UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences

Département de Mathématiques

MEMOIRE DE MASTER

En Mathématiques

Spécialité : MODELISATION STOCHASTIQUE ET STATISTIQUE

HYBRIDATION DES OPERATEURS DE CROISEMENT DANS LES ALGORITHMES EVOLUTIONNAIRES MULTI-OBJECTIF

Par

BOUZID Abdelkader

Promoteur : Mr. AIT AKKACHE Mustapha, USD/ Blida 1.

Blida, 2015.

Remerciements

Je remercie Dieu tout puissant clément et miséricordieux de m'avoir soigné et aidé.

Je tiens, avant tout, à exprimer ma profonde gratitude à Monsieur ***Ait Akkache Mustapha****, professeur à l'université de Blida1, qui a assumé la direction de ce travail. Qu'il veuille bien trouver ici l'expression de ma reconnaissance pour son dévouement, sa patience, sa disponibilité, ses conseils et son aide constant qu'il m'a apporté tout au long de ce travail.*

Je remercie Messieurs le président et membres de jury, les premiers lecteurs de mon mémoire, d' avoir assisté et accepté de juger ce modeste travail.

J'adresse mes vifs remerciements à tous les enseignants et administrateurs du département de Mathématiques qui, par leur enseignement, leur encouragement et leur aide, ont contribués à ma formation durant toutes mes études à l'université de Blida1.

Je m'adresse d'une manière particulière par tout mon profond respect et tous mes remerciements à mon exemplaire, ***Madame, Dr/ OUKID Nadia****, pour ses conseils efficaces et ses encouragements pour faire le mieux.*

Je tiens également à remercier tous mes collègues de la sûreté nationale, en particulier la brigade de l' identité judiciaire pour leurs aide et gentillesse.

Enfin, je tiens à exprimer un grand merci à tous mes amis qui m' ont soutenu et supporté, en particulier ***BOUINOUNE Mahmoud*** *et* ***Tabarli A****.*

À mes parents...

ملخص:

الخوارزميات التطورية مستنبطة من نظرية تطور الكائنات الحية لداروين، و هي طرق احتمالية مستعملة في التوسع الكلي، لحل المسائل في الابعاد الكبرى، و خاصة المسائل متعددة الاهداف.

تعتبر هذه الخوارزميات الاطار الاساسي لهذه المذكرة و لهذا نبدأ عملنا بتقديم الخطوط العريضة للمسائل متعددة الاهداف، ثم نمر الى تقديم اساسيات الخوارزميات التطورية. و بعد ذلك نقدم في هذا العمل خوارزمية تطورية تستعمل الهجانة بين خمس عوامل للتقاطع، و في الاخير نقدم نتائج هجانة هذه العوامل من واحد الى خمسة.

الكلمات المفاتيح: التوسع متعدد الاهداف، الخوارزميات التطورية، الهجانة، التقاطع.

Résumé.

Les Algorithmes Evolutionnaires sont inspirés de l'évolution Darwinienne des populations. Ils sont des méthodes stochastiques d'optimisation globale, dédiés à la résolution des problèmes d'optimisation de grande dimension, et plus particulièrement multiobjectif.

Ces Algorithmes font le cadre principal de ce travail, et pour cela, Nous commencerons tout d'abord par les grandes lignes de l'optimisation multiobjective, remontant brièvement vers la présentation des notions des processus d'évolution. Nous passerons ensuite à présenter un algorithme évolutionnaire se basant sur une hybridation de cinq operateurs de croisement. Enfin, nous illustrerons les résultats d'hybridation de ces operateurs de un à cinq.

Mots Clefs : **Optimisation Multiobjective, Algorithmes évolutionnaires, Hybridation, Croisement.**

Abstract

The Evolutionary Algorithms had inspired by Darwinian's evolution of population. They had stochastic's optimization global methods, used to resolve the problems with high dimension, and multiobjective.

These algorithms make the main framework of this thesis, by this, we start by the headlines of multiobjective optimization, moving slowly to present an evolutionary algorithm based on the hybridization of five of crossover operators. Finally, we illustrate the results of hybridization of these operators for one to five.

***Keywords*:** **Multiobjective optimization, Evolutionary algorithm, Hybridization, Crossover.**

Glossaire

A

AE Algorithmes évolutionnaires.

B

BLX Blend Crossover.

D

DE Differential Evolution.

E

EP Evolutionary Programming.
ES Evolution Strategie.
EXA Archive Externe.

G

GA Genetic Algorithm.
GD General Distance.
GP Genetic Programming.

H

HMC-MOEA Hybrid of Multi-operator Crossover for Multi-Objective Evolutionary Algorithms.

L

LB Lower Bound

M

MOEA Multi-Objective Evolutionary Algorithms.

N

n_exa La taille maximale de l'archive externe.
n_pop La taille de la population.
NSGA Non Sorting Genetic Algorithm.

P

PCX Parent Centric Crossover.
PO Ensemble de solutions Pareto optimales.

PO* Ensemble de solutions potentiellement Pareto optimales trouvées par un algorithme

POM Problèmes d'optimisaton multiobjective.

S

SBX Simulated Binary Crossover.

SM Spacing Metric.

SPX Simplex Crossover.

U

UB Upper Bound.

Z

ZDT Zitzler, Deb, Thiele.

TABLE DES MATIERES

INTRODUCTION

Les ingénieurs se heurtent quotidiennement à des problèmes technologiques de complexité grandissante, qui surgissent dans des secteurs très divers, comme dans le traitement des images, la conception de systèmes mécaniques, la recherche opérationnelle et l'électronique. Le problème à résoudre peut fréquemment être exprimé sous la forme générale d'un problème d'optimisation, dans lequel on définit une fonction objectif, ou fonction de coût, que l'on cherche à minimiser par rapport à tous les paramètres concernés. Par exemple, dans le célèbre problème du voyageur de commerce, on cherche à minimiser la longueur de la tournée d'un "voyageur de commerce", qui doit visiter un certain nombre de villes, avant de retourner à la ville de départ. La définition du problème d'optimisation est souvent complétée par la donnée de contraintes : tous les paramètres (ou variables de décision) de la solution proposée doivent respecter ces contraintes, faute de quoi la solution n'est pas réalisable.

Il existe de nombreuses méthodes d'optimisation "classiques" pour résoudre de tels problèmes, applicables lorsque certaines conditions mathématiques sont satisfaites : ainsi, la programmation linéaire traite efficacement le cas où la fonction objectif, ainsi que les contraintes, s'expriment linéairement en fonction des variables de décision. Malheureusement, les situations rencontrées en pratique comportent souvent une ou plusieurs complications, qui mettent en défaut ces méthodes : par exemple, la fonction objectif peut être non linéaire, ou même ne pas s'exprimer analytiquement en fonction des paramètres ; ou encore, le problème peut exiger la considération simultanée de plusieurs objectifs contradictoires.

L'optimalité d'une solution, dans ce dernier cas, n'est pas clairement définie et plusieurs solutions de compromis entre les performances des objectifs sont possibles. Ces solutions de compromis sont généralement dites les solutions efficaces (aussi appelées Pareto-optimales) du problème. La recherche de ces solutions de compromis et plus particulièrement dans le cas des fonctions objectif non linéaires est très comliquée [28]. Parmi les outils utilisés pour déterminer ces solutions efficaces, on trouve les algorithmes évolutionnaires multi-objectifs [32]. Ces métaheuristiques basées sur la manipulation d'un ensemble de solutions dit

population, en utilisant des opérateurs de variations tels que le croisement et la mutation arrivent souvent à donner une bonne approximation de l'ensemble de solutions de Pareto.

Dans ce travail, nous nous intéressons plus particulièrement à l'opérateur de croisemnt qui permet d'explorer l'ensemble des solutions réalisables de manière stochastique afin d'identifier les solutions efficaces. Dans la litérature, on trouve plusieurs types de croisement, BLX (Blend alpha Crossover), SBX (Simulated Binary Crossover), SPX (Simplex Crossover), PCX (Parent Centric Crossover) et DE (Differential Evolution),....

Le but de ce mémoire est d'étudier la possibilité d'utiliser un opérateur de croisement hybride dans l'espoire d'accélérer la convergence et la diversité dans les algorithmes évolutionnaires multi-objectif.

À cet effet, notre mémoire est organisé da la façon suivante: le premier chapitre est consacré à l'optimisation multi-objectif, le second pour donner le fonctionnement des algorithmes évolutionnaires, le troisième chapitre présente un algorithme évolutionnaire utilisant le mécanisme d'hybridation de cinq opérateurs de croisement, le quatrième chapitre est consacré aux résultats et analyses d'hybridation des opérateurs de croisement et on termine par une conclusion générale.

CHAPITRE 1

Les Problèmes d'Optimisation Multi-objectif

1.1 Introduction

Résoudre un problème d'optimisation consiste à trouver les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise un critère de comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenu la meilleure évaluation au regard du critère défini. Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes appliqués : le traitement d'images, la conception de systèmes, la conception d'emplois du temps, ... La majorité de ces problèmes sont qualifiés de difficiles, car leur résolution nécessite l'utilisation d'algorithmes évolués, et il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable. Lorsqu'un seul critère est donné, par exemple un critère de minimisation de coût, la solution optimale est clairement définie, c'est celle qui a le coût minimal. Mais dans de nombreuses situations, un seul critère peut être insuffisant. En effet, la plupart des applications traitées intègrent plusieurs critères simultanés, souvent contradictoires. Intégrer des critères contradictoires pose un réel problème. Considérons les actions suivantes :

* Louer un appartement bien situé et d'un prix raisonnable.
* Etablir un planning pour les vacances satisfaisant toute la famille.
* Choisir entre plusieurs itinéraires.
* Acheter une voiture.

La plupart des critères liés à ces tâches sont contradictoires. La solution idéale n'existe pas, et il faut donc trouver un compromis. En effet, en considérant deux critères contradictoires a et b, améliorer a détériore forcément b et inversement. Le concept de solution optimale devient alors plus difficile à déffinir. Dans ce cas, la solution optimale cherchée

n'est plus un point unique, mais un ensemble de compromis. Résoudre un problème comprenant plusieurs critères, appelé communément problème multiobjectif, consiste donc à calculer le meilleur ensemble de solutions compromis : le front Pareto. Dans ce chapitre, nous présentons les problèmes d'optimisation multiobjectifs.

1.2 L'Optimisation Multi-objectif

La principale difficulté que l'on rencontre en optimisation monobjectif vient du fait que modéliser un problème sous la forme d'une équation unique peut être une tâche difficile. Avoir comme but de se ramener à une seule fonction objectif peut aussi biaiser la modélisation.

L'optimisation multiobjectif autorise ces degrés de liberté qui manquaient en optimisation monobjectif. Cette souplesse n'est pas sans conséquences sur la démarche à suivre pour chercher un optimum à notre problème enfin modélisé. La recherche ne nous donnera plus une solution unique mais une multitude de solutions. Ces solutions sont appelées solutions de Pareto et l'ensemble de solutions que l'on obtient à la fin de la recherche est la surface de compromis.

C'est après avoir trouvé les solutions du problème multiobjectif que d'autres difficultés surviennent : il faut sélectionner une solution dans cet ensemble. La solution qui sera choisie par l'utilisateur va refléter les compromis opérés par le décideur vis-à-vis des différentes fonctions objectif.

Cependant, lorsque l'on modélise un problème, on cherche souvent à satisfaire plusieurs objectifs. Par exemple, on veut un système performant et on veut aussi que ce système consomme peu. Dans ce cas, on parle de problème d'optimisation multiobjectif **(POM)**. Celui-ci s'écrit de la manière suivante :

$$
\begin{aligned}
&Minimiser\ \overrightarrow{f}(\overrightarrow{x}) \\
&avec\ \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\
&et\ \overrightarrow{h}(\overrightarrow{x}) = 0 \\
&où\ \overrightarrow{x} \in \mathbb{R}^n,\ \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^k,\ \overrightarrow{f} = [f_1, f_2, ..., f_k],\ \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^m,\ \overrightarrow{h}(\overrightarrow{x}) \in \mathbb{R}^p
\end{aligned}
\tag{1}
$$

Comme on peut le voir ici, on n'a plus un seul objectif à atteindre, mais k (le vecteur

$\overrightarrow{f}$ regroupe k fonctions objectif).

Le but que l'on se fixe dans la résolution d'un problème d'optimisation multiobjectif est de minimiser "au mieux" les différents objectifs. Comme on va le voir dans le paragraphe suivant, dans un problème d'optimisation multicritère, on rencontre souvent des objectifs contradictoires. Deux objectifs sont contradictoires lorsque la diminution d'un objectif entraîne une augmentation de l'autre objectif.

1.2.1 La multiplicité des solutions [28]

Lorsque l'on cherche à obtenir une solution optimale à un problème d'optimisation multiobjectif donné, on s'attend souvent à trouver une solution et une seule. En fait, on rencontre rarement ce cas de figure. La plupart du temps, on trouve une multitude de solutions, du fait que certains des objectifs sont contradictoires.

Quand on résoudra un problème d'optimisation multiobjectif, on obtiendra une grande quantité de solutions. Ces solutions, comme on peut s'en douter, ne seront pas optimales, au sens où elles ne minimiseront pas tous les objectifs du problème. Un concept intéressant, qui nous permettra de définir les solutions obtenues, est le compromis.

En effet, les solutions que l'on obtient lorsque l'on a résolu le problème sont des solutions de compromis. Elles minimisent un certain nombre d'objectifs tout en dégradant les performances sur d'autres objectifs.

1.2.2 La Dominance [32]

Lorsque nous avons résolu notre problème d'optimisation multiobjectif, nous avons obtenu une multitude de solutions. Seul un nombre restreint de ces solutions va nous intéresser. Pour qu'une solution soit intéressante, il faut qu'il existe une relation de dominance entre la solution considérée et les autres solutions, dans le sens de la définition suivante:

Définition 1.2.1. *(la relation de dominance)*

On dit que le vecteur $\overrightarrow{x_1}$ *domine le vecteur* $\overrightarrow{x_2}$ *si:*

- $\overrightarrow{x_1}$ *est au moins aussi bon que* $\overrightarrow{x_2}$ *dans tous les objectifs, et,*
- $\overrightarrow{x_1}$ *est strictement meilleur que* $\overrightarrow{x_2}$ *dans au moins un objectif.*

Les solutions qui dominent les autres mais ne se dominent pas entre elles sont appelées **solutions optimales au sens de Pareto** (ou **solutions non dominées**). On définit comme suit l'optimalité locale et l'optimalité globale au sens de Pareto.

Définition 1.2.2. *(optimalité locale au sens de Pareto)*

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

Un vecteur $\overrightarrow{x}$ est donc optimal localement au sens de Pareto s'il est optimal au sens de Pareto sur une restriction de l'ensemble R^n.

Définition 1.2.3. *(optimalité globale au sens de Pareto)*

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

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

Lorsque l'on applique la définition de la dominance, on peut définir quatre régions auxquelles on peut attribuer des niveaux de préférence. Ces régions sont représentées à la figure 1.2.

Par exemple, si ce graphique est centré sur une solution A et que l'on compare cette solution avec une solution B, on aura les possibilités suivantes :

– si la solution B se trouve dans le quadrant 1, alors la solution A est préférée à la solution B;

– si la solution B se trouve dans le quadrant 3, alors la solution A est dominée par la solution B;

– si la solution B se trouve dans l'un des quadrants 2 ou 4, alors, on ne peut pas se prononcer sur la préférence de A par rapport à B ou de B par rapport à A.

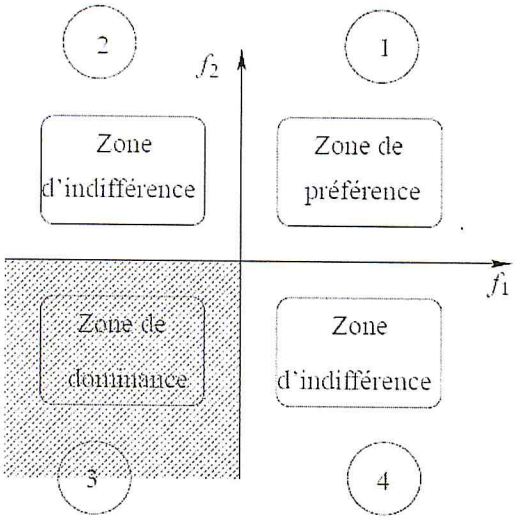


FIGURE 1.1. Les niveaux de préférence dans la relation de dominance.

1.2.3 Procédure de recherche de l'ensemble non-dominé

Dans cette section, M représente la taille d'une population, et N le nombre de générations.

Naive et Lente:

Dans cette aproche, chaque solution i est comparée à toutes les autres solutions jusqu'à ce qu'elle soit dominée par l'une d'elles. Si aucune ne la domine, elle est déclarée non-dominée. La complixité de cette procédure dans le pire des cas est $O(MN^2)$ [18].

Mise à jour continue:

Cette approche est très similaire à la précédente, on lui a simplement rajouté une mémoire qui la rend en général plus efficace- mais pas dans tous les cas.

Chaque solution de P est comparée avec un sous-ensemble partiellement rempli et continûment mis à jour. Au début, l'ensemble P' contient la première solution. Ensuite, chaque solution i sera comparée aux éléments de P'. Toutes les solutions de P' dominées par i, en sont éliminées. Si aucune solution de P' ne domine i, i est ajoutée à P'. La complixité "au pire des cas" de cet algorithme est la même que celle de l'approche précédente, c'est-à-dire $O(MN^2)$. Mais en moyenne cette seconde approche est plus rapide [18].

La plus efficace:

Cet algorithme proposé par Kung et al, en 1975 [12], consiste à ordoner l'ensemble P

par ordre croissant du premier critère et à appeler la procédure récursive suivante:

Procédure FRONT(P) [28]

Notons pi le $i^{ième}$ élément de P, La procédure retourne un ensemble de solutions.

- Si $|P| = 1$, retourne P comme le résultat de la procédure FRONT(P)
- Sinon

1. $A = FRONT(\{p_1, .., p_{|p|/2}\})$ et $B = (\{p_{|p|/2+1}, .., p_{|P|}\})$
2. Pour chaque solution b_i de l'ensemble B, $i = 1, .., |B|$
   - si b_i n'est dominée par aucune solution de A, alors $A = A \cup \{b_i\}$
3. retourner A comme le résultat de la procédure FRONT(P)

La complixité de cet algorithme est $\mathbf{O(N(\log N)^{M-2})}$ pour $\mathbf{M > 4}$, et $\mathbf{O(N(\log N))}$ pour $\mathbf{M = 2, 3}$.

Assignation du rang de Pareto:

On peut établir un classement des solutions en fonction du rang de domination. On attribue dans un ensemble de soltuions le rang 1 aux points qu'ils dominent tous les autres points mais ne se dominent pas entre eux. Ces points sont donc des solutions optimales au sens de Pareto de rang 1 et ils seront supprimés. Le processus d'assignation s'arrête lorsque l'ensemble des points à comparer est vide.

Le pseudo-code de la fonction d'assignation du rang de Pareto est représenté dans l'algorithme suivant. Dans cet algorithme, la variable N désigne le nombre de points de l'ensemble sur lequel on effectue les comparaisons.

Algorithm 1. ***Assigniation du rang de Pareto.*** *[32]*

RangCourant = 1

$m = N$

while $N \neq 0$ *do*

For $i = 1$ *to* m *do*

If X_i *est Non dominé*

$Rang(X_i, t) = RangCourant$

End For

For $i = 1$ *to* m *do*

If $Rang(X_i, t) = RangCourant$

Ranger X_i *dans une population temporaire*

$N = N - 1$

End For

$RangCourant = RangCourant + 1$

$m = N$

End While

1.2.4 Les relations dérivées de la dominance [32]

La relation de dominance n'offre pas de degrés de liberté dans sa définition. Par exemple, il n'est pas possible d'inclure dans la définition de la relation de dominance une préférence d'un objectif par rapport à un autre. C'est pour contrecarrer ce manque de flexibilité que des relations dérivées de la relation de dominance ont été développées. Les solutions que permettent de trouver ces relations dérivées de la dominance sont toutes optimales au sens de Pareto. La grande différence que l'on rencontre avec ces relations est que l'ensemble des solutions que l'on obtient avec ces relations est un sous-ensemble de l'ensemble des solutions obtenues avec la relation de dominance de Pareto.

