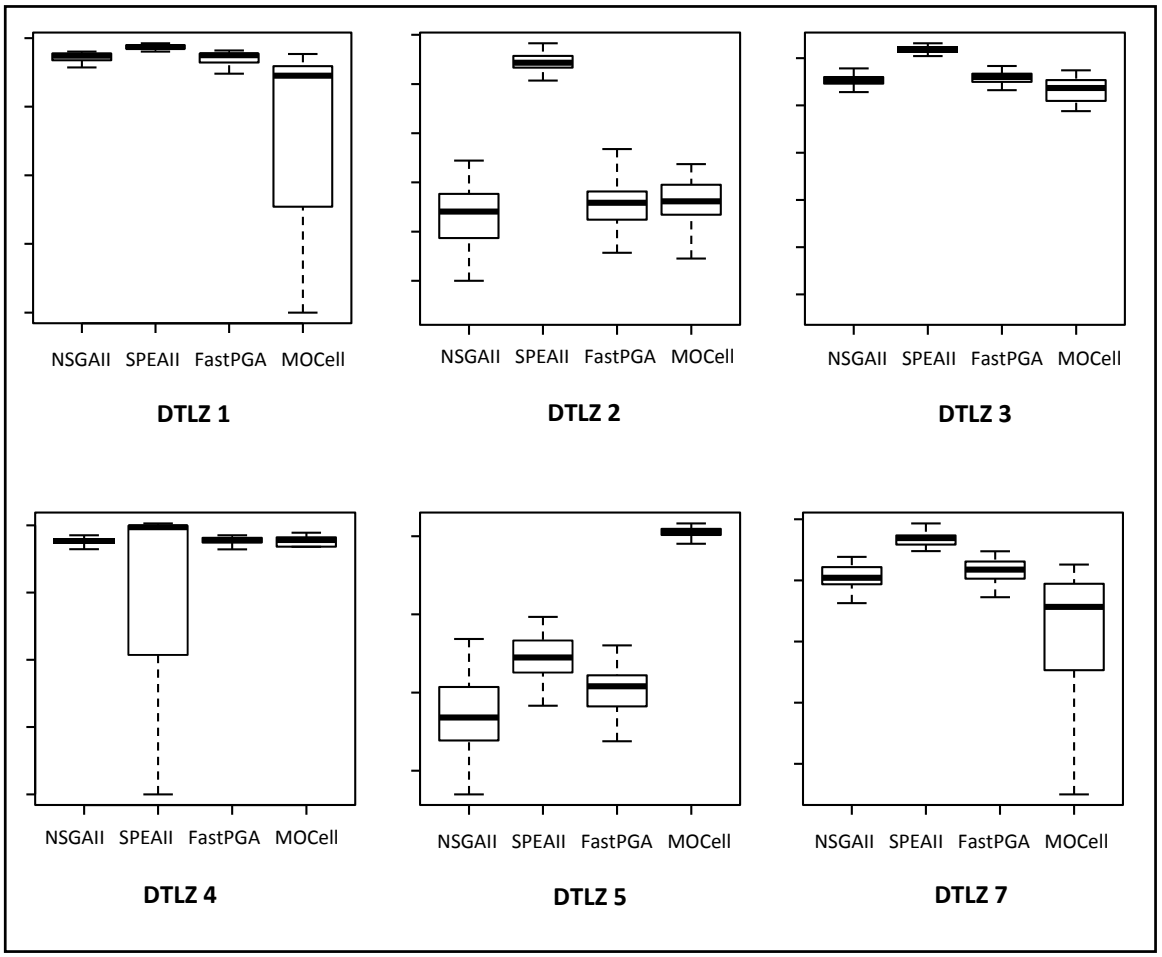

Figure IV-18 - Boxplot de l'indicateur HV pour les fonctions DTLZ

Le tableau (IV-13) (Spread) indique que le SPEA II détient les meilleures distributions des solutions dans l'espace objectif par rapport aux autres AGMOs, les indicateurs de convergence MOCell vers le front réel sont relativement mauvais, la figure suivante illustre les fronts de Pareto du SPEAII et du MOCell pour la fonction DTLZ1 :

