REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DE MATHEMATIQUES ET D 'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'études

Présenté pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Spécialité : Informatique Décisionnelle et Optimisation

Par : BENDAKMOUSSE Faiza

THEME

Algorithme génétique pour la régression du front Pareto en optimisation multi-objectif

Soutenu publiquement le : 03/07/2021 devant le jury composé de :

Dr. HAMAK Allaoua	**Université de M'sila**	**Encadreur**
Dr. MOUHOUB NsserEddine	**Université de M'sila**	**Président**
Pr. GASMI Abdelkader	**Université de M'sila**	**Examinateur**

Promotion : 2020/2021

Dédicace

A mes très chers parents.

Remerciement

Je remercie Dieu le tout puissant qui m'a donné le courage et la force pour aboutir à l'accomplissement de ce travail.

Je voudrais exprimer toute ma reconnaissance et mon respect à Monsieur Alloua Hemmak pour son orientation.

Je remercie les membres du jury pour l'honneur qu'ils m'ont attribué en acceptant d'évaluer et de juger ce modeste travail.

Je remercie mes amies pour leur aide appréciable, leurs encouragements continus et leur soutien moral ininterrompu.

Ce travail n'aurait jamais été possible sans le soutien et l'appui moral des membres de ma famille. Je les remercie tous.

Table des matières

INTRODUCTION GENERALE

Introduction générale

Dans la vie quotidienne on est souvent confronté à toutes sortes de problèmes d'ordre économique, industrielle, militaire, etc. Un problème peut être décrit et représenté sous forme d'un langage formel. Par exemple, plusieurs problèmes peuvent être formulés sous forme d'un problème d'optimisation combinatoire : il s'agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objective sous certaines contraintes.

Proposer des méthodes (ou approches) de résolution pour ce type de problèmes revient à considérer deux points majeurs : la qualité de la solution et le temps d'exécution. En général, la qualité de la solution est elle aussi en fonction du temps.

Généralement, les méthodes que l'on met en œuvre pour résoudre un problème d'optimisation combinatoire dépendent de la complexité de ce dernier. La théorie de la NP-complétude (Garey et Johnson) fournit de précieux renseignements sur le genre de méthodes à adopter en fonction de la difficulté intrinsèque des problèmes. Il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable.

Lorsqu'un seul critère est donné, par exemple un critère de maximisation de profit, la solution optimale est clairement définie, c'est celle qui a le profit maximal. Mais dans de nombreuses situations, un seul critère peut être insuffisant. En effet, la plupart des applications traitées intègrent plusieurs critères simultanés, souvent contradictoires. Intégrer des critères contradictoires est vraiment un problème réel. Considérons par exemples les problèmes suivants :

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

Optimiser un tel problème relève donc de l'optimisation combinatoire multiobjectif. Les premières études concernant l'optimisation combinatoire multiobjectif transformaient les problèmes multiobjectif en une succession de problèmes mono objectif. Pour cela, un ordre d'importance sur les objectifs pouvait être donné, et l'optimisation consistait à optimiser un objectif sans dégrader les valeurs déjà obtenues pour les objectifs plus prioritaires.

Une autre approche consistait en l'optimisation d'une agrégation linéaire des objectifs, chacun pouvant avoir un poids représentant son importance. Lorsque l'on se trouve dans un réel contexte multiobjectif, il n'est pas toujours possible de trouver un ordre

d'importance sur les critères. Il est alors nécessaire de rechercher les solutions de meilleur compromis entre les objectifs. Si cette notion de compromis sera définie plus précisément dans le chapitre 1, il est facile de voir que dans ce contexte la solution recherchée n'est pas une unique solution mais un ensemble de solutions représentant les différents compromis possibles. Ainsi l'optimisation multiobjectif s'intéresse aux particularités liées à l'existence de ces différentes solutions optimales.

Le problème de sac à dos (ou Knapsack problem) est un problème d'optimisation appartenant à la classe des problèmes NP-difficiles. A cause de son utilité, ce problème particulier est de plus en plus utilisé dans le domaine décisionnel.

Le domaine d'application de ce problème (ainsi que ses variantes) inclut des cas du domaine de transport, de la logistique, de la fiabilité ainsi que de la production. Le problème de sac à dos est intensément étudié dans sa version mono objectif depuis plus d'un siècle. L'intérêt qui lui est porté est en particulier dû au fait qu'il se retrouve en tant que sous problème dans de nombreux problèmes d'optimisation.[37]

Dans ce mémoire nous intéressons au problème de sac à dos multiobjectif en variables binaires. Nous étudions un algorithme génétique en s'appuyant sur une procédure de sélection et croisement et mutation. Ainsi, nous présentons une procédure d'agrégation pour transformer problème multi objectif.

Organisation du document

Ce mémoire est structuré de la manière suivante :

- Dans le premier chapitre, je présente l'optimisation multiobjectif. Je donne quelques notions fondamentales concernant l'optimisation multiobjectif telles que la dominance, l'Optimalité de Pareto. Je décris aussi les principales approches de résolution pour ces problèmes.
- Le deuxième chapitre a consacré l'algorithme génétique, Je présente les éléments de cette méthode ainsi que des Principes de fonctionnement.
- Le troisième chapitre contient état de l'art de l'optimisation multiobjectif.
- Dans Le quatrième chapitre, j'étudie le problème de sac à dos multiobjectif, j'ai aussi appliqué la méthode d'agrégation (somme pondérée), et j'ai présenté une description détaillée de différentes étapes de l'algorithme génétique (sélection, croisement, mutation) pour résoudre le problème de sac à dos.
- Dans cinquième chapitre, je proposé deux méthodes pour résoudre le problème du sac à dos multi objectif en 0/1, Premier méthode est l'agrégation, deuxième

méthode est méthode Pareto, j'applique également l'algorithme génétique sur les deux méthodes, et les a comparés.

Le mémoire s'achève par une conclusion générale sur l'ensemble du travail réalisé et des perspectives de recherche induites par les résultats obtenus.

CHAPITRE I

L'OPTIMISATION MULTI-OBJECTIF

Table des Matières

1. Introduction

De nombreux secteurs de l'industrie (mécanique, chimie, télécommunications, environnement, transport, etc.) sont concernés par des problèmes complexes de grande dimension et Multicritères (couts financiers, qualité de service, etc.) pour lesquels les décisions doivent être prises de façon optimale. Les problèmes d'optimisation rencontres en pratique sont rarement uni-objectif. Il y a généralement plusieurs critères contradictoires à satisfaire simultanément. L'optimisation multicritère s'intéressé a la résolution de ce type de problèmes. Elle possède ses racines dans le 19 IIème siècle dans les travaux en économie de Edgeworth et Pareto. Elle a été utilisée initialement en économie et dans les sciences de management et graduellement dans les sciences pour l'ingénieur.

L'optimisation multi-objectif cherche donc à optimiser plusieurs composantes d'un vecteur fonction cout. Contrairement à l'optimisation uni-objectif, la solution d'un Problème Multi-Objectif (PMO) n'est pas une solution unique, mais un ensemble de solutions, connu comme l'ensemble des solutions Pareto optimales (PO). Toute solution de cet ensemble est optimale dans le sens qu'aucune amélioration ne peut être faite sur une composante du vecteur sans dégradation d'au moins une autre composante du vecteur. [15]

2. Problème Multi-Objectif (PMO)

1.2.1 Formulation :

Un PMO (*F, D*) peut être défini comme suit :

$$Min\ F(x) = (f_1(x), f_2(x), ..., f_m(x)) \quad m \geq 2$$
$$avec \quad F = (f_1 : D \rightarrow IR, \ldots, f_m : D \rightarrow IR)$$

m **:** est le nombre de fonctions à optimiser, $x = (x_1, .., x_n)$ est le vecteur des variables de décisions, $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$:est le vecteur des critères à optimiser.

L'image d'une solution x dans l'espace des critères est le point ***y***

$y = (y_1, .., y_m)$ *avec* $yi = fi(x),\ i = 1..m,$

$Y = F(D)$ **:** représente les points réalisables dans l'espace des objectifs. On impose sur cet ensemble une relation d'ordre partiel appelée relation de dominance que nous allons définir dans la section suivante. [2]

1.2.2 La Dominance :

Une solution $y = (y_1, .., y_m)$ domine une solution $z = (z_1, .., z_m)$ **Ssi** $\forall i \in 1, .., m \; f_i(y) \leq f_i(z)$

et $\exists j \in 1, .., m$ tq $f_j(y) < f_j(z)$ Si la solution **y** domine la solution **z** nous notons alors $\mathbf{y} \prec \mathbf{z}$ Notons que pour toute paire de solutions **y** et **z** un et un seul des cas suivants peut se présenter :

- ✓ ***Y*** domine ***z***
- ✓ ***Y*** est dominé par ***z***
- ✓ ***Y*** et ***z*** sont équivalentes au sens de la dominance.

Les solutions équivalentes au sens de la dominance sont appelées dans ce qui suit, solutions équivalentes au sens de Pareto ou solutions Pareto équivalentes ou, encore, solutions non-dominées.

Voici la (**figure1**) qui représente le test de domination

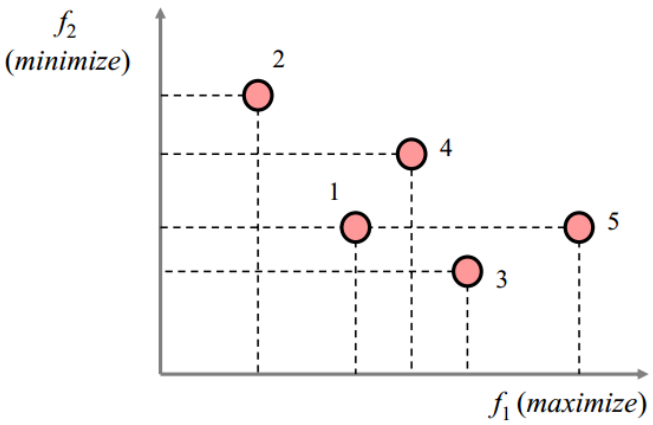


Figure1. Test de dominance. [19]

- ✓ **1** vs **2** : **1** domine **2**
- ✓ **1**vs **5** : **5** domine **1**
- ✓ **1** vs **4** : aucune solution ne domine aucune solution

Propriétés de la relation de dominance :

La relation de dominance telle qu'elle est définie ci-dessus :

- ✓ N'est pas réflexive, car une solution ne se domine pas elle-même.

- ✓ N'est pas symétrique, car on n'a jamais y ≺ z et z ≺ y
- ✓ Est transitive, car si y ≺ z et z ≺ w alors y ≺ w. [2]

1.2.3 Optimalité de Pareto :

La définition de solution Pareto optimale découle directement de la notion de dominance. Elle signifie qu'il est impossible de trouver une solution qui améliore les performances sur un critère sans que cela entraine une dégradation des performances sur au moins un autre critère. Les solutions Pareto optimales sont aussi connues sous le nom de solutions admissibles, efficaces, non-dominées, et non-inferieures. L'ensemble exact de toutes les solutions Pareto optimales sera noté P O. [15]

Optimalité locale au sens de Pareto :

Une solution **y** est optimale localement au sens de Pareto s'il existe un réel $\delta > 0$ tel qu'i n'ait pas une solution z dominant y et vérifiant $|z - y| <= \delta$

Optimalité globale au sens de Pareto :

Une solution **y** est optimale globalement au sens de Pareto, ou optimale au sens de Pareto, ou encore Pareto-optimale s'il n'existe aucun point de l'espace faisable D qui la domine. L'ensemble des solutions Pareto optimale est appelé l'ensemble de Pareto ou également l'ensemble des compromis optimaux.

- ✓ L'image de l'ensemble de Pareto dans l'espace des critères est appelé la surface de Pareto (ou le front de Pareto pour le cas bi-objectif) ou encore la surface des compromis optimaux. [2].

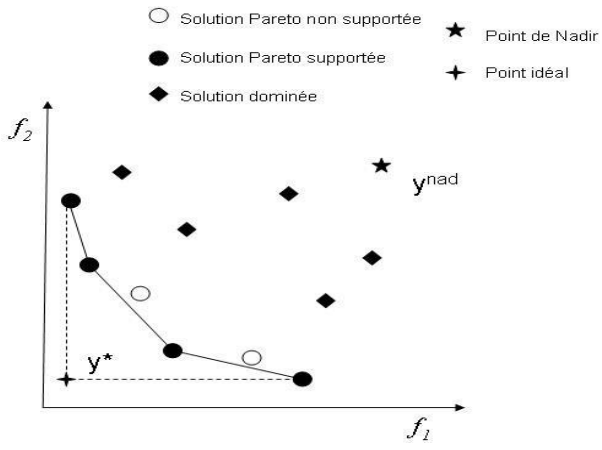


Figure2. Représentation des solutions supportées et non supportées, point idéal et point de Nadir [2]

Point idéal et Point de Nadir :

Le vecteur idéal $y* = (y1*, .., ym*)$ est obtenu en optimisant séparément chaque fonction objectif fi , i.e. $yi* = fi*(x), x \in D$ **(voir figure 2).** Généralement ce vecteur n'appartient pas à l'espace objectif réalisable mais il est dans certains cas utile en tant que référence, par exemple, pour normaliser les valeurs des objectifs. A la différence du vecteur idéal qui représente les bornes inférieures de chaque objectif dans l'espace faisable, le vecteur de Nadir correspond à leurs bornes supérieures sur la surface de Pareto, et non pas dans tout l'espace faisable (voir figure2). Ce vecteur sert à restreindre l'espace de recherche ; il est utilisé dans certaines méthodes d'optimisation interactives. [2].

1.2.4 Convexe :

Certaines méthodes d'optimisation multi objectif nécessitent de travailler sur un espace *Y des* valeurs de F qui soit convexe.

Définition :

Un ensemble ***Y*** est convexe si, pour n'importe quels deux points distincts de cet ensemble, le segment qui relie ces deux points est contenu dans l'ensemble ***S***.

1.2.5 Buts dans Optimisation Multi-Objectif

Trouver un ensemble de solutions aussi proche que possible du front Pareto-optimal

Trouver un ensemble de solutions aussi diversifiées que possible (**figure 3**). **[19]**

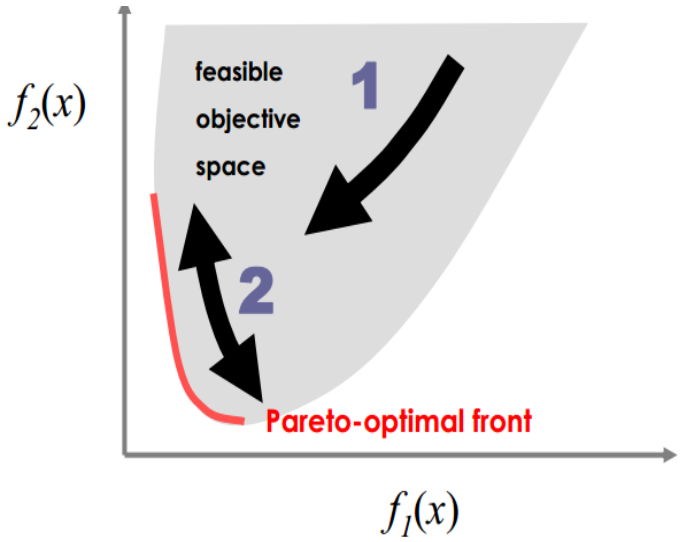


Figure3. Buts dans Optimisation Multi-Objectif. [19]

3. Complexité et difficulté de l'optimisation combinatoire multi objectif :

Les problèmes d'optimisation combinatoire multi objectif (OCMO) sont très complexes.
En effet, la difficulté des problèmes d'optimisation due au phénomène d'explosion combinatoire s'additionne de nouvelle difficulté propre aux problèmes d'optimisation multi objectif.

Un problème d'OCMO est un problème Non Polynomial(NP) si la détermination d'un point non dominé ne peut être faite en temps polynomial pour toutes les instances du problème. Sera fini (1987) a proposé la définition suivante de la NP-complétude pour les problèmes OCMO :

Définition 1 : Pour chaque point **z** existe -t- il une solution x $\in X$ tel que : $F(x) \geq z$?

D'après la définition1, un problème D'OCMO serait NP si les problèmes d'optimisation combinatoire mono objectif qui lui correspondent sont NP. Sera fini démontre aussi
comment certains problèmes d'optimisation mono objectif "facile" deviennent NP-complet dans le cas bi-objectif. C'est le cas pour le problème d'affectation et le problème de plus court chemin. Dans plusieurs OCMO, la détermination d'un ensemble complet est très difficile. Malgré que les solutions supportées sont obtenues par la résolution d'un programme linéaire pondéré s'il existe un algorithme efficace pour la résolution du problème mono objectif associé, mais aussi le calcul des solutions non supportées est très intéressant. Or, la détermination de ces solutions nécessite généralement de résoudre des problèmes NP-difficiles, même si le problème mono objectif considéré peut être résolu en temps polynomial.

Par conséquent, une difficulté pratique dans la résolution des MOCO provient de la détermination d'un ensemble *XNE*. De plus, des expérimentations ont montré que les solutions
non supportées, en plus d'être plus difficiles à obtenir, peuvent également être beaucoup plus nombreuses que les solutions supportées. D'un point de vue théorique, la plupart des OCMO sont NP-complets, P-complets et intraitables. En particulier, Hugot a démontré l'intraitable du problème de sac à dos multi objectifs. Cependant, cette propriété apparaît même si le problème mono objectif correspondant est dans la classe P. [37]

4. Classification des méthodes :

Dans les différentes publications rencontrées, nous avons retrouvé deux classifications différentes des approches de résolution des problèmes multi-objectifs. La première catégorie adopte un point de vue décideur, les approches sont classées en fonction de l'usage qu'on désire en faire. La seconde adopte un point de vue concepteur, les approches sont triées de leur façon de traiter les fonctions objectives. Donc, avant de se lancer dans la résolution d'un PMO, il faut se poser la question du type d'approches de résolution que l'on veut utiliser. Voir la (**figure 4** Classification des méthodes d'optimisation multi-objectif). [27]

Décideur

- ✓ Les méthodes a priori (décideur → recherche)
- ✓ Les méthodes a posteriori (recherche → décideur)
- ✓ Les méthodes progressives ou interactives (décideur ↔ recherche).[4]

1.4.1 Classification « point de vue décideur » :

On distingue trois schémas possibles. Soit le décideur intervient dès le début de la définition du problème, en exprimant ses souhaits et préférences, afin de transformer un PMO en un problème simple objectif. Soit le décideur effectue son choix dans l'ensemble des solution proposées par le solveur multi-objectif. [27]

1.4.1.1 Les approches a priori : le décideur intervient en aval du processus d'optimisation, pour définir la fonction d'agrégation modélisant le compromis que l'on désire faire entre les différents objectifs. Dans ce cas le décideur est supposé connaître a priori le poids de chaque objectif afin de les mélanger dans une fonction unique. Cela revient à résoudre un problème mono-objectif. Cependant dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, parce que les différents objectifs sont non commensurables (exprimés dans des unités différentes). [21]

1.4.1.2 Approches interactives (Ou bien progressives) : Elles combinent de manière cyclique et incrémentale les processus de décision et d'optimisation. Le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi interactivement le compromis entre ses préférences et les résultats. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche. [27]

1.4.1.3 Les approches a posteriori : cherche à fournir au décideur un ensemble de bonnes solutions bien réparties. Il peut ensuite, au regard de l'ensemble des solutions, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contrepartie fournir un ensemble de solutions bien réparties, ce qui peut également être difficile et requérir un temps de calcul important (mais ne nécessite pas la présence du décideur). Nous nous placerons dans le cadre de cette troisième famille de méthodes où la modélisation des préférences n'est pas requise et où le procédé d'optimisation doit être puissant afin de fournir une très bonne approximation de la frontière Pareto**.).** [21]

Concepteur

- ✓ Les méthodes agrégées
- ✓ Les méthodes non agrégées et non Pareto

✓ Les méthodes basées sur Pareto. [4]

1.4.2 **Classification « point de vue concepteur » :**

1.4.2.1 **Les méthodes agrégées :**

Agrégation pondérée :

Cette méthode de résolution de PMO est la plus évidente et probablement, la plus largement utilisée en pratique. Elle consiste à ramener le problème multi objectif au problème de l'optimisation d'une combinaison linéaire des objectifs initiaux. Ainsi, il s'agit d'associer à chaque fonction objectif un coefficient de pondération et à faire la somme des fonctions objectifs pondérées pour obtenir une nouvelle et unique fonction objective.

