UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences

Département de Mathématiques

MEMOIRE DE MAGISTER

En Mathématiques

Spécialité : Modélisation mathématiques pour l'aide à la décision

APPROCHE METAHEURISTIQUE POUR OPTIMISATION MULTIOBJECTIF DISCRETE

Par

DJEMIA NORA

Devant le jury composé de :

N. OUKID	Maître de conférences A, USD de Blida	Présidente
M.E-A. CHERGUI	Maître de conférences A, USTHB	Rapporteur
M. MAHIEDDINE	Maître de conférences A, USD de Blida	Examinateur
M. MOULAÏ	Professeur, USTHB	Examinateur

Blida, 19 Novembre 2012

RESUME

La naissance de l'optimisation multiobjectif remonte à un ouvrage de W. Pareto 1906 sur l'économie politique, dans lequel l'auteur définit pour la première fois ce qu'est un optimum multiobjectif. Ces dernières années les métaheuristiques, notamment les algorithmes évolutionnaires, ont permis l'élaboration de méthodes de résolution très performantes.

Dans cette thèse, nous nous intéressons aux problèmes d'optimisation multiobjectifs en appliquant un algorithme évolutionnaire NSGAII (Fast Nondominated Sorting Genetic Algorithm) au problème de sac à dos multiobjectif, on met en évidence la convergence du front Pareto donné par NSGAII vers celui donné par une méthode exacte en calculant la distance de Tchebycheff séparant les deux fronts.

Les résultats obtenus montrent la performance de l'algorithme évolutionnaire multiobjectif utilisés dans la programmation informatique, aussi bien en termes de convergence qu'en termes de temps d'exécution.

ABSTRACT

The birth of the multi-objective optimization returns back to the book of W. Pareto 1906 about political economy, in which the writer defines for the first time what is multiobjective optimum. These last years the metaheuristic, including evolutionary algorithms permit the elaboration of the elaboration of the resolution of the methods that very efficient.

In this thesis, we are interrelated in the multiobjective optimization problems by applying an algorithm évolutionnaires NSGAII (Fast Nondominated Sorting Genetic Algorithm) to the problem of multiobjective back bag, on high lights the convergence of the Pareto front gives by NSGAII towards that gives that gives an exact method by calculating the distance from Chebyshev separating the two fronts.

The results obtained show the performance of the multi-objective evolutionary algorithm used in computer programming, as will in terms of convergence as in terms of execution time.

ملخص

إن مولد التفاؤل بالتعدد الأهداف يعود إلى العمل السيد Pareto سنة 1906 حول الاقتصاد السياسي حيث تم تعريف الكاتب ولأول مرة حول قضية التعدد الأهداف. في السنوات الماضية تقدمت الطرق باستعمال (metaheuristic) بما في ذلك الخوارزميات التطورية(évolutionnaires les algorithmes), وتطويرها بأساليب ذات كفاءة عالية سمحت التهئ للطرق التصحيح والتجديد.

في هذه الأطروحة ركزنا على المشاكل التعدد الأهداف بتطبيقها على الحقيبة الظهرية باستعمال إحدى الخوارزميات التطورية NSGAII (السريع خوارزمية الوراثية) حيث سلطنا الضوء على التقارب بين الجبهة باريتو والتي قدمت من طرف NSGAII نحو الذي قدم حول الطريقة الصحيحة وذلك بحساب المسافة (Tchebycheff) التي تفرق بينهما.

أظهرت النتائج المحصل عليها أداء مميز لمتعدد الأهداف لاستعمال لخوارزمية التطورية NSGAII في برمجة الكمبيوتر, سواء من حيث التقارب من حيث وقت التنفيذ.

REMERCIEMENTS

Mes profondes gratitudes et mes sincères remerciements sont présentés à mon promoteur Dr. Chergui Mohamed EL Amine à l'USTHB d'Alger pour ces dispositions excellentes, ses préoccupations sérieuses dans la réalisation de ce travail.

Mes respects et vifs remerciements au Pr. Mostafa Blidia, au Pr. Chellali Mustapha et au Dr. Tami Omar de l'université Saad Dahlab de Blida.

Je tiens à remercier vivement Dr. N.OUKID de l'université Saad Dahlab de Blida d'avoir accepté de présider le jury.

Je remercie également Dr. M. Mahieddine de l'université Saad Dahlab de Blida et Dr. M. Moulai de l'USTHB d'Alger d'avoir accepté de faire partie du jury .

Je saisis aussi cette opportunité pour exprimer mes remerciements et ma gratitude à tous ceux qui m'ont aidé de prés ou de loin et dont le soutien m'a été précieux durant la préparation de ce mémoire.

DEDICACES

Je dédie ce mémoire :

- A mon père et à ma mère pour leur patience sans limite, leur sacrifice de tous les instants, qui m'ont tout donné et qui ont été à l'origine de ma réussite.
- A mon marie et mes enfants.
- A mes frères et mes sœurs.
- A tous ce qui m'ont aidée à réaliser ce travail particulièrement Madame H.Harfouch qui m'a aidée et encouragé aux moments où j'en avais besoin, Melle A.Boumesbah, Mme F.Kadik, Mme F.Loranie , Mr L.Ouldrabah et Mr M.Aattalah.

Merci à tous

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS DES TABLEAUX

INTRODUCTION

De nombreux problèmes rencontrés quotidiennement dans divers domaines (transport, télécommunication, économie ...etc) nécessitent la considération de plusieurs objectifs contradictoires simultanément, toutefois, la majorité de ces problèmes ne peuvent pas être résolus de manière pratique par des algorithmes exacts car leur résolution exacte exige souvent un temps de calcul prohibitif, voire même irraisonnable dans le cas d'instances de grandes tailles ou dans le cas d'instances qui sont qualifiées de difficiles (des données dont l'instanciation rend le problème particulièrement difficile à résoudre).

