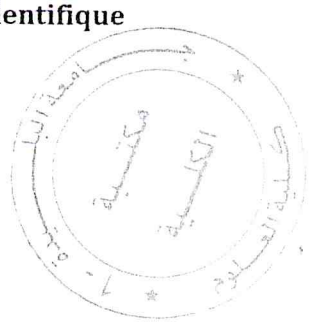
République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de BLIDA 1



Faculté des Sciences

Département de Mathématiques

Mémoire de fin d'études en vue de l'obtention du diplôme de

Master en Mathématiques

Option :

Modélisation Stochastique et Statistique

Thème

Algorithme Evolutionnaire basé sur les Grilles pour l'Optimisation Multi-Objectif

Présenté par :

- Mahieddine Selma
- Ben Ahmed Mohamed

M. Président : O. Tami

M. Examinateur : R. Frihi

M. Promoteur : M. Ait akkache

Promotion : 2015/2016

ملخص

تحقيق التوازن بين التقارب والتنوع يلعب دورا رئيسيا في الهدف المتعدد التطوري. معظم الخوارزميات التطورية الحالية لها أداء جيد في مشاكل مع اثنين أو ثلاثة أهداف، ولكن تواجه صعوبات في الاستيعاب عندما يفوق عدد الأهداف الثلاثة.

نقدم في هذا المشروع، المحاكاة وتنفيذ خوارزمية " GrEA", من أجل حل أمثل لمشاكل مع أكثر من ثلاثة أهداف. وتستند هذه الخوارزمية على شبكة لتقوية ضغط اختيار نحو الاتجاه الأمثل مع الحفاظ على توزيع واسع وموحد بين الحلول.

Résumé

L'équilibre entre la convergence et la diversité joue un rôle clé dans l'optimisation multi-objectif évolutionnaire. La plupart des algorithmes évolutionnaires actuels fonctionnent bien sur les problèmes avec deux ou trois objectifs, mais rencontrent des difficultés dans leur évolutivité quand le nombre d'objectifs dépasse trois.

Nous présentons dans ce projet, la simulation et la mise en œuvre d'un algorithme évolutionnaire GrEA(Grid-Based Evolutionary Algorithm), pour résoudre les problèmes d'optimisation avec plus de trois objectifs. Cet algorithme est basé sur une grille pour augmenter la pression de sélection dans la direction optimale, tout en maintenant une distribution large et uniforme parmi les solutions.

Abstract

Balancing convergence and diversity plays a key role in evolutionary multiobjective. Most current evolutionary algorithms perform well on problems with two or three objectives, but encounter difficulties in their scalability when the number of objectives exceeds three.

We present in this project, simulation and implementation of an evolutionary algorithm GrEA (Grid-Based Evolutionary Algorithm), to solve optimization problems with more than three objectives. This algorithm is based on a grid to strengthen the selection pressure toward the optimal direction while maintaining an extensive and uniform distribution among solutions.

Remerciements

Nous remercions <<Allah>> de nous avoir aidé à réaliser ce projet de mémoire.

Nous tenons, avant tout, à exprimer notre profonde Ogratitude à monsieur Ait Akkache professeur à l'université de Blida, qui a assumé la direction de ce travail. Qu'il veuille bien trouver ici l'expression de notre reconnaissance pour son dévouement, sa patience, sa disponibilité, ses conseils et son aide constant qu'il nous a apporté tout au long de ce travail.

Nous remercions les membres de jury et nous adressons nos vifs remerciements à tous les enseignants qui, par leur enseignement, leur encouragement et leur aide, ont contribués à notre formation durant toutes nos études à l'université de Blida.

Dédicace

A mes très chers parents qui ont été derrière cette réussite.

A mes deux frères Abdeslam et Abdenour.

A tout mes proches, et toute ma grande famille.

A toutes mes amies : B. Khadîdja, B. Mériem, Moufida, Zahra, Zinb, Amal, Amina, Hakima, Samira. B. mohamed, C. mohamed, Mahmoude, M. Billel, Ridha, Brahim, M. Brahim, Isselam.

A toutes les personnes qui m'ont soutenue et encouragée tout an long de cette année

Je dédie ce projet de mémoire.

Selma juventinia

Dédicace

A mes très chers parents qui ont été derrière cette réussite.

A mes deux frères et mes sœurs

A tout mes proches, et toute ma grande famille.

A toutes mes amies

A toutes les personnes qui m'ont soutenue et encouragée tout an long de cette année

Je dédie ce projet de mémoire.

B. Mohamed

Tableau des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

De très nombreux problèmes techniques impliquent le plus souvent l'optimisation simultanée de plusieurs objectifs, généralement contradictoires. Pour ce genre de problèmes, la notion de solution optimale unique n'a pas lieu d'être car il y a des compromis à prendre en considération. L'étude de compromis a donné lieu à la définition de solutions optimales et de dominance au sens de Pareto. L'optimisation multi-objectif est l'appellation donnée aux méthodes de résolution des problèmes citées ci-dessus. Ces méthodes sont des techniques particulièrement adaptées à la recherche de surfaces de Pareto (surfaces de compromis) à la différence de la plupart des méthodes traditionnelles (mono-objectif), qui manipulent en général un point unique dans l'espace de recherche. La majorité de ces problèmes sont qualifiés de difficiles et plus particulièrement dans le cas continu, car leur résolution nécessite l'utilisation des algorithmes évolués (en général, il n'est pas possible de fournir des solutions optimales dans un temps raisonnable).

Nous distinguons deux grandes classes de méthodes: les méthodes mathématiques et les méthodes basées sur les heuristiques « Méta-heuristiques ». Parmi les méta-heuristiques, on trouve la famille des algorithmes évolutionnaires qui ont connus un grand succès dans l'optimisation multi-objectifs.

Les algorithmes évolutionnaires (AE) sont inspirés par des concepts issus de la théorie de l'évolution de Darwin. Ce sont des techniques d'optimisation itérative et stochastique, car ils utilisent itérativement des processus aléatoires. Ils font évoluer un ensemble de solutions (dite population) d'un problème donné en utilisant une fonction d'évaluation (fitness) afin de mesurer ces valeurs, dans l'optique de trouver les meilleurs résultats. Cette particularité donne aux AEs la capacité à trouver plusieurs solutions Pareto optimales en une seule exécution.

Le front de Pareto vise à maximiser la contribution de l'hyper volume de la population en sélectionnant les individus dans un régime permanent du schéma évolutif. Alors qu'un grand coût de calcul est requis dans le calcul exact de cet indicateur qu'est l'hypervolume dans un espace de grande dimension. D'autre part, l'estimation de l'hyper volume a été récemment mise au point par l'échantillonnage de Monte Carlo, conduisant à l'apparition des algorithmes basés sur hyper-volume conçus spécialement pour optimisation de plusieurs objectifs, comme l'algorithme d'estimation d'hyper-volume (HYPE) [23] .

Ce projet propose la simulation d'un AE basé sur une grille [24] (GrMOEA) pour résoudre des problèmes d'optimisation multi objectives continues. Le but de ce document

est d'exploiter le potentiel de l'approche basée sur les grilles pour renforcer la pression de sélection dans la direction optimale tout en maintenant une vaste distribution et uniforme parmi les solutions.

Ce mémoire est organisé en quatre chapitres comme suit:

Le premier chapitre est consacré aux principes de base de l'optimisation multi-objectif. On donnera aussi les phases de résolution d'un problème d'optimisation multi-objectif (MOP) et on finira par quelques méthodes de résolution de problème multi-objectif

Le deuxième chapitre nous nous intéresserons de manière approfondie à l'évolution artificielle, Celui-ci présente des notions nécessaires permettant la compréhension des algorithmes évolutionnaires, nous insisterons aussi sur le rôle de ces algorithmes au cas de l'optimisation multi-objectif.

Le troisième chapitre constitue le noyau de ce mémoire. Nous présenterons l'algorithme évolutionnaire multi- objectif a base grille ainsi que les techniques de simulation.

Le quatrième chapitre des expérimentations sont effectuées sur des problèmes tests pour évaluer notre simulation.

Enfin, on termine par une conclusion générale qui fera la synthèse de notre travail en ouvrant de nouvelles perspectives.

Chapitre 1
Optimisation Multi-Objectif

1.1 Introduction

La plupart des problèmes du monde réel nécessitent l'optimisation simultanée de plusieurs objectifs souvent irrationnels (définis en unités incomparables) et présentent un certain degré de conflit entre eux (par exemple, un objectif ne peut pas être amélioré sans la détérioration d'au moins un des autres objectifs). Ces problèmes sont appelés problème d'optimisation multi-objectif ou multicritères.

Dans l'optimisation mono-objectif, la solution optimale est généralement bien définie, cela n'est pas le cas pour les problèmes multi-objectifs. Au lieu d'une solution optimale, il existe plutôt un ensemble de solutions qui sont des « compromis » ; cet ensemble est généralement dénommé « l'ensemble des solutions optimales de Pareto ». Ces solutions sont optimales dans le sens qu'aucune autre solution dans l'espace de recherche n'est meilleure à elles, lorsque tous les objectifs sont considérés simultanément. [3]

1.2 Optimisation multi-objectif

1.2.1 Définition :

Formellement un problème d'optimisation multi-objectif (MOP) est défini comme suit:

$$\begin{cases} Minimiser\ \boldsymbol{F(x)} = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_M(\boldsymbol{x})]^t \\ sous\ contrainte\ \boldsymbol{x} \in \boldsymbol{\Omega} \end{cases}$$

Le vecteur $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ est le vecteur à n variables de décision. L'ensemble faisable $\boldsymbol{\Omega} \subseteq \boldsymbol{\mathcal{R}^n}$ est implicitement déterminé par un ensemble de K contraintes d'égalité de la forme $h_k(x) = 0$, $(k = 1,2, \dots, K)$ et de J contraintes d'inégalité $g_j(x) \geq 0$ $(j = 1,2, \dots, J)$. La fonction vectorielle $F: \boldsymbol{\Omega} \longrightarrow \boldsymbol{\mathcal{R}^M}$ est composée de M $(M \geq 2)$ fonctions scalaires $f_i: \boldsymbol{\mathcal{R}^n} \longrightarrow \boldsymbol{\mathcal{R}}(i = 1, \dots, M)$. Dans l'optimisation multi-objectif, les ensembles $\boldsymbol{\mathcal{R}^n}$ et $\boldsymbol{\mathcal{R}^M}$ sont appelés respectivement espace de variables de décision et espace des objectifs. La figure 1.1 représente un exemple avec 2 variables de décision et 2 fonctions objectives. [26]

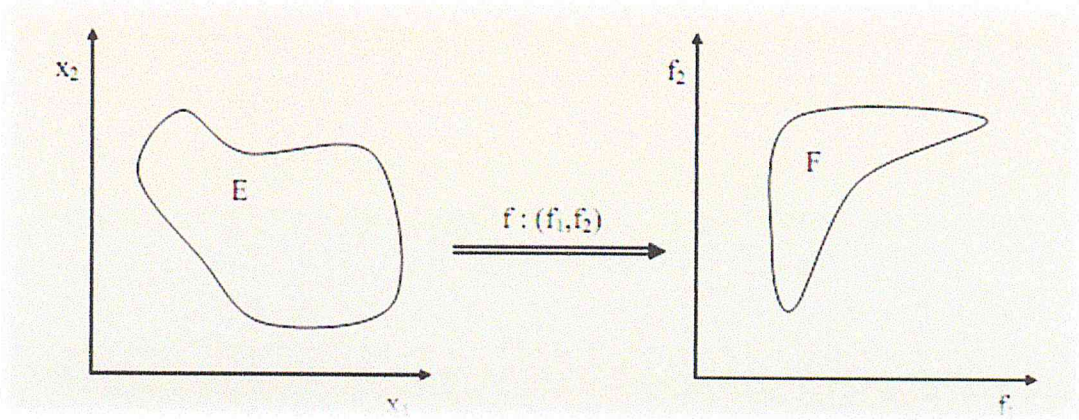


Figure 1.1 : illustration de l'espace faisable et l'espace des objectifs. [2]

1.3 Notions d'optimalité dans un MOP

Afin de définir plus précisément le problème d'optimisation multi-objectif présenté ci-dessus, nous devons établir le sens de la minimisation dans $\mathcal{R}^M$. Autrement dit, il est nécessaire de définir comment des vecteurs $f(x) \in \mathcal{R}^M$ doivent être comparés pour différentes solutions $x \in \mathcal{R}^n$. La relation "inférieure ou égale" utilisée dans l'optimisation mono-objective impose un ordre total dans l'espace des solutions. En revanche, dans les problèmes d'optimisation multi-objectif, il n'y a pas d'ordre canonique sur $\mathcal{R}^M$, et donc, nous avons besoin de définitions plus faibles afin de comparer les vecteurs de $\mathcal{R}^M$. Dans l'optimisation multi-objectif, la relation de dominance Pareto à l'origine proposé par Edgeworth en 1881 [29], et plus tard généralisée par l' économiste Vilfredo Pareto en 1896 [27] est généralement adoptée.

1.3.1 Relation Dominance Pareto

On dit qu'un vecteur $Z^{(1)}$ domine le vecteur $Z^{(2)}$, noté $Z^{(1)} \prec_{Pareto} Z^{(2)}$ si les deux conditions sont vérifiées :

- $\forall i \in \{1,2,\dots,M\}, \quad {Z^{(1)}}_i \leq {Z^{(2)}}_i$.
- $\exists j \in \{1,2,\dots,M\}, \quad {Z^{(1)}}_j < {Z^{(2)}}_j$.

1.3.2 Pareto optimalité :

Une solution $x^* \in \Omega$ est dit Pareto optimale s'il n'existe pas une autre solution $x \in \Omega$ telle que, [2]

$$f(x) \prec_{Pareto} f(x^*).$$

L'ensemble des solutions Pareto optimales et son image dans l'espace objectif est défini comme suit

1.3.3-Ensemble Pareto optimal : L'ensemble optimum de Pareto. P_{Pareto}, est défini par :

$$P_{Pareto} = \{x \in \Omega | \nexists\, y \in \Omega : F(y) \prec_{Pareto} F(x)\}.$$

1.3.4-Front de Pareto : Pour un ensemble optimal de Pareto, P_{Pareto}, le front de Pareto, PF_{Pareto}, est défini comme suit:

$$PF_{Pareto} = \{F(x) = [f_1(x), f_2(x), \dots, f_M(x)]^t | x \in P_{Pareto}\}$$

La figure 1.2 montre un exemple de front de Pareto pour un problème de minimisation à deux objectifs.

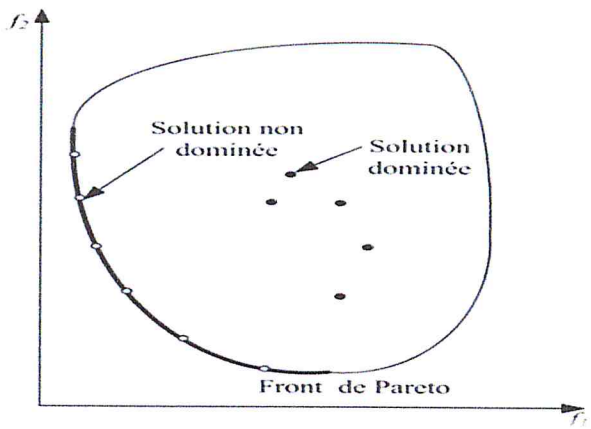


Figure 1.2 : front de Pareto.

L'ensemble de points blancs représentent le front de Pareto.

1.3.4.1-Point idéal et point nadir

On observe deux points caractéristiques associés à une surface de compromis (front de Pareto) :

- **Point idéal**

Les coordonnées de ce point sont obtenues en optimisant chaque fonction objective séparément. On dit aussi que les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des points du front de Pareto (voir figure 1.3).

- **Point nadir**

Les coordonnées de ce point correspondent aux pires valeurs obtenues par chaque fonction objective lorsque l'on restreint l'espace des solutions à la surface de compromis (voir figure1.3).

Le point idéal est souvent utilisé dans les 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.