Dans cette section, l'ensemble S^k désigne l'ensemble des solutions réalisables d'un problème d'optimisation à k fonctions objectif.

Optimalité lexicographique Cette définition de l'optimalité permet d'inclure une préférence entre objectifs [25].

Définition 1.2.4. *(optimalité au sens lexicographique)*

Une solution $\overrightarrow{x^*} \in S^k$ *est optimale au sens lexicographique si:*
$\overrightarrow{x^*} \leq_{lex} \overrightarrow{x}, \forall \overrightarrow{x} \in S^k - \{\overrightarrow{x^*}\}$.

Si $\overrightarrow{x}$, $\overrightarrow{y} \in S^k$, on dit que $\overrightarrow{x} \leq_{lex} \overrightarrow{y}$ s'il existe une valeur d'index q^* telle que$x_q = y_q$ pour $q = 1, \cdots, (q^* - 1)$ et $x_{q^*} < y_{q^*}$. Les relations entre x_q et y_q pour $q \geq q^*$ ne sont pas prises en compte car nous nous arrêtons à l'indice q^* (c'est le premier indice pour lequel $x_q < y_q$).

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

Optimalité extrême Comme pour la relation d'optimalité lexicographique, cette relation permet d'établir une préférence entre critères. Cette préférence est établie en utilisant des poids. Plus un objectif sera important, plus son poids sera élevé [25].

Définition 1.2.5. *(optimalité extrême)*

Une solution $\overrightarrow{x^*} \in S^k$ *est extrême-optimale si, étant donné un vecteur de poids* $\overrightarrow{\lambda} \in \mathbb{R}^k$ *tel que* $\sum_i \lambda_i = 1, \overrightarrow{x^*}$ *est une solution optimale du problème de minimisation monocritère ayant pour fonction objectif*

$$\sum_{i=1}^{i=k} \lambda_i . \overrightarrow{x_i}$$

donc:

$$\sum_{i=1}^{i=k} \lambda_i . \overrightarrow{x_i^*} \leq \sum_{i=1}^{i=k} \lambda_i . \overrightarrow{x_i}, \forall \overrightarrow{x} \in S^k - \{\overrightarrow{x^*}\}$$

Optimalité maximale Cette relation, contrairement aux précédentes, ne permet pas d'introduire une préférence entre objectifs [25].

Définition 1.2.6. *(optimalité maximale)*

Une solution $\overrightarrow{x^*} \in S^k$ *est max-optimale si la valeur du pire objectif est aussi petite que possible :*

$$\begin{aligned} \max\ x_q^* &\leq \max\ x_q \\ |q &\in \{1, ..., k\} \\ |\overrightarrow{x} &\in S^k - \{\overrightarrow{x^*}\} \end{aligned}$$

La a-dominance [13]

Définition 1.2.7. *Une solution* $\overrightarrow{x^*} \in S^k$ *a-domine une solution* $\overrightarrow{x} \in S^k$ *s'il existe un ensemble de combinaisons de* $k+1-a$ *critères (on note* $I_{(k+1-a)}$ *l'ensemble des index correspondant à l'ensemble des combinaisons de ces critères) tel que :*

1. $\overrightarrow{x_j^*} \leq \overrightarrow{x_j}$ *pour tout* $j \in I_{(k+1-a)}$, *et*
2. $\overrightarrow{x_j^*} < \overrightarrow{x_j}$ *pour au moins un* $j \in I_{(k+1-a)}$.

1.2.5 Conditions d'optimalité

Dans cette section, nous citons les conditions nécessaire et suffisante d'optimalité au sens de Pareto formulées pour un problème d'optimisation. Mais avant de citer ces conditions, nous rappelons ici quelques définitions liées à la convexité, qui nous seront utiles par la suite.

Définition 1.2.8. *(La convexité)*

la fonctionf : $R^n - \rightarrow R$ *est convexe si et seulement si pour tout pair de variable* $x_1, x_2 \in R^n$ *et pour tout* $\lambda \in [0, 1]$,

$f(\lambda.x_1 + (1-\lambda).x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$

Rappelons que si une fonction g est concave, l'ensemble de x tels que $g(x) \geq 0$ est un ensemble convexe.

Définition 1.2.9. *Un problème d'optimisation multi-objectif est appelé convexe si et seulement si toutes les fonctions objectif f_m sont convexes et l'espace faisable est un ensemble convexe (c'est-à-dire toutes les fonctions g_j sont non convexs et toutes les fonctions h_j sont linéaires).*

Nous sommes donc prêts maintenant à citer les conditions nécessaire et suffisante d'optimalité. Nous supposons par la suite que toutes les fonctions objectif f_m, $m = 1, ..., M$, ainsi que les fonctions de contraintes g_j, $j = 1, ..., K$ et h_j, $j = 1, ..., L$, sont continûment différentiables.

Théorème 1.2.1. ***(Condition nécessaire de Fritz-John.)***

Si x^* une solution Pareto-optimale d'un problème d'optimisation, Alors, il existe deux vecteurs non nuls $\lambda \geq 0$ et $u \geq 0$ ($\lambda \in \mathbb{R}^M$ et $u \in \mathbb{R}^K$) tels que :

1. $\sum_{m=1}^{M} \lambda_m \nabla f_m(x^*) - \sum_{j=1}^{K} u_j \nabla g_j(x^*) = 0$ *et*
2. $u_j g_j(x^*) = 0$ *pour tout* $j = 1, ..., K$.

La pruve de ce théorème peut être trouvée dans [27].

Théorème 1.2.2. ***(Condition suffisante de Karush-Kuhn-Tucker.)***

Si un problème multi-objectif est convexe et si la condition nécessaire de Fritz-John est vérifiée, alors, la solution x est une solution Pareto-optimale [21].

1.2.6 La surface de compromis

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

Imaginons que nous ayons un problème à deux objectifs (minimiser $f1$ et minimiser $f2$ sous les contraintes $\overrightarrow{g}(\overrightarrow{x}) \leq 0$ *et* $\overrightarrow{h}(\overrightarrow{x}) = 0$) :

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

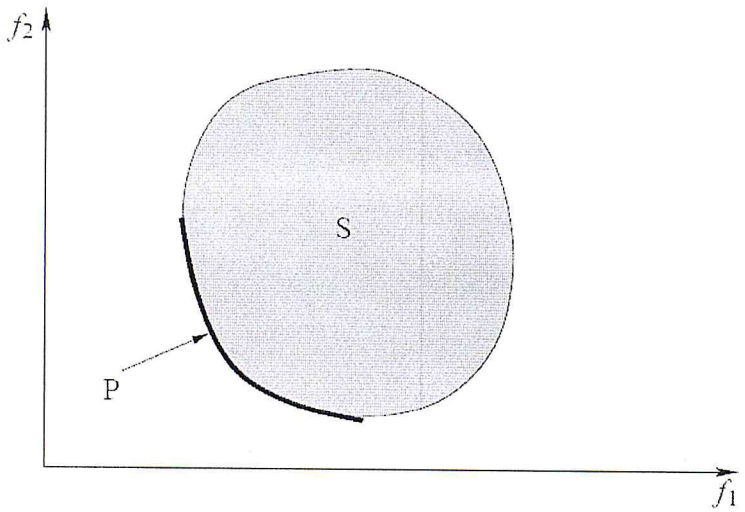


FIGURE 1.2. Représentation de la surface de compromis [32]

– On appelle P la surface de compromis.

On observe deux points caractéristiques associés à une surface de compromis :

Point idéal :

Les coordonnées de ce point sont obtenues en optimisant chaque fonction objectif séparément.

Point "nadir" :

Les coordonnées de ce point correspondent aux pires valeurs obtenues par chaque fonction objectif lorsque l'on restreint l'espace des solutions à la surface de compromis.

Le point idéal est utilisé dans beaucoup de méthodes d'optimisation comme point de référence. Le point nadir, lui, sert à restreindre l'espace de recherche ; il est utilisé dans certaines méthodes d'optimisation interactives.

1.2.7 La représentation de la surface de compromis

Toutes les représentations de la surface de compromis, pour un même problème, ne sont pas équivalentes. En effet, la représentation idéale de la surface de compromis devra être

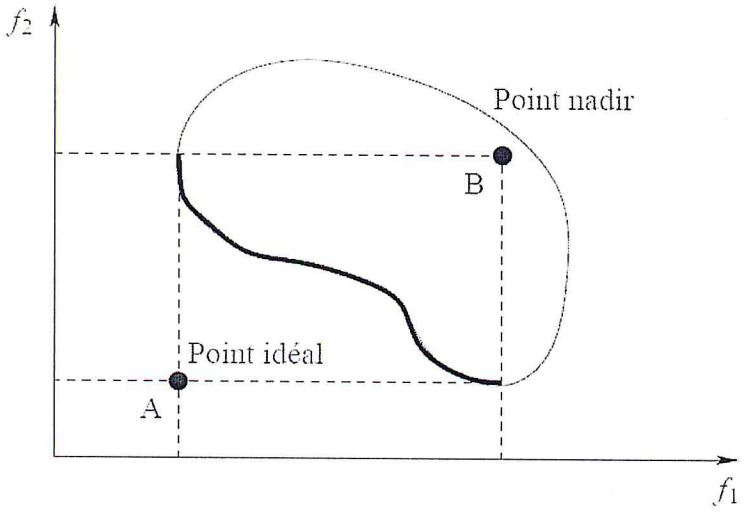


FIGURE 1.3. Représentation du point idéal et du point "nadir" [32]

constituée de points solution de notre problème répartis de manière uniforme sur la surface de compromis.

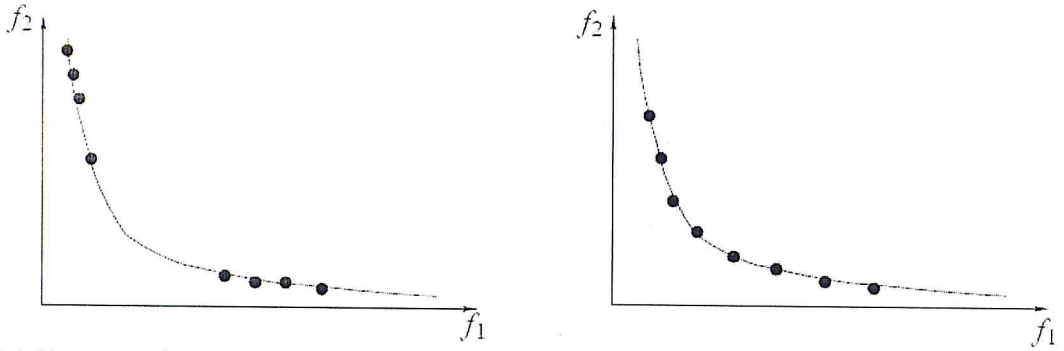
Dans le premier cas, les points représentant la surface de compromis ne sont pas répartis de manière uniforme. L'utilisateur n'aura alors pas en sa possession un ensemble de solutions très utile. En effet, s'il décide que la solution qu'il avait choisie ne lui convient pas, le choix d'une autre solution risque de faire varier brusquement tous ses objectifs, et cette nouvelle solution ne lui conviendra pas non plus. Il est alors probable que la solution offrant le "meilleur" compromis se trouve dans une zone qui ne soit pas représentée par des points solution.

La détermination d'une bonne représentation de la surface de compromis sera un critère de choix d'une méthode d'optimisation multiobjectif.

1.3 Approches traditionelles pour la résolution des (POM)

Il existe un nombre important de méthodes qui peuvent être classées en cinq groupes :

– les méthodes scalaires,



(a) Une mauvaise représentation de la surface de compromis

(b) Une bonne représentation de la surface de compromis

FIGURE 1.4. La représentation de la surface de compromis.

– les méthodes interactives,

– les méthodes floues,

– les méthodes exploitant une métaheuristique,

– les méthodes d'aide à la décision.

Les méthodes de ces cinq groupes peuvent aussi être rangées en trois familles de méthodes d'optimisation multiobjectif [5]

* Les méthodes à préférence **à priori** :

Dans ces méthodes, l'utilisateur définit le compromis qu'il désire réaliser (il fait part de ses préférences) avant de lancer la méthode d'optimisation. On retrouve dans cette famille la plupart des méthodes par agrégation (où les fonctions objectif sont fusionnées en une seule).

* Les méthodes à préférence **progressive** :

Dans ces méthodes, l'utilisateur affine son choix de compromis au fur et à mesure du déroulement de l'optimisation. On retrouve dans cette famille les méthodes interactives.

* Les méthodes à préférence **à posteriori** :

Dans ces méthodes, l'utilisateur choisit une solution de compromis en examinant toutes les solutions extraites par la méthode d'optimisation. Les méthodes de cette famille fournissent, à la fin de l'optimisation, une surface de compromis.

Nous présenterons, dans ce qui suit, les deux méthodes les plus connues et util-

isées : l'optimisation par **pondération des fonctions objectif**, et l'approche par **ε-contraintes**.

1.3.1 Méthode de pondération des fonctions objectif [3]

Cette approche de la résolution d'un problème d'optimisation multiobjectif est la plus évidente. D'ailleurs, on appelle aussi cette méthode l' "approche naïve" de l'optimisation multiobjectif. Le but, ici, est de revenir à un problème d'optimisation monobjectif, dont il existe de nombreuses méthodes de résolution. La manière la plus simple de procéder consiste à prendre chacune des fonctions objectif, à leur appliquer un coefficient de pondération et à faire la somme des fonctions objectif pondérées. On obtient alors une nouvelle fonction objectif.

On part du problème (1) (voir page 11). Le problème (1) se transforme de la manière suivante [3]:

$$\begin{aligned} &\text{minimiser } f_{eq}(\overrightarrow{x}) = \sum_{i=1}^{k} \omega_i f_i(x) \\ &\text{avec } \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ &\text{et } \quad \overrightarrow{h}(\overrightarrow{x}) = 0 \end{aligned} \tag{2}$$

On a $\overrightarrow{x} \in \mathbb{R}^n$, $\overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^m$, $\overrightarrow{h}(\overrightarrow{x}) \in \mathbb{R}^p$.

Fréquemment, les coefficients de pondération respectent la relation suivante :

$\omega_i \geq 0$ pour tous les $i \in \{1, \cdots, k\}$ et :

$$\sum_{i=1}^{k} \omega_i = 1$$

Certains résultats théoriques clarifiant le rapport entre la solution du problème (2) et les solutions Pareto-optimales du problème (1) ont été établis sous forme de deux théorèmes qui sont présentés comme suit:

Théorème 1.3.1. *Si tous les poids ω_i sont positifs, La solution du problème (2) est une solution Pareto-optimale du problème multi-objectif (1), [30].*

Théorème 1.3.2. *Si le problème d'optimisation multi-objectif (1) est convexe et* $\overrightarrow{x}$ *est une de ses solutions Pareto-optimales, il existe un vecteur des poids positifs* $(\omega_1, ..., \omega_k)$ *tel que* $\overrightarrow{x}$ *est la solution du problème (2), [21].*

On peut représenter graphiquement le fonctionnement de la méthode sur un problème à deux objectifs. Le problème est le suivant :

minimiser $f_1(x)$
minimiser $f2(x)$
avec $\overrightarrow{g}(\overrightarrow{x}) \leq 0$
et $\overrightarrow{h}(\overrightarrow{x}) = 0$

Notre nouvelle fonction objectif aura pour expression :

$$f_{eq}(\overrightarrow{x}) = \omega_1 . f_1(\overrightarrow{x}) + \omega_2 . f_2(\overrightarrow{x})$$

Ceci est l'expression d'une droite dans le plan f_1, f_2. En effet, si l'on cherche à minimiser $f_{eq}(\overrightarrow{x})$, on cherche en fait une constante C de

l'équation de droite suivante la plus petite possible :

$$f_2(\overrightarrow{x}) = -\frac{w_1}{w_2} . f_1(\overrightarrow{x}) + C$$

Cette équation de droite correspond à la courbe des isovaleurs de la fonction objectif équivalente.

l'ensemble S correspond à l'ensemble des valeurs du couple (f_1, f_2) respectant les contraintes définies par $\overrightarrow{g}(\overrightarrow{x})$ et $\overrightarrow{h}(\overrightarrow{x})$. Les droites L1 et L2 correspondent à deux couples de coefficients de pondération (w_1, w_2) différents.

Cette méthode consiste à "faire tangenter" la droite L1 et la droite L2 avec l'ensemble S. Le point de tangence est alors la solution recherchée. Si l'on répète ce processus pour plusieurs valeurs des coefficients de pondération, les différentes solutions trouvées forment la surface de compromis. Cette méthode n'est applicable qu'à des ensembles S convexes. Dans le cas contraire, elle ne permet pas de trouver la totalité de la surface de compromis.

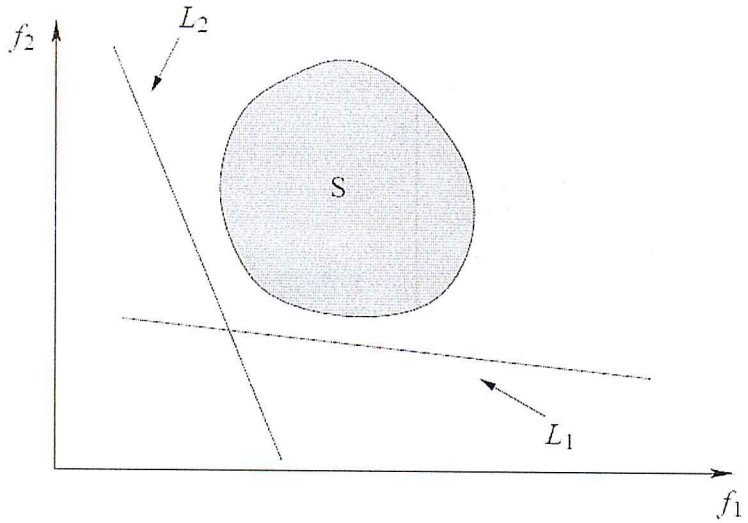


FIGURE 1.5. La méthode de pondération des fonctions objectif.

1.3.2 Méthode de $\epsilon-$contrainte [31]

Cette méthode est basée sur la minimisation d'un objectif f_i en considérant que les autres objectifs f_j avec $i \neq j$ doivent être inférieurs à une valeur ϵ_j. En général, l'objectif choisi est celui que le décideur souhaite optimiser en priorité [31].

$$Mininiser\ f_i(x) \text{ avec}$$
$$f_j(x) \leq \epsilon_j, \forall j \neq i$$

De cette manière, un problème simple monobjectif sous contraintes peut être résolu. Le décideur peut ensuite réitérer ce processus sur un objectif différent jusqu'à ce qu'il trouve une solution satisfaisante.

1.4 Discussion

Les méthodes de résolution "classiques" présentent un certain nombre d'inconvénients. Tout d'abord, elles nécessitent le choix des paramètres qui supposent la disposition à priori de certaines informations liées au problème : une connaissance plus profonde de la

nature du problème est souvent nécessaire afin de prévoir comment tel ou tel choix des paramètres va influencer le résultat. Ce problème est lié au fait que ces approches ne fournissent qu'une seule solution, qui dépend fortement des paramètres choisis au départ.

Mais, même si plusieurs essais sont faits avec des paramètres différents, on ne peut pas être sûre d'explorer l'ensemble de l'espace de recherche efficacement. Une des raisons est l'incapabilité de trouver les régions non-convexes de la surface des compromis lors de l'application de certaines méthodes (par exemple, la méthode d'agrégation pondérée). En plus, vu que les critères de l'optimisation sont nombreux, de trés nombreux essais sont nécessaires pour approcher la surface de Pareto multi-dimensionnelle.

Dans la pratique, il est en général intéressant d'exploiter tous les moyens disponibles pour tirer le maximum du profit de l'étape de l'optimisation. Le champs d'application de nouvelles techniques d'optimisation multi-objectif plus performantes et plus coûteuses, telles que les algorithmes évolutionnaires, s'élargit au fur et à mesure que les experts de divers domaines d'applications se rendent compte de tous les avantages que ces approches peuvent leur apporter. Dans le chapitre 2, nous illustrons le principe de fonctionnement des algorithmes évolutionnaires. Ensuite, nous présentons dans le chapitre 3 comment ces algorithmes peuvent être adaptés pour résoudre un problème multi-objectif.