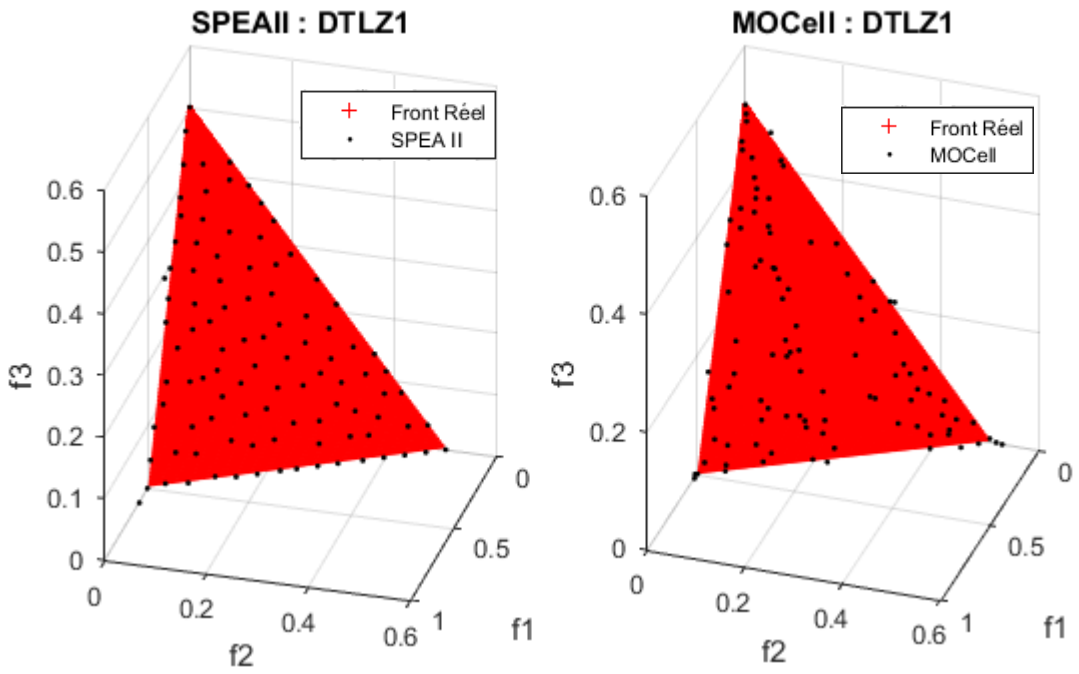

Figure IV-19 - Fronts de Pareto du SPEAII et MOCell : DTLZ1

Confirmant les résultats du Boxplot et de Wilcoxon, la distribution des solutions du SPEAII et largement meilleure que celle du MOCell.

Les tableaux IV-14 et IV-15 indiquent que le FastPGA a des performances acceptables concernant les problèmes à 3 objectifs, le test de Wilcoxon et le Boxplot indiquent que le FastPGA est plus performant que NSGAII dans la majorité de fonctions, nous avons choisi la fonction DTLZ3 qui montre une concurrence entre le FastPGA et le NSGA et le SPEAII.