Un PMO se transforme alors de la manière suivante :

$$\boldsymbol{Min} \sum_{i=1}^{m} \boldsymbol{w_i} * \boldsymbol{f_i}(\mathrm{x}) \qquad m \geq 2$$

Où les poids $\boldsymbol{w_i} \geq \boldsymbol{0}$ sont tels que $\sum_{i=1}^{m} \mathrm{Wi} = 1$

Cette méthode peut, en faisant varier les valeurs du vecteur poids w, retrouver un ensemble de solutions supportées, si le domaine réalisable est convexe. Cependant, elle ne permet pas de trouver les solutions enfermées dans une concavité (les solutions non supportées).

Les résultats obtenus avec de telles méthodes dépendent fortement des paramètres choisis pour le vecteur de poids **w**. Les poids $\boldsymbol{w_i}$ doivent également être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate. Une approche généralement utilisée consiste à répéter l'exécution de l'algorithme avec des vecteurs poids différents. [2]

Goal programming [Charmes 1961] :

Le décideur fixe un but $\boldsymbol{Ti}$ à atteindre pour chaque objectif $\boldsymbol{fi}$. Ces valeurs sont ajoutées au problème comme des contraintes. La nouvelle fonction objective est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre.

Le min max [Coello 1995] :

Elle minimise le maximum de l'écart relatif entre un objectif et son but associé par le décideur. [4]

La méthode ε-contrainte [Ritzel 1994] :

Cette méthode est basée sur la minimisation d'un objectif ***fi*** en considérant les autres objectifs contraints par une valeur ***εj.***[4]

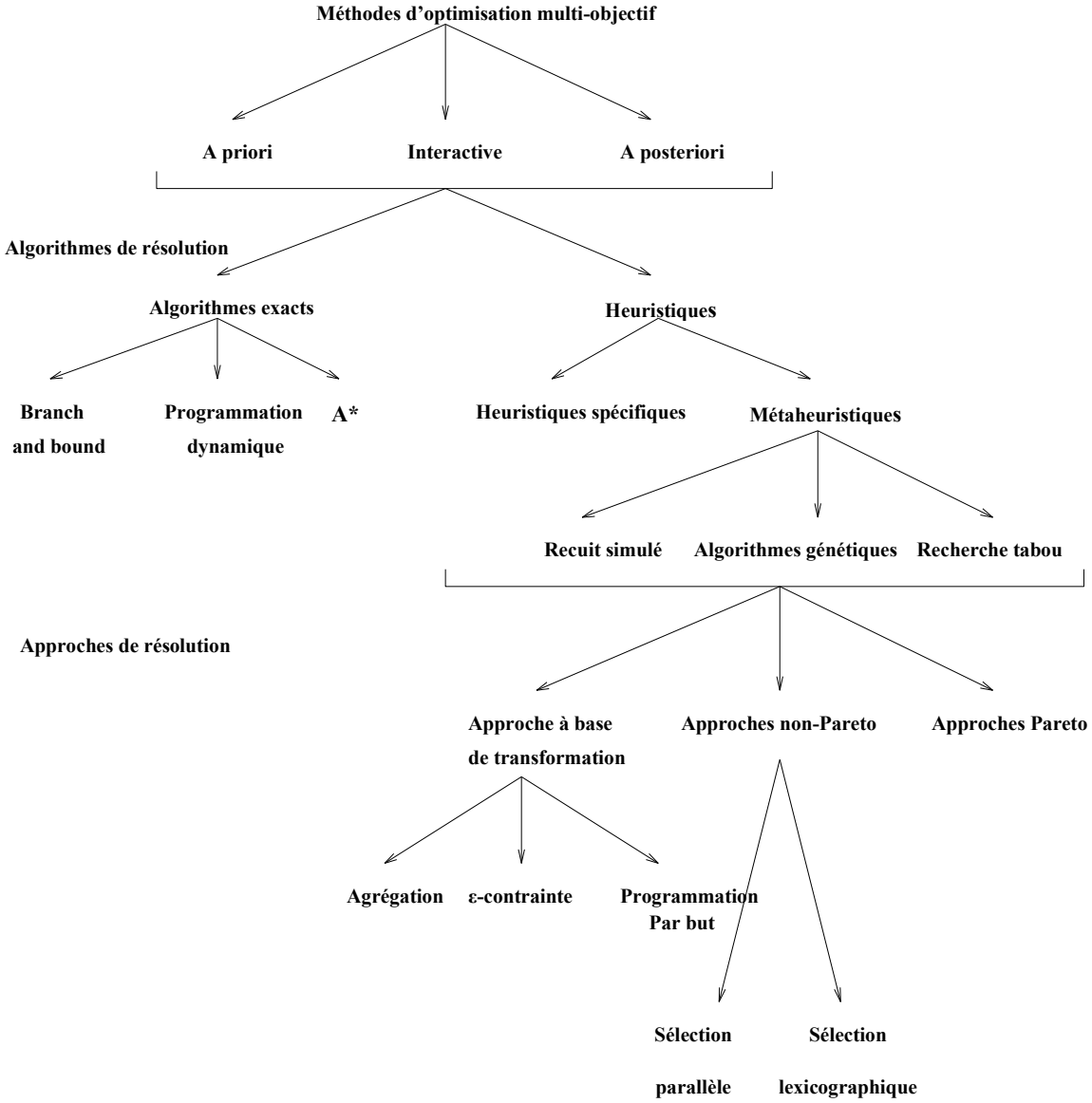


Figure 4 : Classification des méthodes d'optimisation multi-objectif. [15]

1.4.2.2 **Approches non-agrégées :**

Nous présentons dans cette section quelques méta heuristiques appartenant à cette classe de méthodes :

Les algorithmes génétiques :

Les AGs ont été largement utilisés dans la communauté multi-objectif. Ils sont très appropriés pour résoudre des PMOs grâce à l'utilisation d'une population de solutions. Les AGs peuvent chercher plusieurs solutions Pareto-optimales dans la même exécution. [2]

Recuit simulé :

Le premier algorithme multi-objectif basé sur le recuit simulé a été proposé par Sera fini en 1992. La méthode utilise le schéma standard du recuit simulé avec une seule solution courante. Le résultat de l'algorithme est un ensemble de solutions Pareto-optimales contenant toutes les solutions non-dominées générées par l'algorithme. [2]

Recherche tabou :

Grandibleux et al [Gandibleux et al, 1997] ont proposé une version multi-objectif de la recherche taboue. La méthode utilise des fonctions pondérées scalaires dont les poids sont changés périodiquement. La modification du vecteur poids dégrade les poids des objectifs qui ont été nettement améliorés. Deux listes taboues sont utilisées. La première est une liste taboue régulière qui empêche de revenir aux solutions déjà visitées. La deuxième contient les vecteurs poids. [2]

1.4.2.3 Approches Pareto :

Les approches Pareto utilisent directement la notion de dominance dans la sélection des solutions générées, contrairement aux autres approches qui utilisent une fonction d'utilité ou traitent séparément les différents objectifs. Cette idée a été introduite initialement dans les AGs par Goldberg. Le principal avantage de ces approches est qu'elles sont capables de générer des solutions Pareto optimales dans les portions concaves de la frontière Pareto.

Les AGs ont été largement utilisés pour la résolution de PMO, étant donné qu'ils travaillent sur une population de solutions. Deux objectifs doivent être pris en compte dans la résolution d'un PMO :

- ✓ **Converger la frontière Pareto :** la plupart des travaux de recherche sur l'application des AGs aux PMO se sont concentrés sur l'étape de sélection. Dans cette étape, des méthodes de ranking sont appliquées dont le rôle est d'établir un ordre (rank) entre les individus. Cet ordre dépend de la notion de dominance et donc directement de l'optimalité Pareto. Les méthodes de ranking permettent de converger vers les solutions Pareto optimales.

- ✓ **Trouver des solutions diversifiées dans la frontière Pareto :** les méthodes de maintenance de la diversité, par la formation de niches écologiques et d'espèces, peuvent être particulièrement utiles pour stabiliser des sous-populations multiples le long de la frontière Pareto. [15]

1.4.3 Les méthodes exploitant une métaheuristique :

Les métaheuristiques, sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile. Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Ces méthodes ont des inspirations de l'éthologie comme les colonies de fourmis, de la physique comme le recuit simulé, et de la biologie

comme les algorithmes évolutionnaires. Dans la (figure 5) suivante, on voit un classement de ces méthodes selon le principe d'inspiration utilisé, est ce qu'il est basé sur le principe de population de solution ou non, aussi est ce que ces méthodes sont exactes ou non.[9].

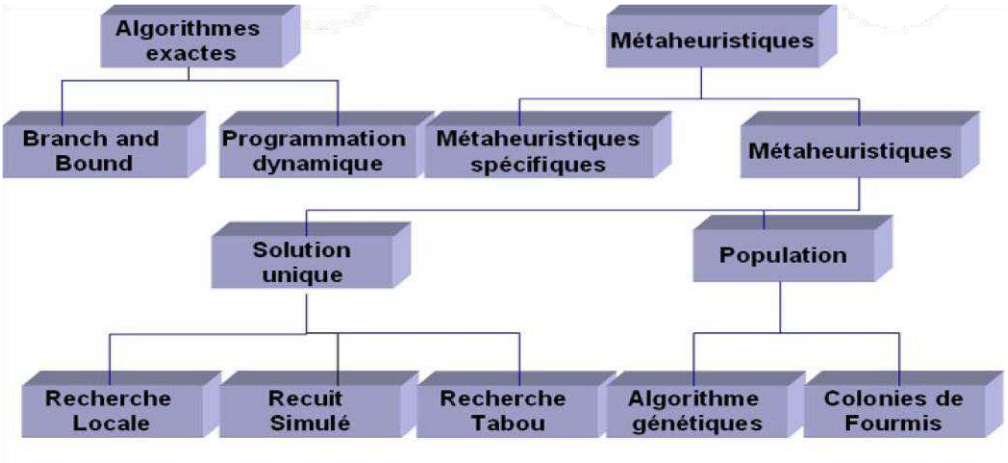


Figure 5. Méthodes basées sur les métaheuristiques.[9]

Si l'on considère les différentes métaheuristiques, on constate que celles-ci se répartissent en trois familles :

- ✓ Les méthodes déterministes de recherche d'optimum local.
- ✓ Le recuit simulé.
- ✓ La méthode P.A.S.A (Pareto Archived Simulated Annealing) .
- ✓ La Recherche Tabou.
- ✓ La méthode de Colonie de Fourmis.
- ✓ Particle Swarm Optimization « PSO ».

1.4.4 **Les algorithmes évolutionnaires** :

En 1860 Charles Darwin publie son livre intitulé « L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature ». Dans ce livre, Darwin rejette l'existence «de systèmes naturels figés », déjà adaptés pour toujours à toutes les conditions extérieures, et expose sa théorie de l'évolution des espèces : sous l'influence des contraintes extérieurs, les êtres vivants se sont graduellement adaptés à leur milieu naturel au travers de processus de reproductions. La sélection naturelle : Sélection des individus les mieux « adaptés » à un milieu donné et qui auront une plus grande faculté de reproduction que les autres. La sélection naturelle soutient donc que les êtres vivants qui s'adaptent le mieux aux conditions naturelles de leur environnement vaincront et survivront. Dans la (figure 6) suivante , nous offrons les différentes branches des algorithmes évolutionnaires[9].

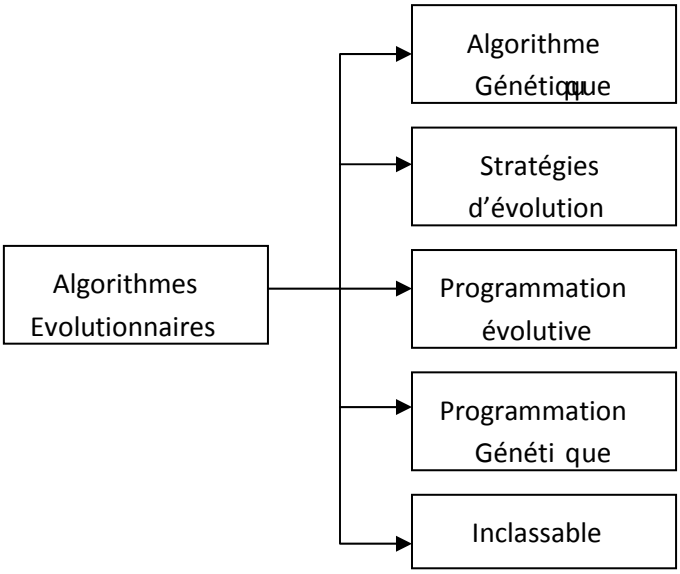


Figure 6. Les différentes branches des algorithmes évolutionnaires [9].

1.4.4.1.1 ***Types d'algorithmes évolutionnaires :***

Algorithmes Génétiques (GA) :

Le mécanisme des AG [Holland, 1962, Goldberg, 1989, Michalewicz, 1996] consiste à faire évoluer, à partir d'un tirage initial, un ensemble de points de l'espace de recherche vers l'optimum du problème. Avec des opérations extrêmement simples et sans se soucier de la sémantique de ces chromosomes codés en combinant ses solutions entre elles pour en former de nouvelles en essayant d'hériter des bonnes caractéristiques des solutions parents. On obtient une solution qui s'approche de l'optimum en un temps acceptable. C'est en effet cette légèreté de mise en œuvre qui fait la puissance et le charme des AGs.[21]

Stratégies d'évolution (ES) :

Inventées par I. Rechenberg et H.P. Schwefel, 1965, Berlin. Les ES ont été mises au point par deux jeunes ingénieurs travaillant sur des problèmes numériques. Un énorme progrès a été apporté par les techniques adaptatives d'ajustement des paramètres de mutation, et ce sont sans contestation les meilleurs algorithmes pour les problèmes purement numériques. [9]

Programmation évolutionnaire (EP) [Fogel, 1966] :

Ce modèle évolutionniste accentue l'utilisation de la mutation et n'utilise pas dans sa version originale la recombinaison des individus par croisement. Développé à l'origine pour l'évolution d'automates à état fini, ce modèle est souvent appliqué à la résolution de problèmes d'optimisation à variables réelles. Dans ce cas, il utilise une mutation qui consiste à ajouter une perturbation Gaussienne à chaque composante du vecteur à variables réelles constituant l'individu. Cette perturbation est basée sur la performance de l'individu : l'idée consiste à faire subir des mutations importantes aux mauvais individus et inversement des mutations faibles aux bons individus. L'opérateur de sélection est de type probabiliste : il

s'agit de la méthode du tournoi basée sur une compétition entre individus choisis aléatoirement. [21]

Programmation génétique (GP) :

Élaborée par J. Koza, 1990, Californie, USA. Apparue initialement comme sous-domaine des GAs, GP est devenu une branche à part entière (conférence, journal, ...). La spécificité de GP est l'espace de recherche, fait d'arbres représentant des programmes complets. GP cherche à atteindre un des vieux rêves des programmeurs, faire écrire le programme par un autre programme. [9]

5. Conclusion :

Un problème multi objectifs ne consiste pas à rechercher la solution optimale, mais l'ensemble des solutions satisfaisantes, pour lesquelles on ne pourra pas effectuer une opération de classement. Dans ce cas il faut utiliser des méthodes capables de trouver l'ensemble de Pareto tout en se basant sur les contraintes et la définition de problème. Dans la littérature, il existe plusieurs méthodes pour ce but, des méthodes non mathématiques et tend vers l'ensemble de solutions dans un temps optimal et d'une manière simple en se basant sur le principe des heuristiques, cette tendance est « **les méta heuristiques** », comme **les algorithmes évolutionnaires**, recuit simulé… [9]

CHAPITRE II

LES ALGORITHMES GENETIQUES

Table des matières

1. Introduction :

Un algorithme génétique **(AG)** est un méta heuristique qui manipule une population de solutions potentielles à la fois. Le mode de fonctionnement d'un **AG** est calqué sur les principes biologiques de la sélection naturelle et de la survie des individus les mieux adaptés à l'environnement **(Darwin, 1876).**La sélection naturelle est basée sur l'idée que les modifications des générations successives sont orientées par les pressions extérieures auxquelles sont soumises les espèces (ex :la limitation des ressources, les modifications de l'environnement, les prédateurs et parasites, etc.).Il en résulte que les individus les mieux adaptés à l'environnement tendent à survivre plus longtemps et à se reproduire plus fréquemment. S'inspirant ainsi de ce mécanisme, **Holland (1975)** a posé les bases de la technique d'optimisation appelée **"*algorithmes génétiques*"**. Mais c'est **Goldberg (1989)** qui, par la suite s'est investi dans l'étude des **AG** et a développé la forme actuelle que nous connaissons. [5]

2. Définition :

Le but principal d'un **AG** est de chercher, dans un espace de solutions E, un élément de cet espace ayant la meilleure *adaptation* à l'environnement posé. Chaque élément de *E* est un individu, noté *x* . Une *population P* est donc un ensemble de *N* éléments (*individus*) de *E : P = (x1, x2,..., xn)*.La mesure du degré d'adaptation de chaque individu constitue la fonction de performance *F (fitness)*. L'objectif d'un **AG** est de faire évoluer cette population *P* afin de trouver le meilleur individu ***x*.*** Dans ce but, à chaque génération ***t,*** les individus de la population *P(i)* sont *sélectionnés, croisés* et *mutés* selon des probabilités prédéfinies respectivement ***pc*** (*probabilité de croisement)* et ***pm*** *(probabilité de mutation*). [18]

3. Les éléments d'un algorithme génétique :

Génotype ou Chromosome :

Le génotype est constitué de gènes situés sur des chromosomes stockés dans le noyau des cellules sous la forme d'une longue chaîne d'acide désoxyribonucléique (ADN).