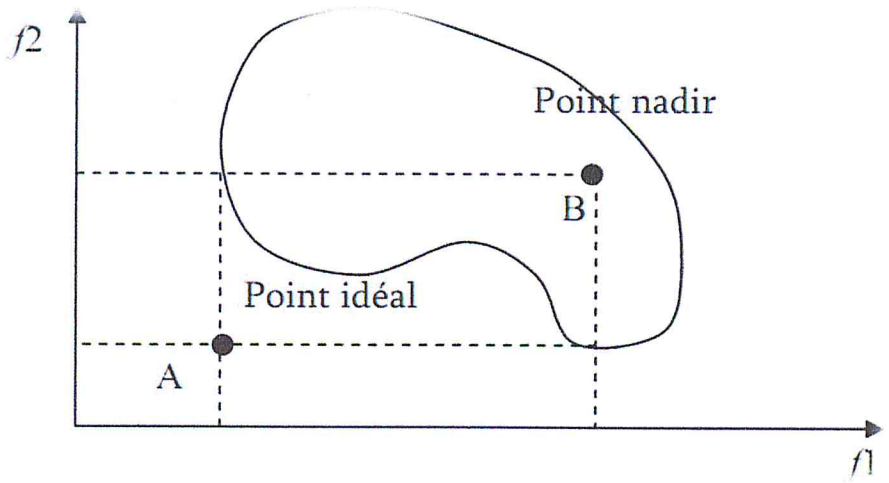


Figure 1.3 : représentation de point idéal et point nadir. [7]

1.4 Les phases de résolution d'un MOP

La résolution de problèmes multi-objectifs relève de deux disciplines assez différentes. En effet, résoudre un problème multi-objectif peut être divisé en deux phases

- **La recherche des solutions de meilleur compromis** : C'est la phase d'optimisation multi-objectif et elle consiste en la détermination du front de Pareto
- **Le choix de la solution à retenir :** C'est la tâche du décideur qui, parmi l'ensemble des solutions de compromis (souvent composé d'un grand ou même un nombre infini de solutions), doit extraire celle(s) qu'il utilisera. On parle alors ici de décision multi-objective et cela fait appel à la théorie de la décision.

Dans le cadre de ce mémoire nous ne parlerons que de la première phase qui consiste en la recherche des solutions de meilleurs compromis.

1.5 Les Méthodes d'optimisation

Les méthodes d'optimisation peuvent être divisées en deux catégories (voir figure 1.4): les méthodes déterministes et les méthodes probabilistes. Les algorithmes déterministes (Exacts) sont le plus souvent utilisés si une relation claire entre les caractéristiques des solutions possibles et leur utilité pour un problème donné existe. Car dans ce cas, l'espace de recherche peut efficacement être exploré en utilisant par exemple un schéma du genre diviser pour mieux régner (Branch and Bound). Dans le cas contraire ou si la dimension de l'espace de recherche est très élevée, il devient plus difficile à résoudre avec une procédure déterministe.

Ainsi, les algorithmes stochastiques (Approché) entrent en jeu. Une famille particulièrement pertinente des algorithmes probabilistes sont des approches fondées sur les méthodes de Monté Carlo. Ils font compenser l'exactitude de la solution par la vitesse de la recherche. Cela ne signifie pas que les résultats obtenus; en utilisant ces procédés stochastiques sont incorrects, car d'autre part, une solution moins bonne par rapport à la meilleure possible, est plus efficace que celle qui nécessite 10^{100}années à trouver. [26]

1.5.1 Heuristique

Une heuristique est une méthode approchée se voulant simple et rapide. Généralement les méthodes heuristiques utilisent des règles simples pour optimiser un ou plusieurs critères. Le principe général de cette catégorie est l'intégration des stratégies de décisions pour avoir une solution approchée de l'optimale en un temps raisonnable.

1.5.2 Méta-heuristique

Une méta-heuristique est une méthode de recherche pour les problèmes de classes très générales. Elle combine des fonctions objectives ou heuristiques d'une manière abstraite et que nous espérons efficace, habituellement sans utiliser de connaissance plus approfondie sur leur structure, i.e. en les traitant comme une procédure de boîte noire.

Cette combinaison est souvent réalisée stochastiquement en utilisant des statistiques obtenues à partir d'échantillons de l'espace de recherche ou basées sur un modèle d'un phénomène naturel ou d'un processus physique. Le recuit simulé, par exemple, décide quelle solution candidate évaluer en fonction de la loi de probabilité de Boltzmann. Les algorithmes évolutionnaires copient le comportement de l'évolution naturelle et traite les solutions candidates en tant qu'individus qui sont en concurrence dans un environnement virtuel. [26]

Une classe importante de méta-heuristiques probabilistes de Monte Carlo est le calcul évolutionnaire. Elle englobe tous les algorithmes qui sont basés sur un ensemble de plusieurs solutions candidates (appelé population) qui sont itérativement affinées. Ce domaine de l'optimisation est également une classe de "Soft Computing" ainsi qu'une partie du domaine de l'intelligence artificielle. Certains de ses membres les plus importants sont les algorithmes évolutionnaires qui seront étudiés dans le chapitre II.

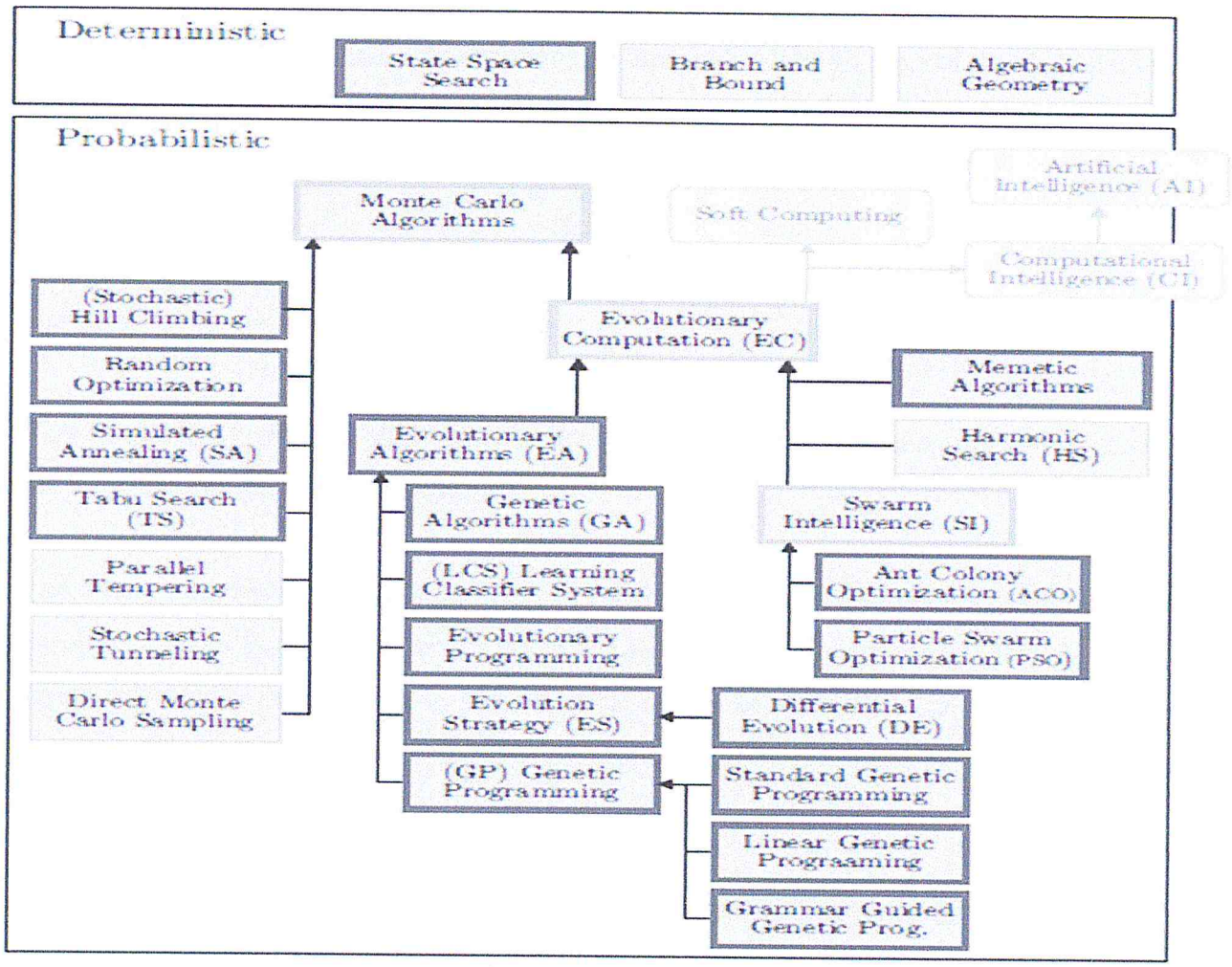


Figure 1.3 : La taxonomie des méthodes d'optimisation globale.[26]

1.6 Approches de résolution d'un MOP

Parmi ces méthodes en trouve :

1.6.1 Approches à base de transformation du problème vers le mono-objectif

Dans la résolution de **MOP**, plusieurs méthodes traditionnelles transforment le **MOP** en un Problème mono-objectif.

soient disponibles. L'autre problème avec cette approche est la détermination des poids, sans avoir des connaissances sur le problème traité.

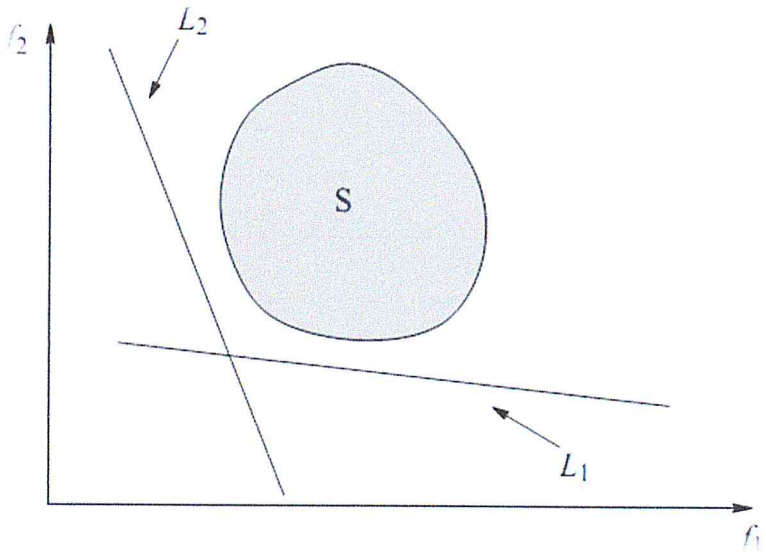


Figure 1.4 : la méthode de pondération des fonctions objectives. [7]

1.6.1.2 La méthode ε-contraintes

Dans cette approche, le problème consiste à optimiser une fonction f_k sujette à des contraintes sur les autres fonctions.

$$(MOP(\varepsilon))\begin{cases} Minimiser\ f_k(x) \\ x \in C \\ s.c \quad f_j(x) \leq \varepsilon_j,\ j = 1,2,\dots,M \text{ où } j \neq k \end{cases}$$

Où $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_{k-1}, \varepsilon_{k+1}, \dots, \varepsilon_M)$.

Ainsi, un problème mono-objectif (objectif f_k) sujet à des contraintes sur les autres objectifs est résolu. Différentes valeurs ε_j peuvent être données pour pouvoir générer différentes solutions de Pareto optimales. La connaissance à priori des intervalles appropriés pour les valeurs ε_j est requise pour tous les objectifs. Pour pouvoir définir les valeurs adéquates pour ε_j , le vecteur idéal doit être calculé pour déterminer les bornes inférieures. On aura donc :

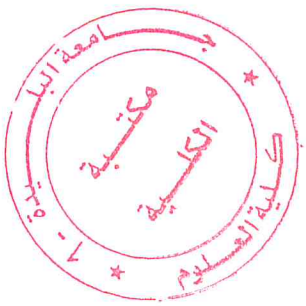
$$\varepsilon_j \leq f_j(x^*) \quad j = 1,2,\dots,k-1,k+1,\dots,M$$

Les approches basées sur la programmation par but trouvent une solution non dominée si le but est choisi dans le domaine réalisable. Cependant, le décideur est chargé de choisir le vecteur but et le poids associés à chaque objectif, ce qui est une tâche difficile sans connaître la structure de l'espace de recherche. Si le domaine réalisable n'est pas facile à approcher, la méthode peut être inefficace.

Ces approches qui consistent à transformer le problème multi objectif du départ (MOP) vers un problème mono-objectif est un processus à répétition souvent très long qui demande des relations claires entre les caractéristiques des solutions possibles et nécessite une connaissance à priori sur les préférences du décideur. Tout cela consolide la tendance qui stipule de résoudre le MOP dans sa forme initiale.

1.7 Conclusion

On a essayé dans ce chapitre de rassembler un état de l'art sur les problèmes d'optimisation multi-objectifs, avec la manière de définir un problème pareil, ses contraintes, ses objectifs ainsi que les méthodes utilisées pour le résoudre tout en respectant le concept de compromis et les frontières de Pareto. On a parlé sur les méthodes utilisées dans le domaine de l'optimisation multi-objectif soit les problèmes mathématiques ou les méta-heuristiques, parmi les méta-heuristiques existantes, on trouve les algorithmes évolutionnaires qu'on présentera en détails dans le chapitre suivant.



Chapitre 2
Evolution Artificiel

2.1 Introduction :

Les algorithmes évolutionnaires sont parmi les méta-heuristiques à base de population, ils sont inspirés de la biologie tout en introduisant le théorème de Darwin dans l'évolution des espèces. Les AE constituent une discipline impliquant la simulation du processus de l'évolution naturelle sur un ordinateur. Ceci peut être vu comme un processus d'optimisation où des individus évoluent dans le temps, afin de devenir de plus en plus adéquats sur un environnement donné, pour le problème à résoudre.
Les algorithmes évolutionnaires ont résolu avec succès des problèmes d'optimisation difficiles dans divers domaines, et ceci a suscité un très grand intérêt pour le domaine de recherche connu sous le nom de calcul évolutionnaire.

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 Ω sont appelés des individus ;
-Un ensemble fini d'individus est appelé population ;
-La fonction objective 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, basée sur leur performance ;
-Le remplacement est le processus de formation d'une nouvelle population à partir des parents et des enfants. [3]

2.3 Principes de fonctionnement des algorithmes évolutionnaires

Comme présenté par l'organigramme de la Figure 2.1, le principe de fonctionnement d'un AE est extrêmement simple. On part d'un ensemble initial d'individus (chacun correspondant à une valeur du jeu de paramètres), nommé « population parent initiale » et générée le plus souvent d'une manière aléatoire dans l'espace de recherche. Ensuite, on évalue de manière exacte la performance de chaque individu en calculant la valeur de la fonction coût correspondant à son jeu de paramètres. L'application des trois opérateurs génétiques de « sélection, croisement et mutation » permet de créer un nouvel ensemble d'individus appelé « population enfant ». Cette population est évaluée à son tour pour indiquer la performance de chacun de ses individus. Cette connaissance permet de décider lesquels des individus enfants méritent de remplacer certains parents. La nouvelle population obtenue, appelée « population parent », constitue la population parent de la nouvelle génération. Si les critères d'arrêt sont vérifiés, on considère la ou les solutions obtenues comme satisfaisantes, autrement, on recommence le cycle jusqu'à satisfaction de ces critères. Le critère d'arrêt d'un AE peut être la convergence de l'ensemble des individus de la génération courante vers un même extremum, dans le cas d'un problème mono-objectif, ou alors vers un même ensemble optimal de Pareto, dans le cas multi-objectifs. Le plus souvent, l'algorithme est arrêté au bout d'un nombre d'itérations (générations) fixé a priori. [32]

Algorithme 2.1 : Principe des approches évolutionnaires

Générer la population initiale$P$
Evaluer les individus de P
tant que le critère d'arrêt n'est pas atteint **faire**
  $P' \rightarrow$ Croiser les individus de P
  $P'' \rightarrow$ Muter les individus de P'
  Evaluer les individus de P''
  Sélectionner la nouvelle génération P à partir de P et P''.
fin

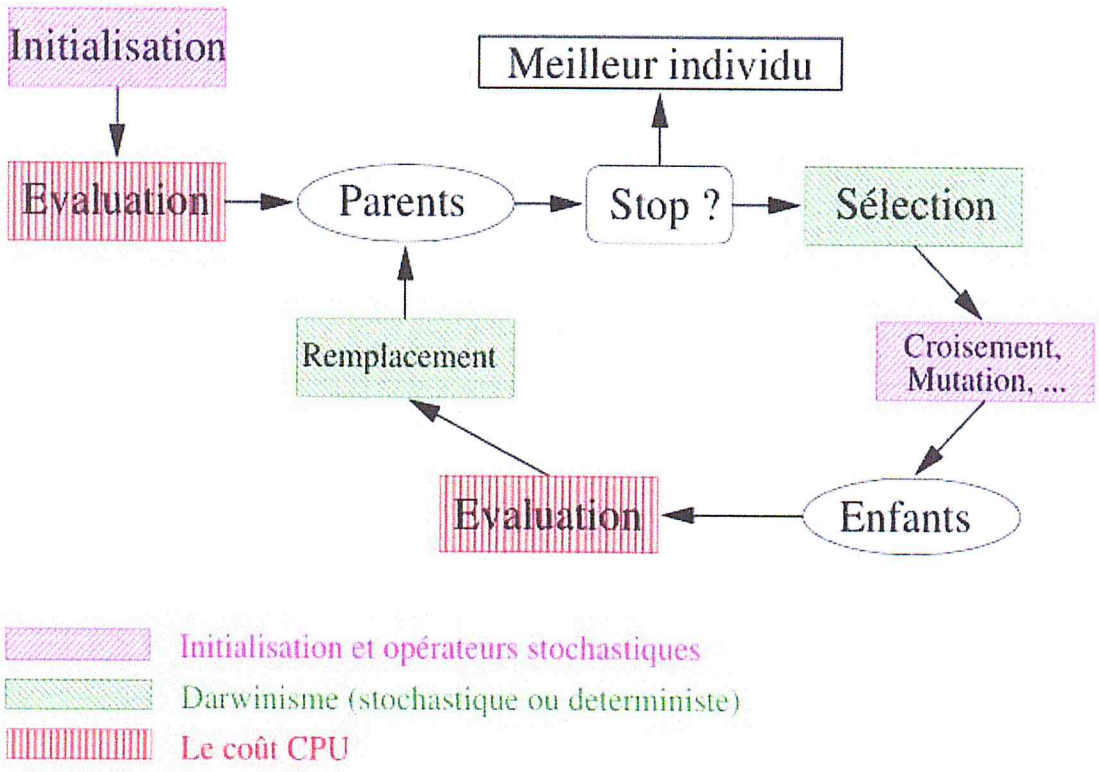


Figure 2.1 : cycle d'un algorithme évolutionnaire.[5]

2.4 Les catégories principales d'algorithmes évolutionnaires

On distingue quatre grandes familles historiques d'algorithmes et les différences entre elles ont laissé des traces dans le paysage évolutionnaire actuel, en dépit d'une unification de nombreux concepts.