En effet, dans le cas de plusieurs critères contradictoires, on ne peut pas trouver une solution unique idéale qui optimise tous ces critères simultanément puisqu'on ne peut pas améliorer l'un des critères sans détériorer au moins l'un des autres critères. Il faut donc trouver des solutions qui réalisent un certain compromis entre tous les critères. L'optimisation multiobjectif a pour objet le développement d'outils adéquats, aussi bien théoriques que pratiques, pour la résolution de ce type de problèmes. Les méthodes pour l'optimisation multiobjectif ont connu un intérêt croissant ces vingt dernières années.

Des méthodes approchées, appelées métaheuristiques, ont été proposées pour essayer de remédier aux difficultés mentionnées ci-dessus. En fait, l'objectif d'une métaheuristique est de trouver des solutions de bonne qualité (une bonne approximation du front Pareto lorsqu'il s'agit d'un problème combinatoire multiobjectif NP-difficile) durant un temps d'exécution (temps CPU) raisonnable. L'intérêt croissant apporté aux métaheuristiques est tout à fait justifié par la croissance rapide de la puissance calculatoire des ordinateurs récents, ce qui a permis de mettre au point des métaheuristiques de plus en plus complexes qui ont fait preuve d'une certaine efficacité lors de la résolution de plusieurs problèmes à caractère NP-difficile. Parmi ces métaheuristiques, on cite les algorithmes évolutionnaires qui s'adaptent bien aux contextes multiobjectifs puisqu'ils font évoluer simultanément toute une population de solutions en utilisant des opérateurs dits évolutionnaires; ce qui permet de trouver tout un ensemble de solutions à chaque exécution.

Des problèmes classiques de l'optimisation combinatoire ont été également étudiés en considérant au moins deux objectifs, nous citons entre autres les problèmes : d'affectation, de plus court chemin, de sac à dos, de voyageur de commerce, de flot dans un réseau et d'ordonnancement. La notion d'optimalité disparaît pour les problèmes de ce type au profit de la notion d'efficacité. Une solution efficace (encore appelée Pareto optimale) est une solution à partir de laquelle, il est impossible d'augmenter la valeur d'un objectif sans diminuer celle d'au moins un autre, l'image associée à une telle solution est non dominée.

Nous nous plaçons dans ce mémoire, dans le contexte de l'optimisation multiobjectif discrète où il s'agit de déterminer l'ensemble des solutions non dominées du problème de sac à dos multiobjectif. Nous nous intéressons aux algorithmes évolutionnaires multiobjectifs, et plus particulièrement l'algorithme : Non Dominated Sorting Genetic Algorithm (NSGAII), en fait, notre objectif est d'adapter cet algorithme (NSGAII) au problème de sac à dos multiobjectif. Une étude comparative est faite entre les résultats obtenus par (NSGAII) et une méthode exacte en calculant la distance entre leurs résultats.

Ce mémoire comporte quatre chapitres :

Dans le premier chapitre, nous introduisons le domaine de l'optimisation combinatoire multiobjectif. Nous donnons d'abord les définitions de base liées à ce domaine. Ensuite, nous présentons un état de l'art des différentes méthodes de résolution destinées aux problèmes d'optimisation combinatoire multiobjectif.

Le deuxième chapitre est consacré à une description de quelques métaheuristiques existantes dans la littérature (algorithme génétique, tabou, recuit simulé…), ensuite les mécanismes spécifiques aux algorithmes évolutionnaires multiobjectifs, nous décrivons, en même temps, quelques célèbres algorithmes évolutionnaires multiobjectifs de la littérature. Enfin nous discutons de la convergence théorique et des structures de données dans le contexte des algorithmes évolutionnaires multiobjectifs.

Dans le troisième chapitre, nous présentons différents types de problèmes de sac à dos : sac à dos monoobjectif unidimensionnel, borné et non borné multidimensionnel, sac à dos multiobjectif et sac à dos multiobjectif multidimensionnel, ainsi que les méthodes de résolutions existantes dans la littérature, en mettant l'accent sur l'algorithme évolutionnaire déjà cité dans le chapitre 2.

Le quatrième chapitre est consacré à l'implémentation de la métaheuristique NSGAII, ainsi qu'à l'expérimentation informatique dans laquelle on met en évidence la convergence du front Pareto donné par NSGAII vers celui donné par une méthode exacte. Ceci est rendu possible à l'aide du calcul de la distance séparant ses deux fronts.

Nous terminons notre travail par une conclusion et des perspectives.

CHAPITRE 1

OPTIMISATION MULTIOBJECTIF

1.1. Introduction

Pendant de nombreuses années, l'intérêt des chercheurs s'était porté sur les problèmes d'optimisation où il s'agit de maximiser ou minimiser une seule fonction objectif par rapport à un ensemble de paramètres ou variables de décision. Un constat qui ne concorde pas toujours avec la réalité, où la majorité des problèmes présentent plusieurs objectifs à optimiser (souvent contradictoires) [3].

Dans ce chapitre, nous présentons ce problème d'optimisation multiobjectif (POMO) qui sera le cadre de ce travail. Nous introduisons d'abord les principales notions liées à ce domaine tel que la dominance, la surface de compromis. Ensuite nous donnons un aperçu des différentes approches de résolution destinées aux problèmes de ce type.