On peut donc décrire l'ADN par des chaînes de quatre caractères ACGT. L'ADN constitue l'ensemble des chromosomes, ou le génome d'un individu. C'est une autre façon de dire « Individu ».

Gène : Un chromosome est composé de gènes. Dans le codage binaire, un gène vaut soit 0 soit 1.

Phénotype : Le phénotype est l'ensemble des protéines et des enzymes qui peuvent être fabriqués à partir de l'ADN. En fait, l'ADN est copié par un messager (ARN) qui au niveau du ribosome, se traduit en chaînes d'acides aminés formant les protéines et les enzymes. [9]

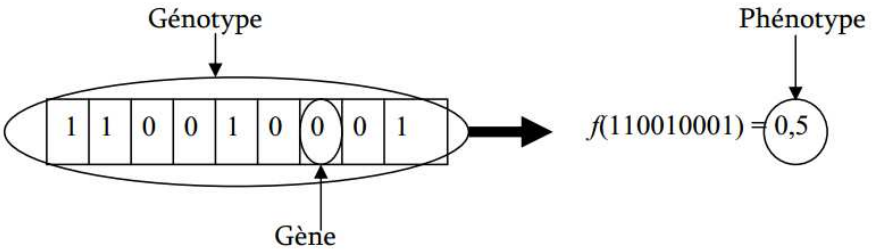


Figure 7 : Le vocabulaire des algorithmes génétiques. [9]

En général, on compte une protéine (un enzyme) par gène. Ce sont les protéines et les enzymes qui dictent la structure et le comportement des cellules qui permettent à un individu de :

- Réaliser des tâches dans son environnement,
- Survivre,
- Reproduire à des taux différents.

Chaque génotype représente une solution potentielle à un problème d'optimisation. La valeur de cette solution potentielle est appelée **le phénotype**.

Individu :

En biologie un individu est une forme qui est le produit de l'activité des gènes. Pour un AG, il est réduit à un chromosome et on l'appelle donc chromosome ou individu pour désigner un même objet. [30]

Population :

C'est l'ensemble des individus, ou encore l'ensemble des chromosomes d'une même génération. Habituellement, la taille de la population reste constante tout au long de l'algorithme génétique. [6]

La fitness ou fonction d'évaluation :

Le calcul de la qualité d'un individu est essentiel aux algorithmes génétiques. Cette fonction donne, en valeur numérique (habituellement réelle), la qualité d'un individu. C'est selon cette valeur numérique que sont calculées les chances de sélection de cet individu. La fonction de *fitness* doit avoir 0 comme plancher, pour ne pas fausser le calcul des pourcentages. [30]

4. Principes de fonctionnement des AG :

Le fonctionnement de tout AG peut être décrit par le principe illustré par la **figure 8**:

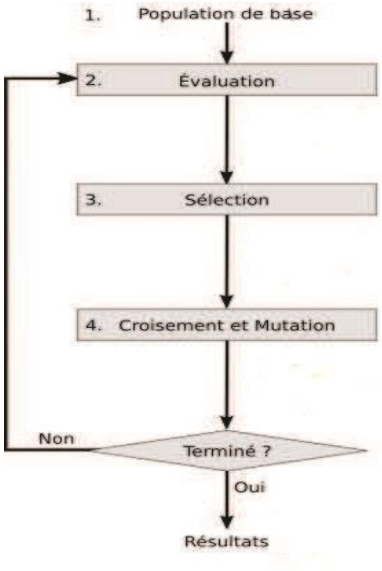


Figure 8 : Les principales étapes d'un algorithme génétique. [28]

A un niveau très général, le fonctionnement d'un AG est alors basé sur les phases suivantes (voir **Figure 9** pour un exemple délibérément trivial que nous allons utiliser pour illustrer cette section) :

1. **Initialisation** (**Figure 9**-étape **(a)** pour t = 0). Une population initiale de chromosomes est tirée aléatoirement.

2. **Évaluation** (premier élément de l'étape **(b)**). Chaque chromosome est décodé, puis évalué.

3. **Sélection** (les derniers éléments de l'étape **(b)** et **(c)**). Création d'une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée.

4. **Reproduction**. Possibilité de *croisement* (étape **(d)**) et de *mutation* (étape **(e)**) au sein de la nouvelle population.

5. **Retour** à la phase d'évaluation (étape **(f)**) tant que la condition d'arrêt du problème n'est pas
satisfaite.

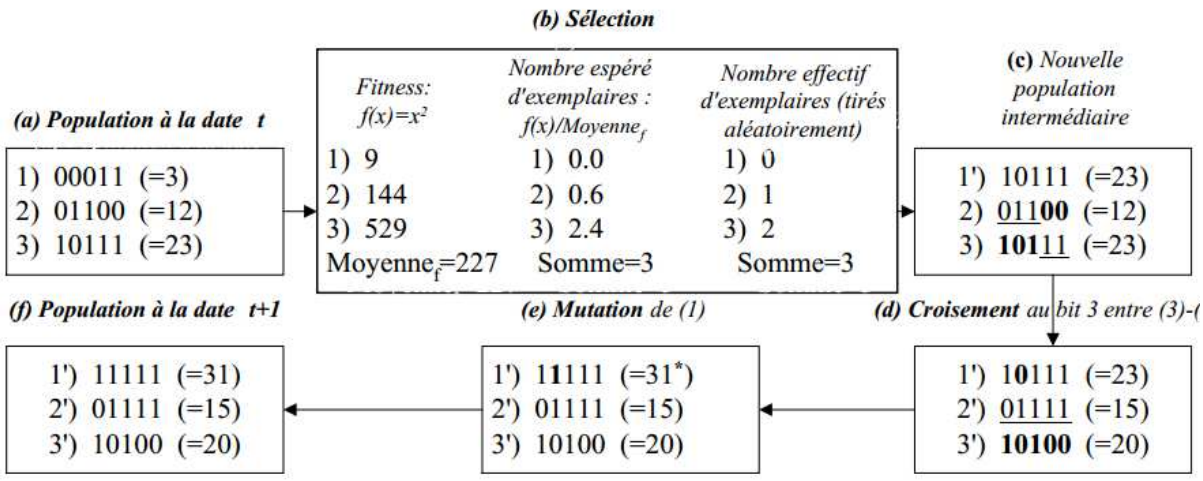
Cet exemple illustre ces différentes étapes. Nous allons maintenant les discuter plus en détail. [31]

Codage et population initiale :

Premièrement, il faut représenter les différents états possibles de la variable dont on cherche la valeur optimale sous une forme utilisable par un AG : c'est le *codage*. Cela permet d'établir une connexion entre les valeurs de la variable et les individus de la population, de manière à imiter la connexion qui existe en biologie entre le génotype et le phénotype. Il

existe principalement deux types de codage : le codage binaire (représentation sous forme de chaîne binaire) et le codage réel (représentation directe des valeurs réelles de la variable). [31]

Définition : Séquence/ Chromosome /Individu (Codage binaire) : Nous appelons une séquence *(*chaîne, chromosome, individu*)* de longueur l ***(A)*** une séquence A = {a_1 ,a_2,a_3 …..a_l } avec ∀i ∈{***1 ,….,l***}, ai ∈ V = {0 ,1} . Un chromosome est donc une suite de bits (formée de zéros et d'uns), appelé aussi chaîne binaire.



Une application des AG à l'optimisation de la fonction $f(x)=x^2$ sur le domaine formé par des entiers entre 0 et 31.
Les entiers sont codés avec une chaîne binaire de cinq bits : 00001=1, 11111=31. L'exemple utilise une population initiale aléatoire de trois individus (chromosomes) et l'AG construit une nouvelle population à travers une reproduction sélective (**étape (b)**), des croisements (combinaison de solutions existantes - **étape (d)**) et des mutations (expériences aléatoires, introduction de la nouveauté - **étape (e)**). L'AG atteint en une étape la solution optimale (x=31) dans cet exemple schématique. Pour chaque chaîne, le croisement, position et le partenaire impliqué sont déterminés de manière aléatoire. La mutation simplement change la valeur du bit concerné : de 0 vers 1 ou de 1 vers 0. **Ce processus est contrôlé par : la taille de la population, la taille des chaînes binaires et les probabilités de croisement et de mutation**

difficile d'expliquer l'importance isolée de chacun de ces opérateurs dans la réussite de l'AG. Pour partie cela tient au fait que chacun de ces opérateurs agit selon divers critères qui lui sont propres (valeur sélective des individus, probabilité d'activation de l'opérateur, etc.). [31]

1.5.1 La sélection :

Cet opérateur est chargé de définir quels seront les individus de P qui vont être dupliqués dans la nouvelle population P' et vont servir de parents (application de l'opérateur de croisement). Soit *n* le nombre d'individus de P, on doit en sélectionner *n/2* (l'opérateur de croisement nous permet de repasser à *n* individus). Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se

reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population. On trouve essentiellement quatre types de méthodes de sélection différentes :

- ✓ La méthode de la "loterie biaisée" (roulette wheel) de GoldBerg.
- ✓ La sélection par tournois.
- ✓ La méthode "élitiste".
- ✓ La sélection universelle stochastique. [32]

1.5.1.1 La sélection par roulette (Wheel) :

C'est une méthode stochastique qui exploite la métaphore d'une roulette de casino. Elle consiste à associer à chaque individu un segment (ou case de la roue) dont la longueur est proportionnelle à sa performance comme l'illustre la (**Figure 10**). La roue étant lancée, l'individu sélectionné est celui sur lequel la roue s'est arrêté. Cette méthode favorise les meilleurs individus, mais tous les individus conservent néanmoins des chances d'être sélectionnés. [18]

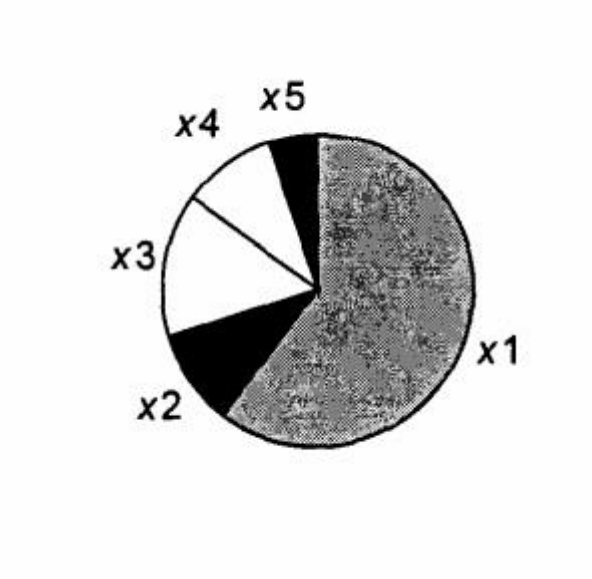


Figure 10 : Roulette pour une population de 5 individus avec ***f(xi) = {60,10,15,10, 5}.*** Pour tirer un individu, on lance la roue, et si elle s'arrête sur le segment i, *xi* est sélectionné. [18]

1.5.1.2 La sélection par tournoi :

La sélection par tournoi fait intervenir le concept de comparaison entre les individus. Son principe est de choisir un groupe de ***q*** individus aléatoirement dans la population, de sélectionner d'une manière déterministe le meilleur dans ce groupe, et de recommencer l'opération jusqu'à l'obtention du nombre d'individus requis. Au cours d'une génération, il y autant de tournois que d'individus à sélectionner. La pression de sélection est ajustée par le nombre de participants ***q*** à un tournoi. Un ***q*** élevé a une forte pression de sélection et

inversement. L'avantage de cette technique de sélection est qu'elle n'est pas coûteuse à mettre en œuvre et à exécuter. [18]

1.5.1.3 **La sélection par Elitisme :**

A la création d'une nouvelle population, il y a de grandes chances que les meilleurs chromosomes soient perdus après les opérations d'hybridation et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions. [6]

1.5.1.4 La sélection universelle stochastique :

Cette méthode semble être très peu utilisée et qui plus est possède une variance faible, donc introduit peu de diversité, nous n'entrerons donc pas dans les détails, on se contentera d'exposer sa mise en œuvre : On prend l'image d'un segment découpé en autant de sous-segment qu'il y a d'individus. Les individus sélectionnés sont désignés par un ensemble de points équidistants. [32]

1.5.2 **Le croisement :**

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus) par le biais de leur combinaison. La population qui résulte de la sélection est divisée en deux sous population de taille et chaque couple formé par un membre provenant de chaque sous population participe à un croisement avec une probabilité donnée (la probabilité de croisement, N/2 pc, souvent supérieur à 60%).

Si le croisement a lieu, sa localisation entre la position 1 et la position *l,* dans le cas du codage binaire, est tirée selon une loi uniforme et les deux individus échangent leurs gènes des deux côtés de cette localisation. Dans notre exemple un croisement localisé à la troisième position a eu lieu entre les individus 2 et 3 :

$$\begin{array}{l} (12=)\boxed{\mathbf{011}\,|\,\mathbf{00}} \\ (23=)\boxed{101\,|\,11} \end{array} \xrightarrow{Croisement} \begin{array}{l} \boxed{\mathbf{011}\,|\,11}(=15) \\ \boxed{101\,|\,\mathbf{00}}(=20) \end{array}$$

Deux nouveaux individus ont été créés par ce biais. Toutefois, un individu sélectionné lors de la reproduction ne subit pas nécessairement l'action d'un croisement. Ce dernier ne s'effectue qu'avec une certaine probabilité. Plus cette probabilité est élevée et plus la population subira de changement. On peut aussi imaginer que le croisement a lieu entre plusieurs zones différentes des chaînes (plutôt que deux comme dans cet exemple).

Quoi qu'il en soit, il se peut que l'action conjointe de la reproduction et du croisement soit insuffisante pour assurer la réussite de l'AG. Ainsi, dans le cas du codage binaire, certaines chaînes peuvent totalement disparaître de la population. Par exemple si aucun individu de la population initiale ne contient de 1 en dernière position de la chaîne, et que ce 1 fasse partie de la chaîne optimale à trouver, aucun croisement ne peut faire apparaître cet élément. Ce dernier ne peut s'introduire dans la population que si l'on permet l'expérimentation au hasard et c'est, entre autres, pour remédier à ce problème que *l'opérateur de mutation* est utilisé. [31]

1.5.3 **La mutation :**

Le rôle de cet opérateur est de modifier aléatoirement, avec une certaine probabilité, la valeur d'un composant de l'individu. Dans le cas du codage binaire, chaque bit $a_i \in \{0\ 1\}$, est remplacé selon une probabilité p_m par son complémentaire $: \tilde{a}_i = 1 - a_i$. La mutation est traditionnellement considérée comme un opérateur intervenant à la marge dans la mesure où sa probabilité est en général fixée assez faible (de l'ordre de 1%). Mais elle confère aux algorithmes génétiques une propriété très importante : tous les points de l'espace de recherche peuvent être atteints. Cet opérateur est donc d'une grande importance et il est loin d'être marginal. Il a de fait un double rôle : celui d'effectuer une recherche locale et/ou de sortir d'une trappe (optima locaux). [9]

6. **Conclusion :**

Dans ce chapitre, nous avons présenté les algorithmes génétiques de manière générale et donné les éléments de base nécessaires à la compréhension de la méthode.

CHAPITRE III

ETAT DE L'ART DE L'OPTIMISATION MULTI-OBJECTIF

Table des Matières

1. Introduction

La naissance de l'optimisation multi-objectif remonte à un ouvrage de W. Pareto 1906 sur l'économie politique, dans lequel l'auteur définit pour la première fois se qu'est un optimum multi-objectif. L'application de l'optimisation multi-objectif aux problèmes d'ingénierie remonte, aux, alentours de la seconde guerre mondiale, elle est longtemps restée une science « anecdotique » à cause de son aspect hermétique, dû à la nécessite de maîtriser un bagage mathématique théorique assez important. Un changement radical est apparu récemment avec les méta heuristique (recuit simule, algorithmes génétiques, etc.) qui sont des méthodes d'optimisation d'un abord plus facile.

Deux approches de l'optimisation multi objective s'affrontent :

- ✓ Résolution vectorielle du problème : on ne modifie pas l'expression du Problème
- ✓ Transformation du problème en un problème d'optimisation mono objectif. [26]

Dans Ce chapitre introduit l'état de l'art de l'optimisation multi-objectif.

2. Méthodes exactes pour l'optimisation multi-objectif :

Concernant les méthodes exactes, plusieurs approches basées sur des procédures de séparation et évaluation (branch and bound), sur l'algorithme A^* et la programmation dynamique ont été proposées pour résoudre de petits problèmes à deux objectifs (problèmes bi-objectifs).

Une approche particulière pour l'optimisation multi-objectif est le "goal programming" (programmation par buts). Une approche intéressante a été proposée par B. Ulungu et J. Teghem pour la recherche du front Pareto de problèmes bi-objectifs. Cette approche a été utilisée efficacement sur des problèmes tels que l'affectation ou le sac à dos bi-objectifs. Cette méthode a ensuite été améliorée afin d'obtenir des fronts complets de façon plus efficace. [20]

1.2.1 Méthode de séparations et évaluations

Une méthode (branch & bound) multi objectifs est présentée et appliquée à l'optimisation combinatoire bi-objectif d'un transformateur de sécurité. De nouveaux critères sont proposés pour le branchement et le rejet. Ils sont basés sur la dominance de Pareto et la métrique de contribution. La comparaison avec un dénombrement exhaustif et un algorithme

génétique de tri non dominé confirme les solutions. Il apparaît que les méthodes exactes et approximatives sont toutes deux très sensibles à leurs paramètres de contrôle. [3]

1.2.2 **Programmation Linéaire Multi objectif**

Le but de ce document de travail est de présenter les différents concepts utilisés en programmation linéaire multi objectif en précisant les types de décision qu'il est possible de prendre, les choix des partenaires et les méthodes qu'il convient de choisir pour résoudre de tels problèmes. Par la suite le problème général est posé et différentes méthodes de résolution sont proposées. [25]

3. La méthode de la somme pondérée pour l'optimisation multi objectifs :

En tant que concept courant dans l'optimisation multi-objectifs, la minimisation d'une somme pondérée constitue une méthode indépendante ainsi qu'une composante d'autres méthodes. Par conséquent, la compréhension des caractéristiques de la méthode de la somme pondérée a des implications de grande portée Cependant, malgré les nombreuses applications publiées pour cette méthode et la littérature traitant de ses pièges en ce qui concerne la représentation de l'ensemble optimal de Pareto, il y a peu de discussion complète concernant la signification conceptuelle des poids et des techniques pour maximiser l'efficacité de la méthode par rapport à un articulation a priori des préférences. Ainsi, dans cet article, nous étudions la signification fondamentale des poids en termes de préférences, l'ensemble optimal de Pareto et les valeurs de la fonction objectif. Nous déterminons les facteurs qui déterminent quel point de solution résulte d'un ensemble particulier de poids. Les déficiences fondamentales sont identifiées en termes d'articulation a priori des préférences. [24]

4. Métas heuristiques pour l'optimisation combinatoire multi-objectif

Cet article présente un état de l'art des méta heuristiques appliquées à la résolution de problèmes d'optimisation combinatoire multi-objectif, suivant une classification proposée. L'objectif principal de telles méthodes est de générer une variété de solutions Pareto-optimales diversifiées dans l'espace de recherche. Une analyse critique de chaque classe de méthode est présentée. Des réponses à des questions ouvertes comme l'évaluation des performances et la comparaison d'algorithmes d'optimisation multi objectif, et l'étude des paysages des frontières Pareto sont abordées. Certains axes de recherche future dans ce domaine tel que la conception d'algorithmes parallèles et hybrides sont aussi identifiés. [15]

5. Contribution Des Meta heuristiques Dans L'optimisation Multi objectif

Les métaheuristiques représentent une alternative indispensable qui a fait preuve de très bons résultats durant ces dernières armées. L'objectif de notre thèse est de présenter les méta heuristiques destinées pour la résolution des problèmes multi objectifs et nous nous sommes intéressés aux Algorithmes Evolutionnaires Multi objectifs (AEMOs) citant VEGA qui est basé sur une approche non scalaire et non Pareto, dans la catégorie des AEMOs basés sur une approche scalaire, on peut citer l'algorithme WBGA. Après ces premiers travaux, une première génération de AEMOs de type Pareto NPGA, SPEA et NSGA-II. Au stade actuel de la recherche, la méthode non dominantes storting genette algorithme Il (NSGA-II) est considérée comme une référence incontournable. [23]

6. Application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectif avec contraintes

Dans cette thèse, deux problèmes de l'industrie automobile sont étudiés. Le premier concerne l'optimisation paramétrique de la forme d'un pare-chocs de voiture, un problème a 10 objectifs issus de 3 domaines mécaniques : crash, acoustique et statique. Le second problème qui se pose lors du calibrage du moteur diesel Common Rail (rampe commune) consiste à minimiser la consommation spécifique du carburant ainsi que le bruit de la combustion tout en respectant les normes européennes de fonctionnement en termes de nuisances à l'environnement.