2.4.1 Algorithmes génétiques (AG) :

Les AGs sont pratiquement les algorithmes les plus connus et utilisés dans le calcul évolutionnaire. Ils ont été développés dans les années 60, pour étudier le processus complexe d'adaptation des espèces naturelles. L'opérateur de croisement est considéré comme étant le plus important des opérateurs. La mutation est appliquée avec des probabilités très faibles et agit en tant qu'opérateur d'arrière-plan. Le codage des solutions consiste en une représentation généralement binaire des individus. [10] [11]

2.4.2 Stratégies d'évolution (SE):

Les SEs ont été mises au point par deux jeunes ingénieurs travaillant sur des problèmes numériques. Le contexte était l'optimisation paramétrique et les schémas d'évolution. Ces techniques ont été développées dans le but de résoudre des problèmes d'optimisation numériques dans un espace de paramètres réels. Le codage des solutions peut être réalisé par des structures de données plus complexes que dans les algorithmes génétiques. Par ailleurs, les opérateurs de mutation ont une place aussi importante que les opérateurs de croisement. [17]

2.4.3 Programmation évolutionnaire (PE) :

La programmation évolutionnaire est apparue dans l'espace des automates à états finis pour l'approximation de séries temporelles. Imaginée au début par Larry Fogel pour la découverte d'automates à états finis, PE a rapidement travaillé sur des espaces de recherche très variés.

La PE est fondée essentiellement sur l'opérateur de mutation et n'utilise pas d'opérateur de croisement. Comme les Stratégies d'évolution, le codage des solutions peut faire intervenir des structures de données complexes. Cette approche a été développée initialement pour faire évoluer des automates à états finis. [12]

2.4.4 Programmation génétique(PG) :

Apparue initialement comme sous domaine des AGs, PG est devenu une branche à part entière (conférence, journal, ...). La spécificité de cette technique consiste à faire évoluer des structures en arborescence représentant des programmes complets. [14]

2.5 Le dilemme exploration-exploitation

A chaque étape de l'algorithme, il faut effectuer le compromis entre explorer l'espace de recherche, afin d'éviter de stagner dans un optimum local, et exploiter les meilleurs individus obtenus, afin d'atteindre de meilleures valeurs aux alentours. Trop d'exploitation entraîne une convergence vers un optimum local, alors que trop d'exploration entraîne la non-convergence de l'algorithme.

Typiquement, les opérations de sélection et de croisement sont des étapes d'exploitation, alors que l'initialisation et la mutation sont des étapes d'exploration. On peut ainsi régler

les parts respectives d'exploration et d'exploitation en jouant sur les divers paramètres de l'algorithme (probabilités d'application des opérateurs, pression de sélection, ...). Malheureusement, il n'existe pas de règles universelles de réglages et seuls des résultats expérimentaux donnent une idée du comportement des diverses composantes des algorithmes. [25]

2.6 Représentation des individus

Dans un algorithme évolutionnaires, les solutions sont représentées par leur génotype (encore appelé chromosome ou individu), qui s'exprime sous la forme d'un phénotype. Le génotype se compose de plusieurs gènes, où chaque gène à une valeur possible et une position spécifique dans son chromosome. Le génotype représente le codage tandis que le phénotype représente la solution elle-même. Ainsi, dans le cas d'un codage direct, le génotype et le phénotype sont similaires. Le génotype contient toutes les informations nécessaires à la définition complète du phénotype. Par contre, ils présentent des différences plus importantes dans le cas d'un codage indirect.

Pour représenter les solutions dans la population, on distingue deux types de codage (génotypes) différentes :

- **Le codage binaire :**

Lorsque la solution peut être représentée par une chaine de bits (nombres binaires), où chaque gène peut prendre seulement les valeurs 0 ou 1.

- **Le codage entier ou réel :**

Cette représentation a été introduite initialement pour les Stratégies d'Evolution, mais son utilisation s'est étendue rapidement aux autres types d'algorithmes évolutionnaires. Là, l'espace de recherche est l'espace des entiers ou celui de réels, dans lesquels les variables ont des intervalles de définition en fonction des contraintes du problème. Chaque individu (solution) se compose d'un (ou plusieurs) vecteur de valeurs entières ou réelles.

2.6.1 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.

- **Croisement intermédiaire**

$$\forall i \{1,2,\dots,n\}, x_i^{'} = \alpha x_{S,i} + (1-\alpha)x_{T,i}.$$

Où α est une variable aléatoire uniforme appartenant à l'intervalle[0,1], ($\alpha = U[0,1]$) et S et T sont deux individus sélectionnés pour l'ensemble des composantes de $\vec{x}$. [3]

- **Blend alpha Crossover (BLX- α)**

A partir de deux parents $X = (x_1, x_2, \dots, x_n)$ et $Y = (y_1, y_2, \dots, y_n)$, BLX génère une nouvelle solution $Z = (z_1, z_2, \dots, z_n)$ où $z_i \in [c_{min} - I\alpha, c_{max} + I\alpha]$.

$c_{max} = max(x_i, y_i)$, $c_{min} = min(x_i, y_i)$, $I = c_{max} - c_{min}$, et α est une constante. [19]

- **Croisement binaire simulé (SBX)**

A partir de deux parents $X = (x_1, x_2, \dots, x_n)$ et $Y = (y_1, y_2, \dots, y_n)$, le croisement SBX génère deux nouvelles solutions $Z^1 = ({z_1}^1, {z_2}^1, \dots, {z_n}^1)$, $Z^2 = ({z_1}^2, {z_2}^2, \dots, {z_n}^2)$ de la manière suivante [19]:

Etape1 : Générer aléatoirement un nombre uniforme $u \in [0\ ,1]$.

Etape2 : Générer un nombre β par :

$$\beta = \begin{cases} (2u)^{1/(1+\eta_c)} \ si\ u < 0.5 \\ (1/2(1-u))^{1/(1+\eta_c)} \ sinon \end{cases}$$

Où η_c désigne l'indice de distribution pour l'opérateur SBX.

Etape3 : générer deux solutions par :

$${z_i}^1 = 0.5[(1+\beta)x_i + (1-\beta)y_i]$$
$${z_i}^2 = 0.5[(1-\beta)x_i + (1+\beta)y_i]$$

2.6.2 Mutation

Après le croisement, un opérateur de mutation est appliqué au sein de la population des enfants avec une certaine probabilité, nommée taux de mutation ou P_m. Cet opérateur joue le rôle d'un (élément) perturbateur, il introduit du bruit, il permet ainsi de maintenir la diversité de la population des enfants et d'explorer l'espace de recherche en évitant à l'algorithme de converger trop rapidement vers un optimum local. Des petits taux de mutation sont recommandés car un grand taux peut occasionner une destruction de l'information utile contenue dans les solutions.

Plusieurs opérateurs de mutation existent pour le codage nous citons :

- **Mutation Gaussienne :**

Le principe de l'opérateur de mutation réelle consiste généralement à ajouter une perturbation arbitraire tirée selon une distribution de probabilité Gaussienne aux différentes composantes de l'individu $X = (x_1, x_2, \dots, x_n)$, où n est le nombre de composantes de X. Cette procédure s'effectue de cette manière:

$$x_i = x_i + \sigma N(0.1),$$

Où σ est l'écart type de la mutation et $N(0.1)$ représente une loi normale centrée réduite. La difficulté de cette approche est l'ajustement des déviations standards des variables Gaussiennes utilisées. En effet, si la déviation standard est trop petite, les déplacements dans l'espace de recherche sont insuffisants au début de l'algorithme, et l'algorithme peut rester au voisinage d'un optimum local et ne permet pas de visiter/découvrir de nouvelles régions de l'espace de recherche. Par contre, si l'écart est élevé, l'algorithme pourra accéder à une région contenant l'optimum, mais la qualité de convergence ne sera pas satisfaisante.

- **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 ces composantes par la formule suivante :

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

Où Δ_{max} est la variation utilisée et δ_q le coefficient de variation, calculé de la manière suivante :

$$\delta_q = \begin{cases} (2u)^{1/(\eta_m+1)} - 1 & si \quad u \leq 0.5 \\ 1 - (2(1-u))^{1/(\eta_m+1)} & sinon \end{cases}$$

Où u est un nombre aléatoire tirer uniformément dans l'intervalle [0, 1], et η_m l'indice de distribution.

2.7 Darwinisme artificiel

2.7.1 Procédures de sélection

La partie darwinienne de l'algorithme comprend les deux étapes de sélection et de remplacement. On distingue deux catégories de procédures de sélection. Ces deux étapes sont totalement indépendantes de l'espace de recherche. Les procédures de sélection les plus fréquemment utilisées :

2.7.1.1 Sélection proportionnelle (par roulette)

Il s'agit de représenter sur une roulette chacun des individus de la population par une section qui est proportionnelle à leur fitness. Ensuite, on lance P fois la roulette et on sélectionne chacun des gagnants.

Cette méthode de sélection favorise les meilleurs individus, mais les mauvais ont tout de même des chances d'être sélectionnés. Par contre, le coût d'exécution et la variance sont élevés. [5]

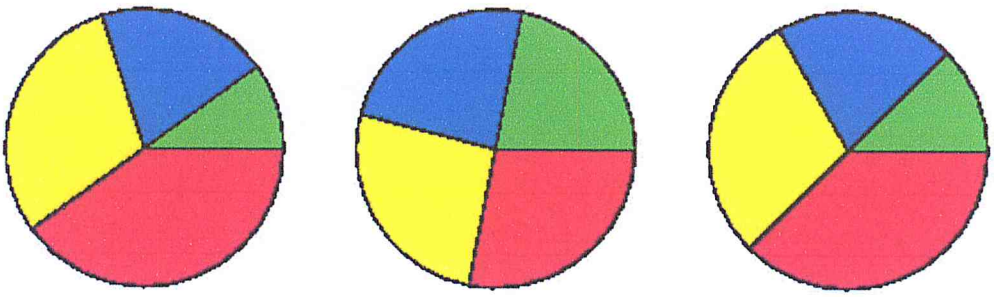


Figure 2.2 : la sélection par roulette.

2.7.1.2 Sélection par tournoi :

La sélection par tournoi n'utilise aussi que des comparaisons entre les individus, et ne nécessite même pas de tri de la population. Elle possède un paramètre T, qui représente la taille du tournoi. Pour sélectionner un individu, on tire de façon uniforme T individus

parmi la population, et on sélectionne d'une manière déterministe le meilleur de ces T individus. Au cours d'une génération il y a autant de tournois que d'individus à sélectionner. Cette technique est caractérisée par une pression de sélection en général plus forte que les méthodes proportionnelles. De plus, elle est la moins chère en terme de coût d'exécution ; elle est peu sensible aux erreurs sur F, facilement paramétrable par la valeur de T, et ne conduit pas à une convergence prématurée. [33]

2.7.2 Remplacement des individus :

Une variété de procédures de remplacement peut être utilisée, le principe étant de remplacer l'ancienne population (population Parent) par une nouvelle (population enfant), obtenue après application des opérateurs de variation stochastiques. Dans les algorithmes génétiques classiques, le remplacement s'effectue d'une manière générationnelle, c'est-à-dire que la population des enfants remplace purement et simplement la population des parents.

Toutefois, Il existe d'autres stratégies de remplacement dont :

-Le remplacement d'un pourcentage des individus de l'ancienne génération par les meilleurs enfants.

-Le remplacement systématique du plus mauvais individu.

-Le remplacement aléatoire.

Le but est d'augmenter la vitesse de convergence de l'algorithme génétique simple, mais il peut cependant engendrer une convergence rapide vers des optima locaux.

2.8 Optimisation évolutionnaire multi-objectif (MOEA)

Dans cette section, on va présenter quelques approches proposées afin d'adapter les algorithmes évolutionnaires au cas de l'optimisation multi-objectif. L'objectif principal de ces différentes approches est le même : obtenir une bonne approximation du front optimal tout en assurant une bonne diversité des solutions le long de ce dernier. Toutefois, ces méthodes sont différentes entre elles dans la façon d'aborder cette problématique en utilisant des procédures de sélection différentes. Avant de présenter les principaux MOEA, nous nous rassemblons dans un premier temps sur ce qui fait la différence entre ces algorithmes à savoir les différentes méthodes de comparaison pouvant être utilisées pour sélectionner les solutions. Comme ces méthodes de sélection

font parfois recourt aux indicateurs de performance, nous commençons d'abord par présenter ces derniers.

2.8.1 Indicateurs de Performance

L'évaluation de la performance des MOEA peut avoir plusieurs aspects, comme par exemple la qualité des résultats obtenus, ou le temps de calcul nécessaire. Nous nous focalisons seulement sur le premier aspect étant donné que le temps d'exécution des algorithmes devient négligeable dès lors qu'il s'agit d'une application réelle ou l'évaluation des objectifs peut durer de quelques minutes à quelques jours. [13]

2.8.1.1 Indicateur Hypervolume

L'indicateur de l'hyper volume mesure le volume de la portion faiblement dominée par un ensemble de point A, dans l'espace des objectifs. Le calcul de ce volume nécessite la désignation d'un point de référence, qui soit de préférence dominé par la totalité des points de l'ensemble A (voir figure 2.3). L'indicateur de l'hypervolume est souvent calculé par rapport à un ensemble de référence R, cet indicateur est noté I_H^- et est défini comme suit :

$$I_H^-(A) = I_H(R) - I_H(A)$$

où plus la valeur de I_H^- est petite, plus la qualité est meilleure. L'indicateur de l'hyper-volume permet de prendre en compte à la fois la convergence de l'algorithme et la diversité des solutions trouvées. Toutefois, le coût de calcul est élevé : la complexité est exponentiellement proportionnelle au nombre d'objectifs. [13]

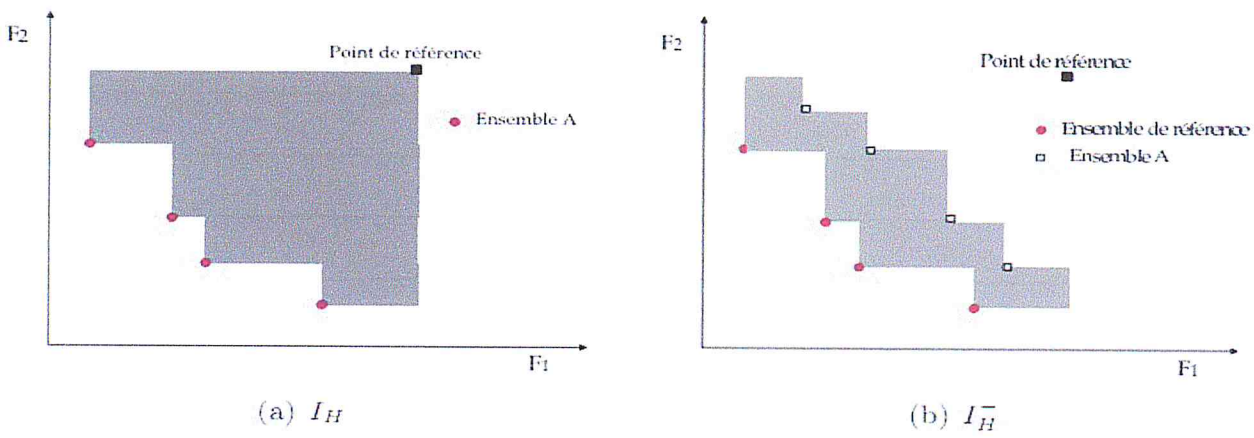


Figure 2.3 : indicateur de l'hypervolume. [13]

Lorsque l'ensemble PF_{Pareto} est connu, les mesures calculent le plus souvent la distance entre l'approximation PF^*_{Pareto} et PF_{Pareto}.