1.2. Problème d'optimisation linéaire monoobjectif

Le problème d'optimisation monoobjectif est défini comme suit :

$$\begin{cases} Minimiser\ f(x) & (une\ seule\ fonction\ objectif) \\ tel\ que\ x \in S \end{cases}$$

où $S = \{x \in \mathbb{R}^n, \boldsymbol{Ax} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ est un ensemble de solutions réalisables, $A \in \mathbb{R}^{m \times n}$ et $\boldsymbol{f}: \mathbb{R}^n \longrightarrow \mathbb{R}$.

Un tel problème peut être résolu par différentes méthodes.

- Méthodes exactes (Simplexe, deux phases, Gomory,…) pour les instances de moyenne et petite taille.
- Méthodes heuristiques (génétique, recuit simulé, tabou,…) pour les instances de grande taille.

1.3. Problèmes d'optimisation multiobjectifs (POMO) :

Les problèmes d'optimisation multiobjectifs sont une généralisation à m fonctions objectifs des problèmes d'optimisation classiques. Ils sont définis formellement comme suit:

$$(\boldsymbol{POMO})\begin{cases} Minimiser\ (f_1(x), f_2(x), \dots, f_m(x)) \\ tel\ que\ \ x \in S \end{cases}$$

La résolution de ce problème consiste à minimiser au mieux ces m fonctions objectifs dans le but de ne pas trop dégrader les valeurs des optima obtenus par rapport à ceux obtenus lors d'une optimisation monoobjectif effectuée objectif par objectif [3].

Ces fonctions f_1, f_2,… f_k désignent les différents objectifs à optimiser. Les variables x_1, x_2,… x_n représentent les variables de décision. L'évaluation d'une solution x est décrite par un vecteur objectif f(x)= [f_1(x), f_2(x),… f_m(x)] . Nous serons donc en face de deux types d'espaces :

- L'espace de décision x de dimension n qui désigne l'ensemble des solutions x=(x_1, x_2,… x_n)
- L'espace des critères (objectifs) de dimension m qui désigne l'ensemble des évaluations possibles correspondant aux vecteurs critères f(x).

Si les m critères et les contraintes sont linéaires en x, on obtient un problème de programmation linéaire multicritère :

$$(\boldsymbol{POMO})\begin{cases} Minimiser\ Z_i(x) = c_i x & i = 1, \dots m \\ Telque \quad x \in S \end{cases}$$

où S={x∈ $\mathbb{R}^n$, Ax ≥ b, x≥ 0} avec les matrices A (m, n), c_i (1, n), b (m×1)

En effet un problème de maximisation peut être aisément transformé en problème de minimisation en considérant l'équivalence suivante :

Maximiser Z(x) ⇔ minimiser –Z(x)

Exemple 1

Le schéma suivant présente une situation de problème combinatoire: on veut acheter une voiture à la mode et en même temps avec un prix raisonnable qui ne peut pas dépasser une certaine limite. Si on maximise le premier critère (une bonne voiture) on va avoir un prix maximal, dans le cas contraire on va aboutir à une mauvaise voiture mais avec un prix minimal dans les limites; on constate sur cet exemple qu'il est difficile de concilier les deux critères.

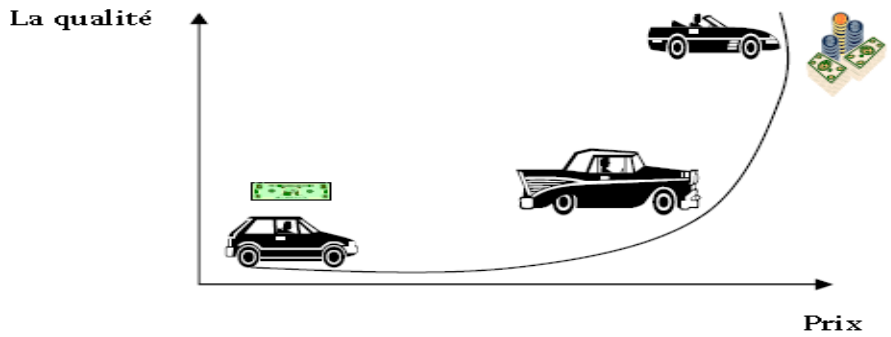


Figure1.1 Une figure illustrant un problème.

1.4. Notions fondamentales

1. 4. 1 Définitions et Notations

Dans ce paragraphe nous définirons les principales notions utilisées dans ce mémoire ainsi que les notations correspondantes.

Espace des décisions

Définition 1

L'espace $\mathbb{R}^n$ dans lequel se situe l'ensemble des solutions S ($S \subseteq \mathbb{R}^n$) est appelée espace des décisions.

Espace des critères

Définition 2

Par espace des critères on entend l'espace $S \subseteq \mathbb{R}^m$, dans lequel se situe $\boldsymbol{Z_S}$:

Dans le cas linéaire, $\boldsymbol{Z_S}$ est l'image de S dans $\mathbb{R}^m$, par l'application linéaire associée à le vecteur C= ($c_1, c_2, \ldots, c_m$) :

$x=(x_1, x_2, \ldots, x_n) \in S \subseteq \mathbb{R}^n \rightarrow Cx = (c_1x, c_2x, \ldots, c_mx) = (z_1, z_2, \ldots, z_n) \in Z_S$

La figure 1.2 représente espace décisionnel et espace des critères d'un problème d'optimisation multiobjectif (exemple avec deux variables de décisions et deux fonctions objectifs).