Une tendance remarquable des Algorithmes Evolutionnaires est que ces méthodes "pénètrent" aujourd'hui dans de nombreux nouveaux domaines d'application malgré l'absence de bases théoriques (notamment, de preuves de convergence) aussi solides que celles qu'on peut trouver pour des approches alternatives. Inspirée par cette observation, la motivation principale de ce travail était de contribuer au développement des Algorithmes Evolutionnaires Multi-objectif de façon à rendre leur application aux problèmes réels la plus efficace possible. Ainsi, une contribution originale de cette thèse consiste à répondre à un manque criant dans ce domaine, le manque de critère d'arrêt plus fin qu'une simple borne sur le nombre d'itérations. Le critère d'arrêt proposé dans ce travail est destiné à optimiser le rapport entre la qualité des solutions et le cout de calcul : dans la pratique c'est ce compromis qui est le plus souvent recherché. De même, un nouvel opérateur de croisement bas sur la

relation de la dominance de Pareto est proposé et nous montrons l'accélération de la progression vers la surface des compromis optimaux qu'il apporte. [14]

7. Evolution des algorithmes évolutionnaires multi-objectifs (Multi-Objective Evolutionnary Algorithms - MOEAs) :

Suite au premier travail recensé dans le domaine, consistant en la proposition de l'algorithme VEGA (Vector Evaluated Genetic Algorithm) par Shaffer en 1984-85, la proposition de Goldberg consistant à utiliser la notion d'optimalité de Pareto dans la sélection a été suivie. Ceci a donné naissance à différents algorithmes évolutionnaires multi-objectifs. Nous pouvons citer MOGA, NSGA ou encore NPGA. Dans ces algorithmes, la qualité d'une solution est évaluée en fonction de sa dominance au sein de la population et la diversité maintenue à l'aide de stratégie de **"niching**". Puis, dans le but d'assurer la convergence vers le front Pareto, la question de préservation de l'élite est devenue fondamentale. Ainsi de nouveaux algorithmes, pouvant faire intervenir ou non des archives de solutions non dominées ont été proposés. Nous pouvons citer parmi ces algorithmes élitistes SPEA, PAES ou encore NSGA-II. Ainsi une classification des Algorithmes Evolutionnaires multi-objectifs est souvent utilisée. Cette classification distingue les algorithmes non élitistes, n'ayant aucun opérateur de préservation de l'élite, des algorithmes élitistes prévoyant un opérateur préservant l'élite des solutions. [20]

8. Algorithme génétique multi-objectifs adaptatif

Propose un nouvel algorithme génétique multi objectifs adaptatif appelé aGAME (***adaptive Genetic Algorithm with Multiple Pareto sets***). GAME utilise une structuration de la population en plusieurs fronts de Pareto, de nouvelles stratégies de sélection et diverses fonctions de fitness. GAME introduit un opérateur d'adaptation dynamique afin d'améliorer, à la fois, les performances et la robustesse de GAME. aGAME alterne entre quatre modes de parcours de l'espace de recherche : initialisation (ou exploration initiale), mode normal, exploration et exploitation. Les changements de modes d´pendent de valeurs d'indicateurs de convergence et de diversité. aGAME est comparé, dans un premier temps, à sa version statique (GAME) afin de valider l'impact de l'opérateur d'adaptation dynamique. Dans un second temps, le nouvel algorithme proposé est comparé aux trois meilleurs algorithmes de la compétition CEC 2009 sur des problèmes bi-objectifs avec des contraintes.[1]

9. Système interactif d'aide à la décision basé sur des algorithmes génétiques pour l'optimisation multi-objectifs

Propose d'intégrer les systèmes interactifs d'aide à la décision, l'optimisation multi-objectifs et les algorithmes génétiques afin de proposer un outil performant permettant la résolution de problèmes d'optimisation multi-objectifs. Dans le SIAD proposé, le traitement d'un problème multi-objectifs se fera en deux phases. La première phase consiste à approximer l'ensemble Pareto optimal. Cette étape sera réalisée à l'aide d'un nouvel algorithme génétique multi-objectifs hybride. Cette approche combine un algorithme génétique basé sur les concepts d'élitisme, de niche et de dominance Pareto avec des opérateurs de recherche locale. La deuxième phase utilise l'expérience du décideur afin d'approfondir la recherche dans une zone plus spécifique de l'ensemble pseudo Pareto Optimal en fonction des préférences exprimées par celui-ci. Pour cela, une approche générique de recherche de solutions de compromis est combinée avec un algorithme génétique. Le SIAD proposé est un outil flexible et facile d'utilisation grâce à son interface homme-machine conviviale. Cet outil ne constitue qu'un support à la prise de décision, la décision finale restant du ressort du planificateur. Un exemple d'application du SIAD proposé a été réalisé pour aborder un problème d'ordonnancement industriel rencontré dans une entreprise de production d'aluminium. *Cette* application montre bien l'intérêt pratique de ce genre de système. [18]

10. Optimisation multi-objectif par colonies de fourmis

D'étudier les capacités de la métaheuristique ACO pour la résolution des problèmes d'optimisation multi objectif. Pour ce faire, il est indispensable de choisir la stratégie phéromone appropriée. De plus, pour ce genre de problèmes, le choix des structures phénoménales est un autre point clé dans la résolution. En effet, pour les problèmes mono objectif, généralement une seule structure de phéromone est utilisée correspondant à l'unique l'objectif. Cependant, lorsqu'on a plusieurs objectifs à optimiser simultanément, il n'est plus évident comment définir les structures phénoménales, comment les associer aux fonctions objectives, ni comment les exploiter lors de la construction de solutions.[2]

11. Contribution à la synthèse et l'optimisation multi-objectif par essaims particulaires de lois de commande robuste RST de systèmes dynamiques

Ces travaux de recherche portent sur la synthèse systématique et l'optimisation de correcteurs numériques à structure polynomiale RST par approches métaheuristiques. Les problèmes classiques de placement de pôles et de calibrage des fonctions de sensibilité de la boucle fermée RST sont formulés sous forme de problèmes d'optimisation multi-objectif sous contraintes pour lequel des algorithmes métaheuristiques de type NSGA-II, MODE, MOPSO et epsilon-MOPSO sont proposés et adaptés. Deux formulations du problème de synthèse RST ont été proposées. La première approche, formulée dans le domaine temporel, consiste à minimiser des indices de performance, de type IAE et MO, issus de la théorie de la commande optimale et liés essentiellement à la réponse indicielle du système corrigé. Ces critères sont optimisés sous des contraintes non analytiques définis par des gabarits temporels sur la dynamique de la boucle fermée. Dans la deuxième approche de synthèse RST, une formulation dans le domaine fréquentiel est retenue. La stratégie proposée consiste à définir et calculer une fonction de sensibilité de sortie désirée en satisfaisant des contraintes de robustesse de type $H\infty$. L'utilisation des parties fixes dans la fonction de sensibilité de sortie désirée assurera un placement partiel des pôles de la boucle fermée RST. L'inverse d'une telle fonction de sensibilité désirée définira le filtre de pondération $H\infty$ associé. [10]

12. Les méthodes hybrides

Afin d'améliorer les performances d'un algorithme, on essaye de le combiner avec une autre méthode. Ce principe général appelé hybridation, peut s'appliquer à plusieurs méthodes. Un cas particulier de l'hybridation entre deux méthodes consiste à combiner un algorithme génétique avec une méthode de recherche locale. Dans une telle hybridation on substitue souvent la mutation par une méthode de recherche locale. Dans le cas des problèmes multi objectifs on peut citer les méthodes hybrides suivantes :

- ✓ La méthode MOTS combinant une population et une recherche Tabou,
- ✓ La méthode PSA combinant un algorithme génétique et le recuit simulé,
- ✓ La méthode M-PAES intégrant un schéma généralisant l'implémentation d'un grand nombre d'algorithmes hybrides pour l'optimisation multi objectif.

1.12.1 **Coopération de méthodes**

Etat de l'art sur la coopération de méthodes

L'idée de faire coopérer différents types de méthode n'est pas nouvelle. Très vite, il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés et on a cherché à profiter des avantages des différentes méthodes. Pourtant l'essentiel des études de coopération ont jusqu'alors porte sur la collaboration entre méthodes heuristiques. Dans son état de l'art sur la coopération des metaheuristiques réalise en 2002, E-G. Talbi propose une taxonomie des méthodes coopératives, en fonction du schéma de coopération utilise. [20]

Coopération entre méthodes exactes et metaheuristiques

Un état de l'art de ces coopérations a été proposé par Stützle et Dumitrescu [Dumitrescu, 2003]. Ils distinguent cinq types d'approches :

- ✓ Utilisation d'un algorithme exact pour explorer des larges voisinages dans une recherche locale.
- ✓ Utilisation des solutions de bonne qualité afin de réduire l'espace de recherche de la méthode exacte.
- ✓ Exploitation des bornes de la méthode exacte pour une heuristique constructive.
- ✓ Utilisation des informations fournies par les relaxations des problèmes linéaires pour orienter un algorithme de recherche local ou constructif.
- ✓ Utilisation d'une méthode exacte pour une fonction spécifique de la métaheuristique. [21]

13. **Conclusion**

L'impossibilité technique de résoudre exactement les problèmes NP-difficile de grande taille et/ou des problèmes avec plus de deux objectifs, impose l'utilisation des heuristiques et en particulier les métaheuristiques (génériques). Néanmoins, les méthodes exactes peuvent être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de

sous-problèmes et résolution heuristique du problème complet. Ces travaux sont généralement efficaces, car les deux méthodes coopérant ont alors des particularités bien différentes, et associent leurs avantages afin d'obtenir de bons résultats. [21]

CHAPITER IV

AlGORITHME GÉNÉTIQUE POUR PROBLÈME DE SAC À DOS MULTI-OBJECTIF

Table des Matières

Introduction

Dans les chapitres précédents, on a étudié les problèmes d'optimisation multiobjectifs, quelques exemples ainsi que les méthodes utilisées pour résoudre ce genre de problèmes. On a constaté que les méthodes les plus répandues sont les algorithmes évolutionnaires à cause de leurs avantages, surtout qu'elles n'étudient plus la nature et la définition du problème mais l'ensemble de ses solutions qui sont représentées d'une manière très simple (binaire, réel, ou programme). Ces méthodes qui favorisent la manipulation de ces solutions par des opérateurs génétiques, on a trouvé que les plus utilisées dans l'optimisation multiobjectifs sont les algorithmes génétiques tout en exploitant un ensemble d'approches, chacune a son principe d'évaluation, d'enregistrement des meilleures solutions.

État de l'art des AGs appliqués à l'OMO

C'est dans le milieu des années 80 [Schaffer, 1985] que, pour la première fois, les AG ont été utilisés pour l'optimisation multi-objectifs. Depuis, plusieurs implementations différentes des AG ont été proposées et appliquées avec succès à divers PMO [Hajela et Lin, 1992 ; Fonseca et Fleming, 1993 ; Horn et al, 1994 ; Srivinas et Deb, 1994 ; Ishibuchi et Murata, 1996 ; Valenzuela-Rendon et Uresti-Charre, 1997]. Récemment, certains chercheurs ont étudié quelques aspects des AG pour l'optimisation multi-objectifs comme la convergence vers la frontière Pareto [Rudolph, 1998] ou Yélitisme [Obayashi et al, 1998 ; Parks et Miller, 1998]. Parallèlement, de nouvelles techniques ont été développées [Zitzler et Thiele, 1998 ; Knowles et Corne, 2000 ; Coello Coello et Pulido, 2001]. [18]

Les approches basées sur la transformation du problème en un problème uni-objectif

Les implementations des AG de cette catégorie d'approche sont basées sur les techniques classiques pour générer des compromis sur l'ensemble PO. Certaines approches comme le Weight-Based Genetic Algorithm (WBGA) [Hajela et Lin, 1992] ou le MultiObjective Genetic Local Search (MOGLS) [Ishibuchi et Murata, 1996], par exemple, utilisent la technique de la moyenne pondérée. Comme chaque individu utilise une 53 combinaison particulière de poids qui est soit choisie aléatoirement, soit encodée dans l'individu, tous les membres de la population sont évalués par différentes fonctions objectifs. En conséquence, l'optimisation est effectuée dans plusieurs directions simultanément [Francisci, 2002]. Cependant, les inconvénients potentiels des techniques classiques, comme la sensibilité à la forme de la frontière Pareto pour certaines, peuvent restreindre l'efficacité de ce type d'AG [Van Veldhuizen, 1999]. [18]

Problème du sac à dos

Le problème du sac à dos (knapsack problème) est un problème académique d'optimisation combinatoire appartenant à la classe des problèmes NP-difficile. On le retrouve sous de nombreuses formes, mono-objectif KP, multidimensionnel [M]KP et multiobjectif MOKP. Le fait qu'on le rencontre dans des domaines d'application aussi différents que l'économie (la répartition de budgets), les transports (chargement de cargaison) et l'informatique répartie (allocation de ressources), etc. lui confère un grand intérêt pratique. Le principe consiste à sélectionner un sous-ensemble d'objets maximisant une fonction dite objectif de manière à transporter la plus grande valeur. Son expression Formelle est la suivante :

$$\begin{cases} \max Z^j(x) = \sum_{i=1}^{n} v_i^j x_i \quad j=1,\ldots,p\ ; \\ \sum_{i=1}^{n} w_i^l x_i \le W^l \quad l=1,\ldots k\ ; w_i^l \ge 0\ ; \\ x_i \in \{0,1\} \end{cases}$$

Un sac de capacité ou volume scalaire W si k =1 ou vectoriel W^l si k >1 à ne pas dépasser.

Une fonction objectif scalaire Z si p =1ou vectoriel Z^j si p >1 à optimiser (maximiser).

n : Nombre d'objets où chaque objet i possède, un poids $\mathbf{w_i}$ si k =1 ou un vecteur de poids $\mathbf{w_i^l}$ si k >1. Et un vecteur de valeur $\mathbf{v_i^j}$ par rapport au critère j.

On définit pour chaque objet i une variable de décision binaire xi.

Si l'objet i est retenu alors $\mathbf{x_i}$ =1 sinon $\mathbf{x_i}$ = 0. [21].

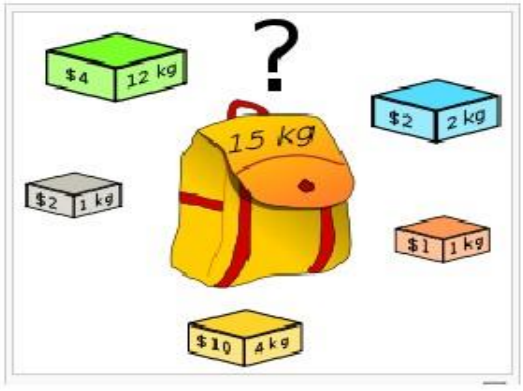


Figure 11 : problème du sac à dos. [37]

k \ *p*	=1	>1
=1	KP (Mono-objectif)	MOKP (Multiobjectif)
>1	[M]KP (Multidimensionnel)	MO[M]KP (Multidimensionnel Multiobjectif)

Tab1 : certains types de problème du sac à dos.

L'espace de recherche est composé de toutes les configurations données par les vecteurs binaires (x_1, x_2…, x_n) vérifiant toutes les contraintes de sac à dos**.** [21]

Problème du sac à dos Multiple (MKP)

Le problème du sac à dos multiple (MKP) est une généralisation du problème standard du sac à dos en variable 0-1 ; nous n'avons donc pas un seul sac à dos mais **m** sac à dos de capacités différentes. [36]

Problème du sac à dos multidimensionnel (MDKP ou MKP)

Le problème du knapsack est dit multidimensionnel en cas où il existe plusieurs sacs à remplir simultanément. Le problème du sac à dos multidimensionnel en variables 0-1 (noté MKP) est une généralisation du problème KP. Il correspond au cas où le nombre de contraintes de capacité est strictement supérieur à 1.

Il existe différentes nominations du MKP. Parmi lesquelles on peut citer : le problème du sac-à-dos multi-contraint ou sac-à-dos multiple. Mais la plus répandue reste celle du sac-à-dos multidimensionnel.

Le MDKP revient à sélectionner un sous-ensemble d'éléments de façon à maximiser le profit total tout en respectant la disponibilité de chaque ressource. [36]

Problème du sac à dos multi objectifs (MOKP) :

Ce problème est parmi les problèmes les plus étudiés dans la communauté multi-objective. L'objectif de ce problème est de maximiser un vecteur de fonction profit tout en satisfaisant un ensemble de contraintes de capacité du sac à dos.

Le problème de sac à dos multi-objectifs multidimensionnel (MOKP) consiste à sélectionner un sous-ensemble d'objets maximisant une fonction multi-objectifs tout en satisfaisant un ensemble de contraintes.

On parle d'optimisation multiobjectifs dans le cas où le problème consiste à optimiser plusieurs fonctions objectives. La notion d'optimalité laisse place à la notion d'efficacité, puisque nous avons plusieurs fonctions objectives à optimiser. L'objectif est donc de trouver l'ensemble des solutions du problème non dominées.