CHAPITRE 2

PROCESSUS D'ÉVOLUTION

2.1 Introduction [28]

Les algorithmes évolutionnaires (**AE**) sont des algorithmes stochastiques d'optimisation inspirés du paradigme darwinien de l'évolution naturelle. Selon la théorie darwinienne, les individus les plus aptes survivent à la sélection naturelle et se reproduisent d'une génération à l'autre. En termes d'optimisation, l'évolution se traduit par un processus itératif de recherche de l'optimum dans l'espace de recherche. Il en existe plusieurs variantes suivant le schéma général suivant :

1. Initialisation : création d'une population aléatoire ou suivant l'expérience du concepteur,
2. Sélection d'une population de parents,
3. Évolution des parents donnant une population d'enfants,
4. Sélection de la nouvelle génération à partir de la génération précédente et des enfants,
5. Revenir à l'étape 2 tant que le critère d'arrêt n'est pas atteint.

Les différences entre les algorithmes évolutionnaires vont apparaître dans le choix des opérations d'initialisation, de sélection et d'évolution, ainsi que dans le codage des individus. Il existe plusieurs familles historiques d'algorithme évolutionnaires qui se sont développées de façon indépendante et peuvent être présentées comme suit:

Algorithmes Génétiques (GA)

Sont apparus dans les années 60 et leur invention est attribuée à **Holland (1962)** [15] popularisée par **Goldberg (1989)** [6]. Les individus d'un algorithme génétique sont historiquement codés sous la forme d'une chaîne de bits. Deux opérations d'évolution sur

ces individus sont effectuées à chaque génération : le croisement, qui à partir de deux parents, crée un ou plusieurs enfants partageant une partie du code des deux parents (une sous-chaîne de bits) ; la mutation, qui consiste en la modification aléatoire d'un ou plusieurs bits sur un individu.

Stratégies d'Evolution (ES)

Ont été développées parallèlement aux algorithmes génétiques, pour résoudre des problèmes numériques d'optimisation dans l'espace des paramètres réels. Leur apparition est attribuée à **Rechenberg (1965)** [14].

* Les principales différences entre les premières stratégies d'évolution et les premiers algorithmes génétiques sont, d'une part, que les ES utilisent un codage réel tandis que les GA utilisent un codage binaire, et d'autre part que les ES n'ont pas d'opérateur de croisement : ils n'effectuent que des mutations. Ces différences tendent cependant à s'estomper puisqu'il existe maintenant des ES avec des opérateurs de croisement et des GA à codage réel.

Programmation Evolutionnaire (EP)

Développée par **L.J. Fogel** [23], se base sur l'évolution d'une population d'automates à états finis, et est utilisée pour la résolution des problèmes de prédiction. La table de transition des automates est modifiée grâce à des mutations aléatoires uniformes dans l'alphabet discret correspondant. L'évaluation de la performance des individus correspond au nombre de symboles prédits correctement. Chaque automate de la population parente génère un enfant par mutation, et les meilleures solutions entre les parents et les enfants sont sélectionnées pour survivre.

Programmation Génétique (GP)

La première utilisation des structures arborescentes dans un algorithme génétique a été suggérée par **Cramer 1985** [26], dans le but de faire évoluer des sous-programmes séquentiels d'un langage algorithmique simple.

L'adoption de cette présentation pour définir la programmation génétique comme un nouvel algorithme évolutionnaire a été faite par **John Koza 1992** [16].

2.2 Vocabulaire et Terminologie

Nous présentons dans cette section le vocabulaire spécifique aux algorithmes évolutionnaires, inspiré du parallèle réalisé avec les principes de l'évolution naturelle.

- Les points de l'espace de recherche D sont appelés des *individus*.
- Un ensemble fini d'individus est appelé *population*.
- La fonction objectif à optimiser est appelée fonction performance, ou fonction *fitness*.
- Le calcul de la performance d'un individu est appelé *évaluation*.
- La *génération* correspond à une population en une certaine itération.
- L'*évolution* est un processus itératif de recherche des individus optimaux.
- Les *opérateurs de variation* sont utilisés pour générer de nouveaux individus et sont le plus souvent catégorisés en deux types d'opérateurs : le *croisement* qui consiste à échanger des parties composantes (gènes) entre deux ou plusieurs individus, et la *mutation* qui consiste à la modification d'un ou plusieurs gènes d'un individu.
- La *sélection* est le processus de choix des individus, basé sur leur performance.
- Le *remplacement* est le processus de formation d'une nouvelle population à partir des parents et des enfants.

2.3 Principe d'un algorithme évolutionnaire

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

À noter que dans la plupart des applications réelles, le coût de calcul des algorithmes évolutionnaires provient essentiellement de l'étape d'évaluation. A titre d'exemple, si l'on

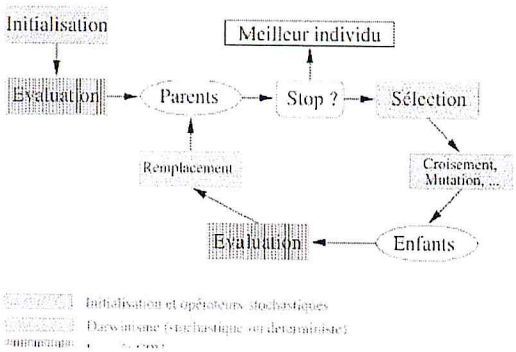


FIGURE 2.1. Principe de fonctionnement d'un AE. [27]

souhaite faire évoluer une population de quelques dizaine d'individus pendant quelques dizaine de générations, quelques milliers de calculs de la fonction performance (ou fitness) doivent être réalisés souvent à travers des évaluations coûteuses.

Schéma général d'un algorithme évolutionnaire [32]

1. Initialisation:
   - Initialiser le compteur des génération $t = 0$
   - Initialiser aléatoirement une population $P(t)$ de taille fixée N
   - Evaluer chaque individu de $P(t)$
2. Sélectionner les individus les plus performants de la population, les recopier pour former une nouvelle population de même taille
3. Créer une nouvelle population en appliquant les opérateurs de variation:
   - Opérateur de croisement
   - Opérateur de mutation
4. Evaluer la *fitness* de chaque individu de la nouvelle population
5. Remplacer certains individus de l'ancienne population par les meilleurs individus de la nouvelle population
6. Si la condition d'arrêt n'est pas vérifiée, aller à l'étape 2, sinon retourner le meilleur individu de $P(t)$

2.4 Génotype : codage des individus et opérateurs génétiques

Les algorithmes évolutionnaires utilisent l'analogie avec les lois de la génomique. On distingue le codage d'une solution (le gène - le génotype) avec sa valeur (l'expression du gène - le phénotype). Les différentes opérations sont réalisées sur le génotype et les individus sont évalués sur leur phénotype. Dans notre modélisation des problèmes d'optimisation, le génotype appartient à l'espace de décision X tandis que le phénotype est dans l'espace des objectifs Y.

2.4.1 Codage des individus

Historiquement, les variables des algorithmes génétiques sont représentées par des chaînes de bits. Ces chaînes peuvent être interprétées de différentes manières. Nous pre nons l'exemple de leur interprétation comme nombres binaires et comme code de Gray. Une alternative est de simplement les coder dans leur représentation réelle.

Codage binaire

Le codage binaire est le plus simple et le plus répandu. Il simplifie les opérateurs de croisement et de mutation. Chaque variable est transformée en une suite de 0 et de 1. Le nombre de bits utilisés dépend évidemment du nombre de valeurs que l'on doit représenter et donc il faut fournir des bornes aux variables ainsi qu'une précision souhaitée.

Codage de Gray

Un code de Gray est un type de codage binaire tel que le passage d'un nombre au suivant ne se fasse qu'en modifiant un seul bit.

base 10	base 2	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111

Codage réel

Le codage réel est le plus simple puisque chaque variable est directement représentée par sa valeur. Un réel est alors représenté par un nombre flottant, un entier par un entier, etc. Le codage réel est mieux adapté aux problèmes d'optimisation continus. Cependant, les opérateurs de croisement et de mutation sont plus délicats à écrire.

Codage spécifique

Dans les problèmes d'optimisation combinatoires, il convient souvent d'utiliser un codage spécifique à la structure du problème. Par exemple, dans le problème de voyageur de commerce, un codage naturel d'une solution est d'utiliser une liste de villes qui représentera l'ordre de parcours. Dans ce cas, le codage est une permutation.

Suivant le codage utilisé, différents opérateurs de croisement et de mutation sont définis.

2.4.2 Croisement

L'opérateur de croisement consiste à partir de deux solutions appelées parents, à créer de nouvelles solutions (généralement deux) appelées enfants. Les nouvelles solutions ainsi créées partagent, en théorie, des caractéristiques des deux parents. Les parents ayant été sélectionnés sur des critères de performance, on espère ainsi créer des enfants combinant les bonnes caractéristiques de chacun des parents.

Un exemple de croisement est le suivant pour le codage binaire:

Croisement en un point

Le croisement en un point est l'opérateur de croisement le plus courant et le plus simple. Il consiste à croiser deux parents qui sont représentés par une chaîne binaire en choisissant un point de croisement entre le début et la fin de la chaîne puis en échangeant, d'un parent à l'autre, la partie située au-delà de ce point.

Le point de croisement est choisi aléatoirement. Évidemment, on peut généraliser ce croisement à plusieurs points.

Exemple de croisement binaire en un point entre deux parents.

2.4.3 Mutation [9]

L'opérateur de mutation consiste à modifier une solution pour créer une nouvelle solution qui lui soit proche. Cela permet, d'une part, de diversifier la population et, d'autre part, d'optimiser localement la population (si la solution mutée est meilleure que la solution d'origine).

Bascule

L'idée est simple et consiste simplement, pour une probabilité p_m de mutation donnée, à parcourir la chaîne binaire codant l'individu et à changer le 0 en 1 ou le 1 en 0, si le nombre tiré aléatoirement est inférieur à p.

Mutation distribuée normalement

Pour un codage réel, un opérateur de mutation simple à mettre en oeuvre est, pour chaque variable, de tirer une nouvelle valeur x' suivant une loi normale de probabilité distribuée autour de la valeur de la variable x.

$$x' = x + y$$

où $y \in \aleph(0, \sigma)$ est une valeur générée avec la loi normale de moyenne 0 et de variance σ déterminée par l'utilisateur.

Mutation polynomiale

La mutation polynomiale, est très proche de la mutation normale. On obtient une nouvelle solution x' par mutation de la solution x, en faisant muter chacune de ses composantes par la formule suivante:

$$x'_i = x_i + \Delta_{\max}\delta_q$$

où $\Delta_{\max}$ est la variation utilisée et δ_q le coefficient de variation, calculé de la manière suivante:

$$\delta_q = \left\{ \begin{array}{ll} (2\mu)^{\frac{1}{\eta m+1}} - 1 & \text{si } \mu \leq 0,5 \\ 1 - (2(1-\mu))^{\frac{1}{\eta_m+1}} & \text{sinon} \end{array} \right\}$$

où μ est un nombre aléatoire tiré uniformément entre 0 et 1, et η_m l'indice de distribution.

2.5 Phénotype : évaluation et sélection des individus

Nous avons vu que le génotype était la représentation de l'espace de décision. Sa structure est liée au codage des variables et aux opérations génétiques définies sur cet espace.

Le phénotype est l'expression du génotype. Il s'agit donc ici d'un ensemble, Y image de X par la fonction objectif f, dans lequel sont évaluées les performances des individus.

2.5.1 Évaluation des individus

Les individus sont évalués et comparés entre eux suivant leur adaptation (fitness en anglais). La mesure d'adaptation peut être différente de la mesure de performance des individus. Dans notre travail, la performance est donnée par la fonction objectif f. De plus, si l'on souhaite obtenir plusieurs solutions, il faut à la fois optimiser les performances, mais également prendre en compte la diversité des solutions dans la fonction d'adaptation.

2.5.2 Sélection

Des individus sont sélectionnés pour se croiser et former les individus de la génération suivante. La méthode la plus simple est évidemment de prendre au hasard les individus dans la population. Les mécanismes de sélection privilégient les meilleurs individus en leur donnant plus de chances de se reproduire que les autres et de se reproduire un plus grand nombre de fois.

Les algorithmes évolutionnaires comportent deux phases de sélection. La première sélectionne les individus pour constituer la population de parents. La seconde sélectionne les individus parmi les parents et les enfants qui feront partie de la génération suivante.

La sélection par tournois

Consiste à confronter deux solutions entre elles. La meilleure est retenue dans la population de reproduction. Si le tournoi est réalisé de façon systématique sur l'ensemble de la population, chaque individu peut participer à exactement deux tournois et peut ainsi être présent en deux, un ou zéro exemplaire dans la population de parents.

La sélection proportionnelle

Consiste à attribuer à chaque individu une probabilité d'être sélectionné en fonction de ses performances. Meilleur est l'individu, plus il a de chances d'être sélectionné. La sélection par roue de fontune est une de ces méthodes. On représente une roulette de casino abstraite divisée en autant de portions qu'il y a d'individus et où la taille des portions dépend des scores des individus. Un nombre est alors tiré aléatoirement (la bille de la roulette).

La méthode de classement

Consiste à trier les solutions suivant leur score et leur attribuer un rang. Une sélection proportionnelle est ensuite effectuée sur la population, mais en prenant en compte les rangs et non les scores.

2.6 Elitisme

Définition 2.6.1. *(élitisme)*

*Soit P^t une population d'un algorithme évolutionnaire donné après t itérations (générations). Soit $P^t(x)$ la probabilité qu'a un individu $x \in P^t$ d'être sélectionné pour une phase de croisement et/ou mutation à la génération t. Alors l'algorithme génétique est dit "**élitiste**" si et seulement si, pour une relation de préférence $\prec$ donnée dans un problème de décision, la condition suivante est vérifiée :*

$\forall t \in \mathbb{N}, \exists x \in P^{*t}$ *tel que* $P^{t+1}(x) > 0$

avec:

$P^{*t} = \{x \in P^t | \nexists x' \in P^t | x' \prec x\}$

$P^t = \cup_{r \leq t} P^r$

L'opération $\cup_{r \leq t} P^r$ désigne l'union de tous les ensembles P^r avec $r \leq t$.

P^{*t} désigne donc tous les individus non dominés de P^t.

L'idée générale qui ressort de ces réflexions sur l'élitisme est qu'il est important de ne pas oublier de conserver, d'une manière ou d'une autre, les bonnes solutions obtenues génération après génération.

En optimisation monobjectif, on a pour habitude de sauvegarder la meilleure solution obtenue au cours du processus d'optimisation. En optimisation multiobjectif, un processus

similaire consiste à placer dans une archive tous les individus non dominés obtenus en cours d'optimisation. Cette archive représente à tout moment la meilleure approximation de la surface de compromis que l'on a obtenue jusqu'ici. De plus, comme cette archive ne sert qu'à sauvegarder les individus non dominés, elle ne modifie pas le comportement de l'algorithme évolutionnaire.

2.7 Maintenir la diversité

Les techniques de maintenance de diversité des individus dans la population sont employées afin de permettre à un EA:

- d'évité la convergence prématurée vers un optimum local,

- d'identifier plusieurs optima ou quasi-optima quand la fonction objectif est multimodale.

Dans la résolution de Problèmes multiobjectifs, il est nécessaire que les solutions trouvées soient Pareto optimales, mais aussi qu'elles soient uniformément réparties dans le sous-espace des solutions Pareto optimales. Pour maintenir une diversité dans la population, Plusieurs approches visant à maintenir la diversité dans la population ont été proposées dans la littérature : crowding, restriction de voisinage, niches écologiques (sharing). Cependant, ces techniques ajoutent un coût calculatoire non négligeable pour l'algorithme, elles doivent donc être choisies avec soin.

2.7.1 Partage (*Sharing*)

Le partage explicite de la performance a été introduit par Goldberg et Richardson [7] en 1987. L'idée consiste à dégrader la valeur de la performance de l'individu en fonction de la densité de la population dans son voisinage. Le but principal est de pénaliser les individus qui sont trop proches les uns des autres.Ainsi, la performance partagée (shared fitness) F' d'un individu x_i est donnée par:

$$F' = \frac{F(x_i)}{\sum_{j=1}^{n} sh(d(x_i, x_j))}$$

où sh désigne la fonction de partage (sharing function) qui dépend de la densité d entre deux solutions. Elle retourne '1' si les deux solutions sont identiques, '0' si la distance entre elles dépasse un certain seuil (σ_{share}) et une valeur intermédiaire pour des degrés de dissimilarité intermédiaire. La fonction de partage la plus utilisée est:

$$sh(d) = \left\{ \begin{array}{ll} 1 - (\frac{d}{\sigma_{share}})^{\alpha}, & \text{si } d \prec \sigma_{share} \\ 0 & \text{sinon} \end{array} \right\}$$

où α est une constante utilisée pour contrôler l'allure de la fonction de partage. Le calcul de la distance entre deux individus peut se faire dans l'espace génotypique (e.g. distance de Hamming) ou phénotypique (e.g. distance Euclidienne), selon le problème traité. Cependant, des études expérimentales faites par Deb et Goldberg en 1989 sur les mêmes testsont montré que le partage dans l'espace phénotypique donne des résultats légèrement meilleurs.

La procédure de partage classique a une complixité élevée ($\mathbf{O(N^2)}$), vu qu'elle nécissite une coparaison deux à deux de tous les individus de la population. Afin de réduire cette complixité, Yin et Germay en 1993, ont proposé de classifier la population en niches avant la procédure de partage. Cette dernière est appliquée ensuite sur chaque niche, ce qui réduit la complixité à $\mathbf{O(N \log (N))}$.

2.7.2 Distance de surpeuplement (Crowding distance)

Holland a été le premier à suggérer l'utilisation de l'opérateur de "crowding" dans la phase de remplacement des algorithmes génétiques (Holland, 1975) [15], pour identifier les situations dans lesquelles de plus en plus d'individus dominent les niches écologiques. Dans la reproduction d'un nouvel individu, l'opérateur consiste à remplacer l'individu existant le plus semblable à l'individu généré, et non pas les parents comme dans les algorithmes génétiques standard.

La technique (crowding distance) utilisée par le NSGAII [20],pour préserver la diversité des solutions sur le front Pareto, s'applique sur le dernier front pour compléter la taille de la population parent pour la génération suivante. L'opérateur de comparaison crowded ($<_n$) guide le processus de la sélection avec la répartition uniforme des solutions Pareto.

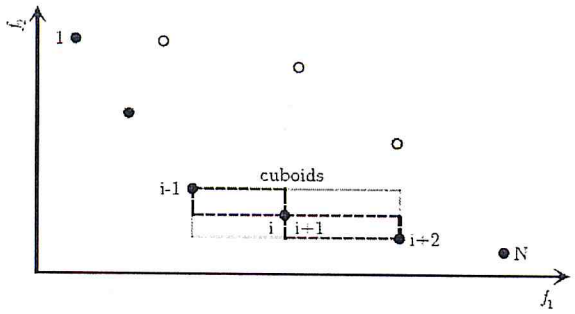


FIGURE 2.2. Exemple du calcul de la distance de crowding. [20]

Un individu i de la population a deux attributs : rang de non domination i_{rank} et distance crowding $i_{distance}$.

Soit deux individus i et j, $i <_n j$ si $i_{rank} < j_{rank}$ ou $(i_{rank} = j_{rank})$ et $(i_{distance} > j_{distance})$. Avec cette relation pour la comparaison de deux solutions non dominées appartenant à deux fronts Pareto, on préfère la solution appartenant au front Pareto d'ordre le plus faible. Sinon, dans le cas où les deux solutions appartenant au même front de Pareto, on choisit la solution qui a la distance crowded la plus élevée.

Procedure Crowding Distance($\mathcal{F}$) [20]

$N = |\mathcal{F}|$

for $i = 1....N$ do

$\mathcal{F}[\mathbf{i}]_{\mathbf{dist}} = 0$

end for

for $m = 1....N$ do

$Sort(\mathcal{F}, \mathbf{m})$

$\mathcal{F}[\mathbf{1}]_{\mathbf{dist}} = \mathcal{F}[\mathbf{N}]_{\mathbf{dist}} = \infty$

for $i = 2....N - 1$ do

$\mathcal{F}[\mathbf{i}]_{\mathbf{dist}} = \mathcal{F}[\mathbf{i}]_{\mathbf{dist}} + \frac{\mathcal{F}[\mathbf{i+1}].m - \mathcal{F}[\mathbf{i-1}].m}{f_m^{max} - f_m^{min}}$

end for

end for

end procedure

2.8 Evaluation des performances

Dans tout problème d'optimisation après la modélisation et le développement d'un algorithme de résolution, vient la phase d'évaluation de la qualité des solutions produites.