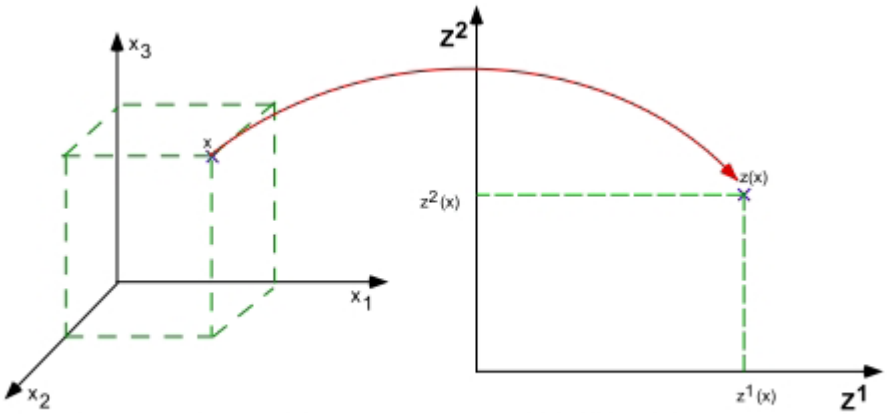


Figure 1.2 Espace des décisions et Espace des critères [1].

Exemple 2 (introduit par [4])

Considérant un problème multiobjectif (2-objectifs) représenté par la figure 1.3 avec c_1= (1,-1) et c_2= (-1,2)

Les vecteurs critères associés aux points extrêmes de S sont comme suit :

x_1= (0,0) → z_1= (0,0)

x_2= (1,2) → z_2= (-1,3)

x_3= (3,3) → z_3= (0,3)

x_4= (4,1) → z_4= (3,-2)

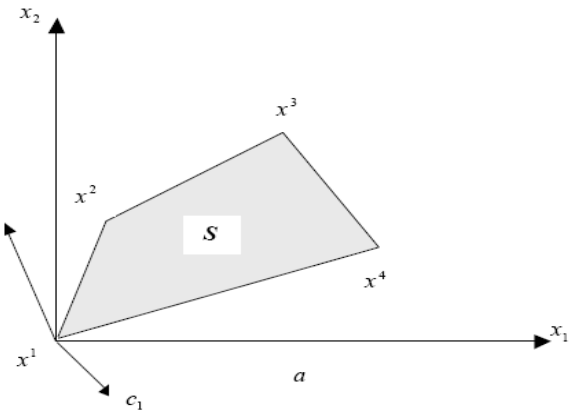


Figure 1.3.a) Espace des décisions.

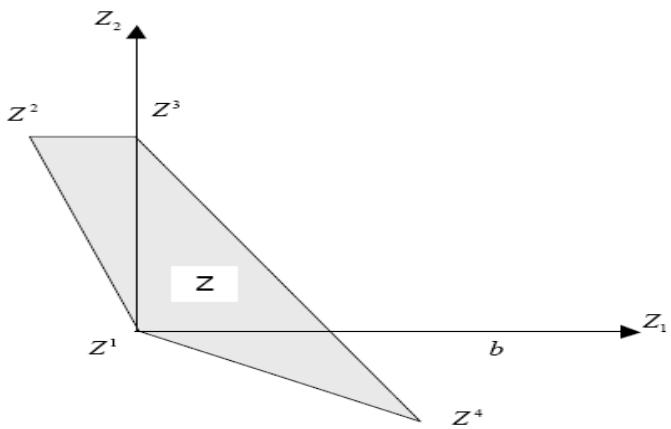


Figure 1.3.b) Espace des critères.

Sur la figure 1.3(b), on remarque que Z est convexe et que les points extrêmes de Z sont les images des points extrêmes de *S*. Avoir des portions du graphe de Z en dehors de la partie positive du plan est interprété par le fait que certaines fonctions objectifs ont des coefficients négatifs.

Toute solution admissible ($x \in S$) ne constitue pas évidemment un bon compromis à considérer ; pour qu'elle le soit, on impose généralement une propriété, basée sur une relation d'ordre et dominance :

1.4.2. Relations d'ordre et de dominance

La relation de dominance nous permet de restreindre l'ensemble des solutions de compromis. Plusieurs types de relations de dominance existent dans la littérature [5]. Mais la plus célèbre et la plus utilisée est la dominance au sens de Pareto qui est défini par :
Le vecteur x domine le vecteur y si :

- x est au moins aussi bon que y en regard à tous les objectifs et
- x est strictement meilleur que y pour au moins un objectif.

Pour définir clairement et formellement cette notion de dominance au sens de Pareto, les relations $=, \leq, <$ usuelles sont étendues aux vecteurs.

Définition 3

Soient x et y, deux vecteurs de même dimension,

x= (x=($x_1, x_2, \ldots x_m$) et y= ($y_1, y_2, \ldots y_m$)

- $x = y \quad ssi \quad \forall\, i \in \{1, \ldots, m\}, \quad x_i = y_i$
- $x \leq y \quad ssi \quad \forall\, i \in \{1, \ldots, m\}, \quad x_i \leq y_i$
- $x < y \quad ssi \quad x \leq y \wedge x \neq y$

$les\ relations \geq et > sont\ définies\ de\ manière\ analogue.$

Pour comparer les points, par contre, elles sont insuffisantes pour comparer des points issus des problèmes multiobjectifs par exemple les points (qui sont ici des vecteurs) x=(1,2) et y=(4,1) sont incomparables à l'aide de ces relations. Nous

définissons donc maintenant la relation de dominance au sens de Pareto qui permet de comparer tous les points (vecteurs) de décision possibles.