On dit qu'une solution x domine fortement une solution y, si elle est meilleure sur l'ensemble des objectifs. Une solution x domine une solution y, s'il existe au moins un critère sur

lequel la valeur de x est strictement supérieure à la valeur de y. Il existe de nombreux concepts concernant la notion d'optimalité.

Le MOKP peut être utilisé pour modéliser de nombreux problèmes réels comme la répartition de budgets et l'allocation de ressources. [36]

Modélisation *:*

Plus formellement, un MOKP est défini comme suit : **m** est le nombre de fonctions objectif, **n** est le nombre d'objets, x_j est la variable de décision associée à l'objet j, **q** est le nombre de contraintes de ressource, w_j est la quantité de la ressource i consommé par l'objet j, b_i est la quantité totale disponible de la ressource i, p_j^k est le profit associé à l'objet j relativement l'objectif k .

$$\begin{cases} Max\left(f_i(x)\right) = \sum_{j=1}^{m} c_j^i x_j \\ C = x / \sum_{j=1}^{m} w_j\, x_j \;\; < W \\ x_j \in \{0,1\};\ \forall j = 1,..,m, i = 1,..,n \end{cases}$$

Avec $x \in C$

w_j: poids de l'élément j et c_j^i : utilité de l'élément j / critère i

$$x_j = \begin{cases} 1\ si\ l'élément j\ est\ dans\ le\ sac \\ 0\ sinon \end{cases}$$. [36]

État de l'art de problème de sac à dos multi-objectif

Le MOKP utilisé dans ce chapitre est celui proposé par Zitzler et Thiele [1999] et Zitzler problèmes les plus étudiés dans la communauté multi-objectifs. Ainsi, nous pourrons comparer les performances de notre approche avec celles des meilleurs algorithmes de la littérature. Le nofej-objectifs problème de sac à dos en 0/1 est défini par un ensemble de j items (ou objets). À chacun des items sont associés Z valeurs de profit ainsi que Z poids. Formellement, le problème peut être exprimé de la manière suivante. Soit un ensemble de J objets et Z sacs à dos avec :

Pzj : profit correspondant à l'objet j en fonction du sac à dos z.

Wzj : poids correspondent à l'objet j en fonction du sac à dos z.

Capz : capacité totale du sac à dos z.

II faut donc trouver un vecteur x = $\{x_1, x_2, \ldots, x_j\}$ £ $\{0,1\}^j$ qui maximise

$$f_z(x) = \sum_{j=1}^{J} p_{z,j} x_j \qquad z = 1,\ldots, Z$$

Sujet à :

$$\sum_{j=1}^{J} w_{z,j} x_j \leq cap_z \quad z = 1,\ldots, Z$$

Le problème consiste donc à sélectionner un sous-ensemble d'objets Xj maximisant f(x)= (f1(x), f2(x), ..., fz(x)) et pour lequel les contraintes de capacité sont respectées.

Plusieurs heuristiques ont été développées pour résoudre ce problème. En recensant la littérature, on retrouve, sans être exhaustif, des méthodes de recherche dans le voisinage comme le recuit simulé [Serafini 1994 ; Ulungu et al. 1999], la recherche avec Tabou [Gandibleux et al. 1996], des méthodes évolutionnaires comme le VEGA [Schaffer 1985], le NSGA [Srivinas et Deb 1994] et le NSGAII [Deb 2000], le SPEA [Zitzler et Thiele 1998] et le SPEA2 [Zitzler et al. 2001], le PMSMO [Zinflou et al. 2006], ou encore le M-PAES [Knowles et Corne 2000]. Plus récemment, des approches hybrides combinant algorithme génétique et méthode de recherche locale ont été proposées pour résoudre le problème comme le Multi-Objective Genetic Local Search (MOGLS) [Jaszkiewicz 2000] ou le Genetic Tabu Search for the Multi-Objective Knapsack Problème (GTSM0Jas) [Barichard 2003]. Le MOGLS utilise une méthode de descente pure comme opérateur de recherche locale alors que le GTSMOs : p, de son côté, utilise un algorithme de recherche avec tabous comme opérateur de recherche locale. Ces deux algorithmes, contrairement à GISMOO, ne sont pas des approches Pareto et ils utilisent une fonction d'agrégation linéaire pondérée pour évaluer la performance des individus trouvés au cours des itérations. Mentionnons que parmi les approches hybrides pour le MOKP, le MOGLS est un des algorithmes les plus réputés et est souvent utilisé dans la littérature pour des fins de comparaison. Toutefois, à notre connaissance, les méthodes hybrides utilisées pour résoudre ce problème se limitent principalement à des approches hybridant algorithmes génétiques et méthodes de recherche locale. [35]

Exemple de problème de sac à dos multi objectif :

- Nombre des objets : n = 5.
- Capacité de sac : w = 15
- Wi : w1=5 ; w2=4 ;w3=6 ; w4=3 ; w5= 8 .
- Les profit : v_{a1} =10 ; v_{a2}= 40 ; v_{a3}=30 ; v_{a4} =50 ; v_{a5} = 70 .
  v_{b1}=15 ; v_{b2}= 35 ; v_{b3}=33 ; v_{b4} =45 ; v_{b5} = 60 .
  v_{c1}=12 ; v_{c2}= 42 ; v_{c3}=35 ; v_{c4} =48 ; v_{c5} = 66 ;

Les objets i	1	2	3	4	5
V_{a1}	10	40	30	50	70
V_{b2}	15	35	33	45	60
V_{c3}	12	42	35	48	66
Wi	5	4	6	3	8

Tab2 : MOKP avec 3 objectifs.

Les éléments d'un AG pour le MOKP

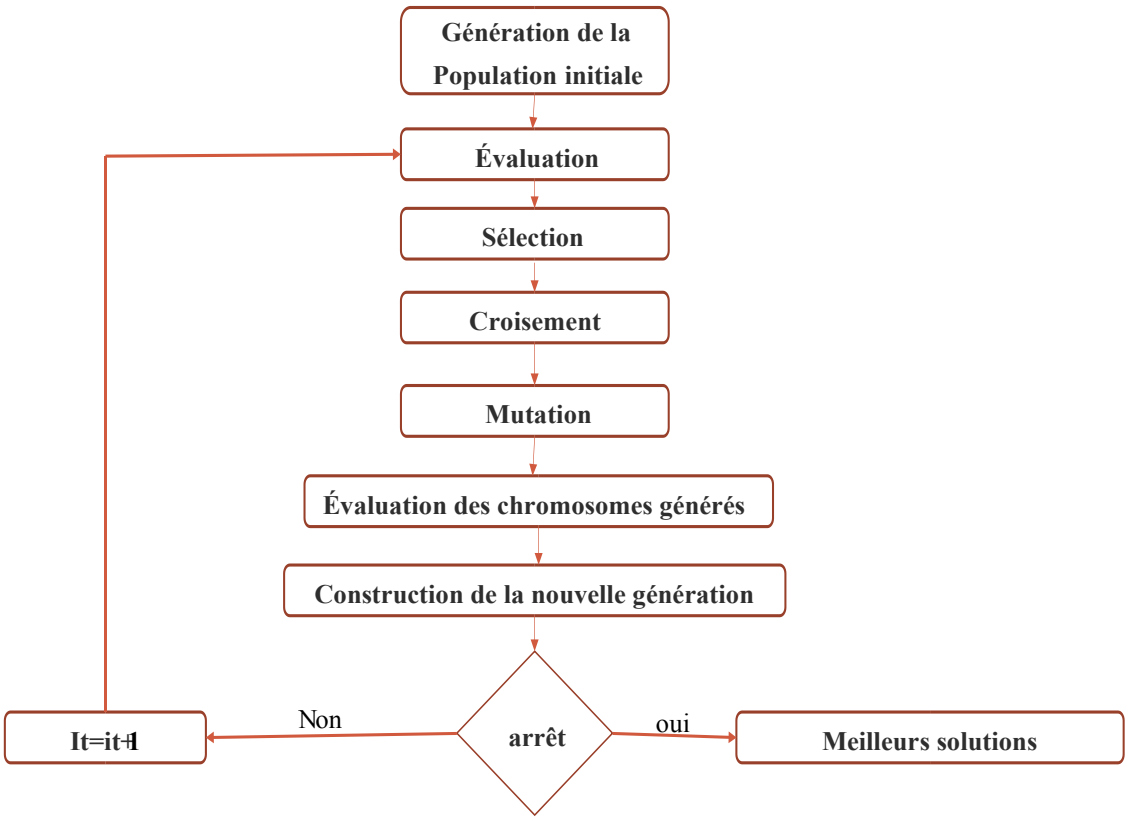


Figure 13 : Organigramme général de l'Algorithme Génétique. [38]

L'organigramme présenté dans la figure montre les déférentes étapes de la démarche de résolution utilisée dans notre travail.

Codage :

La représentation binaire est la première représentation utilisée par les AG. Chaque individu x est représenté sous forme de chaînes de bits, où chaque élément prend la valeur 0 ou 1 :

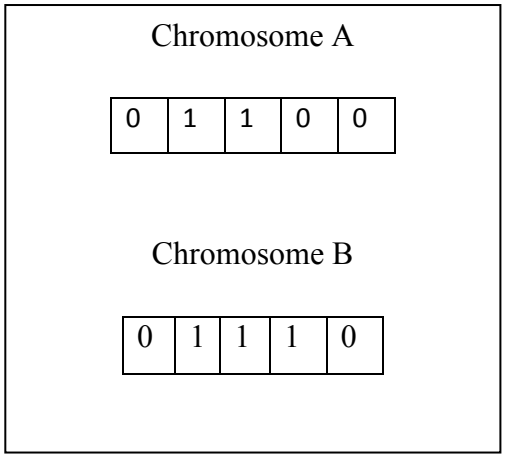


Figure 12 : Schéma du codage binaire.

Fonction fitness :

$$\begin{cases} Max \quad z1 = 10x_1 + 40x_2 + 30x_3 + 50x_4 + 70x_5 \\ Max \quad z2 = 15\,x_1 + 35x_2 + 33x_3 + 45x_4 + 60x_5 \\ Max \quad z3 = 12x_1 + 42x_2 + 35x_3 + 48x_4 + 66x_5 \\ \\ wi = 5x_1 + 4x_2 + 6x_3 + 3x_4 + 8x_5 \leq 15 \\ x \in \{0,1\} \end{cases}$$

Les fonctions objectives peuvent être agrégé en un seul objectif :

$Z(x, \lambda) = \lambda 1\, z_1(x) + \lambda 2 z_2(x) + \lambda 3 z_3(x)$ avec $0 \leq \lambda \leq 1$

Les grandes lignes de l'algorithme le fonctionnement général d'un AG est illustré à la Figure :

Générer aléatoirement une population *Pop* de ***Pop*** individus
TANT QUE Nombre de générations < Nombre de générations Maximum
FAIRE
Évaluation :Calculer la fitness de chaque individu *x*
Sélection : Établir de façon probabiliste les paires d'individus qui vont se reproduire en accordant une plus grande chance aux meilleurs individus
Croisement : Recombiner les paires sélectionnées en fonction de ***pc*** .
Mutation : Effectuer une mutation en fonction de ***pm***
Remplacement : Les nouveaux individus produits représentent la nouvelle population

Algorithme 1 : Un algorithme génétique standard. [35]

Création de la population initiale

Avant de lancer l'Algorithme Génétique, une population initiale doit être générée. Cet ensemble d'individus qui servira de base pour les générations ultérieures doit être non homogène. Afin de satisfaire cette non homogénéité, la génération de la population initiale dans notre application se fait en générant des chromosomes aléatoires au lieu d'ordonnancements réels.

```
Sub inialisation()
        Dim i, j, c As Integer
        ReDim pool(taille, n)
        For i = 1 To taille
            c = 0
            j = Int(n * Rnd()) + 1
            While c + pi(j) <= capacity
                pool(i, j) = 1
                c = c + pi(j)
                Do
                    j = Int(n * Rnd()) + 1
                Loop Until pool(i, j) = 0
            End While
        Next
    End Sub
```

Algorithme 2 : Génération de la population initiale.

Sélection

Pour notre problème on a choisir la sélection par la roue de fortune (roulette Wheel) :

Pour chaque chromosome i on calcule son degré d'adaptation f_i et on pose :

$$p_i = \frac{f_i}{\sum_i f_i} \times 100$$

On crée une roulette biaisée où chaque i occupe une portion p_i.

Pour déterminer les descendants d'une génération de taille n, il faut n tirages

Exemple simple :

Maximiser $z(x)=\lambda 1\ z_1(x)+\lambda 2 z_2(x)+\lambda 3 z_3(x)$ avec $\lambda 1=0.4$; $\lambda 2=0.3$; $\lambda 3=0.3$

Solution	z_1	z_2	z_3	$0.4\ z_1+0.3z_2+0.3z_3$	% total
A =01100	70	68	77	71.5	15.63
B =01110	120	113	125	119.4	26.11
C= 11100	80	83	89	83.6	18.3
D =01001	110	95	108	104.9	22.93
E =10001	80	75	78	77.9	17.03
Total				457.3	100

Tab3 : Exemple de la somme pondérée.

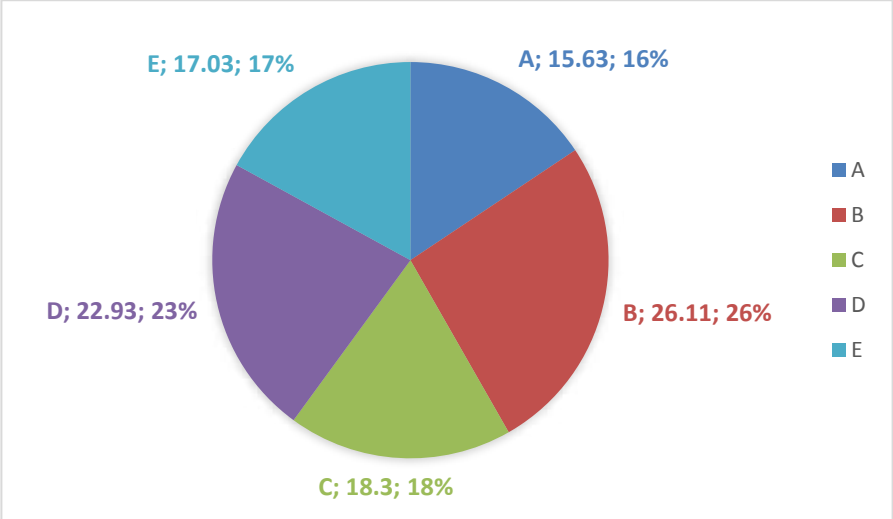


Figure 14 : la roue de fortune (roulette Wheel).

Fonction (roulette Wheel) :

```
Function roulette() As Integer
        Dim i As Integer
        Dim tirage, s1 As Double
        tirage = Rnd() * sum_fitness
        While s1 < tirage
            i = i + 1
            s1 = s1 + fitness(i)
        End While
        roulette = i
    End Function
```

Algorithme 3. Roulette.

Algorithme de sélection :

```
Sub selection()
    Dim i, j, x, y As Integer
    ReDim pool1(taille, n)
    For i = 1 To taille
        x = roulette()
        y = roulette()
        If fitness(x) > fitness(y) Then
            For j = 1 To n
                pool1(i, j) = pool(x, j)
            Next
        Else
            For j = 1 To n
                pool1(i, j) = pool(y, j)
            Next
        End If
    Next
    Erase fitness, pool
End Sub
```

Algorithme 4. Sélection.

Croisement :

Espérance d'amélioration de nouvelles générations

Croisement à 1 point :

- ✓ 2 chromosomes de taille *l*.
- ✓ On choisit aléatoirement un entier *k* entre 1 et *l*-1.
- ✓ *K* représente le point de croisement des deux chaînes

On crée 2 nouveaux individus en échangeant les caractères des chaînes initiales compris entre *k*+1 et *l*

Exemple : *l*=5 et *k*=

A = **0 1 1 | 0 0**
B = **0 1 1 | 1 0**

A' = **0 1 1 | 1 0**
B' = **0 1 1 | 0 0**

Vérifier la contrainte :

$$wi = 5x_1 + 4x_2 + 6x_3 + 3x_4 + 8x_5 \leq 15$$

Avec chromosome A' :

$$wi = 4x_2 + 6x_3 + 3x_4 \leq 15$$

wi=4*1+6*1+3*1<=15

13<=15

Avec chromosome B' :

$$wi = 4x_2 + 6x_3 \leq 15$$

wi=4*1+6*1<=15

10<=15

La probabilité de croisement pc

Pour la sélection des chromosomes à croiser on a basé sur la probabilité de croisement dont sa valeur reflète l'intensité des Changements appliqués sur la population. Généralement, la probabilité de croisement est comprise entre 0,5 et 0,9. Pour notre cas on a choisi pc = 0.8

Algorithme de croisement :

```
Sub croisement()
        Dim i, j, m, x, y, z, z1 As Integer
        Dim tirage As Single
        ReDim pool(taille, n)
        i = 1
        For m = 1 To taille - 1 Step 2
            x = m
            y = m + 1
            r = Rnd()
            If tirage <= pc Then
                z = 1 + Int(Rnd() * n * 0.5)
                z1 = z + Int(Rnd() * n * 0.5)
                For j = 1 To z : pool(i, j) = pool1(x, j) : Next
                For j = z + 1 To z1 : pool(i, j) = pool1(y, j) : Next
                For j = z1 + 1 To n : pool(i, j) = pool1(x, j) : Next
                If Not feasible(i) Then
                    For j = 1 To n : pool(i, j) = pool1(x, j) : Next
                End If
                i = i + 1
                For j = 1 To z : pool(i, j) = pool1(y, j) : Next
                For j = z + 1 To z1 : pool(i, j) = pool1(x, j) : Next
                For j = z1 + 1 To n : pool(i, j) = pool1(y, j) : Next
                If Not feasible(i) Then
                    For j = 1 To n : pool(i, j) = pool1(y, j) : Next
                End If
            Else
            For j = 1 To n : pool(i, j) = pool1(x, j) : Next
            i = i + 1
            For j = 1 To n : pool(i, j) = pool1(y, j) : Next
            End If
        Next
        Erase pool1
    End Sub
```

Algorithme 5. Croisement .

Mutation

Le but de cet opérateur est la diversification de l'espace de solution afin d'éviter une convergence rapide vers un seul type d'individu, cette diversification se fait par l'introduction des perturbations génétiques au niveau des individus.

La mutation uni-point :

Cet opérateur de mutation consiste à inverser de manière aléatoire un gène de l'individu à muter (si le gène sélectionné est à 0, on le met à 1 et inversement)

B'= **0** 1 1 0 0

B''= **1** 1 1 0 0

Vérifier la contrainte :

$$w = 5 * 1 + 4 * 1 + 6 * 1 + 3 * 0 + 8 * 0 \ <= 15$$

$$5 + 6 + 3 <= 15$$

$$\mathbf{15 <= 15}$$

La probabilité de mutation pm

La probabilité de mutation c'est une valeur permet de définir le nombre de chromosomes sur lesquels on va faire la mutation. Un taux élevé de Pm risque de converger vers une solution sous-optimale et à la perte de la population originale, c'est pour cette raison que la probabilité de mutation est généralement faible. Pour notre cas on a choisi pm = 0.1.