La comparaison d'algorithmes pour la résolution exacte des problèmes d'optimisation est triviale. Dans le cas des méthodes approximatives, la comparaison reste triviale pour l'optimisation mono-objectif, mais pas pour l'optimisation multiobjectif. En effet l'existence d'un ensemble de solutions de compromis et l'absence d'ordre total entre les solutions rendent la mesure de qualité d'un front difficile. Si la notion de dominance au sens de Pareto peut être utilisée pour comparer deux solutions, bien que ces deux solutions puissent être incomparables, la comparaison d'un ensemble de solutions est encore plus délicate. L'évaluation d'un algorithme en terme de qualité des solutions obtenues, nécessite soit de pouvoir évaluer un front (métriques absolues) soit de le comparer de façon quantitative avec les fronts produits par d'autres algorithmes (métriques relatives). Malheureusement cette tâche est délicate puisque la notion de qualité d'un front est elle-même multiobjectif (intensification, diversification). De nombreux indicateurs de performances ont été proposés dans la littérature.

Dans cette section, on désigne par PO^* l'ensemble des solutions potentiellement Pareto optimales trouvé par un algorithme, et par PO l'ensemble des solutions Pareto optimales d'un MOP.

2.8.1 Ensemble PO connu

Lorsque l'ensemble PO est connu, les mesures calculent le plus souvent la distance entre l'approximation PO^* et PO.

Proportion d'erreur

Cette mesure compte le nombre de solutions PO^* qui n'appartiennent pas à PO, soit : $ER = \frac{\sum_{i=1}^{|PO^*|} e_i}{|PO^*|}$ où $e_i = 1$ si la $i^{ème}$ solution de $PO^* \in PO$, sinon $e_i = 0$.

Le désavantage de cette méthode est que si aucune solution de PO^* n'appartient à PO, elle n'apporte aucune information au sujet de la proximité relative de PO^* par rapport à PO puisque dans ce cas, quelque soit la distance séparant PO^* de PO, on a $ER = 0$.

Distance générationnelle

Cette mesure calcule la distance moyenne entre les solutions de PO^* et celles de PO. Elle se calcule selon la formule suivante :

$GD = \frac{(\sum_{i=1}^{|PO^*|} d_i^p)^{\frac{1}{p}}}{|PO^*|}$

Pour $p = 2$, le paramètre d_i est la distance Euclidienne (dans l'espace des objectifs) entre la solution $i \in PO^*$ et le membre le plus proche de PO.

$d_i = \min_{k=1}^{|PO^*|} \sqrt{\sum_{j=1}^{p}(f_j^i - f_j^k)^2}$ où f_j^i est la valeur de la $j^{ème}$ fonction objectif de i, et f_j^k est la valeur de la $j^{ème}$ fonction objectif de la $k^{ème}$ solution de PO.

La difficulté avec cette méthode est que, s'il existe un ensemble PO* pour lequel il y a une fluctuation importante dans la distance, la métrique peut ne pas retourner la véritable distance. Dans un tel cas, le calcul de l'écart type de la mesure est nécessaire.

2.8.2 Ensemble PO inconnu

Lorsque l'ensemble PO est inconnu, les mesures permettent le plus souvent de comparer de manière relative deux approximations. Il existe cependant des mesures qui renvoient une évaluation de la qualité d'une seule approximation. Les mesures évaluent la qualité des approximations soit par rapport à la convergence, soit par rapport à la diversification, ou par rapport aux deux buts en même temps.

Mesures évaluant la convergence

La métrique C:

Cette mesure, calcule la proportion de solutions d'un ensemble potentiellement Pareto optimal PO_B^* dominées par des solutions d'un ensemble PO_A^*:

$C(A,B) = \frac{|\{b \in PO_B^* / \exists a \in PO_A^* : a \succ b\}|}{|PO_B^*|}$

$C(A,B) = 1$ signifie que toutes les solutions trouvées par l'algorithme B sont dominées par celles trouvées par l'algorithme A. Tandis que $C(A,B) = 0$ indique qu'aucune solution générée par l'algorithme B n'est dominée par une solution trouvée par l'algorithme A. comme la relation de dominance n'est symétrique, $C(B,A)$ n'est pas forcément égal à $1 - C(A,B)$. Il est donc nécessaire de calculer $C(A,B)$ et $C(B,A)$.

Mesures évaluant la diversité

a) La métrique d'espacement:

Cette métrique, calcule la distance relative entre deux solutions consécutives de PO_A^*:

$S = \sqrt{\frac{1}{|PO_A^*|}\sum_{j=1}^{|PO_A^*|}(d_i - \overline{d})^2}$ où $d_i = \min_{k \in PO_A^* \wedge k \neq i} \sum_{m=1}^{n} |f_m^i - f_m^k|$ et $\overline{d} = \frac{\sum_{j=1}^{|PO_A^*|} d_i}{|PO_A^*|}$.

La distance d_i est la valeur minimale de la somme des différences absolues des valeurs des fonctions objectifs entre la $i^{ème}$ solution et toutes les autres solutions de l'ensemble généré. Il est noter que cette distance est différente de la distance Euclidienne minimale entre deux solutions. Cette métrique calcule les écarts type des différentes valeurs de d_i. Ainsi, si les solutions sont uniformément espacées, la distance correspondante sera faible. Donc, plus un algorithme trouve un ensemble de solutions pour lequel cette mesure est faible, meilleur il est.

b) Métrique maximum spread

métrique mesurant la longueur de la diagonale d'une « hyper boîte » formée par les valeurs des fonctions objectifs extrêmes de l'ensemble potentiellement Pareto optimal généré:

$D = \sqrt{\frac{1}{|PO_A^*|}\sum_{m=1}^{n}(\frac{\max_{i=1}^{|PO_A^*|} f_m^i - \min_{i=1}^{|PO_A^*|} f_m^i}{F_m^{\max} - F_m^{\min}})^2}$

Où $F_m^{\max}$ et $F_m^{\min}$ sont le maximum et le minimum pour le $m^{ème}$ objectif. Le problème de cette mesure est qu'elle ne fournit aucune information sur la distribution exacte des solutions de compromis.

Conclusion sur les mesures d'évaluation des performances

Un ensemble de mesures ont été proposées pour analyser les performances des algorithmes multiobjectifs. Pour une bonne analyse, différentes mesures doivent être utilisées afin de pouvoir analyser à la fois convergence et diversité. L'analyse de performances en multiobjectif est en elle seule un domaine d'étude encore très ouvert puisqu'il n'existe pas de mesure universellement utilisée. Il est important de retenir qu'aucune des mesures existantes ne peut synthétiser en une seule valeur toute l'information contenue dans la surface de compromis. C'est pourquoi il est souvent nécessaire d'utiliser plusieurs de ces mesures conjointement pour espérer évaluer au mieux la surface de compromis.

CHAPITRE 3

HYBRIDATION MULTI-OPÉRATEUR DE CROISEMENT DANS LES ALGORITHMES ÉVOLUTIONNAIRES MULTI-OBJECTIF

Dans ce chapitre, nous proposons un algorithme que nous avos nommé **"Hybrid of Multi-operator Crossover for Multiobjectif Evolutionary Algorithms (HMC-MOEA)"**, qui se base sur une hybridation de cinq opérateurs de croisements. *Nous validons ensuite cette approche et nous comparons les différents types de croisement en se basant sur une analyse d'une série d'expériences indépendantes.*

Cette métaheuristique fait générer des variables aléatoires, à cet effet on commence par une vue générale sur la simulation des variables aléatoires, puis on passe à la présentation de notre algorithme.

3.1 Simulation de variables aléatoires [4]

Il existe des algorithmes qui génèrent des suites de tirages pseudo-aléatoires indépendants de loi $U(0,1)$ uniforme sur $[0,1]$. La plupart des calculettes permettent d'exécuter de tels programmes, souvent baptisés $rand1$. En général leur conception repose sur des propriétés arithmétiques de certaines suites récurrentes. Ces algorithmes sont déterministes, c'est-à-dire qu'il n'ont rien d'aléatoire. Si vous utilisez le même algorithme avec la même donnée initiale, il vous donnera toujours la même suite de nombres. De plus, ces suites de tirages de valeurs numériques sont périodiques, mais avec une période extrêmement grande. C'est la raison pour laquelle ces générateurs sont appelés pseudo-aléatoires plutôt qu'aléatoires.

Description de certains générateurs

Une famille de générateurs populaire est celle des générateurs congruentiels linéaires. Ils génèrent des suites de nombres entiers $(x_n)_{n\geq 1}$ dans l'ensemble $\{0,...,m-1\}$ où m est

un grand nombre. Il suffit ensuite de prendre $u_n = x_n/m$ pour obtenir une suite de tirages $(u_n)_{n\geq 1}$ dans $[0, 1[$ dont les valeurs sont des nombres arrondis avec une précision de l'ordre de $1/m$. La suite $(x_n)_{n\geq 1}$ est solution de l'équation de récurrence:

$$x_{n+1} = ax_n + b \bmod m, \qquad m, n \geq 0$$

en partant d'une donnée initiale entière x_0. On rappelle que $x = r$ modulo m signifie que r est le reste de la division euclidienne de x par m. En d'autres termes $x = qm + r$ avec un quotient q entier et$0 \leq r \leq m - 1$. On constate immédiatement qu'une telle suite est périodique (de période au plus m). Il faut donc que m soit très grand. En choisissant intelligemment a et b, cette période est effectivement m. D'autre part il faut aussi choisir adéquatement les nombres a, b et m pour que la suite simule correctement de très longues séquences (de l'ordre de $m/10$) de tirages uniformes et indépendants. En fait, le choix de ces paramètres est loin d'être évident et est encore l'objet de recherche. La fonction **rand** de **Scilab** utilise les valeurs $m = 231$, $a = 843314861$ et $b = 453816693$. La fonction **grand** de *Scilab* est basée sur un type de générateur déterministe plus performant dont la période $219937 - 1$ est fabuleuse. La plupart des générateurs utilise la date et l'heure de l'ordinateur pour décider de la valeur initiale x_0.

Principe général de la simulation

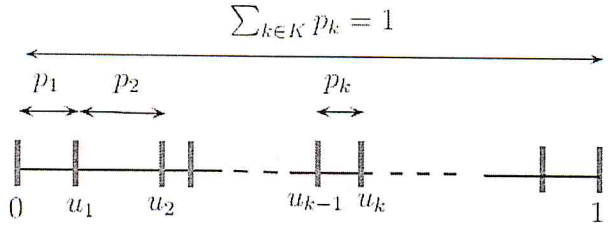
Soit $U_1, U2, ...$ un échantillon de la loi uniforme U(0, 1). On sait que, F^{-1} désignant l'inverse généralisé de la fonction de répartition F de la loi de X

$$X_i = F^{-1}(U), \quad i \geq 1$$

définit un échantillon de la loi de X. C'est-à-dire une famille de copies indépendantes de X. Ce principe s'applique donc lorsqu'on connaît une expression de F^{-1}.

Dans le cas d'une variable discrète, le principe précédent correspond à une manipulation intuitivement claire que nous allons décrire. La méthode est simple. La variable discrète X que nous souhaitons simuler prend ses valeurs dans $\{x_k; k \in K\}$ avec $K \subset \{1, 2, ...\}$. Sa loi s'écrit $\sum_{k\in K} p_k \delta_{x_k}$. On suppose sans perte de généralité que $p_k > 0$ pout tout k.

On partitionne l'intervalle $]0, 1]$ de sorte que



$$]0,1] = \underset{k\in K}{\cup}]u_{k-1}, u_k]$$

avec $u_0 = 0$ et $u_k = \sum_{1\leq i\leq k} p_i$, $k \in K$. La probabilité que la variable U de loi uniforme sur $(0,1)$ tombe dans k-ième boîte $B_k =]u_{k-1}, u_k]$ est

$$P(U \in B_k) = P(u_{k-1} < U \leq u_k) = u_k - u_{k-1} = p_k, \quad k \in K.$$

La variable

$$X = \sum_{k\in K} x_k \mathbf{1}_{\{U\in B_k\}}$$

qui vaut x_k si et seulement si $U \in B_k$, $k \in K$ a pour loi $\sum_{k\in K} p_k \delta_{x_k}$.

3.2 Présentation de l'algorithme (HMC-MOEA)

Cet algorithme est une modification de l'algorithme proposé par Lixing Tang et Xianpeng Wang dans [24]. La différence de notre algorithme avec ce dernier réside dans la procédure de la mise à jour de la population et cela afin de remédier du problème de la convergence prématurée, pour le cas des problèmes discontinus, par exemple ZDT3.

L'hybridation dans cet algorithme réside dans le fait d'utiliser plusieurs opérateurs de croisement, cette utilisation est contrôlée de manière stochastique par un dispositif qui favorise la convergence.

Fonctionnement et schéma général de l'algorithme HMC-MOEA:

Initialement, on commence par la génération d'une population initiale à l'aide d'un processus utlisant l'idée de partitionnement en boites citée dans la section précédente et

qui réspecte les contraintes des bornes supérieures et inférieures des variables de décision, puis on divise cette population selon leurs efficacité dans le premier objectif qui sera l'axe des abssices dans l'espace de recherche. Cette division se fait avec l'affectation des individus dans un sou ensemble nommé $Pinf$ et qui on attribue l'efficacité selon le premier objectif inférieure à $\alpha = \frac{UB-LB}{2}$ où UB et LB sont les bornes supérieures et inférieures respéctivement d'une variable de décision, et le reste des individus de la population seront affectés dans un autre sous ensemble nommé Psup. On garde les individus non dominés au sens de Pareto dans une archive externe qui s'appelle EXA (External archive).

L'évolution de cet algorithme consiste à faire des mis à jous des 4 ensembles créés initialement (population initiale, Pinf, Psup et EXA) en faisant intervenir des opérateurs génétiques (croisement et mutation), et le schéma général de fonctionnement est présenté dans l'algorithme suivant:

Algorithm: Main Procedure of HMC-MOEA

BEGIN

1. Mettre le critère d'arrêt et les tailles des ensembles la population (n_pop), EXA (n_exa), Pinf, Psup, la probailité de mutation et les paramètres de contrôle de chaque opérateurs de croisement.
2. Créer l'ensemble EXA vide.
3. Générer la population initiale par $\boldsymbol{Population\_Generation\_Method}$ décrit dans (section III.3.1)
4. Remplir les ensembles Pinf and Psup comme dans la section III.2.
5. Stocker les solutions non dominées de la population dans EXA.

While (le critère d'arrêt n'est pas satisfait) **do**

1. Repeupler EXA par $\boldsymbol{EXA\_popagating\_Mechanism}$ comme dans la section (III.3.2)
2. Mettre à jour la population par $\boldsymbol{Population\_Update\_Mechanism}$ section (III.3.3)
3. Muter la population par $\boldsymbol{Population\_Mutation}$ section (III.3.4)
4. Mettre à jour $Pinf$ et $Psup$ comme dans la section III.2.

For chaque solution non dominée dans population, mettre à jour EXA par $\boldsymbol{EXA\_Update\_strategy}$ décrit dans la section (III.3.5)

End For

End While

Retourner les solutions non dominées dans EXA.

END

3.3 Définitions des fonctions de l'algorithme (HMC-MOEA)

On va maintenant définir les différentes fonctions utilisées dans l'algorithme et citées dans la section précédente avec plus de détails en commençant par l'initialisation de la population et ensuite la mise à jour des autres ensembles:

3.3.1 Génération de la population initiale ($Population\_Generation\_Method$)

Pour obtenir une bonne diversité dans la population initiale, nous avons procédé comme dans [1] et [8]. Soit $X_i = [x_1^i, x_2^i, .., x_j^i, .., x_{n\_var}^i]^T$ le ième individu dans la population, où (i est l'index des solutions dans la population, et j est celle des variables de décision dans un individu). La procédure de génération est donnée comme suit:

***Etape*01** : Diviser chaque individu en n_sub subdivisions equivalentes. $j = 1$.

***Etape*02** : Si $j > n\_var$ Stop, sinon $i = 1$ et initialiser la probabilité de séléction de la subdivision k de la ième variable comme $p_k = \frac{1}{n\_sub}$.

***Etape*03** : Sélectionner une subdivision avec la roue de fortune, proportionnellement aux probablilités de séléction de chaque subdivision, et générer aléatoirement la valeur de x_j^i qui appartient à la subdivision séléctionnée.

***Etape*04** : Mettre à jour la probabilité de séléction de chaque subdivision k en suivant la procédure:

$$p_k = \left\{ \begin{array}{ll} p_k - 1/n\_pop & \text{si } k \text{ est séléctionnée} \\ p_k + 1/(n\_pop * (n\_sub - 1)) & \text{sinon} \end{array} \right\}$$

***Etape*05** : $i = i + 1$; **Si** $i > n\_pop$; $j = j + 1$ et aller à l'étape 02, **Sinon** aller à l'étape 03.

Méchanisme de choix de l'opérateur de croisement:

Comme il est mentionné plutôt, on a utilisé dans cet algorithme un méchanisme multi-opérateur pour le croisement des individus, et cette idée est tirée de [11]. Après la génération de la population initiale de taille n_pop, on assigne à chaque opérateur $n\_pop/5$ solutions dans la population ce qui rend ces opérateurs de croisement initialement equiprobables. Durant le processus d'évolution, quand un opérateur est séléctionné pour générer une nouvelle solution, l'index de cet opérateur sera assigné à cette nouvelle solution. Soit p_i la probabilité de séléction de chaque opérateur i (l'index $i = 1$ pour l'opérateur BLX_α, $i = 2$ pour SBX, $i = 3$ pour SPX, $i = 4$ pour PCX et $i = 5$ pour DE) et le méchanisme de choix est décrit comme suit:

***Etape*01** : Si l'ensemble EXA est actualisé, on calcule la probabilité de séléction de

chaque opérateur i $(i = 1, ..., 5)$ comme $p_i = \frac{a_i}{|EXA|}$, où a_i est l'ensemble de solutions qui ont l'index de l'opérateur assigné est i, $|EXA|$ est le cardinal de l'ensemble EXA courant.

***Etape*02 :** Mette à jour les probabilités de choix de chaque opérateur par: si $p_i < p_{\min}$ alors $p_i = p_{\min}$ et $p_j = p_j - (p_{min} - p_i)$, où p_j est la probabilité la plus grande, et $p_{\min}$ est une probabilité minimale adaptée pour éviter l'elimination d'un opérateur durant le procéssus d'évolution.

***Etape*03 :** Utiliser la roue de fortune pour séléction un opérateur.

3.3.2 Méchanisme de repeuplement de l'ensemble EXA ($EXA\_popagating\_Mechani$

Cette fonction permet de créer de nouvelles solutions fortement probable qu'elles soient nondominées grace aux opérateurs de croisement qui s'interviennent lors de son exécution, et la procédure est décrite comme suit:

Soit n_exa la taille maximale de l'ensemble EXA.

***Etape*01 :** Si la taille de l'archive externe $|EXA| = 2$ aller à l'étape 02, Sinon aller à l'étape 03.

Etape*02 :** Sélectionner aléatoirement une solution dans EXA, et perturber cette solution pour générer une nouvelle, cette perturbation se fait en seléctionnant aléatoirement une dimention dans la solution choisie par exemple i, et cette dimension sera une valeur générée aléatoirement dans l'intervalle $[LB_i, UB_i]$, avec LB_i et UB_i sont les bornes inférieure et superieure de la dimension i, répéter cette étape n_exa fois pour générer n_exa nouvelles solutions et utiliser la fonction ***EXA_Update_strategy décrite dans la page 53, pour mettre à jour l'ensemble EXA et Stop.

***Etape*03 :** Calculer la distance de crowding (voir section 2.7.2) de chaque solution j notée d_j dans EXA, et déterminer la probabilté de séléction de chaque solution j par $s_j = d_j / \sum_{k \in EXA} d_k$. Ce qui veut dire que la solution ayant la plus petite distance de crowding a la probabilité la plus faible à être sélectionnée pour générer une nouvelle solution.