La dominance au sens de Pareto

Définition 4

Considérons un problème de minimisation. Soient x et y deux vecteurs de décision,

- $x\ domine\ y \quad ssi \quad z(x) \leq z(y) \quad et \quad z(x) \neq z(y)$

$(z_i(x) \leq z_i(y) \quad pour\ tout\ i = 1, \dots k\ et\ z_i(x)\ \ z_i(y)\ pour\ au\ moins\ un\ indice\ i)$

- $x\ domine\ fortement\ y \quad ssi \quad z_i(x) < z_i(y) \quad pour\ tout\ i = 1, \dots, m$

- $x\ domine\ faiblement\ y \quad ssi \quad z_i(x) \leq z_i(y) \quad pour\ tout\ i = 1, \dots, m$

Exemple 3 (introduit par [3])

La figure 1.4 représente un exemple de dominance. On a un problème d'optimisation biobjectif. Les fonctions objectif sont f_1 et f_2 Chaque point y_i est l'image de x_i par f : $y_i = f(x_i)$ avec i =1,…,4 Prenons le point y_1 comme point de référence. Nous pouvons distinguer trois zones :

- La zone de préférence est la zone contenant les points dominés par y_1,
- La zone de dominance est la zone contenant les points dominant y_1,
- La zone d'incompatibilité contient les points incomparables avec y_1

x_1 domine x_2, x_1 dominé par x_3 et x_1 est non dominé (incomparable) avec y_4.

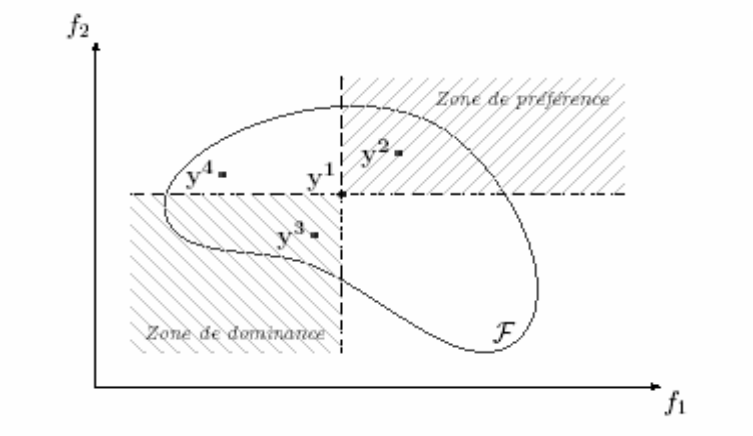


Figure 1.4 Exemple de dominance [3].

Avec ce nouvel outil, nous pouvons maintenant définir l'optimalité dans le cas des problèmes multiobjectifs. Le terme Pareto optimalité est aussi connu sous efficacité ou non infériorité. Nous pouvons définir l'efficacité c'est à dire utilisant la dominance au sens de Pareto [3].

Efficacité

Définition 5 :

Une solution $x^* \in S$ est une solution efficace s'il n'existe pas de $x \in S$ telle que :

$$Z(x) \text{ domine } Z(x^*)$$

Un point est efficace si son image par Z est un vecteur critère non dominé. Une définition équivalente de l'efficacité est :

Définition 6 :

Une solution $x^* \in S$ est une solution efficace si et seulement s'il n'existe pas de $x \in S$ telle que :

$$z_i(x) \leq z_i(x^*) \; ; \quad i=1,\ldots, m \text{ avec au moins une inégalité stricte.}$$

A partir d'un point efficace, il est impossible d'augmenter la valeur d'un des critères sans diminuer la valeur d'au moins un autre critère.

Efficacité faible

Définition 7 :

Une solution $x^* \in S$ est une solution faiblement efficace s'il n'existe pas de $x^* \in S$ telle que **:** $Z(x) < Z(x^*)$

1.4.3. Surface de compromis

La surface de compromis (ou le front de Pareto) est l'ensemble de points de Z tels qu'aucun autre point de z ne les domine. La surface de compromis est aussi appelée l'ensemble des solutions non dominées.

1.4.3.1. Front minimal et front maximal complet

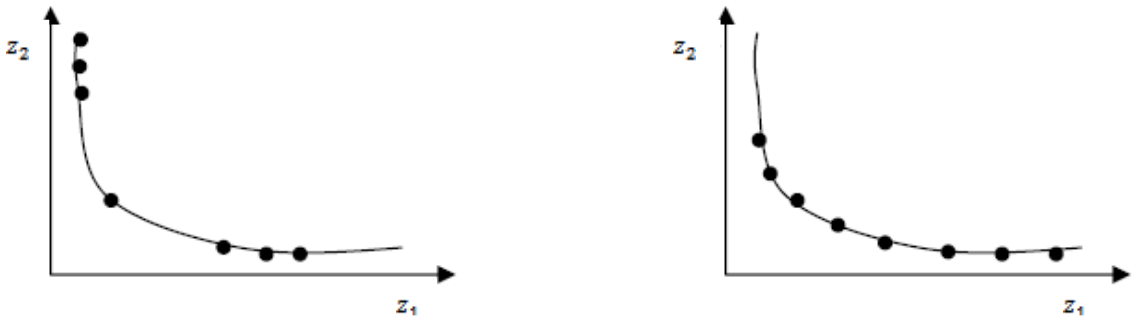
La définition de front se réfère à l'espace des objectifs. Une solution appartient au front si elle n'est dominée par aucune autre solution réalisable. Lorsque deux solutions ont exactement les mêmes valeurs pour l'ensemble des critères

(objectifs), elles sont équivalentes dans l'espace critère, mais peuvent correspondre à deux solutions différentes dans l'espace de décision. Une question importante est de savoir s'il est intéressant de garder ces deux différentes solutions. La réponse peut dépendre du contexte (type de problème étudié) en plus de la volonté des décideurs [13].

Lors de la résolution d'un problème comportant énormément de solutions Pareto,il est préférable de privilégier une bonne approximation de l'ensemble de la frontière et donc favoriser la diversité (du côté objectif) des solutions retenues [12].