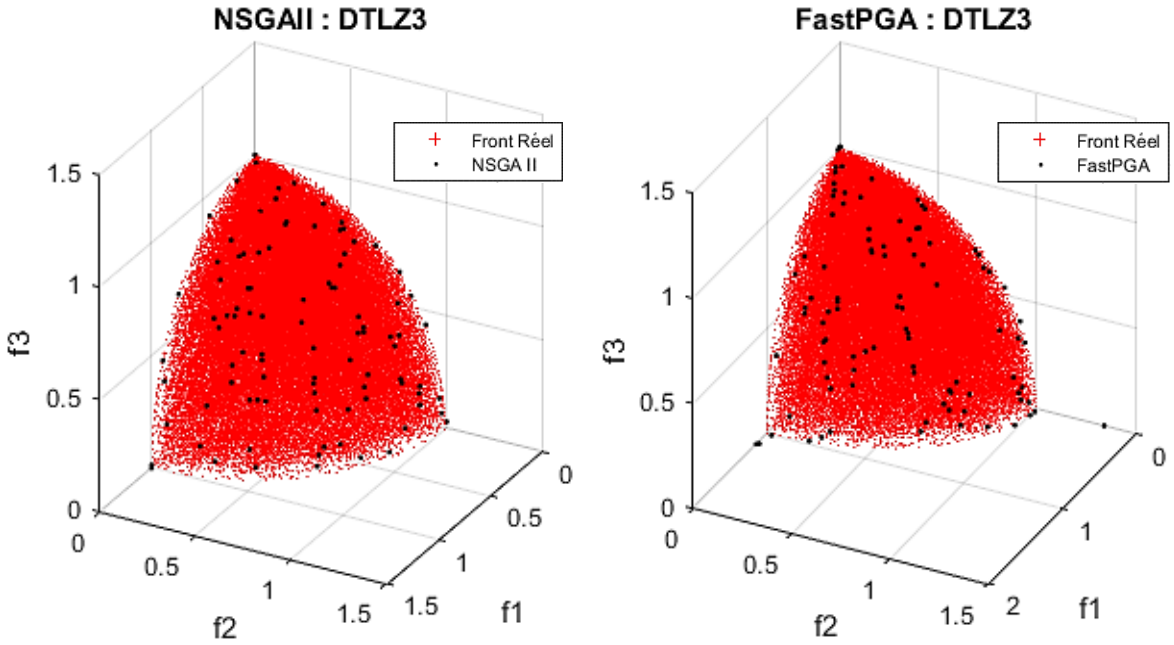

Figure IV-20 - Fronts de Pareto du NSGAII et FastPGA : DTLZ3

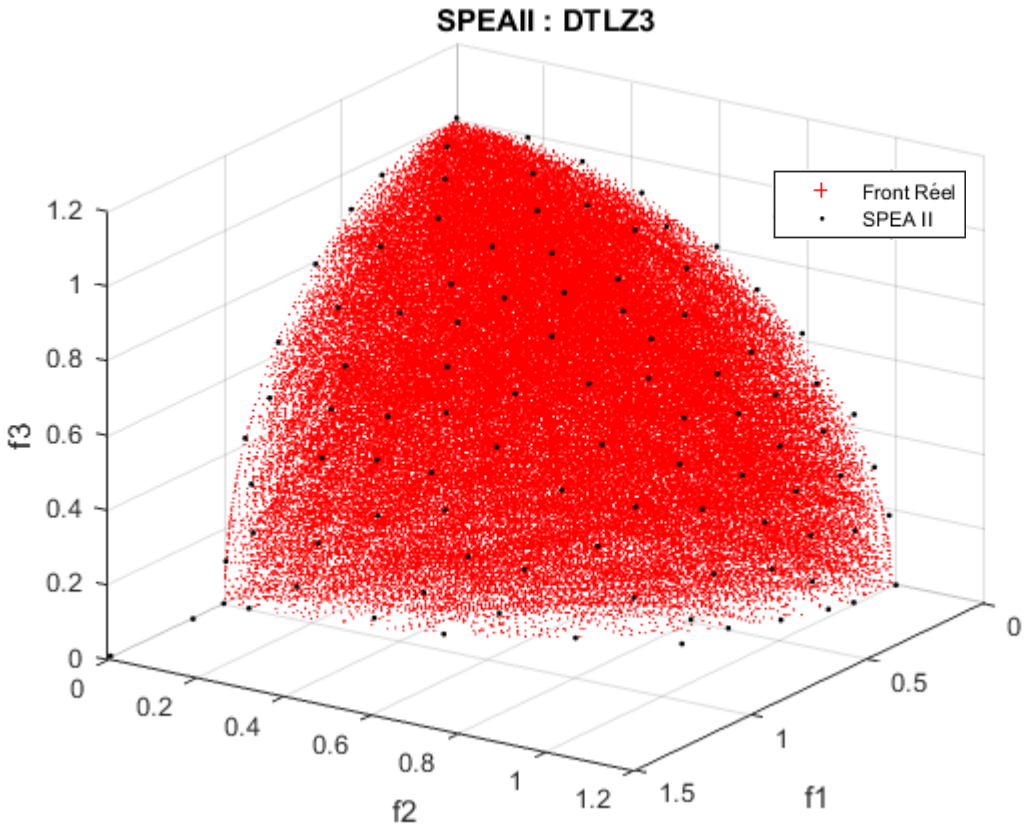

Figure IV-21 - Front de Pareto du SPEAII : DTLZ1

Le NSGA et FastPGA montre des performances similaires avec une légère supériorité du FastPGA en matière de convergence, tandis que le SPEAII affiche des performances nettement supérieures dans tous les indicateurs.

Le MOCell montre des performances uniquement pour la fonction DTLZ5 qui a un front sous forme de courbe dégénérée, les indicateurs montrent que le MOCell n'est pas destiné aux problèmes supérieurs à deux objectifs.