***Etape*04 :** Utiliser le méchanisme de séléction pour séléctionner un opérateur i, Si $i \leq 2$ (càd l'opérateur sélectionné est BLX_α ou SBX) séléctionner aléatoirement deux solutions dans EXA, Sinon séléctionner trois solutions. La séléction des solutions de se fait proportionnellement aux probabilités déterminées à l'étape 03 en utilisant la roue de

wheel.

***Etape*05 :** Croiser les solutions choisies par l'opérateur choisi à l'étape 04 pour générer une nouvelle solution, si l'opérateur choisi est SBX d'où il peut générer deux solutions donc on prend la nondominée.

Etape*06 :** Répéter cette étape n_exa fois pour générer n_exa nouvelles solutions et utiliser la fonction ***EXA_Update_strategy décrite dans la section (3.3.5) pour mettre à jour l'ensemble EXA et Stop.

3.3.3 Actualisation de la population (*Population_Update_Mechanism*)

Dans cette section on va décrire l'idée qu'on a introduit dans cette méthode et qui se base sur la création de deux ensemble $Pinf$ et $Psup$ décrite dans la section (3.2). Après la génération de la population initiale et la détermination des deux sous ensembles $Pinf$ et $Psup$, on procède à la stratégie suivante à chaque fois qu'on a besoin d'actualiser la population:

Soit $N1$ (respectivement $N2$) le nombre de solutions dans EXA dont la valeur de leurs premier objectif est inférieure et (respéctivement supérieure) à $\alpha = \frac{UB-LB}{2}$. initialement on génère un vecteur de probabilité proportionnel aux entiers $N1$ et $N2$, $p = (1/N1+N2)*[0 \quad N1 \quad (N1+N2)]$

***Etape*01 :** Générer un nombre μ aléatoire uniforme dans $[0,1]$, **Si** $\mu \in [0, N1/(N1+N2)]$, donc pour les opérateurs BLX_α et SBX, les parents à croiser consiste sur une solution dans $Psup$ et une autre dans EXA, et pour les autres opérateurs les parents consistent sur deux solution dans $Psup$ et une dans EXA.

Sinon ($\mu \in [N1/(N1+N2), 1]$), refaire la même procédure en utilisant le sous ensemble $Pinf$.

***Etape*02 :** Répéter l'étape 01 jusqu'avoir n_pop solutions dans la population.

***Etape*03 :** Mettre à jour les deux sous ensembles $Pinf$ et $Psup$ par la nouvelle population.

NB: La partition de la population en deux sous ensembles $Pinf$ et $Psup$, assure la création de nouvelles solutions dans tout l'espace de recherche dans la phase de croisement.

et cela réside dans le méchanisme de choix des solutions condidates au croisement.

3.3.4 Fonction de mutation ($Population\_Mutation$)

Pour chaque dimension d'une solution dans la population, on génère un nombre aléatoire rnd dans $[0, 1]$, et si $rnd < p_m$ donc la mutation polynomiale décrite dans la section (2.4.3 page 37) est utilisée pour muter cette dimension. Comme il est décrit dans [22], $p_m = 1/n\_var$ (n_var est le nombre de variables de décision pour un problème donnée).

3.3.5 Mis à jour de l'ensemble EXA ($EXA\_Update\_strategy$)

Pour une solution nondominée i donnée de la population courante à litération t, la procédure de réduction de l'ensemble EXA est donnée comme suit:

***Etape*01 :** Si la solution i est dominée par une solution dans EXA, supprimer la solution i.

***Etape*02 :** Si la solution i n'est pas dominée par aucune solution dans EXA, stocker la dans EXA et supprimer toutes les solutions dominées par i.

***Etape*03 :** Si $|EXA| > n\_exa$ (la taille maximale de EXA), calculer la distance de crowding de chaque solution dans EXA, et supprimer celle ayant la plus petite distance. Répéter cette étape jusqu'à ce que $|EXA| = n\_exa$.

3.4 Opérateurs de Croisement

On va présenter dans cette section les différents opérateurs de croisement utilisés dans cet algorithme.

3.4.1 Blend alpha Crossover (BLX_α)

À partir de deux parents $X = (x_1, x_2, ..., x_n)$ et $Y = (y_1, y_2, ..., y_n)$, le croisement BLX_α [22] génère une nouvelle solution $Z = (z_1, z_2, ..., z_n)$ où $z_i \in [c_{\min} - I\alpha, c_{\max} + I\alpha]$, $c_{\max} = \max(x_i, y_i)$ et $c_{\min} = \min(x_i, y_i)$, $I = c_{\max} - c_{\min}$, α est une constante.

Pour la valeur de α, *Eshelman* et *Schaffer* ont utilisé $\alpha = 0.5$ [22]

3.4.2 Simulated Binary Crossover (SBX)

À partir de deux parents $X = (x_1, x_2, ..., x_n)$ et $Y = (y_1, y_2, ..., y_n)$, le croisement SBX génère deux solutions $Z_1 = (z_1^1, z_2^1, ..., z_n^1)$ et $Z_2 = (z_1^2, z_2^2, ..., z_n^2)$ avec la manière suivante [19]:

***Etape*01 :** Générer aléatoirement un nombre uniforme $\mu \in [0, 1]$

***Etape*02 :** Générer un nombre aléatoire β par:

$$\beta = \left\{ \begin{array}{ll} (2\mu)^{1/(1+\eta)} & \text{si } \mu \leq 0.5 \\ (1/2(1-\mu))^{1/(1+\eta)} & \text{sinon} \end{array} \right\}$$

avec η est l'indice de distribution pour l'opérateur SBX.

***Etape*03 :** Générer deux nouvelles solutions par:

$$\left\{ \begin{array}{l} z_i^1 = 0.5[(1+\beta)x_i + (1-\beta)y_i] \\ z_i^2 = 0.5[(1-\beta)x_i + (1+\beta)y_i] \end{array} \right\}$$

3.4.3 Simplex Crossover (SPX)

SPX est opérateur de croisement multi-parents pour algorithmes génétiques de codage réel [29]. Supposons que l'espace de recherche est $\mathbb{R}^n$, dans ce travail on utilise un espace de recherche $\mathbb{R}^3$ pour cet opérateur, et la procédure de génération est décrite comme suit:

***Etape*01 :** Choisir n+1 parents nommés $X_0, X_1, X_2, ..., X_n$.

***Etape*02 :** Déterminer le centre de gravité de ces n+1 parents par:

$$G = \sum_{i=0}^{n} X_i/n + 1.$$

***Etape*03 :** Générer n nombres aléatoires par l'équation suivante: $r_k = u^{1/(1+k)}$. avec $k = 0, 1, .., n-1$ et $u \in [0, 1]$

***Etape*04 :** Calculer Y_k et C_k par:

$$Y_k = G + \epsilon * (X_k - G) \quad \text{pour } k = 1, .., n-1$$

et

$$C_k = \left\{ \begin{array}{ll} 0 & \text{si } k = 1 \\ r_{k-1}(x_{k-1} - x_k + C_{k-1}) & \text{si } k = 2, ..., n-1 \end{array} \right\}$$

***Etape*05 :** générer la nouvelle solution $Z = x_n + C_n$.

N.B: Il y a une possibilité de définir le simplex générant les nouvelles solutions par l'opérateur SPX en divisant les individus parents en plusieurs subdivisions et refaire le même algorithme su-cité pour chaque subdivision, ce qui donne une bonne couverture du triangle de simplex et évite de générer des solutions proches durant le processus d'évolution et ça assure la diversité des nouvelles solutions générées [2].

3.4.4 Parent Centric Crossover (PCX)

PCX est aussi un opérateur multi-parents [17] et comme l'opérateur qui le précède on travaille toujours dans espace de recherche à 3-dimensions. La procédure de cet espace peut être décrite comme suit:

***Etape*01 :** Choisir μ individus nommés $X_1, X_2, ..., X_n$ et calculer leurs centre de gravité nommé G.

***Etape*02 :** Selectionner aléatoirement un individus nommé X_p parmi les μ parents, ensuite calculer son vecteur directeur $d_p = X_p - G$.

***Etape*03 :** Calculer la distance perpendiculaire D_i des deux vecteurs restants vers le vecteur directeur d_p, ensuite la moyenne des distances D_i notée $\overline{D}$.

***Etape*04 :** La nouvelle solution générée Y est calculée à partir de l'équation:

$$Y = X_p + \omega_\varepsilon . d_p + \sum_{i=1, i \neq p}^{\mu} (\omega_\eta . \overline{D} . e_i)$$

avec ω_ε et ω_η sont des valeurs normalement distribuée avec une moyenne nulle et variance σ_ε^2, σ_η^2 respectivement, et e_i 2 bases orthogonales des deux sous espaces perpendiculaires à d_p.

3.4.5 Differential Evolution (DE) (croisement DE)

L'opérateur DE a montré une bonne performance pour les problèmes test avec un front de pareto compliqué [10]. Dans l'implémentation, cet opérateur choisit initialement 3 parents nommés $X_1 = (x_1^1, x_2^1, ..., x_n^1)$, $X_2 = (x_1^2, x_2^2, ..., x_n^2)$ et $X_3 = (x_1^3, x_2^3, ..., x_n^3)$, d'où la nouvelle solution générée nommée $Y = (y_1, y_2, ..., y_n)$ est calculée par:

$$y_k = \left\{ \begin{array}{ll} x_k^1 + F.(x_k^2 - x_k^3) & \text{avec une probabilité } CR \\ x_k^1 & \text{avec une probabilité } 1 - CR \end{array} \right\}$$

Où CR et F sont des paramètres de contrôle.

Remarque:

Pour les opérateurs de croisement multiparents (SPX et PCX), on a pris que 3 parents dans l'implémentation de ces opérateurs ce qui veut dire que notre travail est réstreint dans l'éspace $\mathbb{R}^3$.

3.5 Présentation des problèmes test [28]

Pour faire notre étude numérique, nous allons présenter dans cette section 03 fonctions bio-objectifs.

3.5.1 Test bio-objectif de Zitzler, Deb et Thiele ($ZDT1$)

Le premier test $ZDT1$ est le plus simple, le front de Pareto correspondant étant continu, convexe et avec la distribution uniforme des solutions le long du front.

$$\boldsymbol{ZDT1}: \left\{ \begin{array}{c} f_1(x) = x_1 \\ f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}] \\ g(x) = 1 + 9\sum_{i=2}^{n} x_i/n - 1 \end{array} \right\}$$

où $x_i \in [0, 1]$ pour tout $i = 1, ..., n$ avec $n = 30$. Les solutions de ce problème sont telles que $0 \leq x_1^* \leq 1$ et $x_i^* = 0$ pour $i = 2, ..., n$. Le front de Pareto de ce problème est présenté dans la figure 3.1.

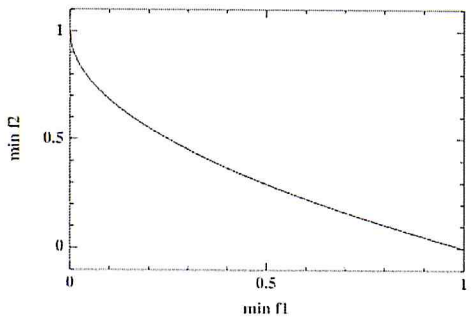


FIGURE 3.1. Le front de Pareto pour le problème ZDT1.

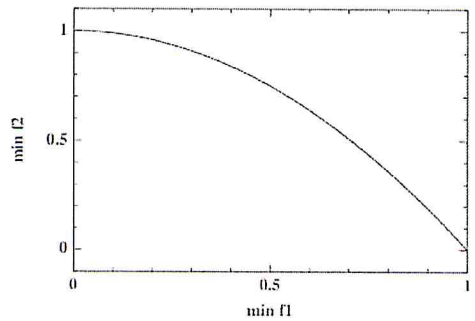


FIGURE 3.2. Le front de Pareto pour le problème ZDT2.

3.5.2 Test bio-objectif de Zitzler, Deb et Thiele ($ZDT2$)

La seule difficulté de ce problème consiste en non-convexité du front de Pareto.

$$\boldsymbol{ZDT2}: \left\{ \begin{array}{c} f_1(x) = x_1 \\ f_2(x) = g(x)[1 - (x_1/g(x))^2] \\ g(x) = 1 + 9\sum_{i=2}^{n} x_i/n - 1 \end{array} \right\}$$

où $x_i \in [0,1]$ pour tout $i = 1, ..., n$ avec $n = 30$. Les solutions de ce problème sont telles que $0 \leq x_1^* \leq 1$ et $x_i^* = 0$ pour $i = 2, ..., n$. Le front de Pareto de ce problème est présenté dans la figure 3.2

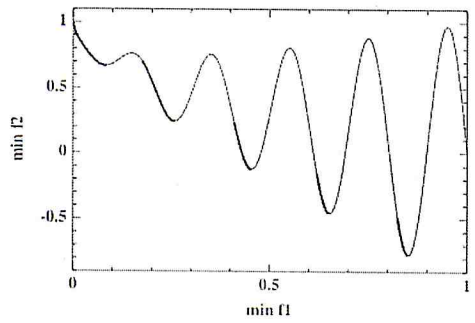


FIGURE 3.3. Le front de Pareto pour le problème ZDT3.

3.5.3 Test bio-objectif de Zitzler, Deb et Thiele ($ZDT3$)

La difficulté de ce problème est que le front de Pareto est discontinu.

$$\boldsymbol{ZDT3}: \left\{ \begin{array}{c} f_1(x) = x_1 \\ f_2(x) = g(x)[1 - \sqrt{x_1/g(x)} - x_1 \sin(10\pi x_1)/g(x)] \\ g(x) = 1 + 9\sum_{i=2}^{n} x_i/n - 1 \end{array} \right\}$$

où $x_i \in [0,1]$ pour tout $i = 1, ..., n$ avec $n = 30$. Les solutions de ce problème sont telles que $0 \leq x_1^* \leq 1$ et $x_i^* = 0$ pour $i = 2, ..., n$. Le front de Pareto de ce problème est présenté dans la figure 3.3.

CHAPITRE 4

RÉSULTATS ET ANALYSES

Dans ce chapitre, on va appliquer l'algorithme $\boldsymbol{HMO-MOEA}$ sur les problèmes test présentés dans la section (3.5) du chapitre précédent. En faisant varier le nombre d'opérateurs hybridés de 1 à 5, selon deux mésures de performance

ant de voir et d'analyser les résultats obtenus lors du changement et modification de la notion d'hybridation appliquée sur l'ensemble des opérateurs de croisement, et finalement donner le meilleur résultat obtenu dans cette hybridation pratiquement par deux mésures de performance (General distance and Spacing metric).

4.1 Initialisation et réglage des paramètres de l'algorithme

Le principe de cette méthode est de faire intervenir plusieurs opérateurs de croisement et de générer des nouvelles solutions selon un processus aléatoire de choix, ce qui nous conduit à donner les paramètres de chaque opérateur ansi que les tailles des ensembles utilisés. Pour ce cela, on distingue deux types de paramètres à initialiser:

4.1.1 initialisation des paramètres de l'algorithme

1. La taille de la population initiale et la population actualisée durant le processus d'évolution, $n\_pop = 100$.

2. La taille de l'archive externe EXA, qui est la même que celle de la population, $n\_exa = 100$.

3. La taille d'un vecteur ou bien nombre de variables de décision de chaque individu notée n_var est spécifique pour chaque problème test.

4. La taille des ensemble P inf et P sup qui vaut 500, ce qui aide à maintenir une archive pour les solutions anciennes et qui apparaissent meilleures ulterieurement.

4.1.2 Paramètres de contrôle des opérateurs génétiques (Mutaion-Croisement)

1. La probabilité minimale dans le méchanisme de choix d'opérateur de croisement notée $p_{\min} = 0.1$.

2. La probabilité de mutation d'un individu est 0.033, et celle d'une dimension est $1/n\_var$.

3. $\alpha = 0.5$ pour l'opérateur de croisement BLX_α, selon la proposition dans [22].

4. $\eta = 20$ pour l'opérateur de croisement SBX, selon [19].

5. $\epsilon = 1$ pour l'opérateur de croisement SPX, selon [29].

6. $\sigma_\eta = \sigma_\varepsilon = 0.1$ pour l'opérateur PCX, selon [2].

7. $CR = 1$ et $F = 0.5$ pour l'opérateur de croisement DE, selon [10].

4.2 Environnement expérimental

Dans le but d'implémentation de notre méthode, et pour pouvoir obtenir des résultats aidant à la comparaison entre les différents types de croisement, on a effectué 500 générations pour chaque problème test en utilisant un ordinateur portatif et particulier ayant les propriètés suivantes: 2.20GHz CPU, 4GB de RAM avec un système d'exploitation Windows 8.

Ce qui concerne l'implémentation, le prmoteur a proposé d'utiliser le language de programmation *Scilab* version 5.5.2, qui est un langage interprété, à typage dynamique, extensible, gratuit à télécharger et facil à manipuler. Il a d'abord été développé sous le nom de Basile (Inria projet Meta2) puis sous le nom de Scilab par le Scilab Group

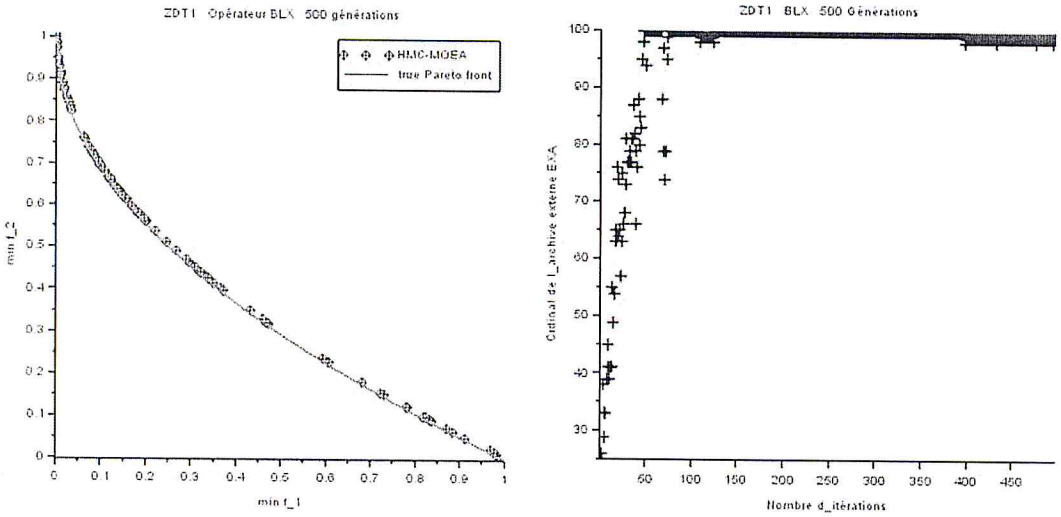


FIGURE 4.1. HMC-MOEA avec l'opérateur BLX.

(Chercheurs du projet Inria Metalau et de l'Enpc Cermics). Son développement actuel est coordonné par un Consortium et un nombre non négligeable de contributions proviennent de contributeurs extérieurs. Scilab est un langage portable. Il est porté sur les différentes variantes d'UNiX, mais aussi sur Windows et MacOS X. Site officiel de ce langage est: **http://www-rocq.inria.fr/scilab/**

4.3 Résultats et analyses

4.3.1 Chaque opérateur seul contre les cinq opérateurs hybridés

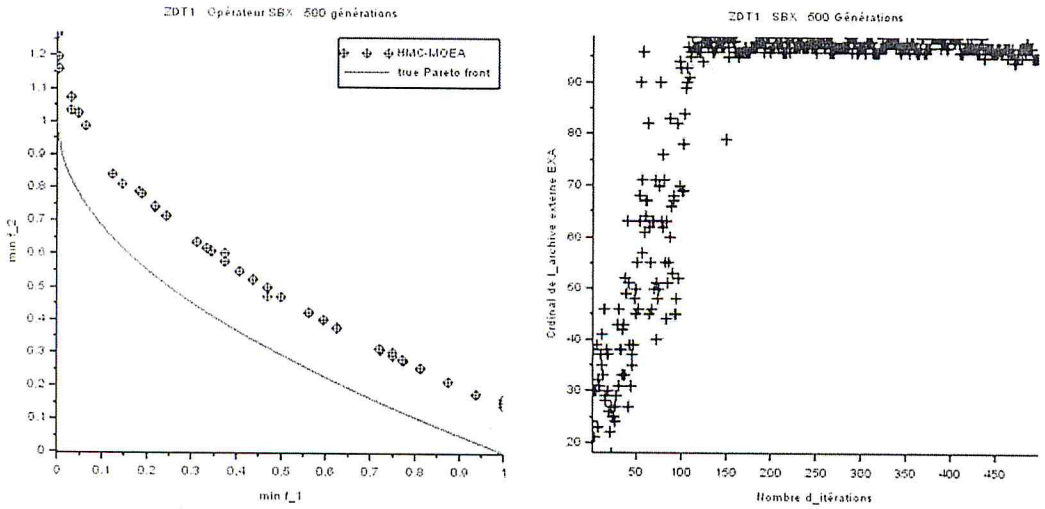


FIGURE 4.2. HMC-MOEA avec l'opérateur SBX.