2.8.1.2-La distance générationnelle (GD)

Cette mesure calcule la distance moyenne entre les solutions de PF^*_{Pareto} et celle de PF_{Pareto}. Elle se calcule selon la formule suivante :

$$GD=\frac{\left(\sum_{i=1}^{|PF^*_{Pareto}|} d_i^p\right)^{\frac{1}{p}}}{|PF^*_{Pareto}|}$$

Pour $p = 2$, le paramètre d_i représente la distance Euclidienne (dans l'espace des objectifs) entre solution $i \in PF^*_{Pareto}$ et le membre le plus proche PF_{Pareto}. [4]

2.8.1.3 La distance générationnelle inversée (IGD)

Cette métrique mesure à la fois la convergence et de la diversité. Soient PF_{Pareto} un ensemble des solutions distribué de manière uniforme dans le vrai front de Pareto et X l'ensemble des solutions non dominées dans le front de Pareto approximé PF^*_{Pareto}.

$$IGD=\frac{\sum_{v \in PF_{Pareto}} d(v,X)}{|PF_{Pareto}|}$$

$d(v,X)$, désigne la distance Euclidienne minimum entre v et les points de X. Pour avoir une faible valeur d'IGD, l'ensemble X devrait être proche de PF_{Pareto}. [22]

2.8.2 Procédures de comparaison

L'utilisation d'un algorithme évolutionnaire dans un contexte multi-objectif nécessite de pouvoir associer une valeur scalaire unique (la fitness), au vecteur des objectifs. Ce principe appelé ranking, consiste à classer les individus en leur donnant un rang. La valeur d'adaptation est alors attribuée à chaque individu en se basant sur son rang. Cette fitness sera utilisée dans l'étape de sélection de l'algorithme (c'est ce mécanisme de sélection de front Pareto qui offre une alternative élégante et efficace aux algorithmes évolutionnaires de s'adapter facilement au cas multi-objectif).

Plusieurs méthodes de ranking ont été utilisées dans la littérature. Cet ordre dépend de la notion de dominance et donc directement de l'optimalité Pareto. La méthode de ranking permet ainsi de converger vers les solutions Pareto optimales. [18]

2.8.2.1 Rang de Pareto

Tous les individus non dominés de la population possèdent le rang 1. Ces individus sont ensuite enlevés de la population, et l'ensemble suivant d'individus non dominés est identifié et on leur attribue le rang 2. Ce processus est réitéré jusqu'à ce que tous les individus de la population aient un rang. Cette méthode de ranking a été utilisée dans les Algorithmes génétiques pour la résolution de plusieurs problèmes (algorithme NSGA). La probabilité de sélection est ensuite affectée à chaque individu en se basant sur le rang. [15]

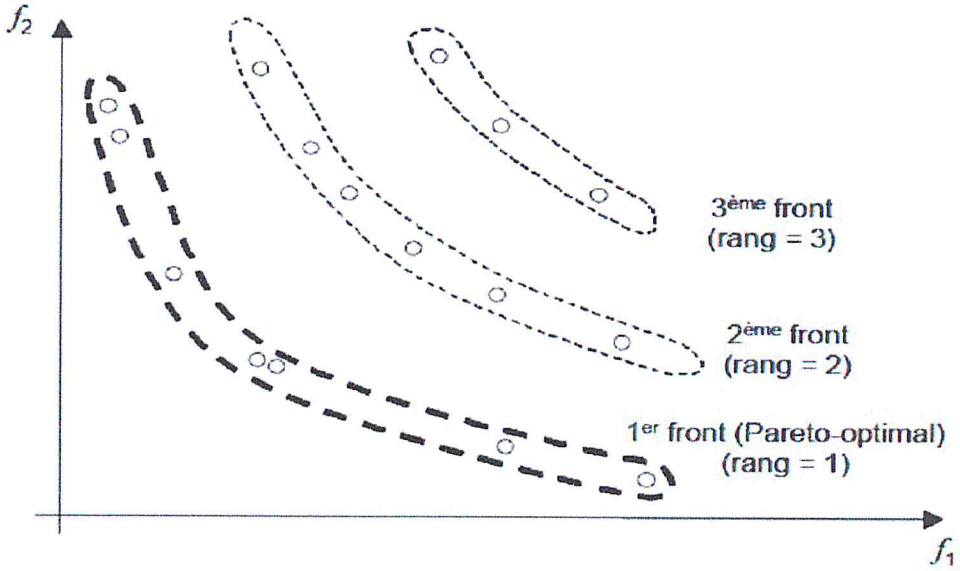


Figure 2.4: classification des individus selon le rang de Pareto.

2.8.2.2 Le Strength

Dans cette méthode, le rang d'un individu est le nombre de solutions dominant l'individu plus un. Considérons par exemple un individu i à la génération t, qui est dominé par p_i^t individus dans la population courante. Son rang dans la population est donné par

$$S(i,t) = 1 + p_i^t.$$

Un individu non dominé de la population possède donc le rang 1. Les rangs associés à cette méthode sont toujours supérieurs à ceux de la méthode NSGA. Ce type de ranking induit donc une plus forte pression de sélection, et peut causer une convergence prématurée. [21]

2.8.3 Les approches évolutionnaire

- NSGA-II : Non-dominated Sorting Genetic Algorithm. [15]
- SPEA2 : Strenght Pareto Evolutionary Algorithm.[21]
- ε-MOEA : ε – Multi-Objective Evolutinary Algorithme.[16]
- NPGA : Niched-Pareto Genetic Algorithm.[3]

2.8.4 Maintenir la diversité

Dans le but d'approximer l'ensemble optimal de Pareto en une seul exécution, les optimiseurs évolutionnaires doivent effectuer une recherche multimodal où de multiple solutions, largement différentes les unes des autres doivent être trouvées. Donc, maintenir une diversité dans la population est crucial pour l'efficacité de l'MOEA. Malheureusement, un simple EA a tendance à converger vers une solution unique et perd souvent les autres solutions en raison de trois effets :

- La pression de la sélection.
- Le bruit de la sélection.
- La perturbation due aux opérateurs.

La pression de la sélection représente le temps au bout duquel la population est complètement remplie par le même individu lorsque seulement la sélection est utilisée. Le bruit de la sélection est relatif à la discordance d'un schéma de sélection, tandis que la perturbation due aux opérateurs représente les effets destructeurs que le croisement et la mutation peuvent avoir (les individus de grandes qualité peuvent être perturbés). [28]

2.9 Conclusion

Nous avons présenté un état de l'art sur les algorithmes évolutionnaires d'une manière générale dans ce deuxième chapitre.

Après avoir rappelé quelque notion de l'algorithme évolutionnaire nous avons passé en revue les principaux algorithmes MOEA de l'état de l'art ainsi que les deux types de codage pour représenter les solutions dans la population. On a parlé aussi sur la partie Darwienne de l'algorithme qui comprend deux étapes importantes la sélection et le remplacement. Ainsi que les indicateurs de performance et procédures de comparaison.

Chapitre 3
Algorithme Evolutionnaire basé sur les Grilles pour l'Optimisation Multi-Objectif

Dans ce chapitre nous présentons un algorithme évolutionnaire multi objectif à base de grille. Une grille a une propriété inhérente réfléchissant l'information de la convergence et de la diversité en même temps. Chaque solution dans la grille a un emplacement déterministe. Cet emplacement peut être apprécié d'une part selon ces coordonnées dans la grille par rapport à celles des autres solutions, et d'autre part par le concept de diversité. Cette diversité peut être estimée sur la base du nombre de solutions dont les coordonnées dans la grille sont identiques ou proches. Par ailleurs, contrairement au critère de Pareto dominance, nous allons voir qu'un critère basé sur une grille peut non seulement qualitativement comparer les solutions, mais donne aussi une différence quantitative par rapport à chaque objectif.

3.1 Mise en place de la grille

Soient P une population d'individus définie sur $\boldsymbol{\Omega(\Omega \subseteq \mathcal{R}^n)}$ et soient $\boldsymbol{F_P}$ l'ensemble des images de la population dans l'espace des objectifs $\boldsymbol{F(\Omega) \subseteq \mathcal{R}^M}$. La mise en place de la grille par rapport au $k^{ème}$ objectif f_k est présentée par la figure 3.1. Tout d'abord, les valeurs minimales et maximales relativement au $k^{ème}$ objectif sont identifiées parmi les valeurs de l'objectif en question appliquée à la population P. Ces valeurs sont désignées respectivement par $min_k(P)$ et $max_k(P)$.

Ensuite, les bornes inférieures et supérieures de la grille par rapport au k^{eme} objectif sont déterminées selon les formules suivantes:

$$lb_k = min_k(P) - (max_k(P) - min_k(P)) / (2 \times div) \quad .$$

$$ub_k = max_k(P) + (max_k(P) - min_k(P)) / (2 \times div).$$

Où div est le nombre de division par rapport à chaque axe de l'espace des objectifs.

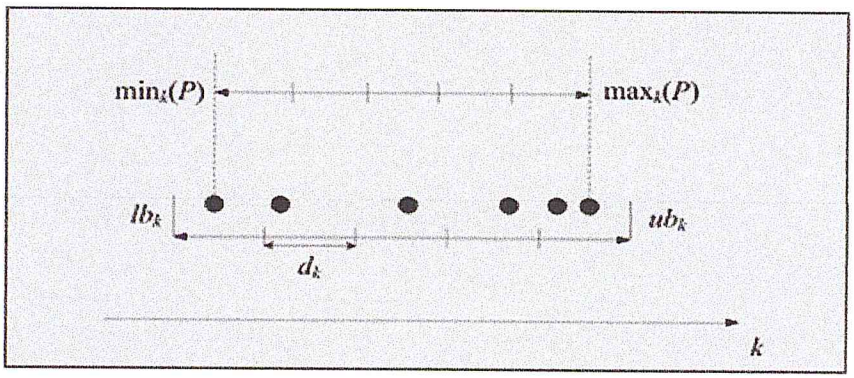


Figure 3.1 : la mise en place de la grille relativement au k^{eme} objectif. $(div = 5)$ ici les coordonnées grille sont respectivement 0, 1, 2, 3, et 4.

La largeur de hyper box d_k, par rapport au k^{eme}objectif, est définie par la formule ci-dessous :

$$d_k = (ub_k - lb_k)/\, div$$

$$G_k\,(x) = \lfloor (f_k(x) - lb_k\,)/d_k \rfloor$$

$\lfloor . \rfloor$ désigne la partie entière. $G_k\,(x)$ est la coordonnée par rapport à la grille d'un individu relativement au k^{eme}objectif, $f_k\,(x)$ est la valeur objective réelle selon *le* k^{eme} objectif.

La grille semble être un moyen naturel de combiner la convergence et la diversité dans la recherche évolutive puisque d'un côté : chaque individu a un emplacement déterminé en elle. Ces coordonnées reflètent le nombre d'objectifs pour lesquels un individu est mieux que d'autre. D'un autre côté, le nombre d'individus dans chaque hyperbox est un moyen de mesurer la diversité parmi la population (voir figure **3.2**). Cependant la dominance de Pareto ne parvient pas a tenir compte de ces deux aspects.

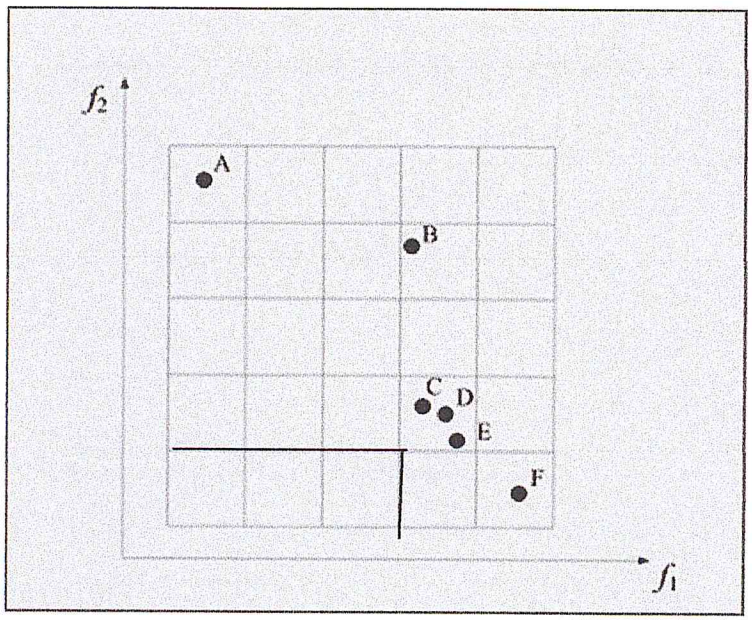


Figure 3.2 : Illustration des individus dans la grille d'un espace bi-objectif.

L'hyperbox de coordonnée (3 ,1) est plus encombré car il contient le plus grand nombre d'individus (**C, D** et **E**).

Grace à ces idées et à ces motivations que l'algorithme évolutionnaire multi-objectif basé sur la grille à vue le jour.

3.2 Grid Based Evolutionary Algorithm for many-objective optimization (GrMOEA)

Algorithme **3.1** donne la forme générale du **GrMOEA** [24]. La procédure de base de l'algorithme est similaire à la plupart des générations d'algorithmes évolutionnaire multi objectif, comme par exemple le *NSGA II* [15]. Premièrement, N individus sont générés aléatoirement pour former une population initiale P. Ensuite, une mise en place de la grille sur la base des données sur la population actuelle. L'évaluation des individus de P est faite en fonction de leur emplacement dans la grille. La sélection pour la variation est faite sur la base de cette évaluation. La variation donne naissance à une population d'enfants. Enfin, une procédure de mise à jour de la population pour choisir les N meilleures individus pour la survie.

Algorithme 3.1 : GrMOEA

Input : N " *taille de la population* "

Output : P " *la population* "

1: $P \leftarrow intialiser(P)$

2 :***tantque*** *le critère d'arrêt n'est pas atteint* ***faire***

3: mise en place de la grille. $(grid\_setting)$

4: évaluation de la population. $(fitness\_assignment(P))$

5:$P' \leftarrow$ selection pour la variation (P). $(mating\_selection)$

6:$P'' \leftarrow variation(P')$

7:$P \leftarrow$ mise à jour de la population $(P \cup P'')$, $(enviromnetal\_selection)$

8:***fin***

9:*returne* P

3.3 Concepts et définition des fonctions de GrMOEA

Dans cette section, nous introduisons quelques définitions utilisées dans la mise en œuvre de notre GrMOEA. Pour cela on suppose que notre grille est bien sur place par rapport à la population P.

- **Grille dominance**

Définition 1 : soit $x, y \in P$, $x \prec_{grid} y$: si et seulement si :

$$\forall i \in (1, 2, \dots, M): G_i(x) \leq G_i(y)$$

$$\exists j \in (1, 2, \dots, M): G_j(x) < G_j(y)$$

Où , $x \prec_{grid} y$ indique que x grille-domine y, M est le nombre d'objectifs. Apparemment, le concept de grille-dominance est le même que celui de la dominance de Pareto si les cordonnées grille des individus sont remplacés par leur valeurs objectives réelles. Leur relation spécifique est la suivante : Si une solution Pareto-domine une autre solution, celle-ci ne sera pas grille-dominée par cette dernière et vice versa.

La relation de grille-dominance est également similaire à la relation de ϵ −dominance, étant donné que les deux sont une forme décontractée de la Pareto dominance mais une

différence importante est que le degré de relaxation de la grille dominante est déterminé par l'état de l'évolution de la population.

En outre, l'utilisation de la grille dominance dans cette étude est totalement différent de celui de ϵ-dominance dans le ϵ-MOEA [16]. Dans les ϵ-MOEA, ϵ-dominance est utilisée pour déterminer la survie des individus. Seul les individus non dominés peuvent être conservés dans l'ensemble de l'archive. Cependant, dans GrMOEA, la grille dominance est principalement utilisée pour empêcher les individus d'être archivé plutôt que leurs concurrents. Cela signifie avec la grille dominance des individus de qualité inferieure peuvent aussi avoir la chance d'entrer dans l'ensemble de l'archive. Cela est utile pour le maintien de la diversité dans une certaine mesure.

- **Grille différence: (grid difference)**

Définition 2 : soit $x, y \in P$, la Grille différence de deux individus est donnée par :

$$GD(x,y)=\sum_{k=1}^{M}|G_k(x)-G_k(y)|$$

La grille différence est liée au nombre de divisions div, et elle varie de 0 à $M(div-1)$.

3.3.1 L'évaluation de la population ($fitness\_assignment$)

L'évaluation de chaque individu de la population sera calculée en fonction de sa position dans la grille. Pour ce faire, trois critères sont définies à savoir : le classement-grille(grid ranking GR), la distance d'encombrement-grille (grid crowding distance(GCD)), et la distance par rapport au grille-point de coordonnées (grid coordinate point distance(GCPD)). Dans ce qui suit, nous allons les présenter dans l'ordre.

- **Grid ranking (GR)**

GR est un estimateur simple de convergence, il permet de distinguer le rang des individus à la lumière de leur coordonnées. Il est défini comme la somme des grilles coordonnées de l'individu:

$$GR(x)=\sum_{k=1}^{M}G_k(x)$$

Où $G_k(x)$ désigne la grille coordonnée d'individu par rapport au k^{eme}objectif, M est le nombre d'objectifs.