1. Représentation de la surface de compromis

Toutes les représentations de la surface de compromis, pour un même problème, ne sont pas équivalentes si on ne donne pas tout le front de Pareto. En effet, la représentation idéale de la surface de compromis devra être constituée de points solution de notre problème répartis de manière uniforme sur la surface de compromis (voir la figure ci-dessous).



Une mauvaise représentation de la surface de compromis

Une bonne représentation de la surface de

Figure 1.5 Représentation de la surface de compromis

Dans le premier cas, les points représentant la surface de compromis ne sont pas répartis de manière uniforme. L'utilisateur n'aura alors pas en sa possession un ensemble de solutions très utiles. En effet, s'il décide que la solution qu'il avait choisie ne lui convient pas, le choix d'une autre solution risque de se faire varier brusquement tous ses objectifs, et cette nouvelle solution ne lui conviendra pas non plus. Il est alors probable que la solution offrant le < meilleur > compromis se trouve dans une zone qui ne soit pas représentée par des points solution [12].

Point idéal

Définition 8

Le vecteur défini par :

$$\underline{z} = \left(\min_{x \in S} z_1(x), \dots, \min_{x \in S} z_m(x)\right) \in \mathbb{R}^m$$

constitue le point idéal ; en général: $\underline{z} \notin Z_s$.

Les coordonnées de ce point sont obtenues en optimisant chaque fonction objectif séparément. On dit aussi que les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des points de la surface de compromis (frontière Pareto).

Point Nadir

Définition 9

Soit *Eff(P)* l'ensemble des solutions efficaces de *(P)*.

Le vecteur défini par :

$$\overline{z} = \left(\max_{x \in Eff(P)} z_1(x), \dots, \max_{x \in Eff(P)} z_m(x)\right) \in \mathbb{R}^m$$

constitue le point Nadir ; en général : $\overline{z} \notin Z_s$.

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

Il existe plusieurs approches pour estimer le point nadir, la plus simple consiste à utiliser une matrice carrée de dimension m appelée matrice des gains et donnée par :

$$G = \begin{bmatrix} \underline{z}_1 & z_{12} \cdots & z_{1m} \\ z_{21} & \underline{z}_2 \cdots & z_{2m} \\ \vdots & \vdots \ \vdots & \vdots \\ z_{m1} & z_{m2} \cdots & \underline{z}_{mm} \end{bmatrix}$$

où $\underline{z}_i = \min_{x \in S} z_i(x) = z_i(\overline{x}_j) \ \forall \ \mathrm{ij} = 1, \dots, \mathrm{k}$ avec $z_{ij} = z_i(\overline{x}_j)$ et $\overline{x}_j$: est la solution optimale obtenue en minimisant le critère z_i sur S, $\forall \ \mathrm{j} = 1, \dots, \mathrm{m}$

Remarque

-La matrice des gains n'est pas toujours univoquement déterminée ; en effet, si la solution optimale x_k^* n'est pas unique pour un critère m, la colonne correspondante de la matrice des gains dépendra de la solution choisie [4].

- Le point nadir est alors estimé par le point de $\mathbb{R}^k$ suivant :

$$\check{z}_i = \max_{j=1,\dots,k} G_{ij}, \mathrm{i} = 1, \dots, \mathrm{m}$$

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

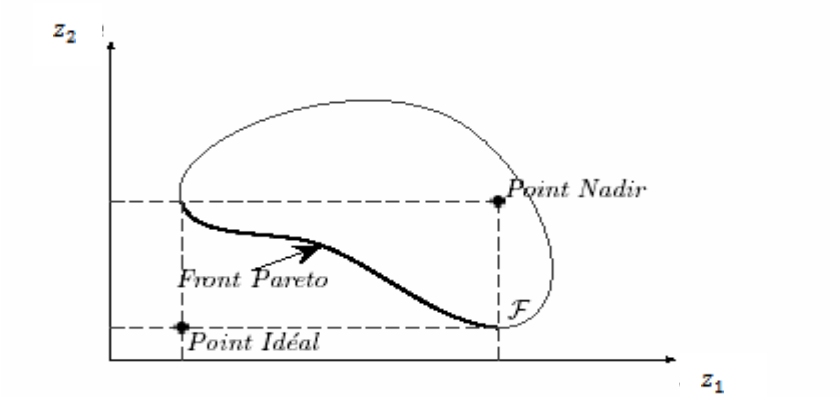


Figure 1.6 Front Pareto, Point Nadir et Point Idéal [3].

Dans l'exemple de la figure 1.6, le problème considéré est un problème de minimisation avec deux objectifs. Deux points particuliers apparaissent clairement : le point idéal et le point Nadir, ces deux points sont calculés à partir du Front Pareto. Le point idéal (resp. le point Nadir) domine (resp. est dominé par) tous les autres points de la surface de compromis.