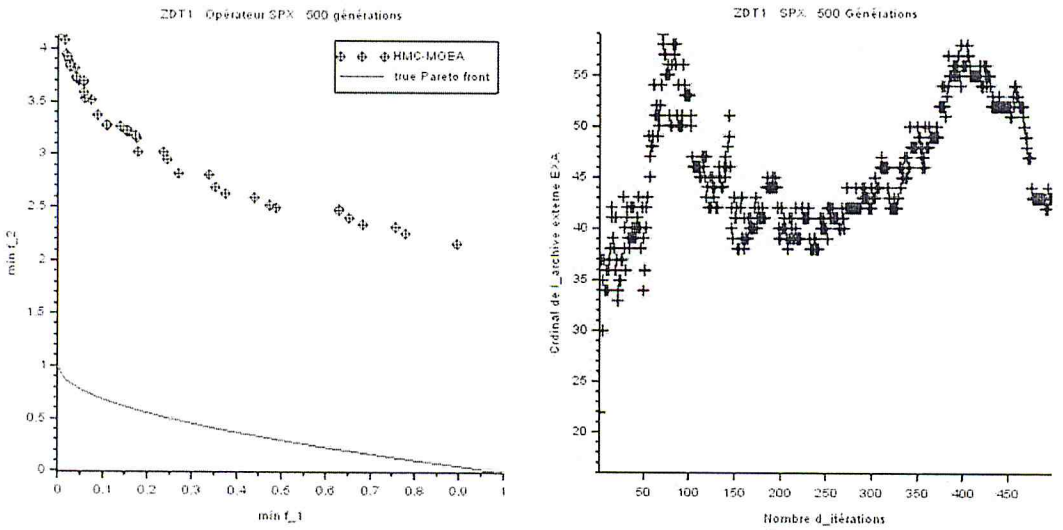


FIGURE 4.3. HMC-MOEA avec l'opérateur SPX.

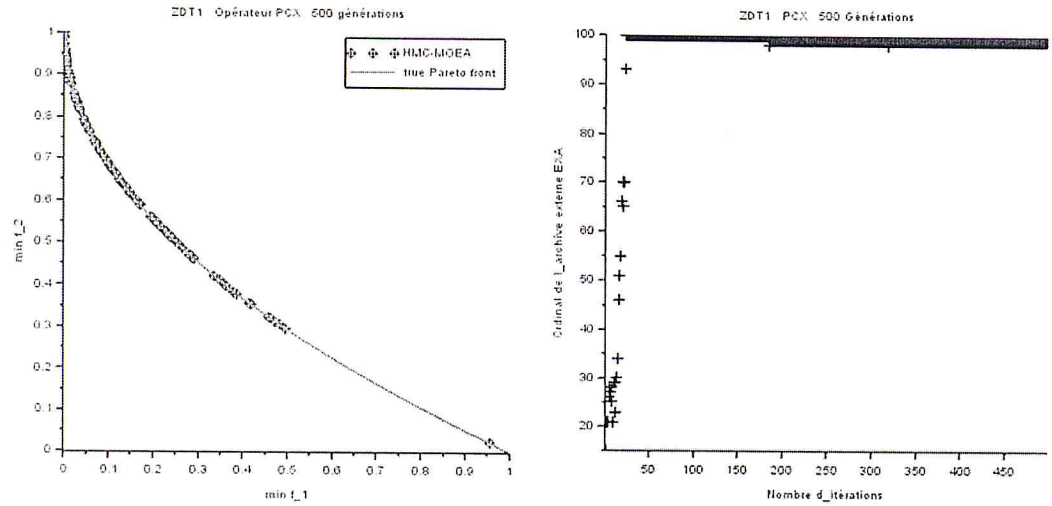


FIGURE 4.4. HMC-MOEA avec l'opérateur PCX.

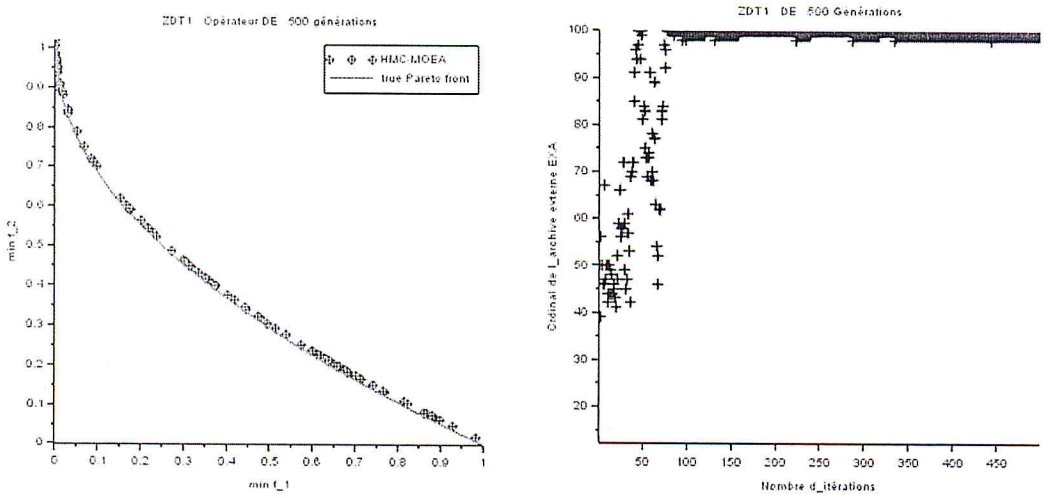


FIGURE 4.5. HMC-MOEA avec l'opérateur DE.

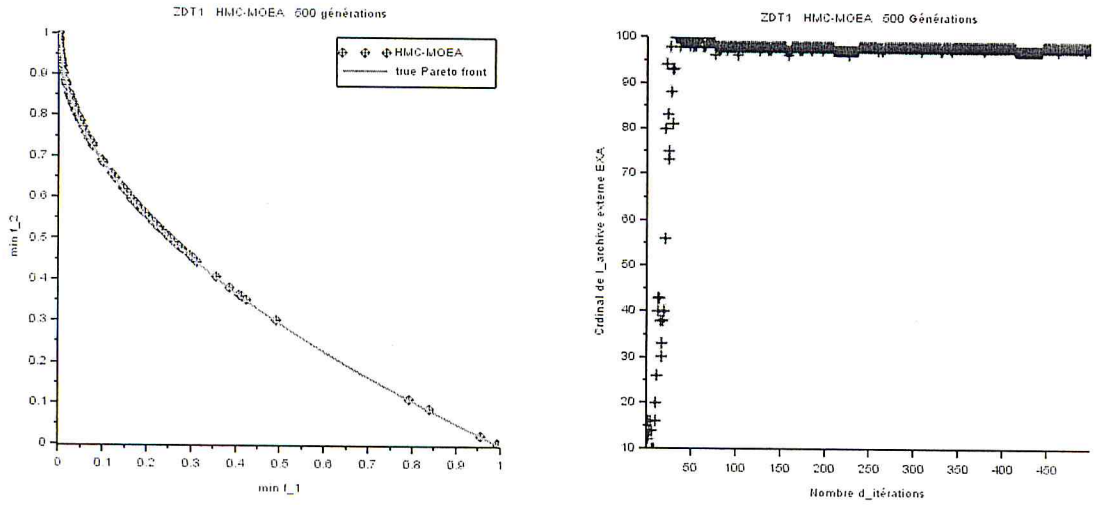


FIGURE 4.6. HMC-MOEA avec les cinq opérateurs.

Les figures (4.1 jusqu'au 4.18) illustrent les solutions obtenues après 500 générations dans l'espace de recherche ansi que l'évolution du cardinal de l'archive externe EXA en fonction du nombre d'itérations en utlisant chaque opérateur de croisement tout seul et l'hybridation des cinq à la fois dans l'algorithme HMC-MOEA. Après le calcul des mésures de performance GD et SM, les résultats son récapultés dans les deux tableaux suivat:

	BLX	SBX	SPX	PCX	DE	$HMC-MOEA$
ZDT1	0.00312	0.00974	0.0626	**0.00161**	0.00446	0.00282
ZDT2	0,0099279	0.0231152	0.2637805	**0.0041266**	0.0537813	0,0042093
ZDT3	0.0048352	0.0081728	0.0607459	**0.0026452**	0.0069682	0.0028182

Tableau 4.1. La distance générationnelle pour chaque opérateur.

	BLX	SBX	SPX	PCX	DE	$HMC-MOEA$
ZDT1	0.0717	0.0743	0.18	**0.0604**	0.0798	0.0624
ZDT2	0,0603305	0.0334832	0.1490311	0.0623951	**0.0148252**	0,0687104
ZDT3	0.1502310	0.1474468	0.2810967	**0.1312251**	0.1840094	0.1401032

Tableau 4.2. La métrique d'espacement pour chaque opérateur.

Déscription et analyse:

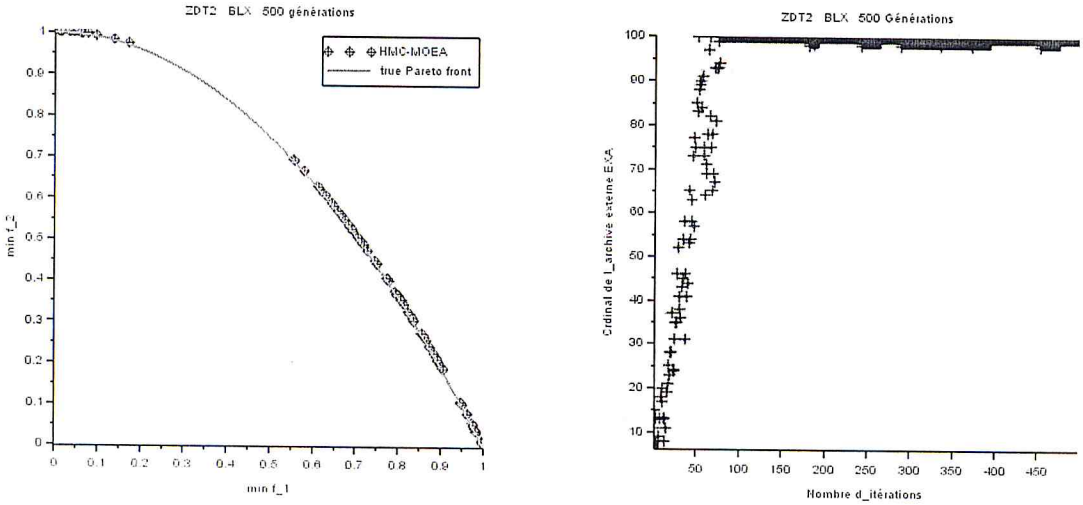


FIGURE 4.7. HMC-MOEA avec l'opérateur BLX, appliquée sur ZDT2.

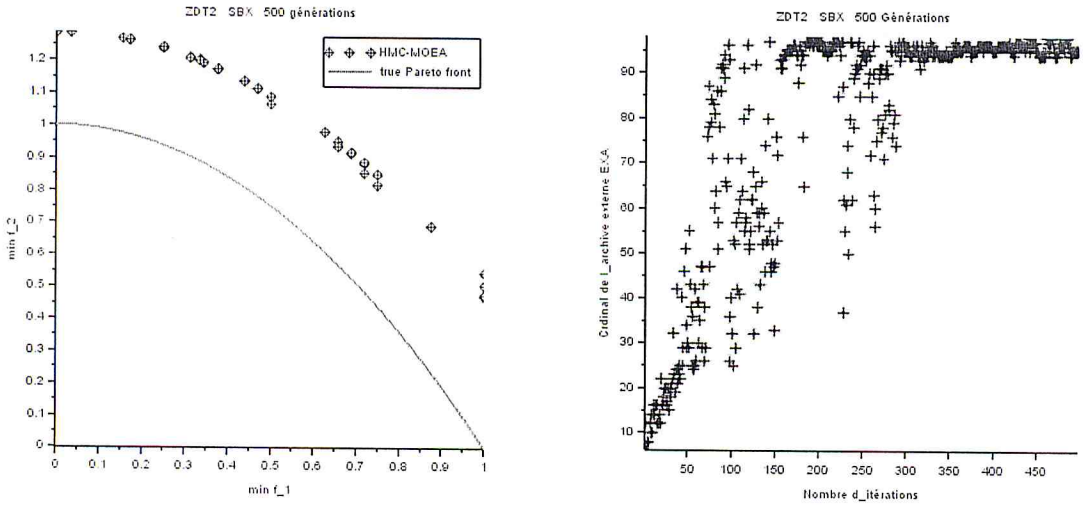


FIGURE 4.8. HMC-MOEA avec l'opérateur SBX, appliquée sur ZDT2.

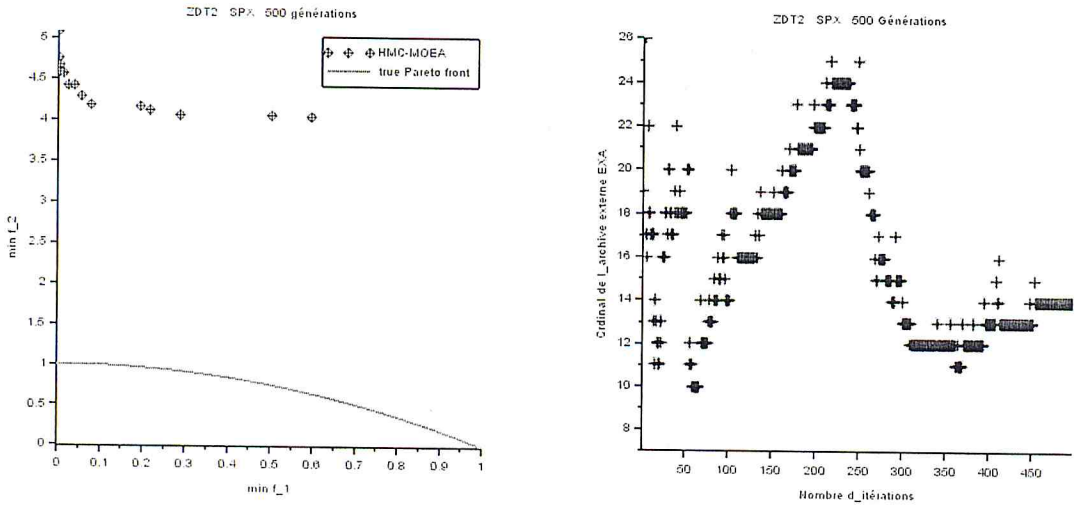


FIGURE 4.9. HMC-MOEA avec l'opérateur SPX, appliqué sur ZDT2.

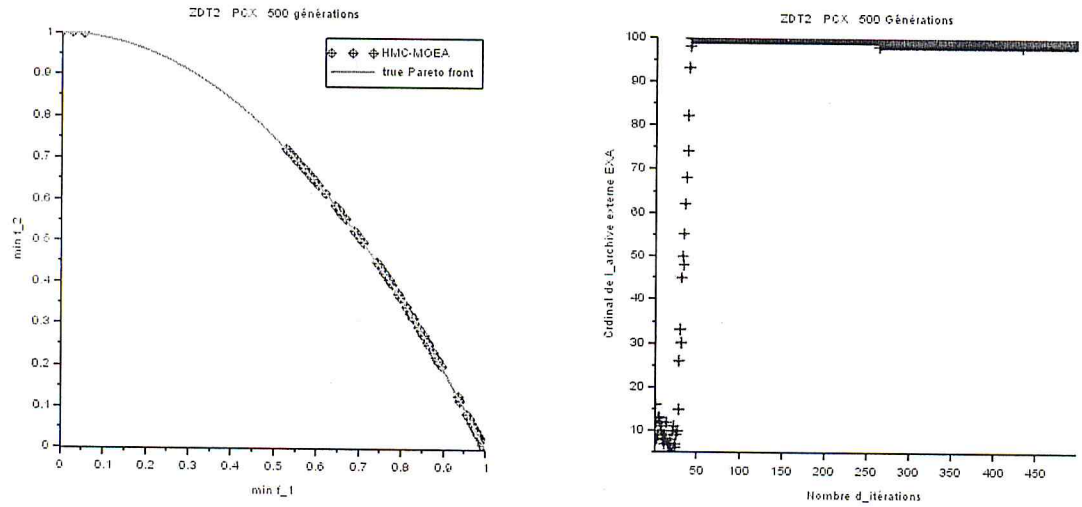


FIGURE 4.10. HMC-MOEA avec l'opérateur PCX, appliqué sur ZDT2.

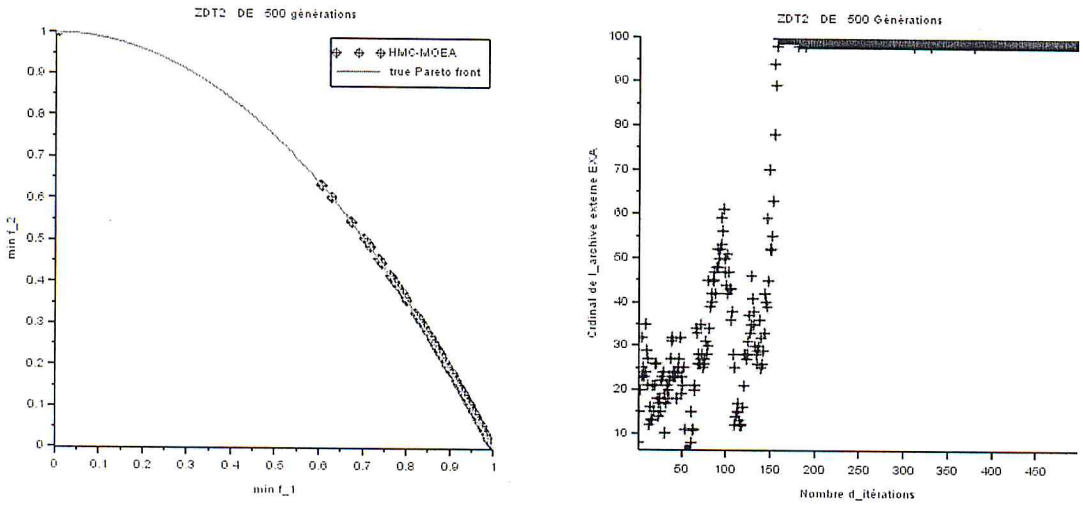


FIGURE 4.11. HMC-MOEA avec l'opérateur DE, appliqué sur ZDT2.

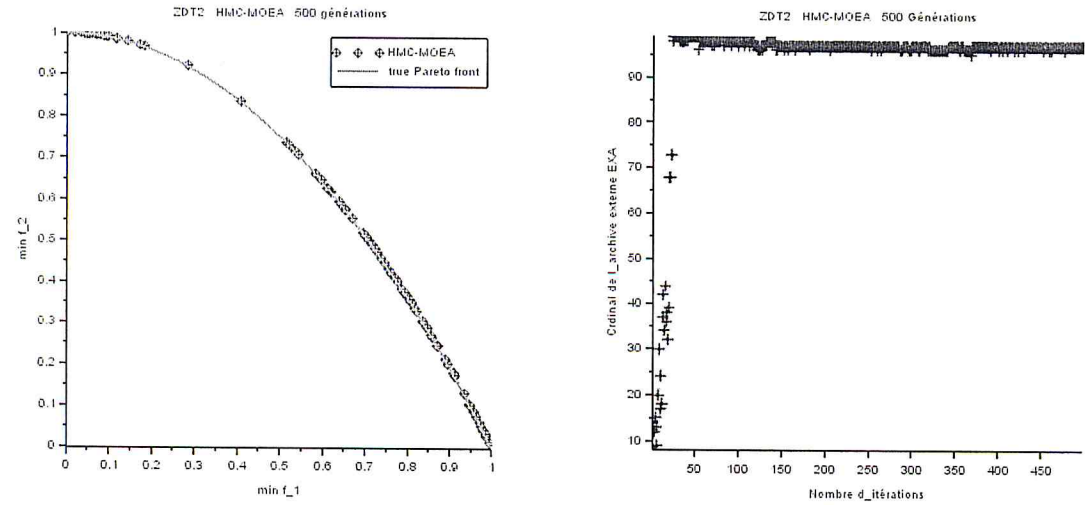


FIGURE 4.12. HMC-MOEA avec les cinq opérateurs, appliqué sur ZDT2.