De toute évidence, un individu avec un petit GR est préférable. Comme les grilles coordonnées d'un individu sont de l'évolution de la population et du nombre de divisions dans différents objectifs '*div*'. Cette normalisation permet d'avoir un processus raisonnable de comparaison des grilles coordonnées de différents individus. Car, la somme des grilles coordonnées GR d'un individu$x$ refléterait son degré d'évolution par rapport aux autres individus dans la population courante P. GR est carrément lié à deux facteurs précis : le nombre d'objectifs et son niveau par rapport à chaque objectif. Dans un sens, GR préfère les individus avec une bonne performance dans plusieurs objectifs (voir l'exemple de la figure 3.3). Un individu avec une meilleure performance dans plusieurs objectifs aurait une probabilité plus élevée pour atteindre une valeur inférieure de GR. D'autre part, la différence dans un seul objectif entre deux individus est également une partie importante dans l'issue de la valeur de GR.

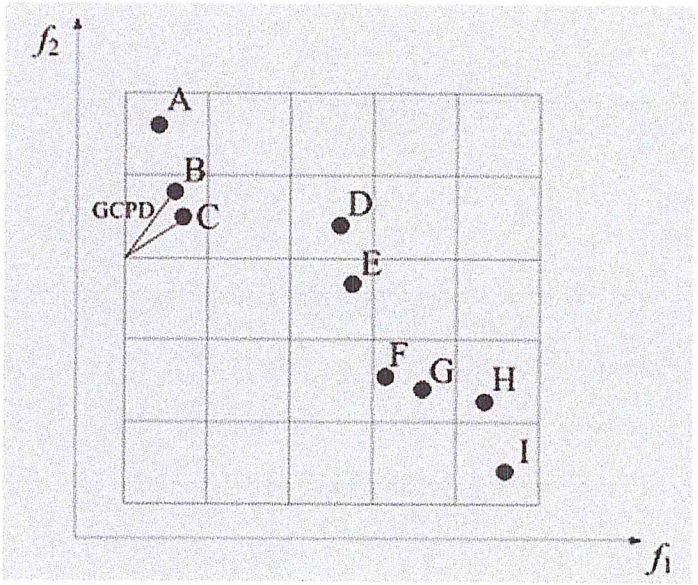


Figure 3.3 : illustration de l 'évaluation (fitness assignement).

Pour les individus non dominée D et A, on remarque que l'avantage de D en f_2 est moins que son désavantage en f_1, cela permet à D d'avoir une valeur GR pire que celui de A (5 contre 4).

En conclusion, GR peut être considéré comme un compromis naturel en intégrant le nombre d'objectifs pour lesquels une solution est meilleure que d'autres et les écarts

dans les valeurs de ces objectifs. Ainsi, cela permet d'améliorer de façon significative la pression de sélection dans la direction d'optimisation.

- **Grid crowding distance(GCD)**

GCD est défini par le nombre de ses voisins :

$$GCD(x) = \sum_{y \in N(x)} (M - GD(x, y))$$

Où $N(x)$ est l'ensemble des voisins de x, et M le nombre des objectifs. Une solution y est considérée comme un voisin d'une solution x si $GD(x, y) < M$.

Dans la figure 3.3, le voisin de l'individu D est E et la valeur $GCD\ de\ D$ est 1, les voisins des individus de G sont F et H, et donc la valeur GCD de G est 2. Notez que la taille des voisins est liée au nombre d'objectifs M. Cela permet donc de fournir une distinction plus fine sur le degré de surpeuplement chez les individus.

- **Grid coordinate point distance (GCPD)**

Bien que GR et GCD ont déjà fourni une mesure approximative pour les individus en termes de convergence et de la diversité, ils peuvent également échouer dans la discrimination des individus, par exemple, les individus situés dans le même hyperbox.

Ici, empruntant l'idée à ε-MOEA, c'est-à-dire, nous calculons la distance entre l'individu et le point, représenté par ses coordonnées dans la grille, pour affiner l'estimation de la convergence. Plus précisément, la distance par rapport au grille-point de coordonnées (Grid coordinate point distance (GCPD)), est définie comme suit:

$$GCPD\ (x) = \sqrt{\sum_{k=1}^{M} ((f_k(x) - (lb_k + G_k(x) \times d_k))/d_k)^2}$$

Où $G_k\ (x)$ désigne la grille-coordonnées d'un individu par rapport au k^{eme} objectif, $f_k(x)$ représente la valeur réelle de k^{eme} objectif, lb_k désigne la borne inférieure de k^{eme}objectif, $d_k$correspond à la largeur de hyperbox dans le k^{eme}objectif et M est le nombre d'objectifs.

Les individus B et C sur la figure 3.2 illustrent ce cas. De toute évidence, une courte $GCPD$ des individus est préférable lorsque les deux autres critères sont incomparable.

En conclusion, ces trois critères ci-dessus permettent de faire des comparaisons entre les individus à des niveaux différents lors de l'évaluation. De cette façon, nous serons en mesure d'établir un ordre complet entre les individus pour la poursuite du processus de sélection.

3.3.2- Sélection pour la variation (Mating selection)

Une sélection pour la variation vise à faire une bonne préparation pour échanger les informations des individus. Cette sélection joue un rôle important dans les algorithmes évolutionnaires. Généralement, elle est mise en œuvre en sélectionnant des solutions prometteuses de la population actuelle pour former des groupes pour le croisement. Ici, nous utilisons des tournois de type binaire comme stratégie de sélection. Cette stratégie est basée sur la relation de dominance pour sélectionner les individus pour la variation.

- **Sélection par tournoi**

Algorithme 3.2 donne une procédure détaillée de cette stratégie. Premièrement, deux individus sont choisis au hasard dans la population. Si l'un grille-domine ou Pareto domine l'autre, le premier est choisi. Dans le cas contraire, cela indique que ces deux solutions sont non dominées. Dans ce cas, nous préférons l'individu avec une valeur de GCD inférieure. Enfin, si GCD ne parvient toujours pas à distinguer entre les deux solutions, le choix dans ce cas sera aléatoire.

3.3.3 La mise à jour de la population (Environmental selection)

La mise à jour de la population vise à obtenir une bonne approximation du front de Pareto. Cela se traduit par la mise en œuvre d'un ensemble d'archives bien distribué. Cela se fait en sélectionnant les meilleures solutions de la population précédente et la population nouvellement créée. Un moyen simple de faire la sélection est basée sur l'évaluation des solutions. Cependant, un inconvénient de cette méthode est qu'elle peut

conduire à la perte de diversité puisque les solutions adjacentes ont souvent des valeurs fitness similaires.

Algorithme 3.2 : Sélection par tournoi

p,q deux individus choisi aléartoirement dans la population

1:***si*** $p < q$ *or* $p <_{grid} q$ ***alors***

2:***returne p***

3: ***sinon si*** $q < p$ *or* $q <_{grid} p$ ***alors***

4:*returne q*

5:***sinon si*** $GCD(p) < GCD(q)$ ***alors***

6:*returne p*

7:***sinon si*** $GCD(q) < GCD(p)$ ***alors***

8:*returne q*

9:***sinon si*** $random(0,1) < 0.5$ ***alors***

10:*returne p*

11:***sinon***

12:*returne q*

13:***fin si***

3.3.3.1 Ajustement de l'évaluation (*fitness adjustment*):

L'algorithme GrMOEA sélectionne les individus par hiérarchie en les comparant selon les trois critères de la fitness : GR, GCD, et GCPD. GR est le principal critère, GCD est considéré comme le second et il est activé lorsque les valeurs GR des individus sont incomparables (à savoir, l'égalité) et lorsque les deux premiers critères ne parviennent pas à distinguer les individus, le troisième GCPD est utilisé pour casser cette égalité. Ici, nous nous concentrons sur l'ajustement du critère principal.

Quand un individu est sélectionné dans l'archive, les valeurs GR des individus qui lui sont attachées seront pénalisées. Cependant, la mise en œuvre de la pénalisation de GR (à savoir, la détermination des individus concernés et à quel point) n'est pas une tâche

triviale. Plusieurs facteurs cruciaux doivent être pris en considération pour parvenir à un bon équilibre entre la convergence et la diversité dans l'archive.

Une peine sévère devrait être imposée aux individus qui ont les même coordonnées dans la grille que l'individu choisi.

Les individus grille-dominé par l'individu choisi devraient être plus lourdement pénalisé que les individus qui ne sont pas de grille-dominée par ce dernier.

Afin d'éviter d'autres surpeuplement, les voisins de l'individu choisi doivent être pénalisés et le degré de la pénalité devrait être réduite en rapport avec la distance qui le sépare de l'individu choisi.

Lors de l'exécution de la pénalité sur les voisins de l'individu choisi, les individus qui sont grille dominée par ces voisins sont à leur tour pénalisés. Car cela peut en grande partie améliorer la convergence de l'archive.

- **L'Algorithme" *GR_adjustment*" :**

Garder à l'esprit, les facteurs cités précédemment, une procédure d'ajustement GR est présenté dans l'algorithme 3.3. De toute évidence, les individus peuvent être classés en trois groupes dans le processus GR-ajustment:

Les individus dont les grille-coordonnées sont égales à celle de l'individu pris (lignes 1-3), ceux qui sont grille-dominée par l'individu choisi (lignes 4-6), et ceux qui ne sont pas grille-dominée par l'élément prié et qui ont des grille-coordonnées différentes de celle de ce dernier (lignes 7-22). Les degrés de pénalité respectifs corresponds à $M + 2, M$, et dans $[0, M - 1]$, où M est le nombre d'objectifs.

Spécifiquement, pour les individus dans le dernier groupe, un voisin p, de l'individu choisi q, est soumet à une pénalité au moins d'ordre $M - GD(p, q)$(lignes 11 et 12), et en conséquent les individus grille dominée par p est soumet à une pénalité de degré supérieur ou égale à celle de p(lignes 13-17). De cette façon, on peut empêcher des individus d'être archivés plutôt que leurs concurrents qui sont meilleures au sens de la grille-dominance.

Algorithme 3.3 : *GR_adjustment*(P, q)

require: $P(ensemble\ candidat), q(individu\ choisi), M(nombre\ d'objectifs)$,

$PD(p)(degré\ de\ la\ peine\ maximale\ de\ p)$, $E(q) := \{p \in P | GD(p,q) = 0\}$,

$G(q) = \{p \in P | q <_{grid} p\}, NG(q) = \{p \in P | q \not<_{grid} p\}, N(q) = \{p \in P | PGD(p,q) < M$

1: ***pour tout*** $P \in E(q)$ ***faire***

2: $GR(p) \leftarrow GR(p) + (M + 2)$

3: ***fin pour***

4: ***pour tout*** $p \in G(q)$ ***faire***

5: $GR(p) \leftarrow GR(p) + M$

6: ***fin pour***

7: ***pour tout*** $p \in NG(q) \wedge p \notin E(q)$ ***faire***

8: $PD(p) \leftarrow 0$

9: ***fin pour***

10: ***pour tout*** $p \in N(q) \cap NG(q) \wedge p \notin E(q)$ ***faire***

11: ***if*** $PD(p) < M - GD(p,q)$ ***alors***

12: $PD(p) \leftarrow M - GD(p,q)$

13: ***pour tout*** $r \in G(p) \wedge r \notin E(q) \cup G(q)$ ***faire***

14: ***si*** $PD(r) \leftarrow PD(p)$ ***alors***

15: $PD(r) \leftarrow PD(p)$

16: ***fin si***

17: ***fin pour***

18: ***fin si***

19: ***fin pour***

20: ***pour tout*** $p \in NG(q) \wedge p \notin E(q)$ ***faire***

21: $GR(p) \leftarrow GR(p) + PD(p)$

22: ***fin pour***

3.3.3.2 Procédure de mise à jour de la Population

Algorithme 3.4 donne la procédure principale de mise à jour de la Population. Elle est semblable à celle de NSGA – II, la première étape de GrMOEA consiste à diviser l'ensemble des solutions candidates F selon le principe de dominance de Pareto. Cette étape réparties les solutions en un nombre de classes distincts F_i de façon suivante : tous les individus non dominés de F appartient à F_1 ; ensuite, tous les éléments de $F \setminus F_1$ sont placés dans F_2 etc. L'étape est terminée quand toute la population est triée des niveaux de non dominance ($F = \bigcup_{i=1}^{r} F_i$). Le front en litige F_k (c'est à dire $|F_1 \cup F_2 \cup \cdots \cup F_{k-1}| \leq N$ et $|F_1 \cup F_2 \cup \cdots \cup F_{k-1} \cup F_k| > N$, où N désigne la taille de l'archive), est déterminé et les $k-1$ première classes d'individus sont versé dans l'archive (lignes 2-6). En effet, étant donné que les solutions (dans le cas d'un nombre d'objectif important) sont Pareto non dominée, la classe en litige correspond souvent à la première, c'est-à-dire $k = 1$.

Dans l'algorithme 3.4, la fonction **Initialization** (ligne 8) est utilisée pour initialiser les informations des individus dans la procédure de mise en place de la grille située dans la ligne 7. L'évaluation des individus pour la convergence (à savoir, GR et GCPD). Il est nécessaire de souligner que la valeur de densité initiale des individus (à savoir, GCD) est mise à zéro dans la fonction. Contrairement à l'estimateur de convergence, qui peut être directement calculé par le propre emplacement d'un individu, la diversité doit être estimée par rapport aux autres individus. Il peut être inutile de considérer la relation d'encombrement entre les individus de l'ensemble des candidates plutôt que dans l'ensemble de l'archive, puisque ce dernier représente la seule population à être conservé. Ici, GrMOEA estime la densité des individus en calculant leur degré d'encombrement dans l'archive (ligne 13).

Les algorithmes 3.5 et 3.6 donnent, respectivement le pseudo-code de fonctions Initialisation et du calcul de GCD.

La fonction ***Findout_best*** (ligne 10) dans l'algorithme 3.4 est conçue pour déterminer le meilleur individu dans le front considéré. Le pseudo-code est illustré dans l'algorithme 3.7. Comme indiqué précédemment, la fonction compare hiérarchiquement les trois critères GR, GCD, et GCPD. Une valeur plus faible est préférable selon tous les critères.

Algorithme 3.4 : *environmental _selection*(P)

require : N *est la taille de l'archive*

1: *générer un ensemble vide* Q *archive*

2: $(F_1, F_2 \ldots\ldots F_i, \ldots\ldots) \leftarrow Pareto\_nondominé\_trier(P)$

3: $Q \leftarrow F_1 \cup F_2 \cup \ldots\ldots \cup F_{i-1}$

4: ***si*** $|Q| = N$ ***alors***

5: *returne* Q

6: ***fin si***

7: $Grid\_setting\ (F_i)$

8: $initialization\ (F_i)$

9: ***tantque*** $Q < N$ ***faire***

10: $q \leftarrow Findout\_best(F_i)$

11: $Q \leftarrow Q \cup \{q\}$

12: $F_i \leftarrow F_i \setminus \{q\}$

13: $GCD\_calculation(F_i, q)$

14: $GR\_adjustment\ (F_i, q)$

15: ***fin***

16: *returne* Q

Algorithme 3.5 : *intializaition*(P)

1: ***pour tout*** $p \in P$ ***faire***

2: $GR(p)$

3: $GCPD(p)$

4: $GCD(p) \leftarrow 0$

5: ***fin pour***

Algorithme 3.6 : *GCD_calculation*(P, q)

require : P *est l'ensemble des candidats* , q *(picked individual)*

$N(q) := \{p \in P | \, GD(p,q) < M\}$

1: ***pour tout*** $p \in N(q)$ ***faire***

2: $GCD(p) \leftarrow GCD(p) + (M - GD(p,q))$

3: ***fin pour***

Algorithme 3.7 : *findout_ best* (P)

q *(meilleure solution dans* P*),* pi *(la* $i -$ *eme solution dans* P*)*

1: $q \leftarrow p_1$

2: ***pour*** $i = 2$ *à* $|P|$ ***faire***

3: ***si*** $GR(p_i) < GR(q)$ ***alors***

4: $q \leftarrow p_i$

5: ***sinon si*** $GR(p_i) = GR(q)$ ***alors***

6: ***si*** $GCD(p_i) < GCD(q)$ ***alors***

7: $q \leftarrow p_i$

8: ***sinon si*** $GCD(p_i) = GCD(q)$ ***alors***

9: ***si*** $GCPD(p_i) < GCPD(q)$ ***alors***

10: $q \leftarrow p_i$

11: ***fin si***

12: ***fin si***

13: ***fin si***

14: ***fin pour***

15: *returne* q

L'implémentation de notre algorithme n'a d'autre objectif que sa simulation et l'évaluation de ces performances dans des situations bien considérées.

3.4 La simulation

La simulation est l'un des outils permettant de simuler des phénomènes réels. Les objectifs principaux de la simulation sont l'évaluation des protocoles, l'architecture des réseaux et prévision leur fonctionnement.

3.4.1 Simulation de variables aléatoires

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 permette d'exécuter de tels programmes, souvent baptisés RAND. En général leur conception repose sur des propriétés arithmétiques de certaines suites récurrentes. Ces algorithmes sont déterministes, c'est-à-dire qu'ils 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 la quelle ces générateurs sont appelés pseudo-aléatoires plutôt qu'aléatoires.