IV.6 – Conclusion

Dans ce dernier chapitre, nous avons mis en épreuve quatre algorithmes génétiques multi-objectifs de notre choix, plusieurs types de fonctions de benchmark ont été utilisés, avec contraintes, sans contraintes deux objectifs et sans contraintes trois objectifs, nous avons déduit les caractéristiques suivantes :

Le NSGA II est performant moyennement sur les problèmes à deux objectifs et moins performant sur les problèmes à trois objectifs.

Pour le MOCell le résultat est clair, parmi les quatre AGMOs il est le meilleur concernant les problèmes à deux objectifs, tandis que pour les fonctions à trois objectifs, les résultats des tests étaient relativement mauvais.

Le FastPGA comment le NSGA II, montre des performances moyennes et constantes pour les problèmes à deux objectifs et ceux à trois objectifs.

Le SPEA II a de bonnes performances avec les problèmes à deux objectifs et d'excellents résultats pour les problèmes à trois objectifs, il est le plus recommandé pour les problèmes à plus de deux objectifs.

Conclusion générale

Nous avons étudié dans ce travail l'une des méthodes les plus utilisées dans le domaine d'optimisation mono et multi-objectif qu'est les algorithmes génétiques. D'abord nous avons présenté les AGs dans leur forme la plus simple afin de donner une idée générale sur la méthode, puis on a entamé le sujet de l'optimisation multi-objectif où on a donné la description mathématique d'un problème d'optimisation multi-objectif. Dans la partie principale de notre travail, on a décrit d'une manière détaillé les quatre algorithmes génétiques (NSGAII, SPEAII, FastPGA et MOCell). Ensuite, nous avons entamé une étude comparative entre les AGMOs choisis en considérant plusieurs types de fonctions de test utilisées dans ce domaine

- le NSGAII en incorporant une nouvelle méthode élitiste et de tri rapide a prouvé son efficacité en matière de rapidité d'exécution et de convergence pour tous les types de problèmes étudiés.

- Le SPEAII utilise une méthode d'attribution de fitness efficace, ainsi qu'un archive pour sauvegarder les solutions non-dominées et avec sa nouvelle méthode de clustering, cet AGs montre des performances relativement meilleures que ses concurrents et particulièrement pour les problèmes à plus de deux objectifs, mais nécessite un temps d'exécution relativement long.

- L'opérateur de régulation de de la population et la méthode de classification des solutions donnent au FastPGA une capacité de convergence supérieure pour tous les types de problèmes étudiés, ses performances sont comparables au NSGAII.

- La structure cellulaire de la population de MOCell lui donne les meilleures performances en convergence vers le front réel et en distribution des solutions sur le front pour les problèmes à deux objectifs. Tandis que pour les problèmes à trois objectifs, les performances du MOCell sont minimales.

Le choix d'un AG repose sur le problème étudié, notre travail facilite ce choix en étudiant théoriquement et expérimentalement[7] quatre types d'algorithmes génétiques et nous avons donné les performances de chacun pour chaque type de problème d'optimisation.

[7] Uniquement sur des fonctions de test

Bibliographie

[1] Adorio, E. P., & Diliman, U. (2005). Mvf-multivariate test functions library in c for unconstrained global optimization.

[2] Alba, E., & Dorronsoro, B. (2009). *Cellular genetic algorithms* (Vol. 42). Springer Science & Business Media,13-20

[3] Baker, J. E. (1985, July). Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications* (pp. 101-111).

[4] Baker, J. E. (1987, July). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms* (pp. 14-21).

[5] Black, P. E. (2007). big-O notation. *Dictionary of Algorithms and Data Structures*, *2007*.

[6] Coello, C. A. C., & Lamont, G. B. (2004). *Applications of multi-objective evolutionary algorithms* (Vol. 1). World Scientific.

[7] Corne, D. W., Knowles, J. D., & Oates, M. J. (2000, January). The Pareto envelope-based selection algorithm for multiobjective optimization. In *Parallel Problem Solving from Nature PPSN VI* (pp. 839-848). Springer Berlin Heidelberg.

[8] Darwin, C. (1859). On the origins of species by means of natural selection. *London: Murray*, 247.

[9] Deb, K., & Agrawal, R. B. (1994). Simulated binary crossover for continuous search space. *Complex Systems*, *9*(3), 1-15.

[10] Deb, K., & Goyal, M. (1996). A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, *26*, 30-45.

[11] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, *6*(2), 182-197.

[12] Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002, May). Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA)* (pp. 825-830). Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA).

[13] Eskandari, H., Geiger, C. D., & Lamont, G. B. (2007, January). FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In *Evolutionary Multi-Criterion Optimization* (pp. 141-155). Springer Berlin Heidelberg.