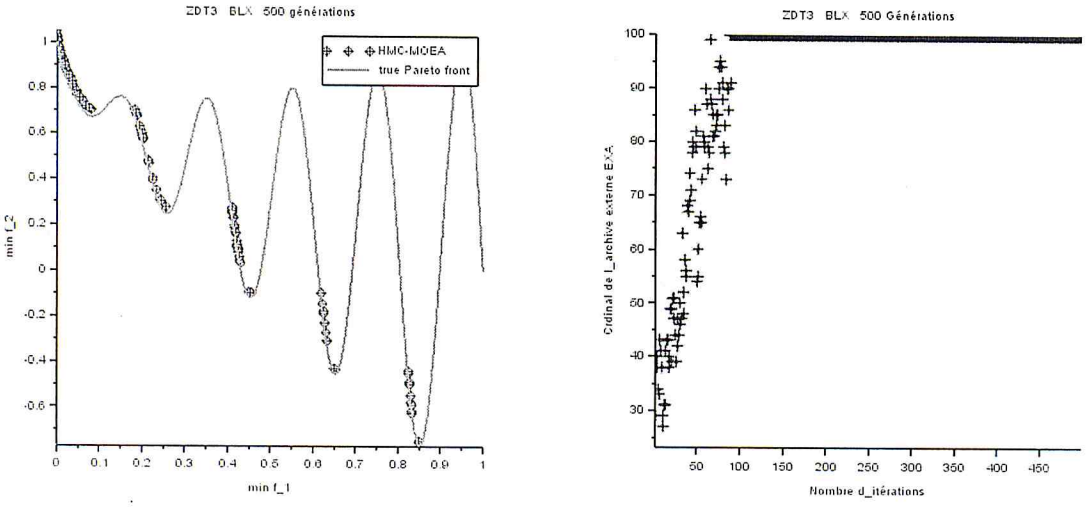


FIGURE 4.13. HMC-MOEA avec l'opérateur SBX, appliqué sur ZDT3.

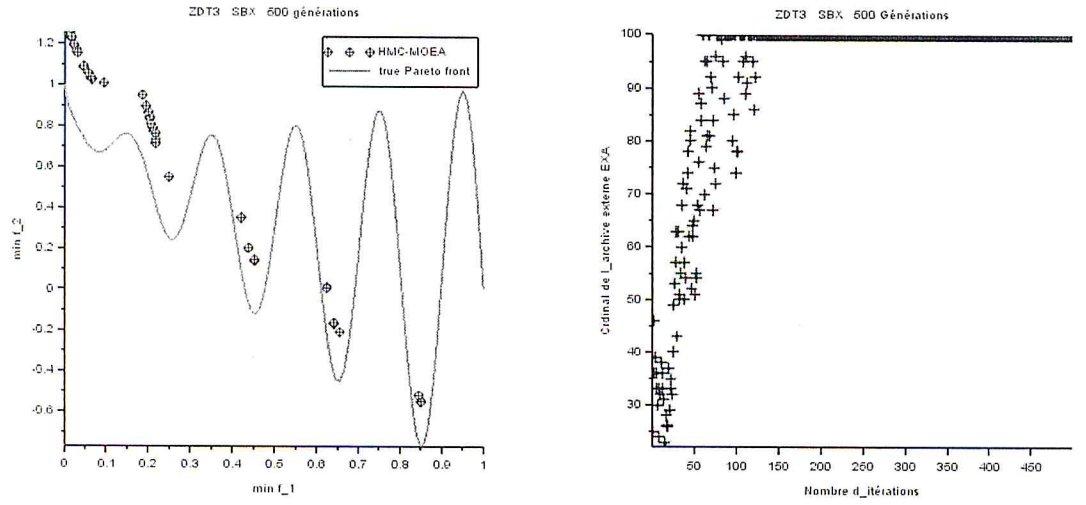


FIGURE 4.14. HMC-MOEA avec l'opérateur SBX, appliqué sur ZDT3.

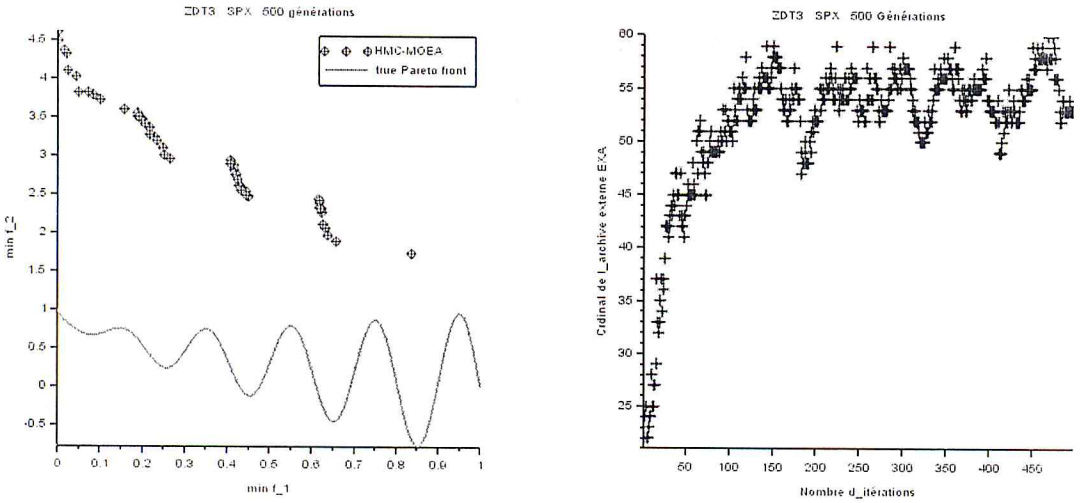


FIGURE 4.15. HMC-MOEA avec l'opérateur SPX, appliqué sur ZDT3.

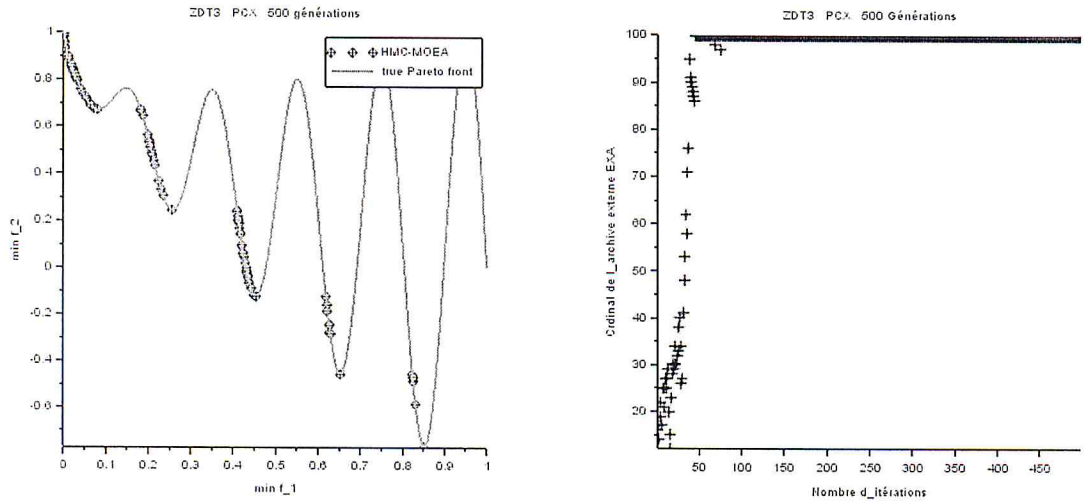


FIGURE 4.16. HMC-MOEA avec l'opérateur PCX, appliqué sur ZDT3.

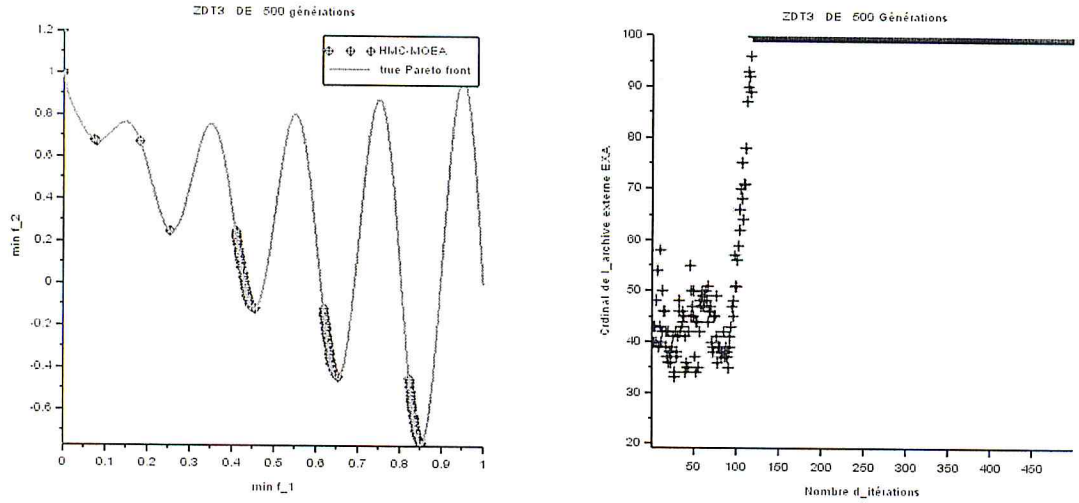


FIGURE 4.17. HMC-MOEA avec l'opérateur DE, appliqué sur ZDT3.

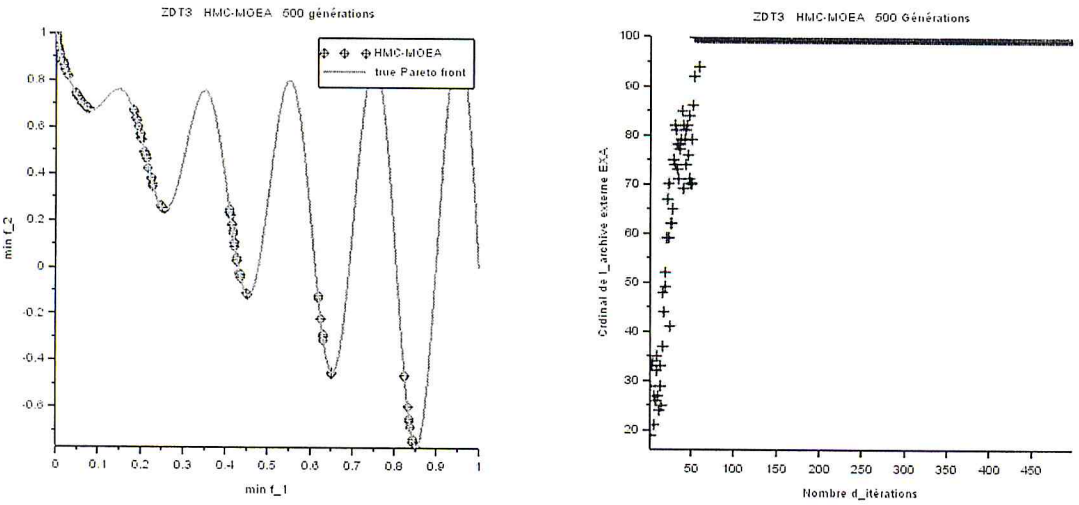


FIGURE 4.18. HMC-MOEA avec les cinq opérateurs, appliqué sur ZDT3.

On peut lire à travers les résultats obtenus dans les deux tableaux (4.1 et 4.2) pour les problèmes test ZDT1, ZDT2 et ZDT3 que:

L'opérateur BLX a pu atteindre le front de Pareto dans 500 générations et de générer un nombre maximum de solutions non dominées dans prèsque 50 générations, mais le plus remarquable est que le front de Pareto n'est pas vraiment couvert et ce qui augmente la métrique d'espacement (SM), par contre l'opérateur SBX n'a pas pu atteindre le front de Pareto avec une petite perturbation dans le maintien du nombre maximum de EXA, et le même résultat pour l'opérateur SPX qui a donné des solutions loins du front de Pareto avec une grande perturbation pour le cardinal de l'ensemble EXA. L'opérateur PCX a montré une bonne éfficacité par rapport aux opérateurs précédents dans toutes les normes que se soient l'atteinte du front de Pareto, la bonne distubition des solution ou bien le cardinal de EXA après moins de 50 générations. L'opérateur DE a donné prèsque les mêmes résultats que PCX avec moins de performance pour l'atteinte du front Pareto. L'hybridation des cinq opérateurs a donné de bons résultats mais moins de performance en la comparant avec l'opérateur PCX ce qui est clairement présenté dans les tableaux su-cités.

Conclusion:

L'hybridation des cinq opérateurs a donné de meilleurs résultats que l'application des opérateurs de croisemnt chacun tout seul, et l'exception est faite pour l'opérateur PCX qui a montré une bonne performance dans le sens de la convergence vers le front de Pareto et l'espacement entre les individus le long de ce front.

D'autre part, notre modification dans la mise à jour de la population a pu éviter le problème de la convergence prématurée dans le problème discontinu ZDT3.

On peut remarquer que le classement des opérateurs selon l'ordre de performance est PCX, DE et SBX, à cet effet, nous allons hybrider les trois opérateurs dans la partie suivante:

4.3.2 Hybridation des opérateurs PCX, DE et SBX

Dans cette section, on présente dans les tableaux (4.3 et 4.4) tous les résultats obtenus aprés l'exécution de l'algorithme HMC-MOEA, avec toutes les hybridations effectuées avec

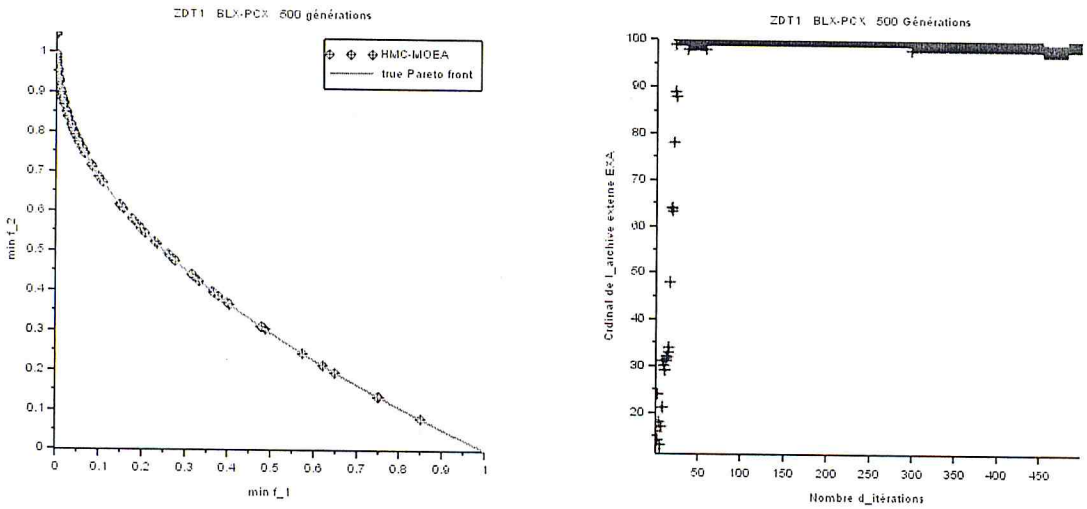


FIGURE 4.19. HMC-MOEA avec l'hybridation PCX-SBX, appliqué sur ZDT1.

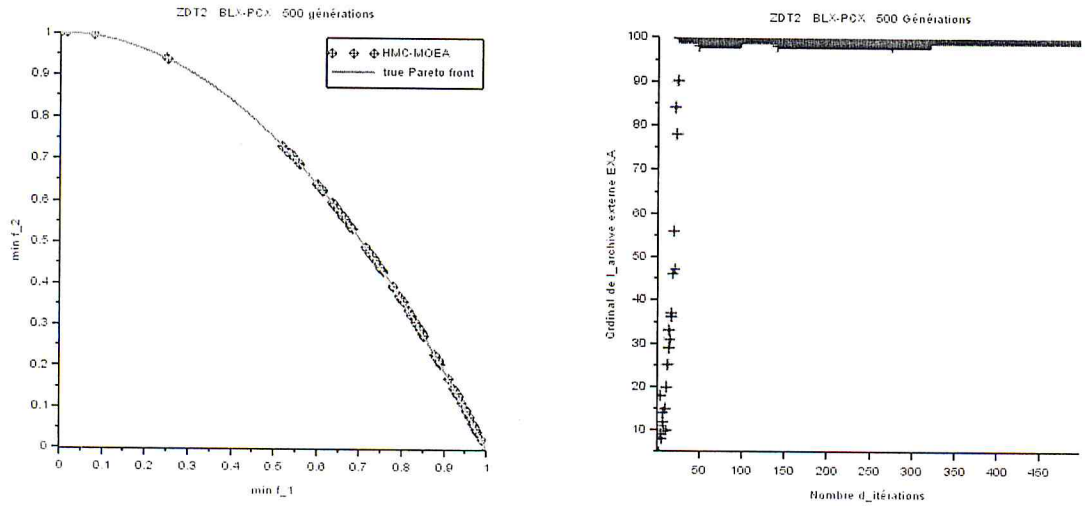


FIGURE 4.20. HMC-MOEA avec l'hybridation PCX-SBX, appliqué sur ZDT2.

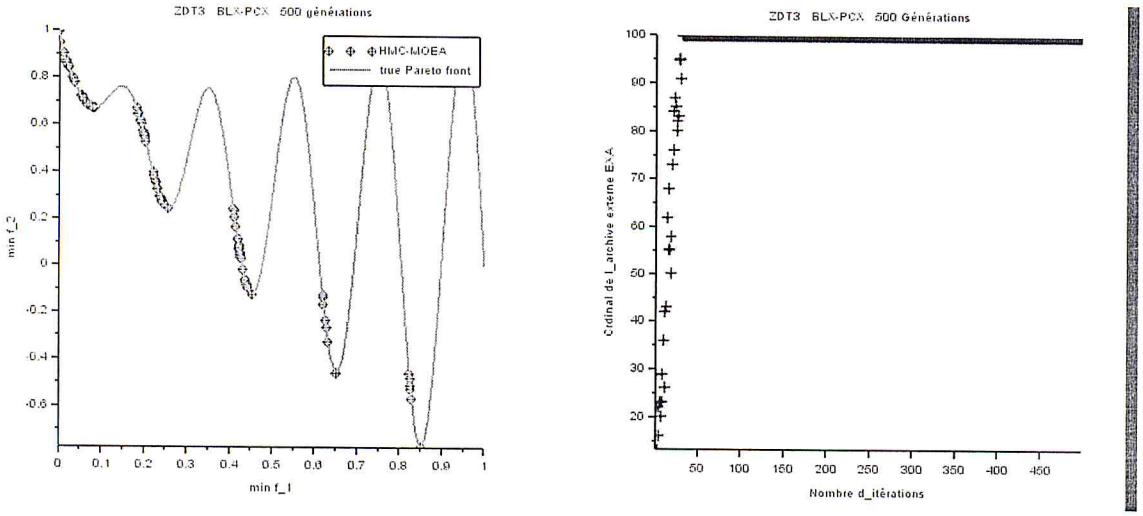


FIGURE 4.21. HMC-MOEA avec l'hybridation PCX-SBX, appliqué sur ZDT3.