3.4.2 Description de certain générateur

Une famille de générateurs populaires est celle des générateurs congruentiels linéaires. Ils génèrent des suites de nombres entiers $(x_n)_{n\geq 1}$, dans l'ensemble $\{1,2,\dots,m-1\}$ 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 :

$$x_{n+1} = ax_n + b \text{ modulo } m, n \geq 0$$

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 (celle de la petite école) 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. [30]

3.4.3 La méthode d'inversion

Soit F une fonction de répartition inversible, alors si U suit une loi uniforme sur [0 , 1], la variable aléatoire $X = F^{-1}(U)$ a pour fonction de répartition F, En effet, pour tout réel x :

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P\big(U \leq F(x)\big) = F(x)$$

où l'on a appliqué respectivement l'aspect bijectif de F, sa croissance et la forme spécifique de la fonction de répartition de la loi uniforme.

Pour ce qui nous concerne, l'intérêt de ce résultat est manifeste : pour peu que F soit facilement inversible, alors pour générer une variable aléatoire de loi F, il suffit de générer une variable uniforme U et de lui appliquer F^{-1}. [31]

3.4.4 Principe général de la simulation

Soit $U_1, U_2, \ldots$, un échantillon du loi uniforme sur [0, 1], on sait que F^{-1} désignant l'inverse généralisé de la fonction de répartition F de la loi de X :

$$X_i = F^{-1}(U_i), i \geq 1.$$

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}.

3.5-Conclusion

Ce chapitre a présenté une stratégie qui utilise les propriétés de la grille pour traiter les problèmes multi-objectifs. La méthode utilisée construit une grille adaptative selon la population évolutive et les informations sur l'état des individus sont déterminées par cette dernière.

D'une part, trois critères hiérarchiques basés sur la grille (GR, GCD et GCPD) sont incorporés dans l'aptitude à établir un ordre complet entre les individus. D'autre part, une technique d'ajustement adaptative de l'évaluation dans la sélection pour la mise à jour de la population est introduite pour maintenir la diversité de l'ensemble des archives.

Chapitre 4
Implémentation Tests et Résultats

L'objectif de ce présent chapitre est de donner une synthèse des résultats obtenus par l'algorithme GrMOEA dans la résolution d'un ensemble de problèmes tests « DTLZ » [1]. Ces résultats vont nous permettre d'évaluer les performances des différents choix considérés dans cet algorithme.

4.1 Problèmes tests

Concernant le choix des benchmark DTLZ, leur choix est motivé par le fait de tester notre algorithme par rapport à des caractéristiques bien définies. Les problèmes testes DTLZ2, DTLZ4 sont utilisés pour tester la capacité d'un algorithme à faire face aux problèmes avec des formes et positions différentes, et DTLZ1, DTLZ3 afin décrier plus d'obstacles pour un algorithme ceci entraine un ralentissement dans convergence vers le front de Pareto. Les tableaux suivants représentent les paramètres de ces problèmes test :

Tableau 1 : réglage de paramètre pour les problèmes test.

Le nom du problème	Nombre d'objectifs (M)	Nombre de variables (n)	Paramètre
DTLZ1	4, 6, 8 et 10	$M-1+k$	$k=5$
DTLZ2	4, 6, 8 et 10	$M-1+k$	$k=10$
DTLZ3	4, 6, 8 et 10	$M-1+k$	$k=10$
DTLZ4	4, 6, 8 et 10	$M-1+k$	$k=10$

4.2-Initialisation et paramétrage du GrMOEA

On s'intéressera ici essentiellement au choix judicieux des paramètres, pour détailler les procédures de notre démarche.

4.2.1 Paramètre *div* dans GrMOEA

Tableau 2 : le choix du paramètre *div* pour les différents problèmes test.

Problème	$M = 4$	$M = 6$	$M = 8$	$M = 10$
DTLZ1	10	10	10	11
DTLZ2	10	8	7	8
DTLZ3	10	11	10	11
DTLZ4	10	8	7	8

4.2.2-Paramètre de croisement et mutation : la probabilité de croisement est $p_c = 1.0$ et de mutation $p_m = 1/n$ où n désigne le nombre de variables de décisions, nous utilisons l'opérateur du croisement binaire simulé (SBX) et la mutation polynomiale pour le croisement et la mutation, les indices de distributions sont : $\eta_c = 20$ et $\eta_m = 20$.

4.2.3-Taille de population : la taille de la population initiale et la population actualisée pendant le processus d'évolution est $N = 100$ pour un nombre d'objectif M=4, 6, 8 et $N = 50$, pour M=10.

4.2.4 Le langage de programmation

Pour la réalisation de notre projet nous avons utilisé le langage de programmation Scilab version 5.2.2, qui est un langage interprété, à typage dynamique extensible facile à manipuler.

Le Scilab est un logiciel libre de calcul scientifique développé depuis 1990 par l'INRIA (Institut National de Recherche en Informatique et Automatique) et l'ENPC (Ecole Nationale des Ponts et Chaussées). Il est disponible gratuitement sur le site Internet **www.scilab.org**. Dans sa syntaxe, son architecture et son fonctionnement, Scilab est très proche du logiciel Matlab commercialisé par (Mathworks Inc). Depuis 2003, le développement de Scilab se fait dans le cadre d'un groupe d'entreprises et d'organismes de recherche constitué de : Axs Ingenierie, CEA, CNES, Cril Technology, Dassault-Aviation, EDF, ENPC, Esterel Technologies, INRIA, PSA Peugeot Citroën, Renault, Thales.

4.3 Résultats expérimentaux et discussion

Pour évaluer les performances de l'algorithme GrMOEA, nous utilisons une mesure de qualité largement utilisés, la distance générationnelle inversé (IGD), déjà présentée auparavant. En effet si la valeur d'IGD est faible alors les solutions donne par l'algorithme sont proches de front de Pareto.

Plusieurs essais ont été effectués pour GrMOEA, nous avons réalisé pour chaque essai n_{itr} = 30.000 itérations pour chacun des problèmes DTLZ1, DTLZ3 et n_{itr} =10.000 itérations pour chacun des problèmes DTLZ2, DTLZ4. Le tableau ci-dessous donne pour chaque fonction et pour chaque valeur de M la valeur moyenne de IGD et entre parenthèses la meilleure valeur de IGD obtenue:

Tableau 3 : les résultats d'IGD par GrMOEA.

Objectifs	M=4	M=6	M=8	M=10
DTLZ1	0.0742 (0.0213)	0.1268 (0.1035)	0.1902 (0.0868)	0.1813 (0.1134)
DTLZ2	0.1458 (0.1178)	0.3075 (0.1005)	0.4144 (0.315)	0.4422 (0.302)
DTLZ3	0.1671 (0.1443)	0.531 (0.3412)	0.6217 (0.4045)	0.8426 (0.3075)
DTLZ4	0.0775 (0.0198)	0.3032 (0.0448)	0.4132 (0.0722)	0.5149 (0.0984)

Pour étudier le comportement de l'algorithme en fonction du nombre d'itérations, nous avons choisi de s'intéresser à la variation de la valeur de l'IGD en cours d'exécution. Les courbes ci-dessous montrent une allure convergente à partir d'un certain rang.

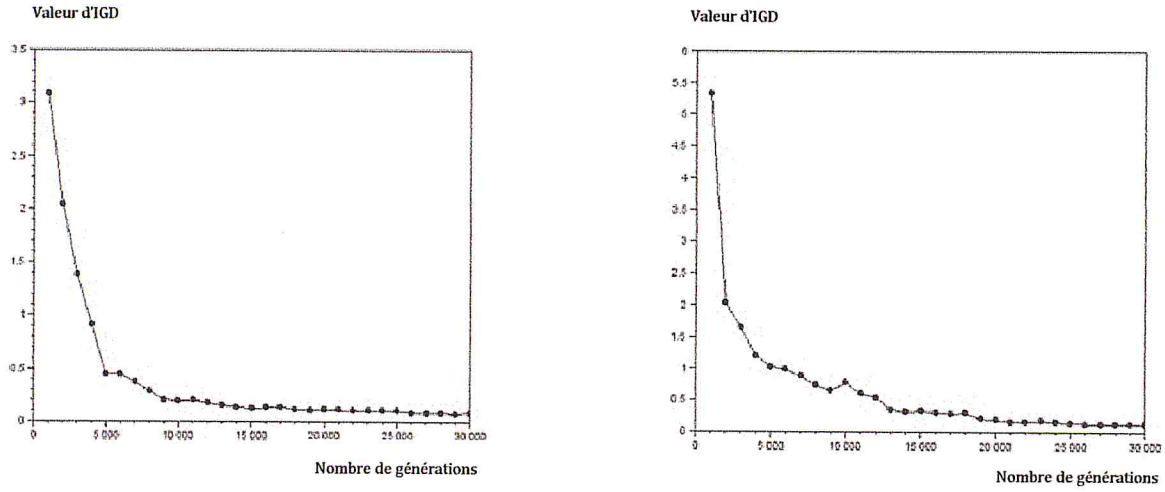


Figure 4.1: l'IGD en fonction du nombre d'itération du problèmeDTLZ1 pour 4 et 6 objectifs.

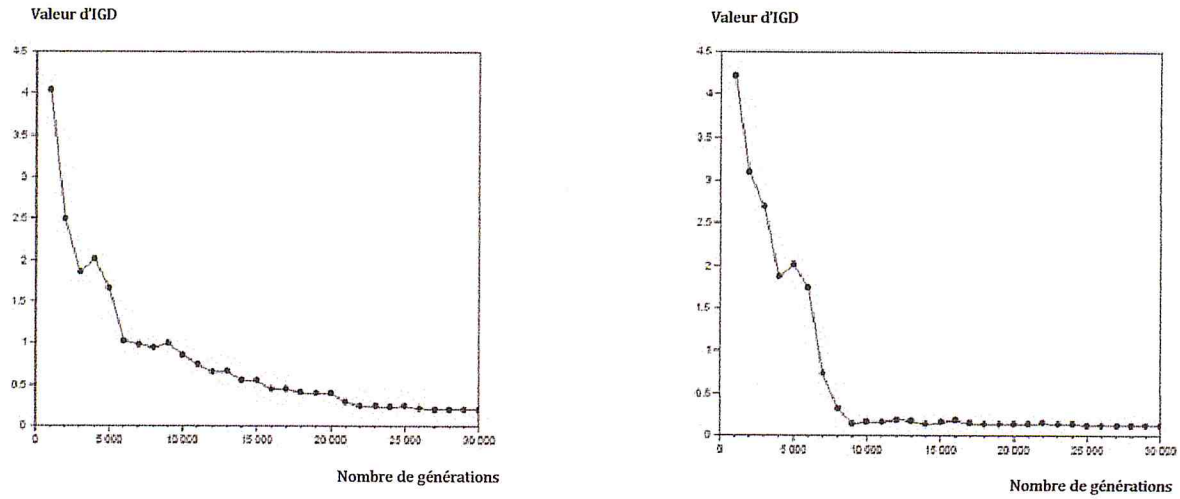


Figure 4.2: l'IGD en fonction du nombre d'itération du problème DTLZ1 pour 8 et 10 objectifs

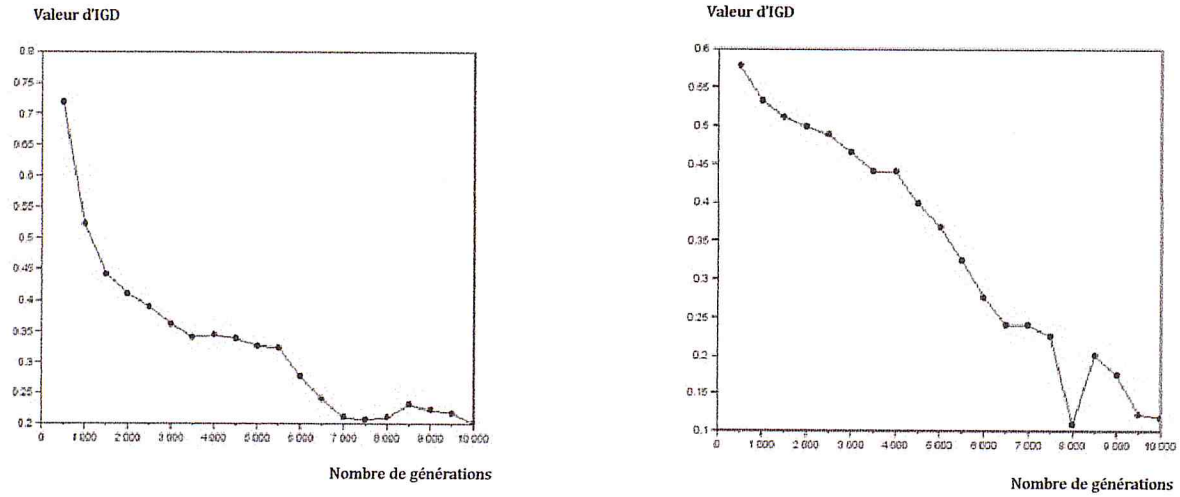


Figure 4.3: l'IGD en fonction du nombre d'itérations du problème DTLZ2 pour 4 et 6 objectifs

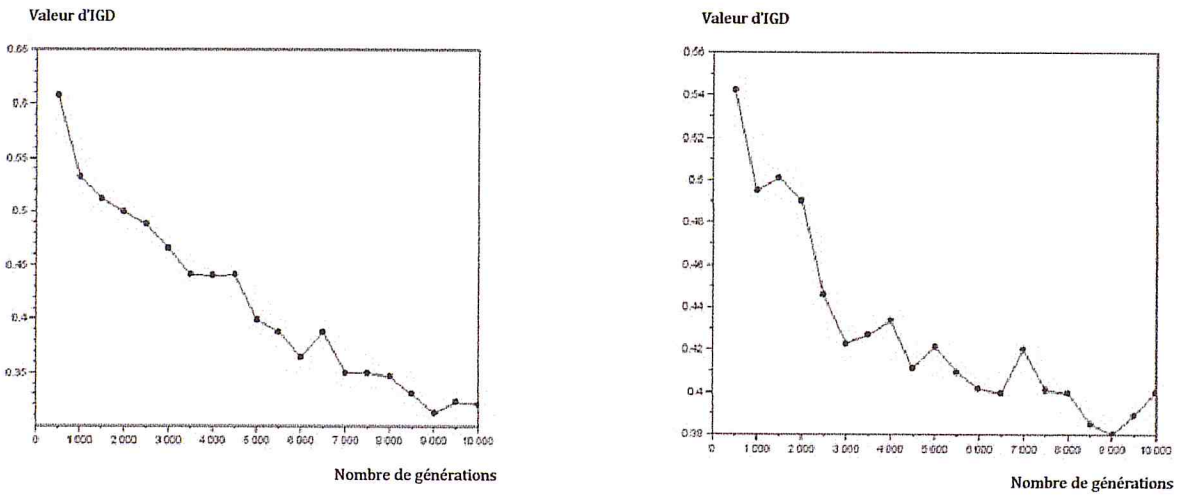


Figure 4.4: l'IGD en fonction du nombre d'itérations du problème DTLZ2 pour 8 et 10 objectifs

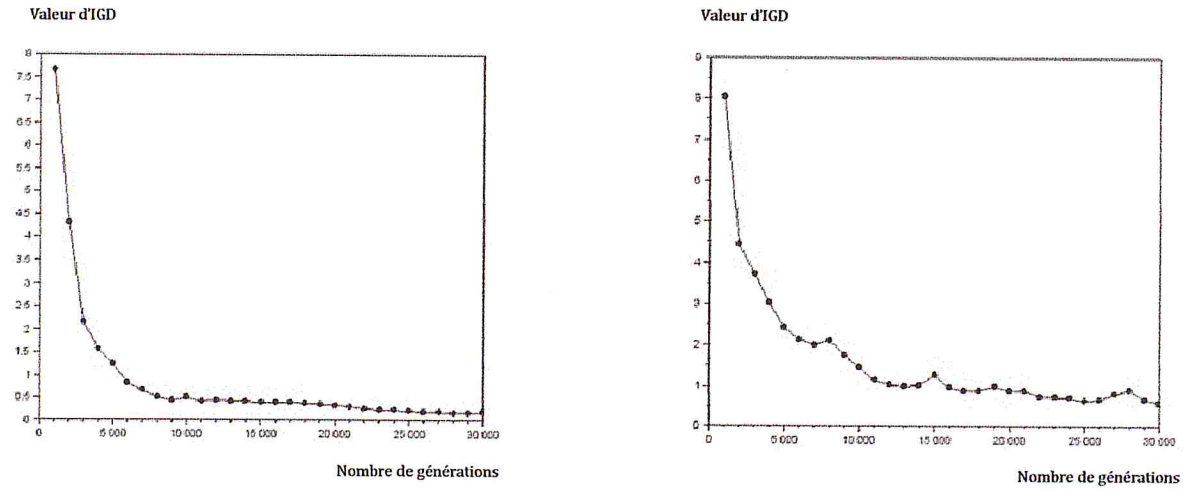


Figure 4.5: l'IGD en fonction du nombre d'itérations du problème DTLZ3 pour 4 et 6 objectifs