Bien que ces points ne soient pas forcément compris dans la zone réalisable, ils servent souvent de pôle d'attraction (resp. de répulsion) lors de la résolution du problème. La figure 1. 6 nous indique aussi que le Front Pareto peut avoir des propriétés particulières quant à sa forme. La principale caractéristique utilisée pour

comparer les formes de ces courbes est la convexité [3]. Nous rappelons alors la définition de la convexité :

Convexité

Définition10 :

Un ensemble A est convexe, si l'équivalence suivante est vérifiée :

$$\forall \boldsymbol{x \epsilon A} \wedge \boldsymbol{x' \epsilon A} \ \boldsymbol{et} \ \forall \boldsymbol{\lambda} \in [\mathbf{0},\mathbf{1}] \Leftrightarrow \boldsymbol{\lambda x} + (\mathbf{1} - \boldsymbol{\lambda})\boldsymbol{x'} \in \boldsymbol{A}.$$

La convexité est le premier indicateur de la difficulté du problème. En effet, certaines méthodes sont dans l'incapacité de résoudre des problèmes non convexes de manière optimale. Mais il existe d'autres indicateurs tout aussi importants, notamment la continuité et la nature des variables de décision (entières ou réelles), [3]

1.5 Optimisation et aide à la décision

La résolution d'un problème optimisation multiobjectif (POMO) revêt deux aspects de difficultés distinctes : l'optimisation et l'aide à la décision. L'aspect optimisation, concerne le processus de recherche ou d'approximation de l'ensemble des solutions optimales (Pareto,...), l'aspect aide à la décision s'adresse au problème de sélection d'une solution représentant un bon compromis entres les différents critères. Une prise de décision humaine est donc nécessaire pour gérer les interdépendances entre les objectifs. Selon la façon suivant laquelle les deux aspects sont combinés, trois approches principales sont possibles [6].

1. Aide à la décision à priori

Cette approche consiste à combiner les différentes fonctions objectifs en une seule fonction dite fonction d'utilité (ou fonction coût). Ce procédé suggère que le décideur à une connaissance des différents poids des objectifs ainsi que la fonction d'utilité. Le problème est alors traité par un algorithme d'optimisation monoobjectif classique. L'utilisateur définit le compromis qu'il désire avant de lancer la méthode d'optimisation. Toutes les méthodes agrégatives sont des méthodes à préférence à priori.

2. Aide à la décision à posteriori

Dans ce cas l'algorithme d'optimisation est lancé en premier générant plusieurs solutions. Le décideur choisit alors parmi celles-ci la solution qui lui convient le plus.

3. Aide à la décision interactive :

Une coopération entre le solveur et le décideur est installée. A partir des connaissances acquises lors de la recherche, le décideur formule ces préférences. Ces dernières sont introduites par le solveur ultérieurement. Ce processus est réitéré plusieurs fois.

1.6. Méthodes de résolution

Les méthodes d'optimisation multiobjectif peuvent être classées selon le schéma suivant [6] :

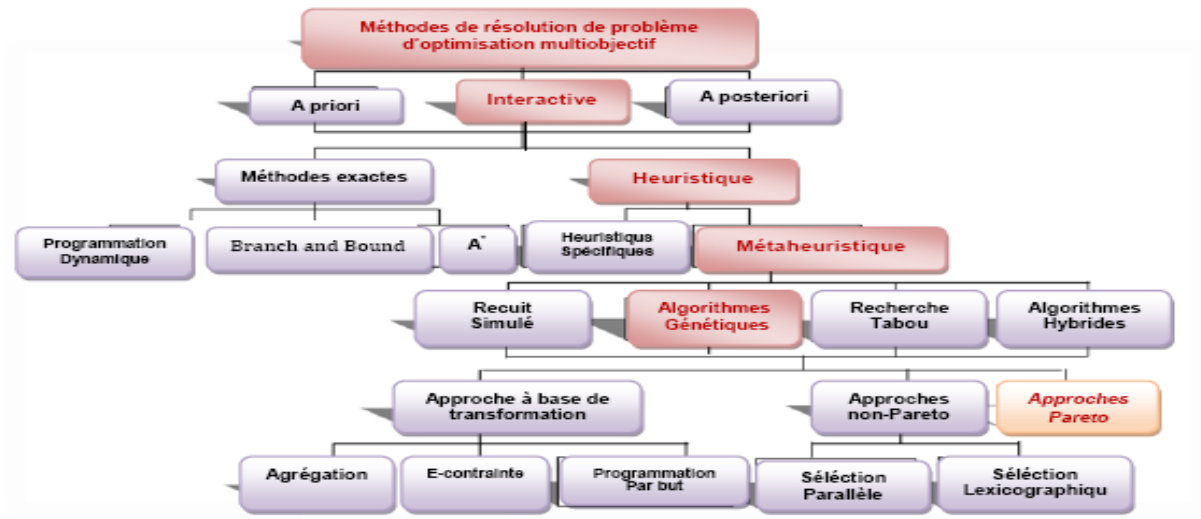


Figure 1.7 Classification des méthodes d'optimisation multiobjectif.

Dans la littérature, on distingue deux classes de méthodes différentes :

1-Les méthodes exactes

Telles que Branch and Bound, l'algorithme A^* et la Programmation dynamique ont subit des révisions afin de les adapter au cas multiobjectif. Cependant ces méthodes sont destinées à des problèmes de petites tailles. Elles permettent théoriquement de trouver une solution optimale grâce à un

parcours exhaustif. Leur efficacité est mise en question dès que la taille du problème augmente et que le nombre de critères retenus croît. Elles ne sont pas adaptées à des problèmes NP-difficiles [3].

2-Les méthodes heuristiques

Tentent d'approcher l'ensemble des solutions optimales. Elles s'appuient sur les connaissances du domaine (heuristiques spécifiques) ou sur des algorithmes (métaheuristiques). Ces dernières sont applicables à une large gamme de problèmes et leur efficacité demeure relativement bonne quand la taille et le nombre d'objectifs augmentent [3].