Algorithme de mutation

```
Sub mutation()
        Dim tirage As Single
        Dim randombit As Single
        Dim i As Integer
        For i = 1 To taille
            tirage = Rnd()
            If tirage <= pm Then
                randombit = Int(n * Rnd()) + 1
                pool(i, randombit) = 1 - pool(i, randombit)
                If Not feasible(i) Then pool(i, randombit) = 1
pool(i, randombit)
            End If
        Next
    End Sub
```

Algorithme 6. Mutation.

Conclusion

Dans ce chapitre, nous avons défini le problème du Sac à dos multi-objectif, nous avons aussi appliqué la méthode d'agrégation et procédé à la conception de notre algorithme, dont on a présenté une description détaillée de différentes étapes de l'algorithme génétique (sélection, croisement, mutation) pour résoudre le problème de sac à dos.

CHAPITER V
RÉSULTATS ET SYNTHÈSE

Table des Matières

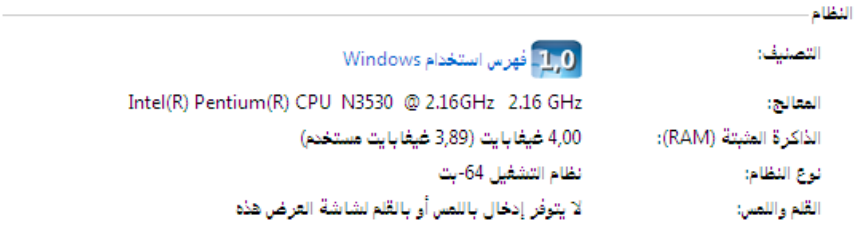
Introduction

Comme pour beaucoup d'algorithmes, les paramètres de l'algorithme génétique doivent être soigneusement choisis pour obtenir de bons résultats, donc dans ce chapitre, nous allons procéder des tests pour savoir quels sont les bons paramètres de l'algorithme génétique à opter pour fournir des bonnes solutions ou des solutions proches de l'optimum du problème du problème de sac à dos multi objectif.

Environnement de développement

Pour la réalisation de ce travail, nous avons eu recours aux environnements suivants :

Environnement matériel



Environnement logiciel

- ✓ Windows 7.1 comme système d'exploitation.
- ✓ Langage Visual basic comme langage de programmation.

Langage de programmation Visual basic

Visual Basic (VB) est un langage de programmation événementiel de troisième génération ainsi qu'un environnement de développement intégré, créé par Microsoft pour son modèle de programmation COM. Visual Basic est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données en utilisant les technologies DAO, ADO et RDO, ainsi que la création de contrôles ou objets ActiveX. Les langages de script tels que Visual Basic for Applications et VB Script sont syntaxiquement proches de Visual Basic, mais s'utilisent et se comportent de façon sensiblement différente. [39]

L'environnement :

Microsoft Visual Studio Express est une version "allégée" disponible gratuitement à des buts éducatifs. Elle reprend, en limitant les fonctionnalités les plus avancées, l'interface de Visual Studio, et en limitant l'usage à un seul langage de programmation par installation. Une version "CTP" de Microsoft Visual Studio 2010 Express est sortie le 14/03/2010 pour

développer les applications pour Windows Mobile 7 incluant une première version 4.0 du Framework .NET. [39]

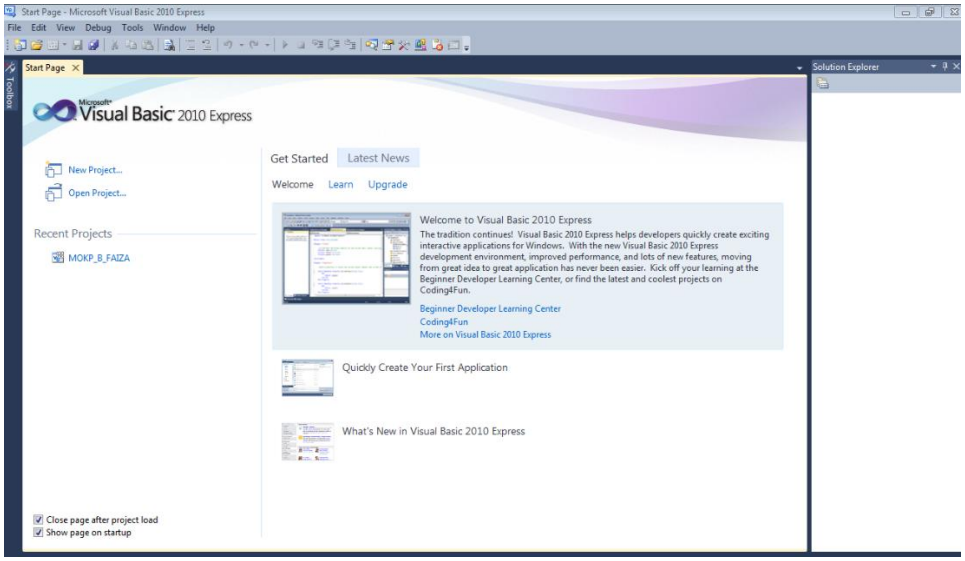


Figure 20. Page de démarrage de Visual basic express 2010.

Implémentation

Pour bien mener l'étude analytique de l'algorithme proposé dans le chapitre précédent,

La **figure 21** illustre l'interface de cette implémentation où on doit générer les données, choisir les différents paramètres de l'algorithme puis afficher les résultats associés.

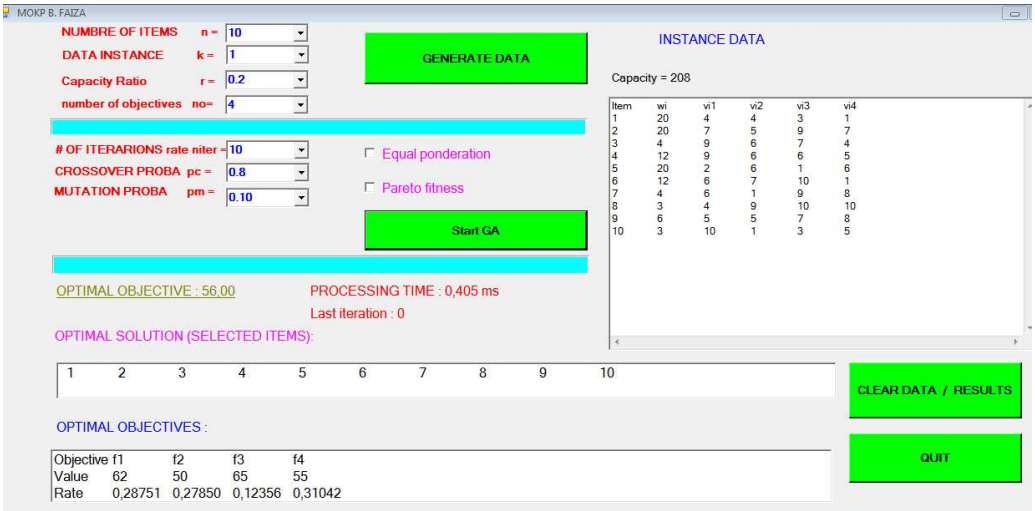


Figure 21. Interface graphique de notre application.

Eléments d'interface et code

Fonction Fitness

Dans notre programme, nous avons créé deux fonctions objectifs :

a) **Fonction d'agrégation :**

$$\boldsymbol{Max} \sum_{i=1}^{no} \lambda_i * f_i(\mathrm{x}) \;\; ; \quad no \geq 2$$

Tel que

$$\sum_{i=1}^{no} w_i * x_i \leq W$$

Le vecteur des scalaires λ

Si nous choisissons l'option 'E*qual ponderation*', le programme calcule λ pour que les valeurs λ soient égales :

λ = 1 / no (Nombre d'objectifs)

Sinon, on calcule des coefficients λi aléatoires tels que $0<\lambda i\leq 1$.

b) Fonction Pareto :

Cette fonction utilise directement la notion de dominance dans la sélection des solutions générées, Nous calculons le nombre de solutions dominantes.

$$\boldsymbol{Max} \sum_{i=1}^{n} v_i * x_i$$

Sujet à :

$$\sum_{i=1}^{n} w_i * x_i \leq W .$$

- ✓ Les poids w_i des objets sont aléatoirement générés entre [1, 20].
- ✓ Les valeurs sont aléatoirement générées entre [1, 10].
- ✓ La moyenne de la Capacité de sac à dos a été utilisée : W = ($\sum w_i$ * r).

Pour savoir quels sont les bons paramètres de l'algorithme génétique utilisé permettant d'obtenir des bonnes solutions, il faut faire des tests. On essayera donc d'améliorer les solutions obtenues par l'ajustement des différents paramètres de l'algorithme génétique.

Pour ce test, on fixe les probabilités de croisement (p_c) et de mutation (p_m) et la taille de population aussi le nombre d'itérations (Critère d'arrêt). Les valeurs de p_c, p_m et le nombre d'itérations sont représenté dans le tableau suivant :

Pc	Pm	Nombre d'itérations	Taille de population
0.8	0.10	10*n	10*n

Tab 4. Paramètres de l'AG.

Phase de test

Premier cas :

Optimal objectif :

- ✓ Nombre des objets : n = 10*n.
- ✓ Capacité (ratio) : r = 0.2.
- ✓ Nombre d'objectifs : no = 3.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	29	29	27	31
2	24	22	25	25
3	16	13	16	15
4	28	28	29	29
5	21	18	17	14
6	17	17	14	23
7	24	22	23	19
8	29	28	30	28
9	24	20	27	26
10	27	27	26	28
Average	**23.9**	22.4	23.4	23.8

Tab 5 : fonction objectif optimal**.1.4.3.2**

✓ Nombre des objectifs : no = 5.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	20	20	22	20
2	22	22	23	20
3	19	17	19	19
4	17	17	16	18
5	25	25	21	22
6	21	20	17	24
7	28	26	23	24
8	24	24	22	26
9	21	21	19	22
10	20	20	16	17
Average	**21.7**	21.2	19.8	21.2

Tab 6 : fonction objectif optimal**.1.2.4.3**

✓ Nombre des objectifs : no = 10.

Pour les paramètres ci-dessus on obtient les résultats suivants :

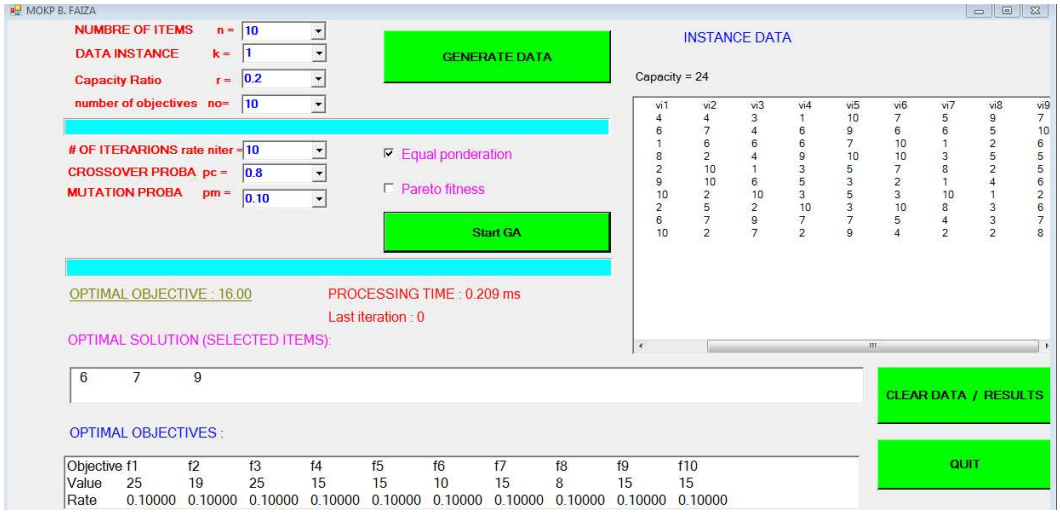


Figure 22: Equal ponderation avec capacity ratio r=0.2, n=10, no=10**.**

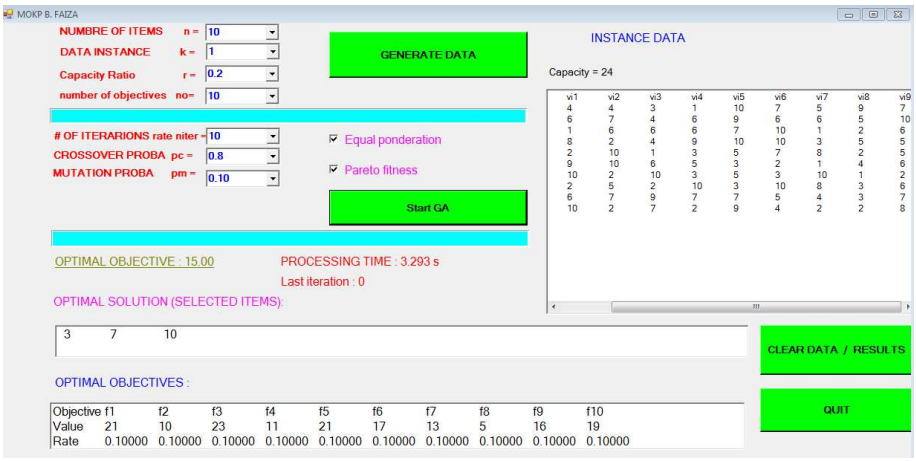


Figure 23: Equal ponderation**,** Pareto fitness avec capacity ratio r=0.2, n=10, no=10**.**

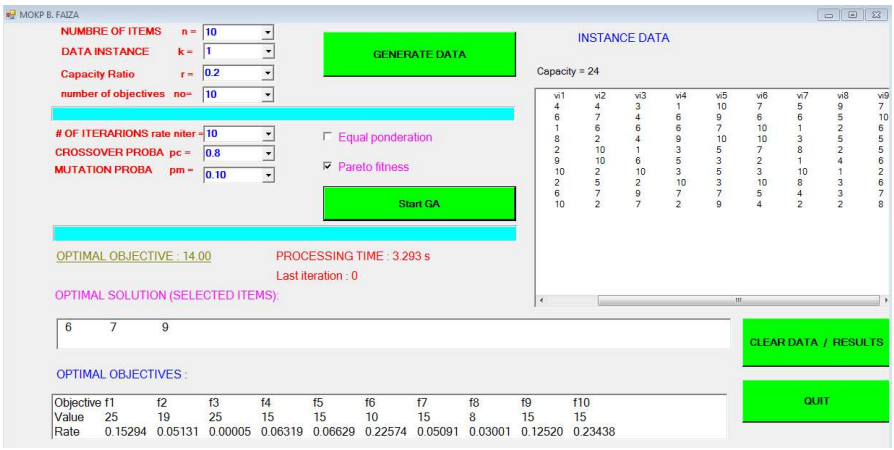


Figure 24: Different Equal ponderation**, Pareto** fitness avec capacity ratio r=0.2, n=10, no=10**.**

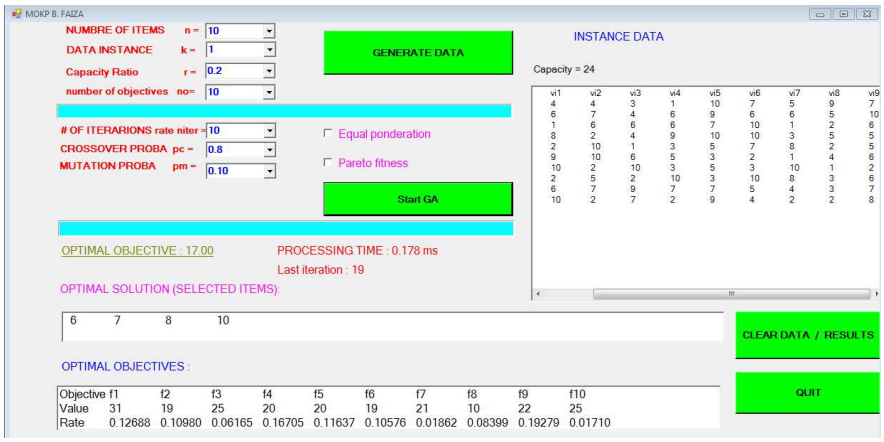


Figure 25: Different Equal ponderation**,** different Pareto fitness avec capacity ratio r=0.2, n=10, no=10**.**

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Prato fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	16	15	14	17
2	18	16	17	17
3	22	22	23	18
4	17	19	14	15
5	25	25	17	23
6	17	17	18	18
7	24	24	17	18
8	18	21	16	21
9	20	14	16	17
10	23	20	20	21
Average	**20**	19.3	17.2	18.5

Tab 7 : fonction objectif optimal**.1.2.4.3**

Temps execution:

- ✓ Nombre des objets : N = 10.
- ✓ Capacité (ratio) : r = 0.2.
- ✓ Nombre des objectifs : no = 3.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Agérate fitness	Equal ponderation Pareto fitness	Different ponderation Agérate fitness	Different ponderation Pareto fitness
1	0.116	1.185	1.151	0.133
2	0.13	1.165	1.149	0.133
3	0.174	1.187	1.193	0.141
4	0.177	1.159	1.247	0.145
5	0.15	1.181	1.174	0.139
6	0.162	1.227	1.185	0.166
7	0.151	1.191	1.169	0.149
8	0.147	1.154	1.177	0.13
9	0.128	1.164	1.18	0.135
10	0.128	1.163	1.155	0.137
Average	0.146	1.178	1.178	**0.141**

Tab 8: progressing time s.**4.1.2.3**

- ✓ Nombre des objectifs : no = 5.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Prato fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	0.153	1.809	1.802	0.159
2	0.157	1.875	1.831	0.16
3	0.164	1.827	1.829	0.167
4	0.162	1.899	1.839	0.181
5	0.161	1.812	1.821	0.152
6	0.17	1.863	1.868	0.158
7	0.183	1.786	1.897	0.162
8	0.168	1.827	1.818	0.164
9	0.169	1.892	1.862	0.154
10	0.165	1.856	1.878	0.161
Average	0.165	1.845	1.844	**0.162**

Tab 9: progressing time s.**4.1.3.2**

- ✓ Nombre des objets : N = 10.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness	Equal ponderation Pareto fitness	Different ponderation Aggregate fitness	Different ponderation Pareto fitness
1	0.21	3.382	3.347	0.207
2	0.233	3.373	3.35	0.213
3	0.207	3.496	3.321	0.208
4	0.207	3.477	3.444	0.211
5	0.217	3.522	3.491	0.213
6	0.185	3.512	3.534	0.209
7	0.194	3.392	3.3	0.199
8	0.193	3.531	3.425	0.198
9	0.194	3.572	3.365	0.194
10	0.197	3.446	3.499	0.204
Average	**0.204**	3.470	3.408	0.206

Tab 10: progressing time s.**1.4.3.2**

Premier test :

But de test : Premièrement on cherche : quelle est le meilleur taux parmi les moyennes avec les différentes valeurs de **no.** Deuxièmement, comparez le temps d'exécution pour chaque fonction.