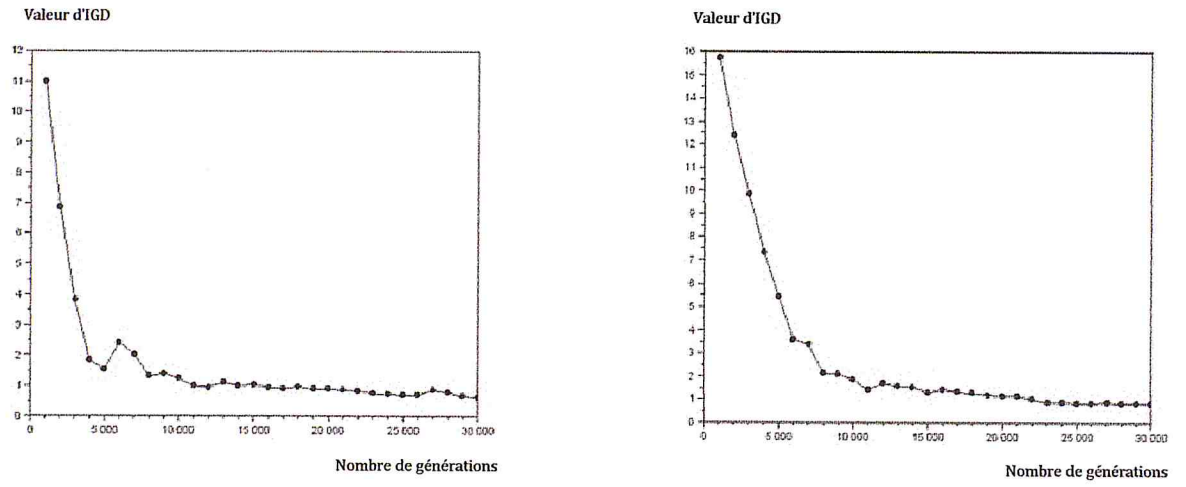


Figure 4.6: l'IGD en fonction du nombre d'itérations du problème DTLZ3 pour 6 et 10 objectifs

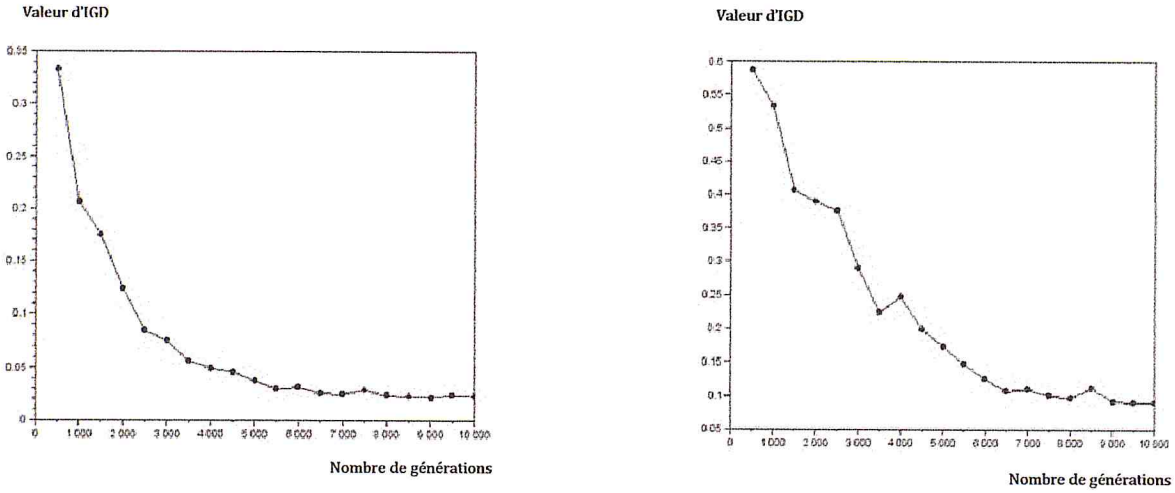


Figure 4.7: l'IGD en fonction du nombre d'itérations du problème DTLZ4 pour 4 et 6 objectifs

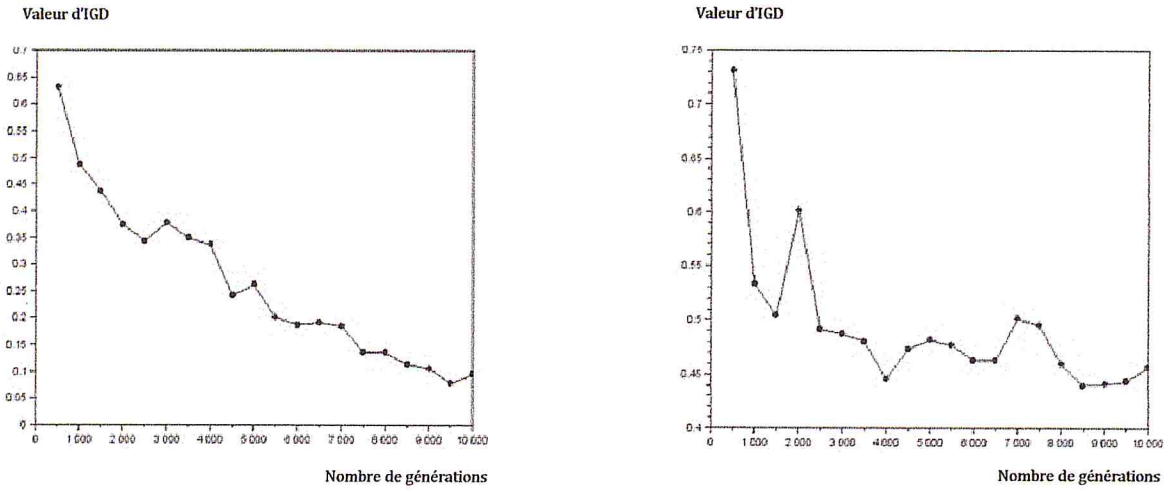


Figure 4.8: l'IGD en fonction du nombre d'itérations du problème DTLZ4 pour 8 et 10 objectifs

4.3.1-Interprétation des résultats : A travers les résultats du tableau 3 et les figures 4.1-4.8, on peut en conclure que, l'algorithme GrMOEA a pu approcher le front de Pareto pour le problème DTLZ1, en effet la valeur d'IGD devient faible à partir 10000 générations ceci montre que notre algorithme nous donne de bon résultats pour toute les tailles du problème. Contrairement à DTLZ1, l'algorithme converge lentement pour le problème DTLZ3 et pour M=4, 6, et 10. Pour DTLZ4 il est clair que GrMOEA converge rapidement et génère un nombre maximum de solutions non-dominées aux environ de 500 itérations. Comme on le voit sur les figures 4.7-4.8, les valeurs d'IGD sont assez fiables et cela est un indicateur de convergence vers le vrai front de Pareto et d'une diversité bien répartis. L'algorithme donne des résultats moins bons pour le problème DTLZ2 que ceux obtenus pour le DTLZ4, malgré qu'on moyenne le GrMOEA approche le front de Pareto, ce qui plais de favorablement en termes d'efficacité de l'algorithme.

D'autre part nous avons remarqué parfois que la valeur d'IGD diminue jusqu'à une certaine valeur et après l'IGD commence à augmenter. Nous croyions que cela peut être dû à l'application de l'opérateur de mutation au moment de la convergence, cela peut provoquer la sauvegarde de solutions de moindre qualité parmi la population. Pour confirmer/ infirmer ce que nous avançant, nous avons choisi de tester une modification en jouant sur l'utilisation de l'opérateur de mutation dans notre algorithme. En effet si le nombre d'itération décidé apriori est n_{itr}, nous appliquant la mutation uniquement sur les $n_{itr}/2$ premières itérations avec la probabilité p_m. Les mesures de performances (IGD) pour ce cas, sont présentées dans le tableau suivant :

Tableau 4 : les résultats d'IGD par GrMOEA modifié.

Objectifs	M=4	M=6	M=8	M=10
DTLZ1	0.2142 (0.1246)	0.2768 (0.1413)	0.4155 (0.2425)	0.4549 (0.317)
DTLZ2	0.254 (0.1637)	0.3751 (0.323)	0.3211 (0.216)	0.7245 (0.6153)
DTLZ3	0.4432 (0.2175)	0.6274 (0.4722)	0.6429 (0.5134)	0.8549 (0.6075)
DTLZ4	0.2134 (0.1384)	0.3243 (0.1448)	0.4251 (0.3074)	0.5667 (0.2341)

4.3.2 GrMOEA modifié :

La comparaison des valeurs des IGD de notre algorithme avec et sans modification est illustrée dans les histogrammes suivants :

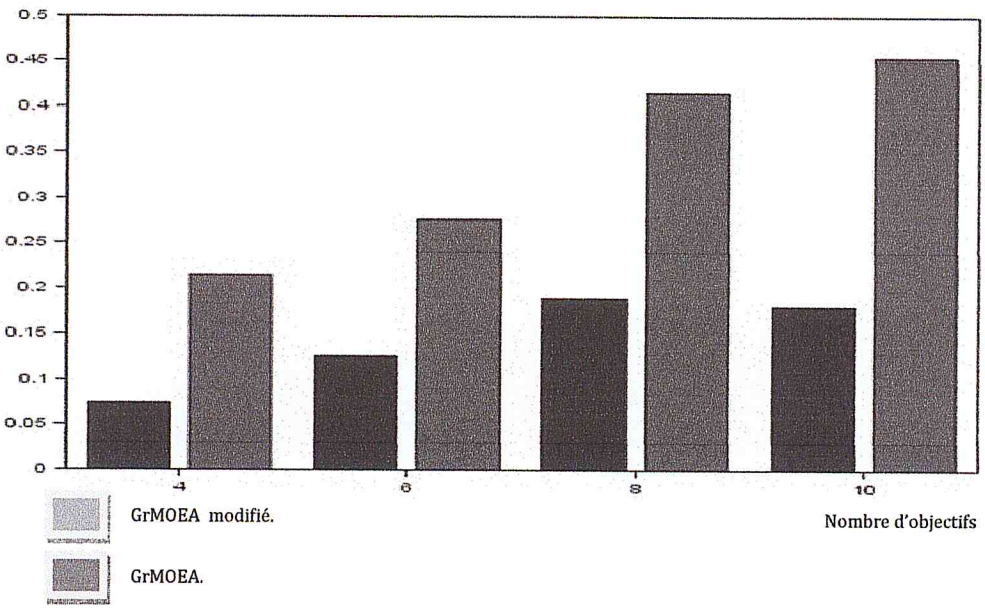


Figure 4.9 : la valeur moyenne d'IGD de GrMOEA avec/sans modification pour le problème DTLZ1

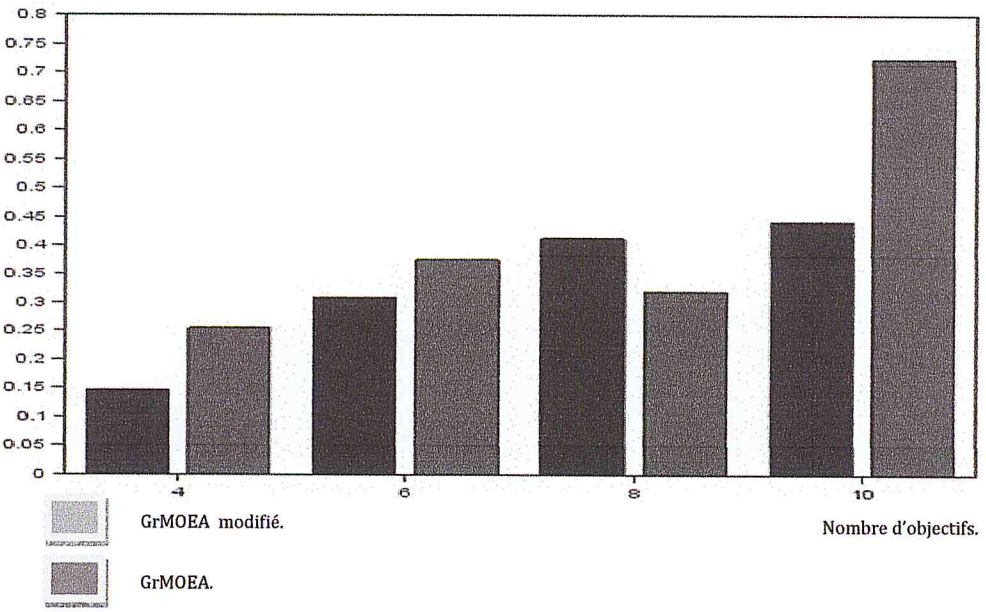


Figure 4.10 : la valeur moyenne d'IGD de GrMOEA avec/sans modification pour le problème DTL

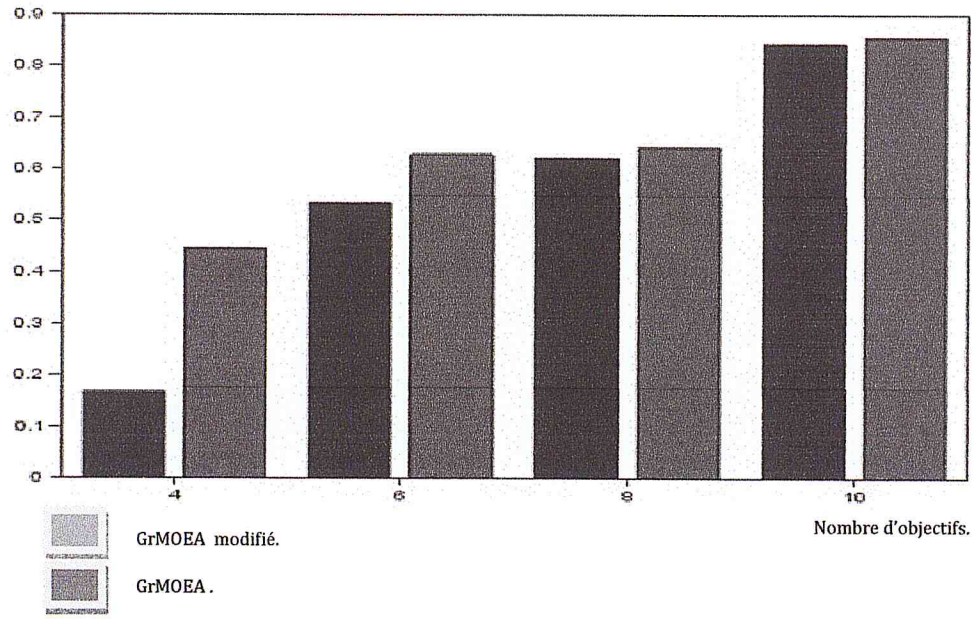


Figure 4.11 : la valeur moyenne d'IGD de GrMOEA avec/sans modification pour le problème DTLZ3

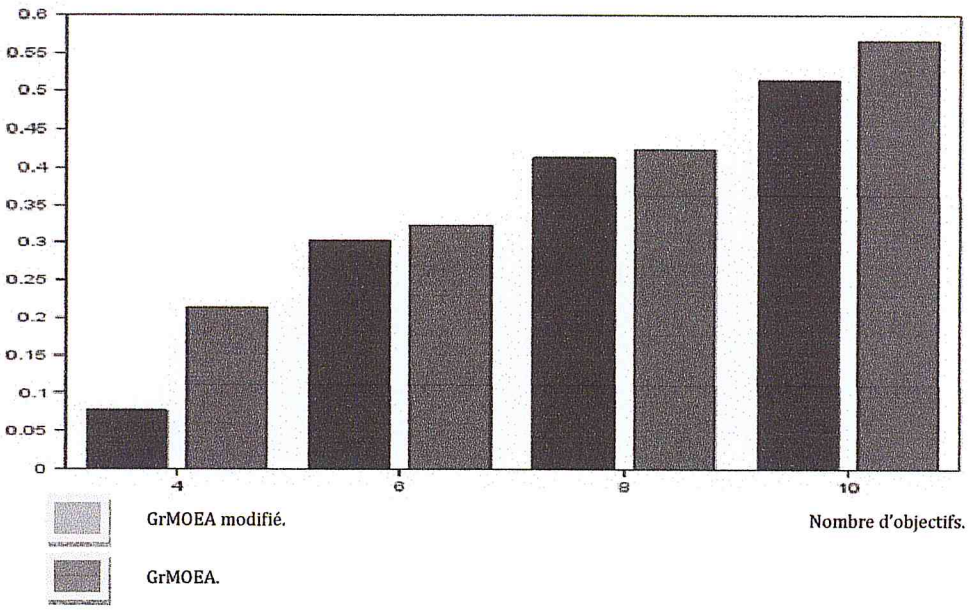


Figure 4.12 : la valeur moyenne d'IGD de GrMOEA avec/sans modification pour le problème DTLZ4

Les résultats obtenus montrent clairement que notre modification n'apporte aucune amélioration mais plutôt les résultats obtenus récoltés sont bon que l'algorithme initiale.