[14] Fonseca, C. M., & Fleming, P. J. (1993, June). Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. In *ICGA* (Vol. 93, pp. 416-423).

[15] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675-701.

[16] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). *Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.*

[17] Goldberg, D. E. (1990). Real-coded genetic algorithms, virtual alphabets, and blocking. *Urbana*, *51*, 61801.

[18] Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, *1*, 69-93.

[19] Holland, J. (1975). Adaptation in artificial and natural systems. *Ann Arbor: The University of Michigan Press*.

[20] Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994, June). A niched Pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (pp. 82-87).

[21] Knowles, J., & Corne, D. (1999). The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (Vol. 1). IEEE

[22] Krause, E. F. (1973). Taxicab geometry. *The Mathematics Teacher*, 695-706.

[23] Man, K. F., Tang, K. S., & Kwong, S. (1996). Genetic algorithms: concepts and applications [in engineering design]. *Industrial Electronics, IEEE Transactions on*, *43*(5), 519-534.

[24] Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 50-60.

[25] Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., & Alba, E. (2009). Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, *24*(7), 726-746.

[26] Osyczka, A., & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural optimization*, *10*(2), 94-99.

[27] Sareni, B., & Krähenbühl, L. (1998). Fitness sharing and niching methods revisited. *Evolutionary Computation, IEEE Transactions on*, *2*(3), 97-106.

[28] Schaffer, J. D. (1985, January). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985* (pp. 93-100).

[29] Silverman, B. W. (1986). *Density estimation for statistics and data analysis* (Vol. 26). CRC press.19-21

[30] Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221-248.

[31] Tanaka, M., Watanabe, H., Furukawa, Y., & Tanino, T. (1995, October). GA-based decision support system for multicriteria optimization. In Systems, Man and Cybernetics, 1995. Intelligent Systems for *the 21st Century., IEEE International Conference on* (Vol. 2, pp. 1556-1561). IEEE.

[32] Tukey, J. W. (1977). Exploratory data analysis.39-42

[33] Tutte, W. T. (1966). *Connectivity in graphs* (Vol. 15). University of Toronto Press.1-4

[34] Van Veldhuizen, D. A., & Lamont, G. B. (1998). *Multiobjective evolutionary algorithm research: A history and analysis*. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

[35] Wright, S. (1946). Isolation by distance under diverse systems of mating. *Genetics*, *31*(1), 39.

[36] Zitzler, E., & Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: the strength Pareto approach.

[37] Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *evolutionary computation, IEEE transactions on*, *3*(4), 257-271.

[38] Zitzler, E., Deb, K. & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, *8*(2), 173-195.

[39] Zitzler, E., Laumanns, M. & Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm.

[40] Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760-771.

Etude comparative des algorithmes génétiques multi-objectifs

Résumé : Ce travail est sous forme d'une étude comparative entre quatre algorithmes génétiques multi-objectifs : le NSGA II, le SPEA II, le FastPGA et le MOCell. Elle est menée en utilisant des fonctions de test afin d'évaluer les performances (convergence et diversité) de chaque algorithme. Nous avons utilisé des méthodes de statistiques afin d'interpréter les résultats et de les comparer afin de déterminer quel sont les Algorithmes génétiques les mieux adaptés pour chaque type de problème. Ce mémoire est divisé en quatre parties la première est une généralité sur les algorithmes génétiques simples, les deuxième et troisième parties sont une étude détaillée sur l'optimisation multi-objectif par algorithmes génétiques, la quatrième partie est l'étude comparative par simulation.

Mots clés : Algorithmes génétiques, NSGA II, SPEA II, FastPGA, MOCell, étude comparative, optimisation, multi-objectif, mono-objectif.

Comparative study between multi-objective genetic algorithms

Abstract : This work is comparative study between four multi-objective genetic algorithms: the NSGA II, SPEA II, FastPGA and the MOCell. This study is conducted using benchmark functions in order to evaluate each genetic algorithm's ability to maintain diversity and convergence. We used statistical methods to interpret the obtained results and compare them in order to figure out which are the best-suited genetic algorithms for each type of problem. This work is divided into four parts, the first is a general definition of simple genetic algorithms, the second and the third parts are a detailed study of multi-objective optimization by genetic algorithms, and the last part is the comparative study by simulation.

Keywords : Genetic algorithms, NSGA II, SPEA II, FastPGA, MOCell, comparative study, optimization, multi-objective, mono-objective.