Les **figures 26** et **27** montrent la variation du résultat obtenu lorsque l'on modifie le nombre des fonction objectif **no**

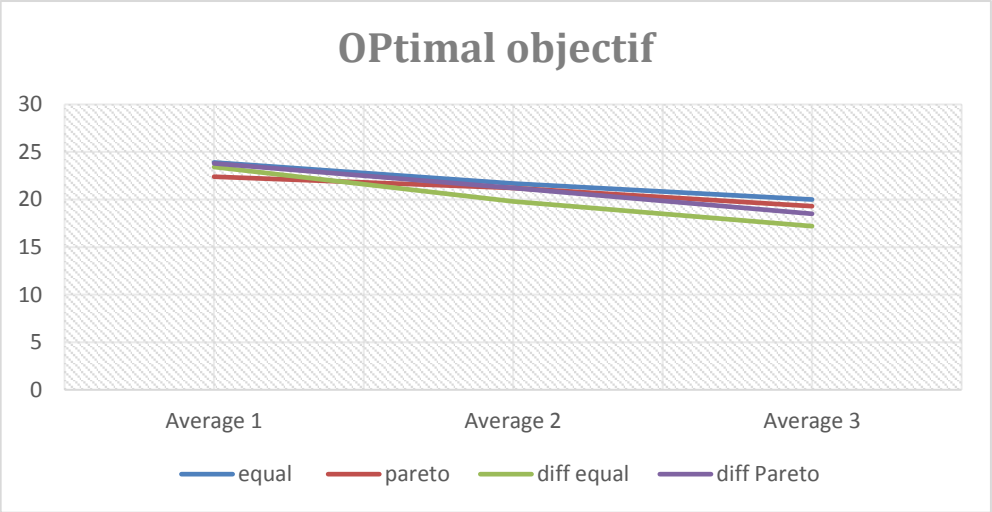


Figure 26: Comparisons: values optimal objectif

Equal ponderation, Pareto fitness, diff equal ponderation, diff Pareto fitness, n=10.r=0.2. no=3,5,10.

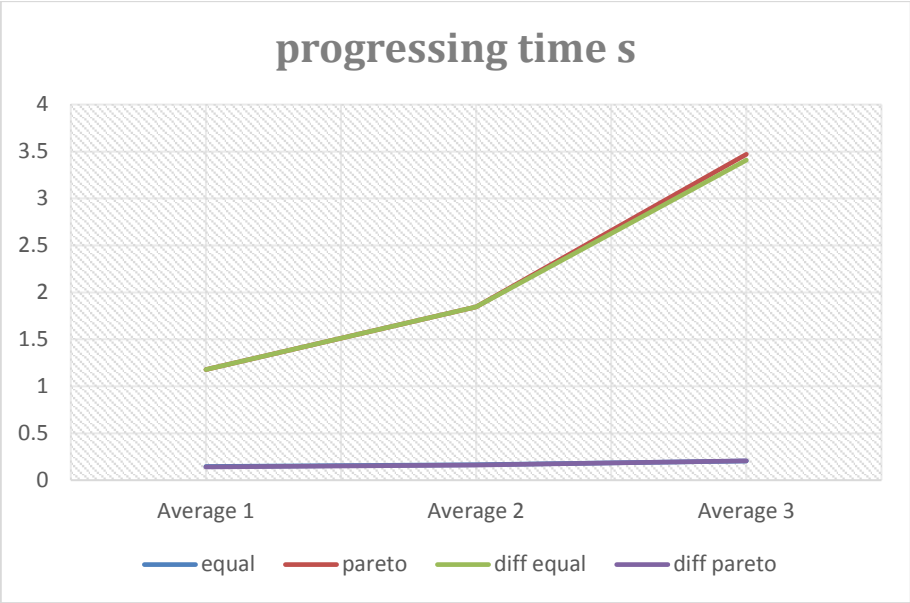


Figure 27: Comparaison le temps d'exécution de fonction optimal equal, Pareto, diff equal, diff Pareto, n=10.r=0.2.no=3,5 et 10.

Constat :

Dans ce cas, on note que le meilleur taux avec la croissance de **no** aussi en termes de temps d'exécution est :

Le cas où le nombre des fonction objectif (no) égal à 3 :

a) Fonction d'agrégation avec λ = 1 / no.
b) Fonction d'agrégation avec λ aléatoire.
c) Fonction Pareto fitness.
d) Fonction Pareto fitness avec λ = 1 / no.

Le cas où le nombre des fonction objectif (no) égal à 5 et 10 :

a) Fonction d'agrégation avec λ = 1 / no.
b) Fonction Pareto fitness avec λ = 1 / no.
c) Fonction d'agrégation avec λ aléatoire.
d) Fonction Pareto fitness.

Deuxième cas :

Optimal objectif :

- ✓ Nombre des objets : N = 10.
- ✓ Capacité (ratio) : r = 0.8.
- ✓ Nombre des objectifs : no = 3.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation aggregate fitness 3	Different ponderation Pareto fitness 4
1	49	49	49	52
2	48	48	51	47
3	46	45	47	50
4	54	54	54	54
5	43	43	44	42
6	48	48	45	46
7	51	52	45	46
8	50	50	49	52
9	49	49	51	50
10	47	45	45	47
Average	48.5	48.3	48	**48.6**

Tab 11 : fonction optimale avec 3 objectifs.**4.1.2.3**

- ✓ Nombre des objectifs : no = 5.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness	Equal ponderation Pareto fitness	Different ponderation Aggregate fitness	Different ponderation Pareto fitness
1	47	47	49	44
2	48	50	45	42
3	45	45	46	45
4	43	43	48	48
5	47	47	45	44
6	57	57	55	58
7	52	53	46	48
8	51	51	54	48
9	52	52	54	52
10	41	40	37	43
Average	48.3	**48.5**	47.9	47.2

Tab 12 : fonction optimale avec5 objectifs.**2.1.3.4**

- ✓ Nombre des objectifs : no = 10.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	45	43	42	40
2	48	48	37	39
3	47	47	45	43
4	47	47	42	40
5	50	50	48	44
6	46	46	44	40
7	51	51	44	46
8	47	45	36	46
9	52	52	44	48
10	46	46	49	42
Average	**47,9**	47,5	43,1	42,8

Tab 13 : fonction objectif optimal avec 10 objectifs**.1.2.3.4**

Temps execution:

- ✓ Nombre des objets : N = 10.
- ✓ Capacité (ratio) : r = 0.8.
- ✓ Nombre des objectifs : no = 3.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	0.147	1.177	1.169	0.14
2	0.134	1.149	1.162	0.138
3	0.149	1.162	1.162	0.144
4	0.138	1.156	1.166	0.17
5	0.138	1.166	1.156	0.13
6	0.132	1.185	1.177	0.142
7	1.131	1.175	1.171	0.14
8	0.138	1.16	1.178	0.144
9	0.139	1.161	1.157	0.134
10	0.144	1.158	1.17	0.131
Average	0.239	1.165	1.167	**0.141**

Tab 14: progressing time savec 3 objectifs.**4.1.2.3**

- ✓ Nombre des objectifs : no = 5.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness	Equal ponderation Pareto fitness	Different ponderation Aggregate fitness	Different ponderation Pareto fitness
1	0.157	1.838	1.841	0.179
2	0.169	1.837	1.978	0.17
3	0.175	1.827	1.86	0.171
4	0.172	1.84	1.853	0.179
5	0.181	1.886	1.976	0.181
6	0.174	1.839	1.808	0.175
7	0.173	1.82	1.86	0.182
8	0.173	1.853	1.834	0.192
9	0.176	2.065	2.084	0.172
10	0.173	1.799	1.846	0.174
Average	**0.172**	1.860	1.89	0.177

Tab 15: progressing time savec 5 objectifs.**1.4.3.2**

- ✓ Nombre des objectifs : no = 10.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness	Equal ponderation Pareto fitness	Different ponderation Aggregate fitness	Different ponderation Pareto fitness
1	0.209	3.446	3.367	0.204
2	0.207	3.446	3.475	0.206
3	0.204	3.456	3.461	0.205
4	0.217	3.386	3.452	0.214
5	0.234	3.494	3.462	0.21
6	0.216	3.448	3.501	0.21
7	0.209	3.467	3.454	0.212
8	0.212	3.477	3.479	0.213
9	0.224	3.448	3.367	0.212
10	0.226	3.384	3.453	0.213
Average	0.215	3.445	3.447	0.209

Tab 16: progressing time s avec 10 objectifs.

Deuxième test :

Le but de test : nous voulons connaitre le résultat de la mise à jour de la valeur de capacité (ratio) à r = 0.8.

Les **figures 28** et **29** montrent la variation du résultat obtenu lorsque l'on modifie la valeur **r :**

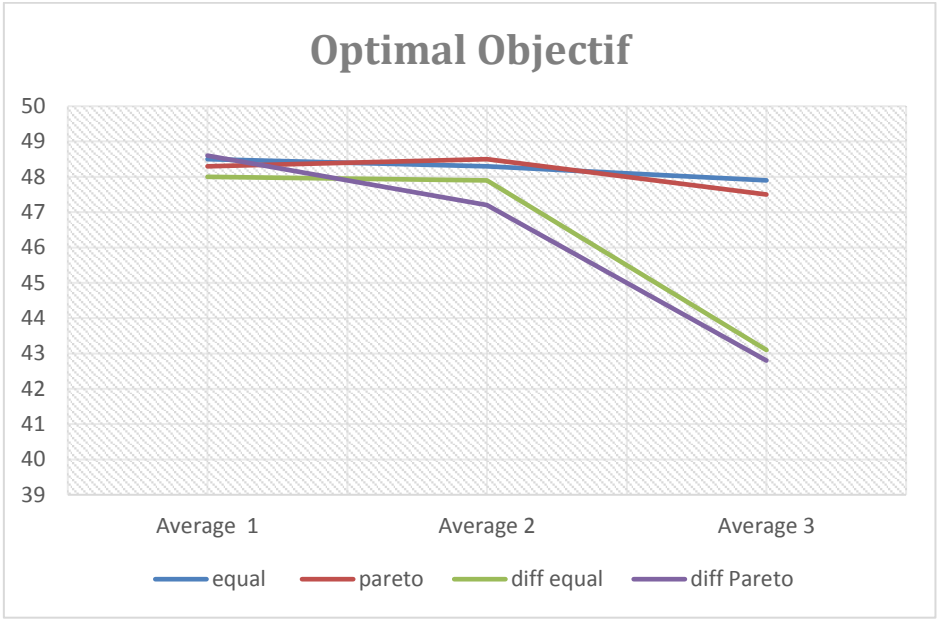


Figure 28: Comparison function objectif optimal equal ponderation, Pareto fitness, diff equal ponderation, diff Pareto fitness, n=10.r=0.8. no=3, 5,10.

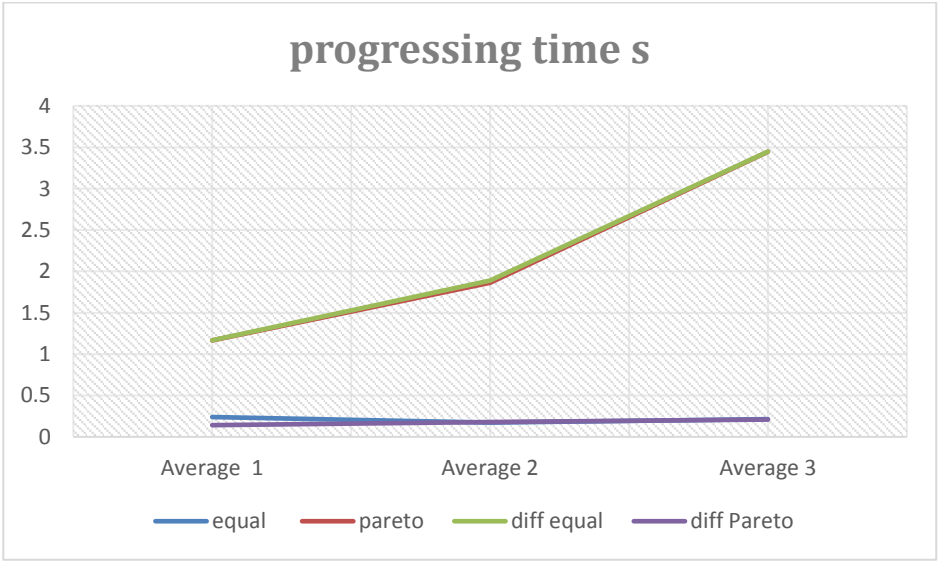


Figure 29: Comparaison le temps d'exécution de fonction optimal equal, Pareto, diff equal, diff Pareto, n=10, r=0.8, no=3,5,10.

Constat :

Dans ce cas, on note que le meilleur taux avec la croissance de **no** est :

Le cas où le nombre des fonction objectif (no) égal à 3, et r =0,8 :

a) Fonction d'agrégation avec λ aléatoire.
b) Fonction d'agrégation avec λ = 1 / no.
c) Fonction Pareto fitness avec λ = 1 / no.
d) Fonction Pareto fitness.

Le cas où le nombre des fonction objectif (no) égal à 5 et r =0,8 :

a) Fonction Pareto fitness avec λ = 1 / no.
b) Fonction d'agrégation avec λ = 1 / no.
c) Fonction Pareto fitness.
d) Fonction d'agrégation avec λ aléatoire.

Le cas où le nombre des fonction objectif (no) égal à 10 et r =0,8 :

e) Fonction d'agrégation avec λ = 1 / no.
f) Fonction Pareto fitness avec λ = 1 / no.
g) Fonction Pareto fitness.
h) Fonction d'agrégation avec λ aléatoire.

Dans tous les cas, la fonction Pareto fitness prend le plus de temps.

Troisième cas :

Optimal objectif :

- ✓ Nombre des objets : n= 20.
- ✓ Capacité (ratio) : r = 0.2.
- ✓ Nombre des objectifs : no = 3.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	44	42	46	47
2	48	47	48	44
3	42	37	48	45
4	41	33	39	39
5	35	40	41	42
6	54	41	47	50
7	41	35	45	43
8	56	44	42	56
9	30	33	28	33
10	32	35	36	38
Average	42.3	38.7	42	**43.7**

Tab 17 : fonction objectif optimal avec 20 objets et 3 objectifs et r=0.2**.4.1.3.2**

- ✓ Nombre des objectifs : no = 5.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal Ponderation Aggregate fitness 1	Equal Ponderation Pareto fitness 2	Different Ponderation Aggregate fitness 3	Different Ponderation Pareto fitness 4
1	41	44	41	37
2	46	39	43	34
3	42	46	41	42
4	43	42	40	42
5	46	34	52	48
6	46	34	34	44
7	43	37	37	44
8	48	41	48	42
9	39	43	47	43
10	41	36	30	40
Average	**43.5**	39.6	41.3	41.6

Tab 18 : fonction objectif optimal avec 20 objets et 5 objectifs et r=0.2**.1.4.3.2**

Troisième test :

But de test : voyez comment les résultats changement du nombre des objet.

La **figures 30** montrée la variation du résultat obtenu lorsque l'on modifie la valeur **n :**

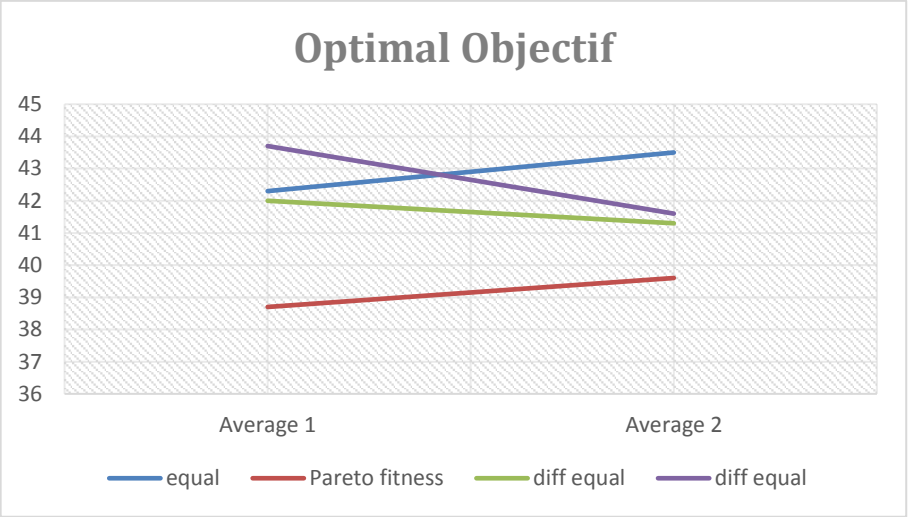


Figure 30: Comparison functions objectif optimal equal ponderation, Pareto fitness, diff equal ponderation, diff Pareto fitness, n=20.r=0.2. no=3,5.

Constat :

Dans ce cas, on note que le meilleur taux avec la croissance de **no** est :

Le cas où le nombre des fonction objectif (no) égal à 3, et r =0,2 et n =20 :

a) Fonction d'agrégation avec λ aléatoire.
b) Fonction d'agrégation avec λ = 1 / no.
c) Fonction Pareto fitness.
d) Fonction Pareto fitness avec λ = 1 / no.

Le cas où le nombre des fonction objectif (no) égal à 5 et r =0,2 et n =20 :

a) Fonction d'agrégation avec λ = 1 / no.
b) Fonction d'agrégation avec λ aléatoire.
c) Fonction Pareto fitness.

d) Fonction Pareto fitness avec λ = 1 / no.

Quatrième cas :

Optimal Objectif :

- ✓ Nombre des objets : n= 20.
- ✓ Capacité (ratio) : r = 0.8.
- ✓ Nombre des objectifs : no = 3.