De ces résultats on peut dire que la mutation joue un rôle déterminant au cours de tout le processus dans le processus d'évolution de l'algorithme GrMOEA.

4.4 Conclusion

Nous avons présenté dans ce chapitre les paramètres utilisés dans nos simulations ainsi que les résultats obtenus sur les différents problèmes tests « DTLZ ». L'objectif a été d'explorer les performances de l'approche et de voir le comportement de l'algorithme GrMOEA avec une modification que nous avons proposé dans le but de faire face à un comportement négatif de la version initiale de GrMOEA. Numériquement, notre proposition a montré des faiblesses plus importantes.

La version initiale de GrMOEA donne des résultats qui sont très proche du front de Pareto, en effet, les valeurs d'IGD obtenues par l'algorithme sont bonnes pour M=4, 6, 8. Mais GrMOEA est un mauvais choix si la taille du problème est supérieure ou égale à 10 car la valeur d'IGD augmente avec le nombre d'objectif.

Conclusion générale

Les algorithmes évolutionnaires (AEs) sont des algorithmes de type stochastique visant à résoudre une large gamme de problèmes complexes en temps réel. Ils sont des méthodes pratiques connus pour leur flexibilité et leur robustesse. 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 exploite le potentiel d'une grille pour gérer les problèmes d'optimisation à plusieurs objectives. L'algorithme utilisé, GrMOEA, est principalement caractérisé par:

- L'exécution du calcul est centré sur l'individu au lieu du calcul de grille à l'inverse de la ε - dominance.[16]
- Le renforcement de la pression de sélection vers le front optimale en introduisant trois relations dans la grille : GR, GCPD et grille dominance;
- L'estimation de la densité des individus en utilisant les voisins adaptatifs dont la gamme varie selon le nombre d'objectifs;
- Le réglage de la fitness des individus dans le processus de sélection de l'environnement ;

Nous avons proposé une simulation de l'algorithme GrMOA pour résoudre les problèmes d'optimisation multi-objectifs. Cette méthode a été validée sur des fonctions tests (DTLZ). Pour évaluer les performances de l'algorithme GrMOEA, nous utilisons une mesure de qualité très utilisée : la distance générationnelle inversé (IGD).

L'analyse des résultats obtenus lors du processus de simulation , mettent en évidence le bon comportement de GrMOEA et nous a montré qu'il est possible de gagner énormément en temps de calcul tout en restant dans une marge d'erreur acceptable sur la précision.

Notre modification n'apporte aucune amélioration où les résultats obtenus sont moins bons que l'algorithme initial mais avec un temps de calcul relativement réduit.

On peut conclure que l'approche utilisée est encore très récente, et que l'adaptation de cette approche est nouvelle dans le domaine, il y aura toujours des améliorations à faire afin de raffiner de plus en plus l'approche adaptée que ce soit pour GrMOEA ou bien pour le multi-objectifs.

Annexe

Fonction test multi objectif de Deb et Al. [1]

Dans cette annexe, nous allons présenter tests multi-objectif, qui ont servi dernièrement de la base commune pour la comparaison des MOEA existants et pour l'évaluation de nouvelles techniques.

Problème test DTLZ1

Comme un problème de test simple, nous construisons un problème de M-objectif avec un front de Pareto-optimale linéaire :

$$\begin{cases} \min\ f_1(X) = \frac{1}{2} x_1 x_2 \ldots . x_{M-1}(1 + g(X_M)) \\ \min\ f_2(X) = \frac{1}{2} x_1 x_2 \ldots . (1 - x_{M-1})(1 + g(X_M)) \\ \vdots\ \vdots \\ \min\ f_{M-1}(X) = \frac{1}{2} x_1 (1 - x_2)(1 + g(X_M)) \\ \min\ f_M(X) = \frac{1}{2}(1 - x_2)(1 + g(X_M)) \\ 0 \leq x_i \leq 1\ , pour\ tout\ i = 1 \ldots n \end{cases}$$

Les dernière $k = n - M + 1$ variables sont représentés comme des variables de X_M, la fonction $g(X_M)$ exige $|X_M| = k$ variables et doivent prendre toute fonction avec $g \geq 0$, où :

$$g(X_M) = 100\left[|X_M| + \sum_{x_i \in X_M}\left((x_i - 0.5)^2 - cos\left(20\pi(x_i - 0.5)\right)\right)\right]$$

Le front de Pareto optimal correspond à $x_i = 0.5$ pour tout $x_i \in X_M$ et es valeurs de fonction objectif sont situés sur l'hyperplan linéaire $\sum_{m=1}^{M} f_m^* = 0.5$, la seule difficulté fournie par ce problème est la convergence dans le front de Pareto optimal.

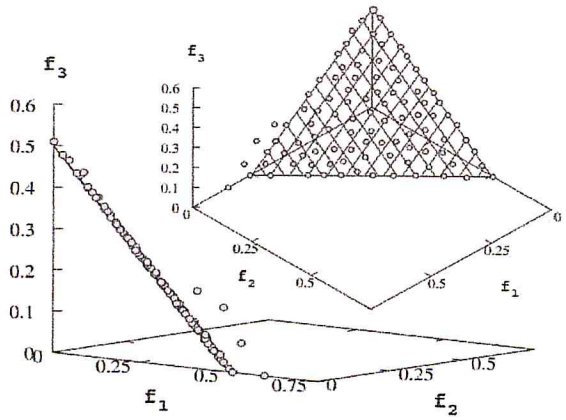


Figure A.1 : La population de SPEA2 pour le problème de test DTLZ1 après 300 générations pour M=3,

Problème test DTLZ2

$$\begin{cases} \min\ f_1(X) = (1 + g(\boldsymbol{X_M}))\, cos(x_1\pi/2) \dots cos(x_{M-1}\pi/2) \\ \min\ f_2(X) = cos(x_1\pi/2) \dots cos(x_{M-2}\pi/2) sin(x_{M-1}\pi/2) \\ \vdots \\ \min\ f_M(X) = (1 + g(\boldsymbol{X_M})) sin(x_1\pi/2) \\ g(\boldsymbol{X_M}) = \sum_{x_i \in \boldsymbol{X_M}} (x_i - 0.5)^2 \\ 0 \le x_i \le 1\ , pour\ tout\ i = 1 \dots n \end{cases}$$

Le front de Pareto optimal correspond à $x_i = 0.5$ pour tout $x_i \in \boldsymbol{X_M}$ et toutes les valeurs d'objectif doit satisfaire $\sum_{m=1}^{M} {f_m^*}^2 = 1$.

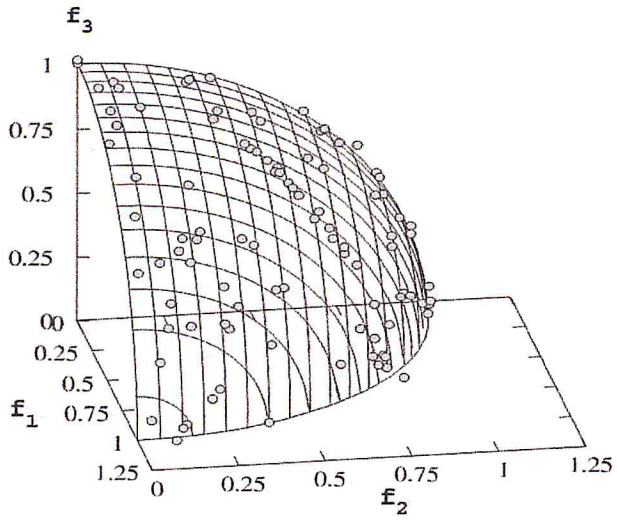


Figure A.2 : La population de NSGA2 sur le problème de test DTLZ2 après 300 générations.

Problème test DTLZ3

$$\begin{cases} \min\ f_1(X) = (1 + g(\boldsymbol{X_M}))\, cos(x_1\pi/2) \dots cos(x_{M-1}\pi/2) \\ \min\ f_2(X) = cos(x_1\pi/2) \dots cos(x_{M-2}\pi/2) sin(x_{M-1}\pi/2) \\ \vdots\ \vdots \\ \min\ f_M(X) = (1 + g(\boldsymbol{X_M})) sin(x_1\pi/2) \\ g(\boldsymbol{X_M}) = 100\left[|\boldsymbol{X_M}| + \sum_{x_i \in \boldsymbol{X_M}} \left((x_i - 0.5)^2 - cos\big(20\pi(x_i - 0.5)\big)\right)\right] \\ 0 \le x_i \le 1\ , pour\ tout\ i = 1 \dots n \end{cases}$$

Le front de Pareto optimal correspond à $x_i = 0.5$ pour tout $x_i \in \boldsymbol{X_M}$.

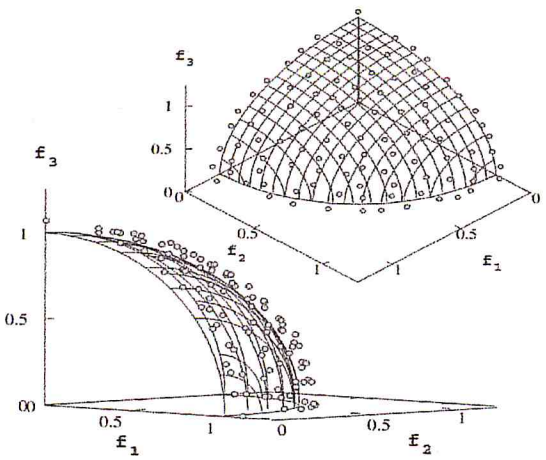


Figure A.3 : La population de SPEA2 sur le problème de test DTLZ3 après 300 générations.

Problèmes test DTZ4

$$\begin{cases} \min\ f_1(X) = (1 + g(X_M))\, cos(x_1{}^{\alpha}\pi/2) \ldots cos(x_{M-1}{}^{\alpha}\pi/2) \\ \min\ f_2(X) = cos(x_1{}^{\alpha}\pi/2) \ldots cos(x_{M-2}{}^{\alpha}\pi/2) sin(x_{M-1}{}^{\alpha}\pi/2) \\ \vdots\ \vdots \\ \min\ f_M(X) = (1 + g(X_M)) sin(x_1{}^{\alpha}\pi/2) \\ g(X_M) = \sum_{x_i \in X_M} (x_i - 0.5)^2 \\ 0 \le x_i \le 1\ , pour\ tout\ i = 1 \ldots n \end{cases}$$

Le paramètre $\alpha = 100$ et Le front de Pareto optimal correspond à $x_i = 0.5$ pour tout $x_i \in X_M$.

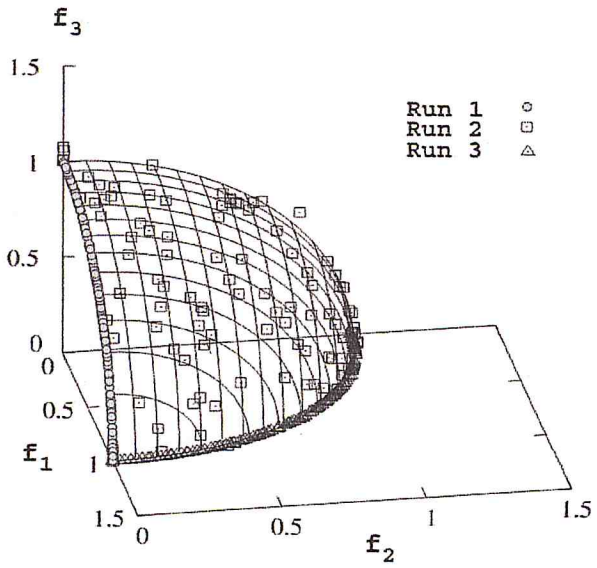


Figure A.4 : La population de SPEA2 sur le problème de test DTLZ4 après 300 générations.

Bibliographie

[1] : A. Abraham, L. Jain et Goldberg, Evolutionary Multi objective Optimization-Theoretical Advances and Applications. Edition Sprinjer. 2005.

[2] : A. Berro, algorithme évolutionnaire pour l'optimisation multi objectif . Séminaire du 4 novembre 2008.-LAAS Laboratoire d'analyse et d'architecture des systèmes.

[3] :O. Roudenko, application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectif avec contraintes. Paris : Ecole polytechnique 2004.

[4] : Veldhuizen, On measuring multi-objective evolutionary algorithm performance D.A.V Veldhuizen and G.B.Lamont. In 2000 Congress of evolutionary computation. Piscataway. New jersey. Volume 1, page 204-211, July 2000.

[5]: M. Schoenauer. Les Algorithmes évolutionnaires : état de l'art et enjeux. Rapport technique, Algorithms seminar 2001-2002 INRIA(2003), pp 113-118.0, 2003. vii, 13.

[6]: Thomas Bàck, David B. Fogel, et Zbigniew Michalewicz, éditeurs. Evolutionary Computation 1 : Basic Algorithms and Operators. Institute of Physics Publishing, Bristol, UK. 2000.

[7]: Y. Collette, P. siarry, Optimisation Multiobectif , Edition EYROLLES. 2002.

[8]: E. Talbi, Métaheuristique pour l'optimisation combinatoire multi-objectif. Laboratoire d'informatique Fondamentale de Lille .1998.

[9]: N. Srinivas and K. Deb. Multiobjective optimization using non-dominated sorting in gebetic algorithm. Evolutionary Computing 2(8): 221-248. 1995.

[10]: D.E. Goldberg. Genetic Algorithms and Walsh Functions : Part I, a Gentle Introduction, In Complex Systems, (1989).

[11]: D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, MA : Addison-Wesley Publishing Co. (1989).

[12]: Th. Back, D.B. Fogel and Z. Michalewicz. "Handbook of Evolutionary Computation", Oxford University Press (1997).

[13]: J. Knowles, L.Thiele et E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK repport 214, (TIK), ETH Zurich, 2006. 16

[14]: J.R. Koza. Genetic Programming : On the Programming of Computers bymeans of Natural Evolution, MIT Press, Massachussetts, (1999).

[15]: K. Deb, A. Pratap, S. Agarwal et T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. IEEE Transactions on Evolutionary Computation, vol. 6, pages 182–197, 2002. 18, 19, 84.

[16]: K. Deb, M. Mohan et S. Mishra. Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In C. Fonseca et al., editeur, EMO'03, pages 222–236. LNCS 2632, Springer Verlag, 2003. 22, 38

[17]: H.P. Schwefel. "Numerical Optimization of Computer Models", John Wiley & Sons, New York, 2nd edition, (1995).

[18]: C. Fonseca Multiobjective genetic algorithms with applications to control engineering problems. 1995.

[19]: L. Tang, X. Wang, A Hybrid Multiobjective Evolutionary Algorithm for Multi objective Optimization Problems. IEEE Transaction on Evolutionary Computation, vol 17. NO 1, 2013.

[20]: H. Hadman : On the distruption-level of polynomial mutation for evolutionary multi-objective optimization algorithm, Computing and informatics, vol, 29, page 783-800, 2010.

[21]: Eckart Zitzler, Marco Laumanns et Lothar Thiele. SPEA2 : Improving the Strength Pareto Evolutionary Algorithm. Rapport technique 103, TIK, ETH Zurich, 2001. 18, 20.

[22]: G. Yen, Z. He, Performance Metric Ensemble for Multi objective Evolutionary Algorithms. IEEE Transaction on Evolutionary Computation, vol. 18, NO. 1, 2014.

[23]: J. Bader and E. Zitzler, HypE: An algorithm for fast hypervolume based many-objective optimization, Evol. Comput., vol. 19, no. 1, pp. 45–76, Jul. 2011.

[24]: S. Yang, M.Li, X.Liu, and J. Zheng , A Grid-Based Evolutionary Algorithm for Many-Objective Optimization. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 17, NO. 5, OCTOBER 2013

[25]: J. Heitkötter, D. Beasley ; The Hitch-Hiker's Guide to Evolutionary Computation; FAQ for comp.ai.genetic; 20 September 2000.

[26]: T.Weise, Global optimization Algorithm theory and application. September 2009.

[27]: E. Alba, B. Dorronsoro, Cellular Genetic Algorithms. Vol 42, Edition Sprinjer. June 2002.

[28]: [28]: D. Francisci, Algorithme Evolutionnaire et Optimisation Multi-objectifs en Data Mining. Informatique, Signaux et Système du Sophia Antipolis, UMR 6070. Mars 2002.

[29]: J. Aikins, H. Shrobe, Proceedings of the Seventh Conference on Innovative Application of Artificial Intelligent. IAAI. 1995.

[30] : C. Léoranrd , Cours de Probabilité et Simulation. Version 2.0. Université Paris Ouest.

[31]: L. Elle, B. Lapeyre, Introduction aux Méthodes de Monte-Carlo. Septembre 2001.

[32] : M. Ejday, Optimisation Multi-Objectifs à base de Métamodèle pour les Procédés de Mise en Forme. Doctorat ParisTech. 2011.

[33] : M. Haj-Rachid, C. Blooch, W. Ramdhan-Cherif, P. Chationnay, Différentes opérateurs évolutionnaires de permutation: sélections, croisements et mutations. LIFC. 2010.