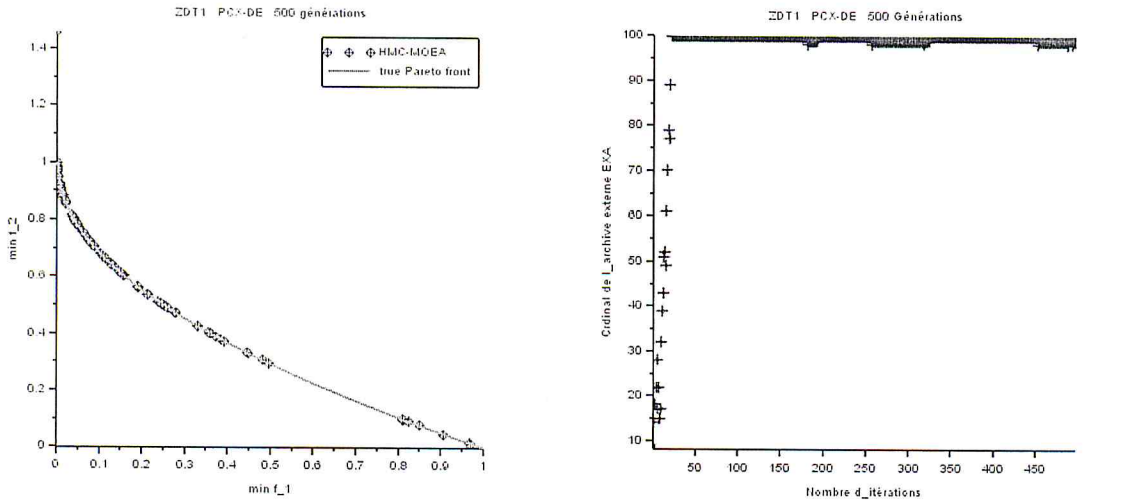


FIGURE 4.22. HMC-MOEA avec l'hybridation PCX-DE, appliqué sur ZDT1.

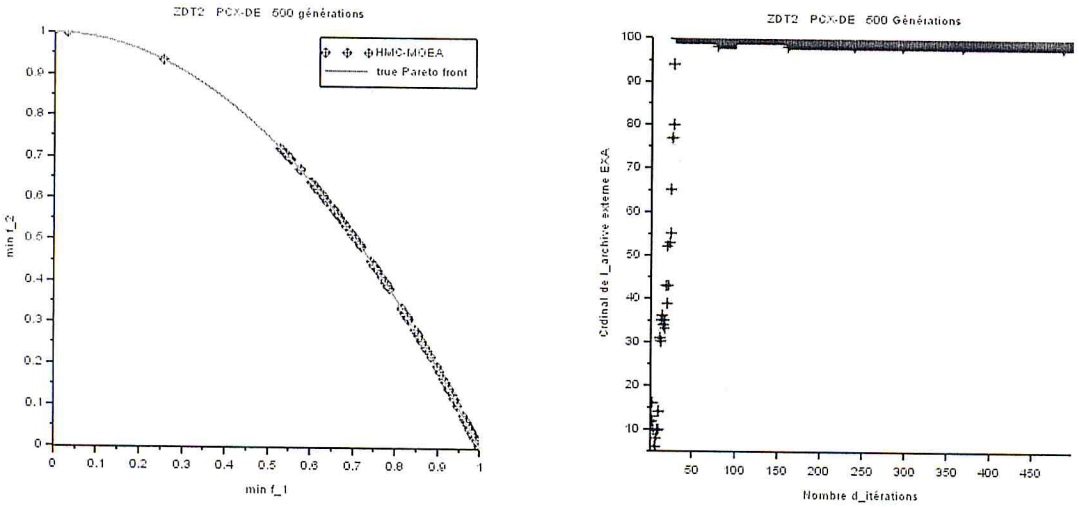


FIGURE 4.23. HMC-MOEA avec l'hybridation PCX-DE, appliqué sur ZDT2.

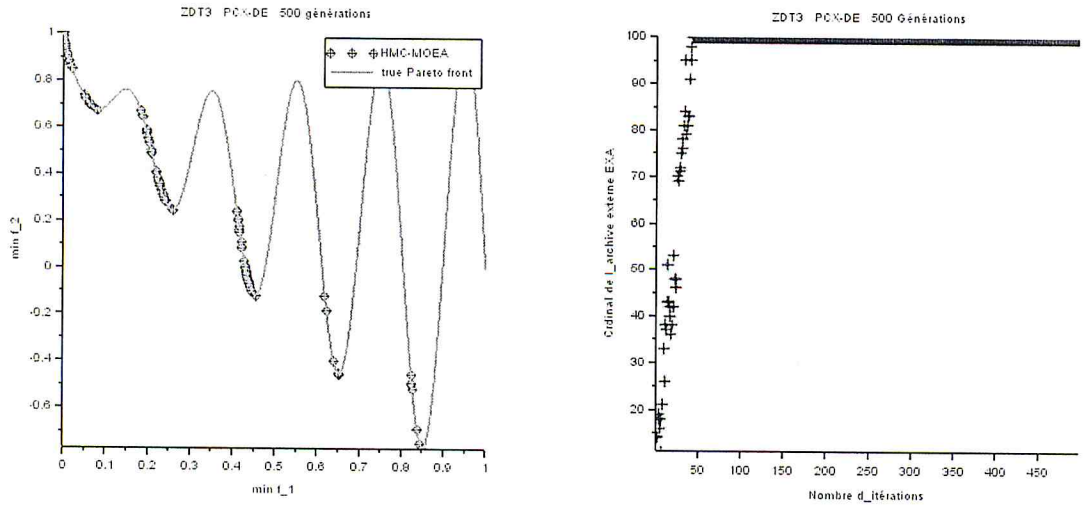


FIGURE 4.24. HMC-MOEA avec l'hybridation PCX-DE, appliqué sur ZDT3.

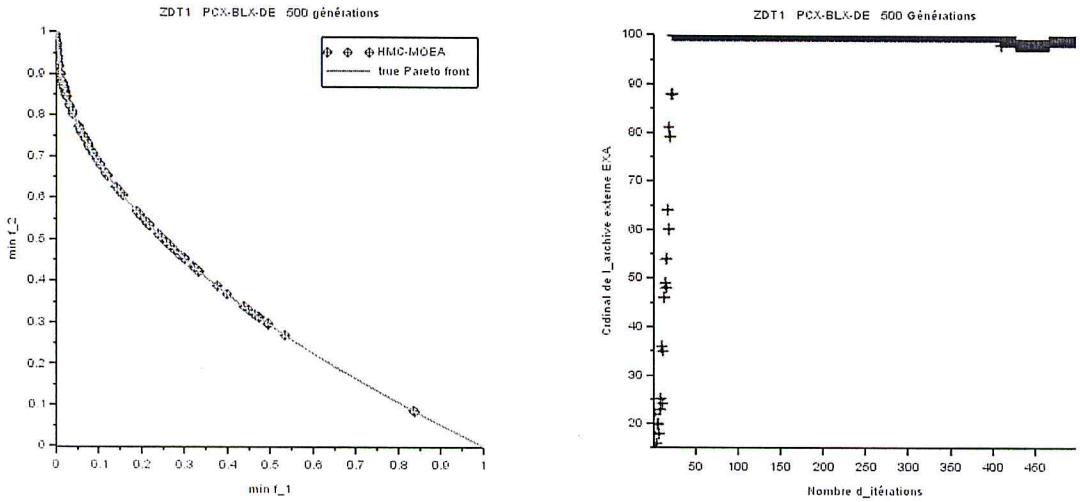


FIGURE 4.25. HMC-MOEA avec l'hybridation PCX-BLX-DE, appliqué sur ZDT1.

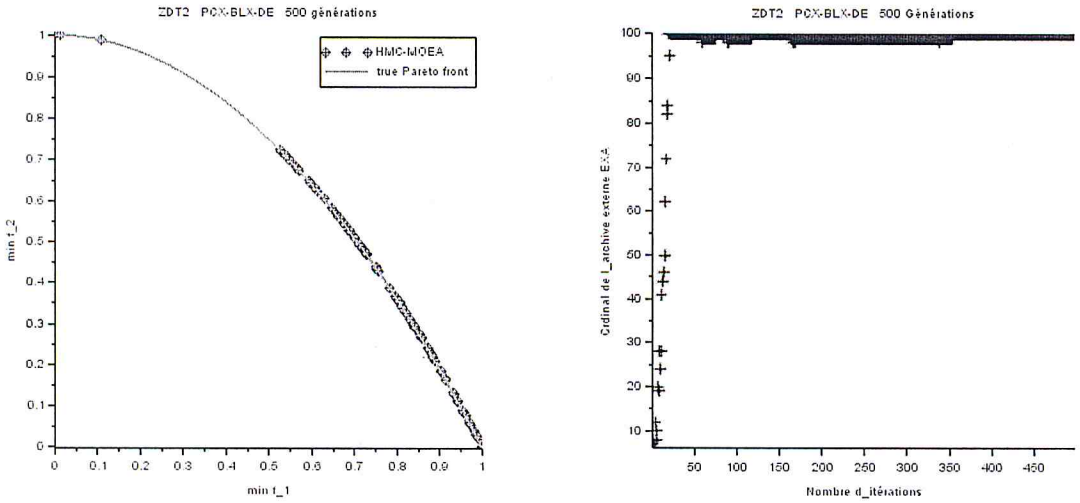


FIGURE 4.26. HMC-MOEA avec l'hybridation PCX-BLX-DE, appliqué sur ZDT2.

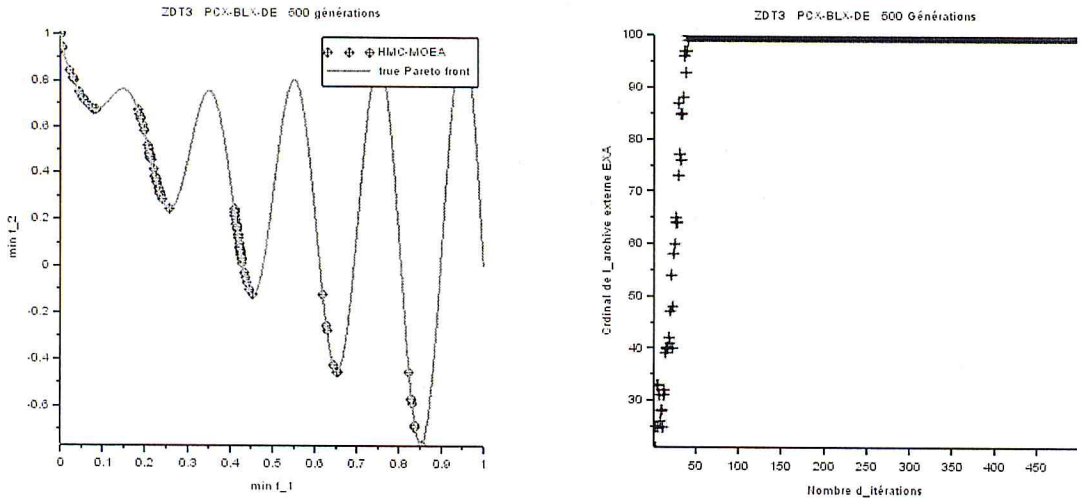


FIGURE 4.27. HMC-MOEA avec l'hybridation PCX-BLX-DE, appliqué sur ZDT3.

les cinq opérateurs de croisement.

	$ZDT1$	$ZDT2$	$ZDT3$
BLX	0.00312	0.00992	0.00483
SBX	0.00974	0.02311	0.00817
SPX	0.0626	0.26378	0.06074
PCX	**0.00161**	0.00412	0.0026452
DE	0.00446	0.0537813	0.0069682
$PCX - BLX$	0.0021149	0.0048095	0.0020363
$PCX - DE$	0.0022514	0.0039891	0.0024094
$PCX - BLX - DE$	0.0017659	**0.0038513**	**0.0013552**
$HMC - MOEA$	0.00282	0.0042093	0.0028128

Tableau 4.3. La distance générationnelle.

	$ZDT1$	$ZDT2$	$ZDT3$
BLX	0.0717	0.0603305	0.1502310
SBX	0.0743	0.0334832	0.1474468
SPX	0.18	0.1490311	0.2810967
PCX	0.0604	0.623951	0.1312251
DE	0.0798	**0.0148252**	0.1840094
$PCX - BLX$	0.0610864	0.0631439	0.1322456
$PCX - DE$	0.0518878	0.0596800	0.1366096
$PCX - BLX - DE$	**0.0516314**	0.0618472	**0.1293423**
$HMC - MOEA$	0.0624	0.0687104	0.1401032

Tableau 4.4. La métrique d'espacement.

Déscription et analyse:

La combinaison des trois opérateurs de croisement a donné de meilleurs résultats que les premières expériences avec et sans hybridation, l'exception est faite dans la distance générationnelle avec l'opérateur PCX appliqué sur le problème ZDT1, et la métrique d'espacement avec l'opérateur DE appliqué sur le problème ZDT2. Les meilleurs résultats sont obtenus avec l'hybridation des trois opérateurs PCX, BLX et DE à la fois.

Conclusion Générale

Les algorithmes évolutionnaires (AEs) sont des méthodes pratiques et robustes pour automatiser la recherche de bonnes solutions. Ils sont proposés comme un moyen de trouver des solutions proches des optima globaux pour des problèmes complexes dans un temps beaucoup plus court que ce qui serait requis par l'évaluation de toutes les solutions possibles. Les AEs ont été appliqués avec succès à une variété de problèmes d'optimisation difficiles. Malgré leurs capacités méthodiques, ils souffrent du problème de réglage/contrôle de paramètres. Un point-clé consiste à trouver un moyen d'ajustement automatique des paramètres durant l'exécution de l'algorithme. Ce concept doit être appliqué pour améliorer la recherche et obtenir une convergence efficace.

Ce mémoire trouve naturellement sa première source d'inspiration dans les travaux traitant des algorithmes évolutionnaires et des métaheuristiques pour l'optimisation multiobjectif.

Le premier objectif de ce travail était de remédier le problème de la convergence prématurée dans les problèmes discontinu, et le deuxième objectif consiste à trouver une meilleure hybridation en combinant cinq opérateurs de croisement (BLX, SBX, SPX, PCX et DE).

Pour le premier objectif, on a introduit une modification sur la mise à jour de la population adaptée dans l'algorithme HMOEA (Hybrid Multiobjective Evolutionary Algorithm) présenté par Lixing Tang et Xianpeng wang [24]. Cette modification consiste à actualiser la population en faisant le croisement entre des individus choisis dans l'archive externe EXA et d'autres choisis dans deux sous ensembles Pinf et Psup faisant une partition de la population courante. Le choix des individus dans Pinf (respectivement Psup) se fait avec la roue de fortune proportionnellement au nombres des individus ayant leur premier objectif inférieur (respectivement supérieur) à un constant α. D'après les résultats obtenus dans les graphes de l'espace de recherche du problème discontinu ZDT3, les solutions sont distribuées sur le front de Pareto, ce qui veut dire que notre premier objectif est atteint.

Le deuxième objectif de notre travail consiste à trouver une meilleure hybridation des opérateurs de croisement. Pour cela, on a effectué des expériences en appliquant notre algorithme avec les différentes combinaisons d'hybridation de ces opérateurs, et la com-

paraison se base sur les valeurs de la distance générationnelle et la métrique d'espacemet obtenues pour chaque hybridations. Les résultats obtenus après 500 générations montrent que la meilleure hybridation est celle de la combinaison (PCX-BLX-DE) et l'exeption est faite dans la distance générationnelle avec PCX appliqué sur ZDT1 et dans la métrique d'espacement avec DE appliqué sur ZDT2. Les mauvais résultats sont obtenus en appliquant le croisement SPX tout seul, et ça revient à la nécessite d'un nombre assez grand de génarations pour cette opérateur, ce qui amoindrit la performance des hybridation prenant SPX dans leur combinaison.

La notion d'hybridation des opérateurs de croisement est très performante et efficace dans le sens de la convergence des solutions vers le front de Pareto dans algorithmes évolutionnaire, mais le point-clé consiste à choisir soigneusement les opérateurs de croisement à hybrider, en respectant la convenance entre ces opérateurs dans leurs vitesse de convergence et le nombre de générations nécessaires pour atteindre le front de Pareto.

Bibliographie

[1] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham, "AbYSS: Adapting scatter search to multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 12, pp. 439–457, Aug. 2008.

[2] Croisement SPX: rapport interne, Faculté de Mathématiques, USTHB 2014.

[3] C. A. Coello Coello, *An updated survey of G.A. based multiobjective optimization techniques*, Rapport technique Lania-RD-98-08, Xalapa, Veracruz, Mexico, décembre 1998.

[4] Suquet, Simulation, Agrégation externe, Université des Sciences et Technologies de Lille, 2005-2006.

[5] D. A. Van Veldhuizen, *Multiobjective evolutionary algorithms : classifications, analyses and new innovations*, Ph. D., Graduate School of Engineering. Air Force Institute of Technology, Wright Patterson AFB, Ohio, USA, janvier 1999.

[6] D. E. Goldberg. Genetic algorithms in search, Optimization and Machine learning. Addison Wesley, 1989.

[7] D. E. Goldberg and J. Richardson. Genetic algorithm with sharing for multi-modal functions optimization. Proceedings of the 2nd International Conference on Genetic algorithms, pages 41-49. Lawrence Erlbaum Associates, 1987.

[8] F. Glover, M. Laguna, and R. Martí, "Scatter search," in *Advances in Evolutionary Computing: Theory and Applications*, A. Ghosh and S. Tsutsui, Eds. Berlin, Germany: Springer-Verlag, 2003, pp. 519–537.

[9] H. Hamdan: On the disruption-level of polynomial mutation for evolutionary multi-objective optimisation algorithms, Computing and Informatics, vol. 29, pages 783-800, 2010.

[10] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto set, OEA/D and NSGA-II," IEEE Trans. Evol. Comput., vol. 13, pp. 284-302, Apr. 2009.

[11] H. S. Yoon and B. R. Moon, "An empirical study on the synergy of multiple crossover operators," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 212–223, Apr. 2002.

[12] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of a vectors. Jornal of the Association of Computing Machinery, 22(4): 469-476, 1975.

[13] I. Othmani, *Optimisation multicritère : Fondements et Concepts*, Thèse de doctorat, université de Grenoble, mai 1998.

[14] I. Rechenberg. Evolution Strategie. Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution. Fromman-Hozlboog Verlag, Stuttgart, 1973.

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

[16] J. R. Koza. Genetic Programming: On the Programming of Computers by means of Natural Evolution, pages 98-105, IEEE service center, 1999.

[17] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter evolution," Evol. Comput., vol. 10, no. 4, pp. 371-395, 2002.

[18] K.Deb. Multi-objective optimization using Evolutionnary algorithms. New York: John Wiley, 2001.

[19] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.

[20] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, Apr. 2002.

[21] K. Miettinen. Nonlinear Multiobjective Optimization. Kluwer, Boston, 1999.

[22] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 187–202.

[23] L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial Intelligence through Simulated Evolution. New-York: John Wiley, 1966.

[24] L. Tang and X. Wang, A Hybrid Multiobjective Evolutionary Algorithm for Multiobjective Optimization Problems, IEEE Trans. On Evolutionary Computation. Vol 17, pages 20-46, February 2013.

[25] M. Ehrgott, *A characterization of Lexicographic Max-ordering Solutions*, Methods of Multicriteria Decision Theory : Proceedings of the 6th Workshop of the DGOR Working Group Multicriteria and Decision Theory, Egelsbach, Häsel-Hohenhausen, pages 193-202, 1997.

[26] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J. J Grefenstette, editor, Proceedings of th 1st International Conference on Genetic Algorithms, pages 183-187. Laurence Erlebaum Associates, 1985.

[27] N. O. D. Cunha and E. Polak. Constrained minimization under vector-evaluated criteria in finit dimensional spaces. Jornal of Mathematical Analysis and Applications, 19(1): 103-124, 1967.

[28] O. Roudenko, Application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectif avec contraintes, Thèse de Doctorat, Ecole Polytechniques, France, 2004.

[29] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multiparent recombination with simplex crossover in real coded genetic algorithms," in *roc.GECCO*, 1999, pp. 657–664.

[30] V. Chankong and Y. Y. Haims. Multiobjective Decision Making Theory and Methodology. North-Holland, New-York, 1983.

[31] Y. Y. Haimes, L. S. Lasdon and D. A. Wismer. On a bicriterion formulation of the pboblems of integrated systems identification and system optimization. IEEE Trans. On systems, Man and Cybernetics, SMC-16: 122-128, 1986.

[32] Y. Collette, P. Siarry, Optimisation Multiobjectif, Edition EYROLLES, 2002.