Pour les paramètres ci-dessus on obtient les résultats suivants :

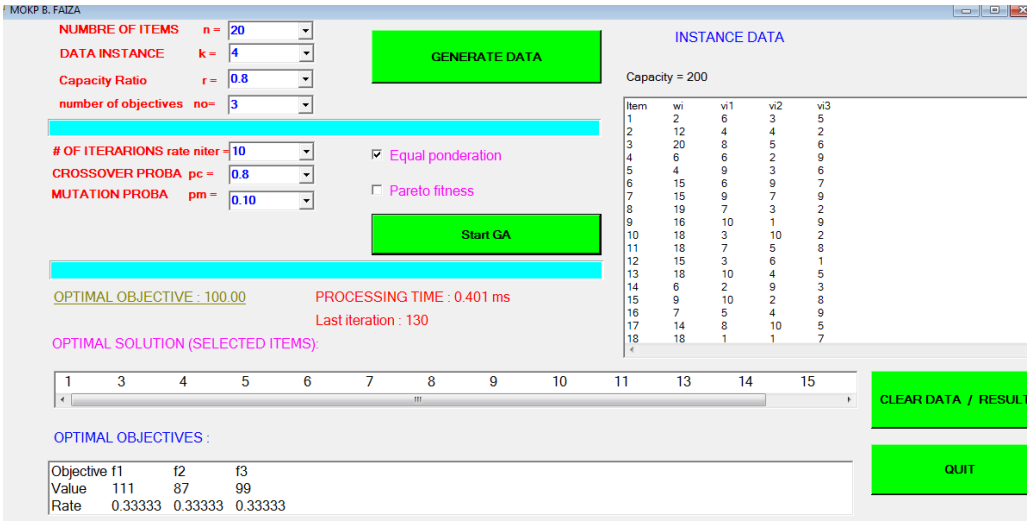


Figure 31: Equal ponderation avec capacity ratio r=0.8, n=3, no=20**.**

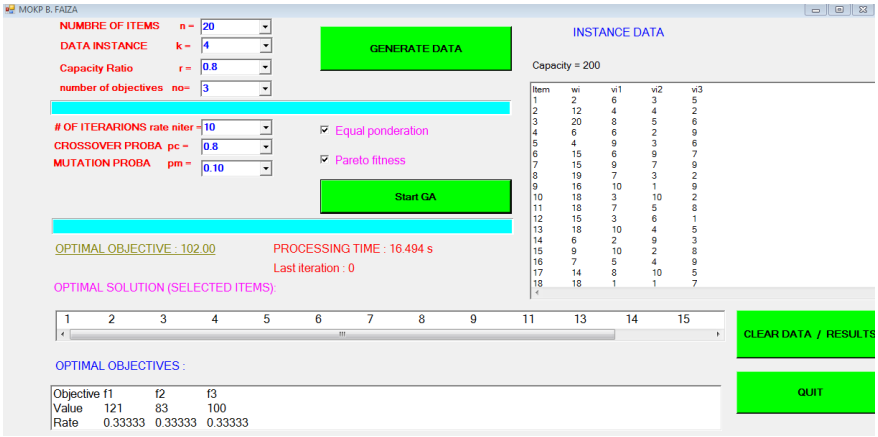


Figure 32: Equal ponderation**,** Pareto fitness avec capacity ratio r=0.8, n=3, no=20

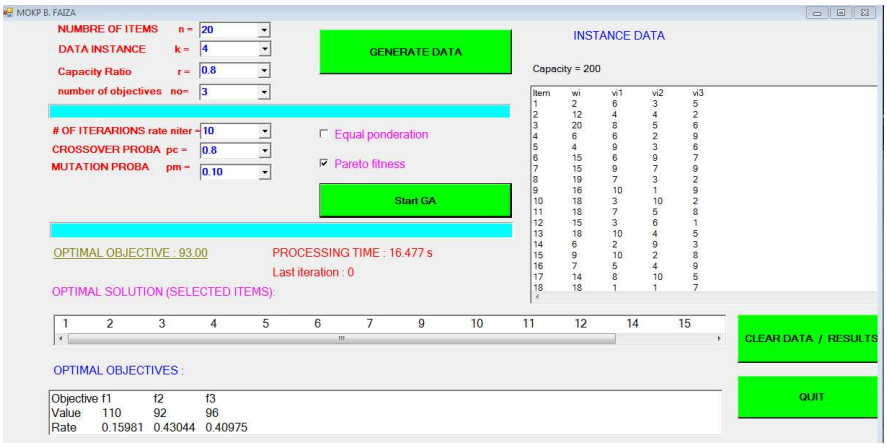


Figure 33: Different Equal ponderation**, Pareto** fitness avec capacity ratio r=0.8, n=3, no=20**.**

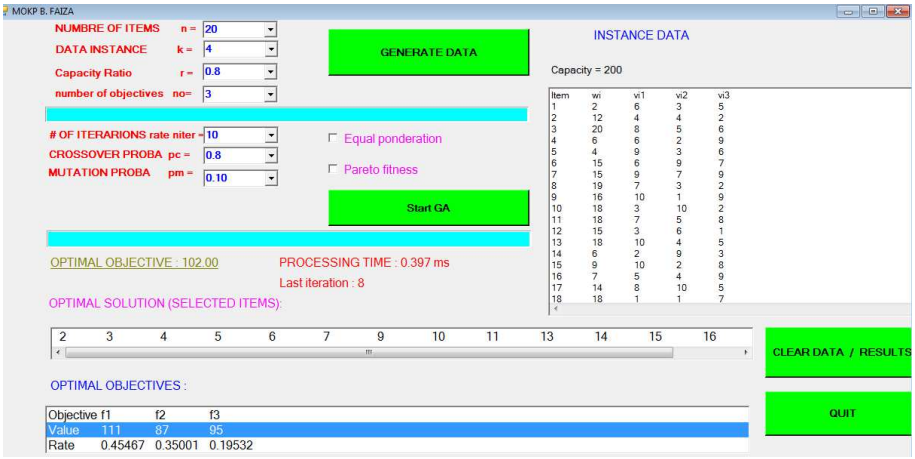


Figure 34: Different Equal ponderation**,** different Pareto fitness avec capacity ratio r=0.8, n=3, no=20**.**

Instance	Equal Ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	97	98	100	103
2	102	102	103	107
3	104	104	112	108
4	100	102	93	102
5	93	96	95	98
6	109	113	111	111
7	99	99	85	92
8	110	110	106	103
9	88	88	92	90
10	94	93	87	95
Average	99.6	100.5	98.4	**100.9**

Tab 19 : fonction objectif optimal avec 20 objets et 3 objectifs et r=0.8**.4.2.1.3**

- ✓ Nombre des objectifs : no = 5.

Pour les paramètres ci-dessus on obtient les résultats suivants :

Instance	Equal Ponderation Aggregate fitness 1	Equal ponderation Pareto fitness 2	Different ponderation Aggregate fitness 3	Different ponderation Pareto fitness 4
1	100	100	93	98
2	94	95	93	96
3	104	105	95	101
4	104	104	93	101
5	105	104	84	97
6	107	107	104	105
7	98	97	97	96
8	100	101	104	95
9	96	91	97	95
10	96	98	98	98
Average	**100.4**	100.2	95.8	98.2

Tab 20 : fonction objectif optimal avec 20 objets et 5 objectifs et r=0.8**.1.2.4.3**

Quatrième test :

Le but de test : nous voulons connaitre le résultat de la mise à jour de la valeur de capacité (ratio) à r = 0.8.

Les **figures 35** montrent la variation du résultat obtenu lorsque l'on modifie la valeur **r :**

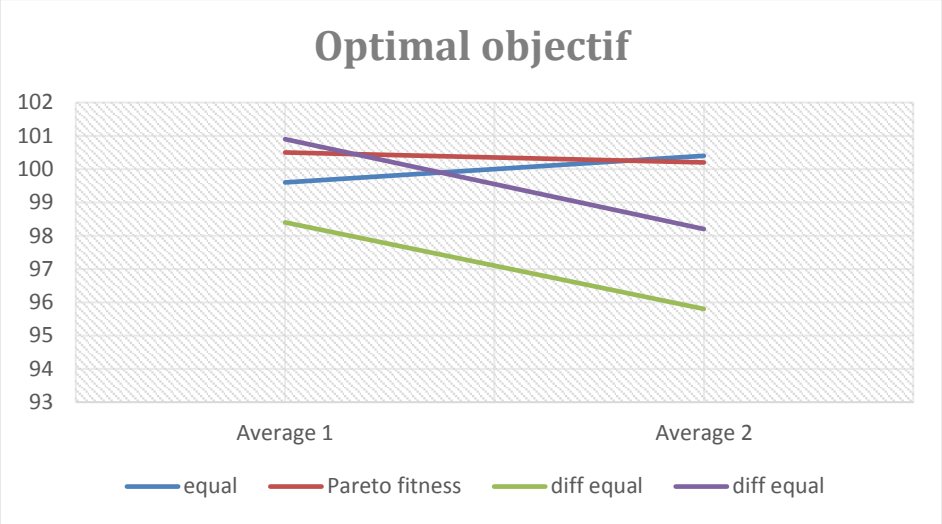


Figure 35: Comparison functions objectifs optimal equal ponderation, Pareto fitness, diff equal ponderation, diff Pareto fitness, n=20.r=0.8. no=3,5.

Constat

Dans ce cas, on note que le meilleur taux avec la croissance de **no** est comme suit:

Le cas où le nombre des fonctions objectifs (no) égal à 3, et r =0,8 et n =20 :

a) Fonction d'agrégation avec λ aléatoire.
b) Fonction Pareto fitness avec λ = 1 / no.
c) Fonction d'agrégation avec λ = 1 / no.
d) Fonction Pareto fitness.

Le cas où le nombre des fonction objectif (no) égal à 5 et r =0,8 et n =20 :

a) Fonction d'agrégation avec λ = 1 / no.
b) Fonction Pareto fitness avec λ = 1 / no.
c) Fonction d'agrégation avec λ aléatoire.
d) Fonction Pareto fitness.

Analyse des résultats

Nous remarquons dans le test de la fonction d'agrégation et fonction Pareto que la fonction d'agrégation a obtenu de très bons résultats et a pris moins de temps par rapport à son homologue la fonction de Pareto. Par ce que la fonction d'agrégation transforme le problème multi objectif pour problème mono objectif puis trouver des solutions optimales, Quant à la fonction de Pareto, elle dépend du calcul du nombre de solutions dominantes, ce qui est une solution inefficace et chronophage surtout lorsque le nombre de fonction objectifs augmente.

On remarque la différence quand on change les valeurs :

Les valeurs λ : soient égales tel que $0<\lambda\leq1$ est très efficace avec un plus grand nombre de fonction objectif, λ aléatoire tel que $0<\lambda\leq1$ est moins efficace lorsque le nombre de fonction objectif augmente.

Capacité (ratio) : quand r =0,2 les valeurs de fonction optimale sont petites, quand r =0,8 les valeurs de fonction optimale sont grandes.

Conclusion

Dans ce chapitre, nous avons proposé deux méthodes pour résoudre le problème du sac à dos multi objectif en 0/1, Premier méthode est l'agrégation méthode de résolution de PMO est la plus évidente et probablement, la plus largement utilisée en pratique. Elle consiste à ramener le problème multi objectif au problème de l'optimisation d'une combinaison linéaire des objectifs initiaux. Ainsi, il s'agit d'associer à chaque fonction objectif un coefficient de pondération et à faire la somme des fonctions objectifs pondérées pour obtenir une nouvelle et unique fonction objectif, deuxième méthode est méthode Pareto utilisé directement la notion de dominance dans la sélection des solutions générées et basé sur calculer le nombre des solution dominantes, nous appliquons également l'algorithme génétique sur les deux méthodes,

Algorithme génétique GA pour bien explorer l'espace de recherche. Ce qui s'avère intéressant pour les problèmes de grandes tailles.

Algorithme génétique appliquée à chaque nouvelle solution produite pour mieux exploiter les bonnes solutions trouvées.

CONCLUSION GÉNÉRALE

Conclusion générale

Dans ce mémoire, nous avons présenté l'application de l'algorithme génétique sur le problème de sac à dos multiobjectif en variables binaires, ainsi q'une méthode d'agrégation appliquée avant la résolution de ce problème.

D'abord, nous avons abordé et étudié l'optimisation multiobjectif. Ensuite nous avons étudié la méthode d'agrégation qui permette de réduire la taille d'un problème, ainsi nous avons présenté l'algorithme génétique de manière générale et donné les éléments de base nécessaires à la compréhension de la méthode, ainsi nous avons présenté l'état de l'art de l'optimisation multiobjectif.

Ainsi, nous avons présenté le problème de sac-à-dos et ses différentes variantes, en nous focalisant sur le cas multiobjectif, et applique la méthode d'agrégation avec l'algorithme génétique au problème de sac à dos.

Enfin, nous avons proposé deux méthodes pour résoudre le problème du sac à dos multi objectif, Premier méthode est l'agrégation, deuxième méthode est méthode Pareto utilisé directement la notion de dominance dans la sélection des solutions générées.

Afin de tester l'efficacité de méthode, nous avons étudié et programmé ces méthodes, qui a été implémenté sur machine sous le logiciel Visual basic, puis une étude comparative à été menée sur un ensemble de problèmes tests, dont les résultats numériques montrent l'efficacité de notre approche, en termes de Nombre de variables et qualité de l'ensemble des solutions.

Perspectives

Nous nous sommes focalisés dans ce mémoire sur l'optimisation combinatoire multiobjectif, dont l'objectif est de fournir l'ensemble le plus complet possible des solutions Pareto à un problème donné. En guise de perspectives, on peut citer :

- ✓ Adapter l'algorithme génétique au cas multiobjectif.
- ✓ Développer d'autres mécanismes de réduction de problème.
- ✓ Améliorer la vitesse de l'algorithme, essayer d'hybrider l'algorithme génétique avec méthode d'agrégation.

REFERENCES BIBLIOGRAPHIQUES

[1] A.Wahabou et B. Christelle et C. Damien et S. François, Algorithme génétique multi-objectifs adaptatif, https://hal.archives-ouvertes.fr/hal-01223029 , 1 Nov 2015.

[2] A. Inès, Optimisation multi-objectif par colonies de fourmis : cas des problèmes de sac à dos, 27 Jun 2011.

[3] B. Stephane et T. Tuan-Vu, Pareto-based branch and bound algorithm for multiobjectifs optimization of a safety transformer, HAL Id: hal-01913787 https://hal.archives-ouvertes.fr/hal-01913787 , 21 Nov 2018

[4] B. Alain, Algorithme évolutionnaire pour l'optimisation multiobjectif , ,4novembre 2008

[5] J. Dipama , optimisation multi-objectif des systèmes énergétiques, avril 2010.

[6] H. Allaoua , Cour de Algorithmes génétiques , Université Mohamed Boudiaf M'sila ,2021

[7] A.Konak,W.coit, E.smith ,Multi-Objective Optimization Using Genetic Algorithms: A Tutorial,

[8] K. Ashis, M. Yogomaya, K. Anil, Multi-Objective Genetic Algorithm: A Comprhensive Survey, https://www.researchgate.net/publication/331590182 , February 2012

[9] L. SAADI, Optimisation MultiObjectifs par Programmation Génétique, 08/07 /2007.

[10] M. Riadh, Contribution à la synthèse et l'optimisation multi-objectif par essaims particulaires de lois decommande robuste RST de systèmes dynamiques,4 Apr 2017.

[11] T. El-ghazali, Metaheuristiques pour l'optimisation combinatoire multi-objectif : Etat de l'art.

[12] Z. Arnaud, Système interactif d'aide à la décision basé sur des algorithmes génétiques pour l'optimisation multi-objectifs, , 29 juin 2004

[13] D. Clarisse, Optimisation Combinatoire Multi-Objectif : Apport des méthode coopératives et contribution à l'extraction de connaissances ,https: //www.researchgate.net/publication/30519113 , January 2005

[14] S. MAHDI, Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Méta/Exacte, Université Mentouri de Constantine,

[15] Z. Arnaud, ALGORITHMES EVOLUTSONNA1RES POUR L'ORDONNANCEMENT INDUSTRIEL : APPLICATION À L'INDUSTRIE AUTOMOBILE, UNIVERSITE DU QUÉBEC, DECEMBRE 2008.

[16] F. Hamdaoui, Contribution à La Résolution Du Problème Du Sac A Dos Multidimensionnel A Choix Multiples , **U**niversité des **S**ciences et de la **T**echnologie d'**O**ran **M**ohamed **B**oudiaf, Le 01 juin 2016.

[17] S. ZOUAOUI, Approche multicritère pour le problème de sac à dos, UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE , 23/10/2011

[18] A. HANNACHE et A. LEMMOUIS, Résolution d'un problème d'ordonnancement de type job shop avec contrainte de transport, UNIVERSITE ABOU BERK BELKAID – TLEMCEN.

[19] https://www.techno-science.net/glossaire-definition/Visual-Studio.html.

[20] K. Mahdi**,** L'optimisation multi objectif et l'informatique quantique**,** Université Mentouri**.**

[21] M.Prévot, Programmation Linéaire Multiobjectif, HAL Id: hal-01544116 https://hal.archives-ouvertes.fr/hal-01544116 **,** 21 Jun 2017

[22] S. Amazou et M.djddai**,** Optimisation bi-objectif pour un ordonnancement intégré destâches de production et de maintenance**,** Université M'hamed BOUGARA – Boumerdès

[23] H**.** Hanaâ, hybridations d'algorithme métha heuristique en optimisation globale et leues applications**,** Université Mohammed V - Agdal, Rabat École Mohammedia d'Ingénieurs**,** le 29 Juin 2013

[24] Y .Murat et V. Thomas, Présentation des algorithmes génétiques et de leurs applications en économie, https://www.researchgate.net/publication/5085159 , · Fivrier 2004

[25] **S.**ZOUAOUI**,** Approche multicritère pour le problème de sac à dos, UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE, 23/10/2011.

Résumé : Dans ce mémoire, j'intéresse à l'étude d'une méthode approché pour la résolution d'un problème d'optimisation combinatoire multiobjectif, cas du problème de sac à dos multiobjectif en variables binaires. Cette méthode est appelée l'algorithme génétique, je étudiée d'abord une méthode d'agrégation somme pondéré qui permet de réduire la taille d'une problème, appliquée au problème de sac à dos multiobjectif, puis j'ai appliquée l'algorithme génétique avec l'opérateur de sélection, croisement et mutation. Enfin, j'attache à décrire l'application de l'algorithme génétique dans notre problème, ainsi la méthode d'agrégation appliquée.
Mots clés : Optimisation combinatoire multiobjectif, problème de sac à dos, l'algorithme génétique, méthode d'agrégation.

Abstract: In this dissertation, I am interested in the study of an approximate method for solving a multiobjective combinatorial optimization problem, case of the multiobjective knapsack problem in binary variables. This method is called the genetic algorithm, I first studied a weighted sum aggregation method that can reduce the size of a problem, applied to the multiobjective knapsack problem, and then I applied the genetic algorithm with the operator of selection, crossing and mutation. Finally, I try to describe the application of the genetic algorithm in our problem, as well as the applied aggregation method.
Keywords : Combinatorial multiobjective optimization, knapsack problem, genetic algorithm, aggregation method.

ملخص: في هذه الأطروحة ، أنا مهتم بدراسة طريقة تقريبية لحل مشكلة تحسين اندماجية متعددة الأهداف ، حالة مشكلة الظهر ثنائية المتغير متعددة الأهداف . الطريقة تسمى الخوارزمية ال الوراثية، لقد قمت أولاً بدراسة طريقة تجميع المجموع الموزون التي يمكن أن تقلل من حجم المشكلة ، مطبقة على مشكلة الظهر متعددة الأهداف ، ثم قمت بتطبيق الخوارزمية الوراثية مع عامل الاختيار والعبور والطفرة. أحاول وصف تطبيق الخوارزمية الوراثية في مشكلتنا ، وكذلك طريقة التجميع المطبقة.

الكلمات المفتاحية : تحسين إندماجي متعدد الأهداف،مشكلة حقيبة الظهر،الخوارزمية الوراثية، طريقة التجميع.