1.6.1. Les méthodes exactes

1.6.1.1 Algorithme A* :

Stewart et White proposèrent une version multiobjectif de l'algorithme A*(A*MO). L'algorithme A* est maintenu dans sa forme originale. Le poids d'un arc (u, v) dans le graphe de recherche, correspond à un vecteur (c_1, c_2,..., c_n) où c_i désigne le coût relatif à l'objectif f_i induit par le passage de u à v. Le coût d'un chemin, correspond à la somme vectorielle des poids des arcs qui le composent. Les fonctions k^*, g*, h^* prennent une forme non scalaire où $k^*(u, v)$, désigne l'ensemble des coûts non dominés des chemins reliant u à v, g*(u) : représente le coût de la recherche ayant aboutit à la solution intermédiaire u, $h^*(u)$: désigne l'ensemble des vecteurs coûts non dominés parmi l'ensemble $\{k^*(u, v) / v \in T\}$ où T représente l'ensemble des états terminaux. Les nœuds du graphe sont triés et parcourus comme pour l'algorithme A* classique. L'ordre de tri est décrit par les relations de dominance entre les vecteurs coûts f. pour plus de détail, le lecteur peut consulter [7].

1.6.1.2 Programmation dynamique

La technique de programmation dynamique multiobjectif a été implémentée par Carraway, Morin et Moskowitz pour la résolution du problème de routage dans les réseaux [8].

L'application de la programmation dynamique pour l'optimisation multiobjectif est rare car ceci est difficile quand le nombre d'objectifs à optimiser est élevé (>2).

La difficulté est liée au principe de monotonie exigée par la méthode. Ce principe réclame un grand volume d'espace de stockage qu'il faudra utiliser pour sauvegarder l'ensemble des résultats des étapes précédentes, en plus d'un temps de calcul très élevé [3].

1.6.2. Les méthodes de résolution approchées

Trois techniques sont élaborées pour la résolution approchée : on procède soit par transformation du problème en un problème monoobjectif, soit on se basant sur la notion d'optimalité Pareto ou finalement par traitement séparé des différents objectifs (non Pareto).

1.6.2.1 Approche à base de transformation (Approche Scalaire)

Cette approche transforme un problème multiobjectif en un problème mono-objectif dans le but d'utiliser l'une des nombreuses méthodes de résolution disponibles pour l'optimisation mono-objectif parmi ces méthodes ont peut avoir :

a-d'agrégation des objectifs : (introduit par [9])

Cette méthode consiste à transformer un problème multiobjectif en un problème à un objectif en agrégeant les différents critères sous la forme d'une somme pondérée :

$$(P_\lambda)\begin{cases} Min\ f_\lambda(x) = \omega_i f_i(x) \\ tel\ que \\ x \in S_r, \omega_i \geq 0\ et\ \sum_{i=1}^{k} \omega_i = 1 \end{cases}$$

Les w_i appelés poids, déterminent le degré de priorité d'un objectif donné.

S_r : représente le domaine réalisable.

Cette approche a l'avantage évident de pouvoir réutiliser tous les algorithmes classiques dédiés aux problèmes d'optimisation à un seul objectif. C'est souvent la première approche adoptée lorsqu'un chercheur se retrouve devant un nouveau problème multiobjectif [12].

Cependant cette approche souffre de deux inconvénients :

1- le premier est dû au fait que pour avoir un ensemble de points bien répartis sur le Front Pareto, les différents vecteurs λ_i doivent être choisis judicieusement. Il est donc nécessaire d'avoir une bonne connaissance du problème [12].

2- le deuxième inconvénient provient du fait que cette méthode ne permet pas, de calculer intégralement la surface de compromis lorsque celle-ci n'est pas convexe [12].

b- la méthode E-contrainte (introduit par [10])

Une autre façon de transformer un problème d'optimisation multiobjectif en un problème à un seul objectif est de convertir k-1 des k objectifs du problème en contraintes et d'optimiser séparément l'objectif restant. Le problème d'optimisation s'écrit alors :

$$\begin{cases} Min\ f_p(x) \\ avec\ f_i(x) \leq \varepsilon_i, \quad \forall i \in [1,\dots,n], i \neq p \\ x \in S \end{cases}$$

Une fois que la première fonction objectif choisi est optimisée, on essaye d'optimiser une autre fonction objectif en prenant comme borne pour la première fonction objectif la valeur maximale trouvée durant la première étape et ainsi de suite jusqu'à ce qu'on passe par toutes les fonctions objectifs du problème.

Cette méthode est autant plus intéressante quand l'utilisateur désir privilégier un objectif donné. Par contre, elle perd de son intérêt quand le nombre d'objectifs est élevé. Aussi d'autres inconvénients surgissent à savoir :

- la connaissance à priori des intervalles appropriés pour les valeurs ε_i, est exigée. Ceci induit le calcul des bornes inférieures des différents objectifs.

- les contraintes rajoutées compliquent la résolution du problème.

c- Programmation par but **:**(introduit par [11])

Cette approche consiste à transformer le problème multiobjectif en un problème à un seul objectif où l'on cherche à minimiser l'écart relatif par rapport à un point de référence appelée but, fixé par la méthode ou le décideur [12].

Cette méthode transforme donc le POMO en un problème d'optimisation monoobjectif de la manière suivante :

- on choisit un vecteur de fonctions objectifs initial $\mathcal{B}$,

- on choisit aussi une direction de recherche ω (vecteur de coefficients de pondération des fonctions objectifs),

- on cherche ensuite à minimiser un coefficient scalaire λ qui représente l'écart par rapport à l'objectif initial B que l'on s'est fixé.

$$\begin{cases} Minimiser\ \lambda \\ Tel\ que\ f_1(x) - \omega_1.\lambda \leq \mathcal{B}_1 \\ \vdots \\ f_m(x) - \omega_m.\lambda \leq \mathcal{B}_m \\ et\ que\ \ g(x) \leq 0 \\ avec\ x \in \mathbb{R}^n, f(x) \in \mathbb{R}^m, g(x) \in \mathbb{R}^q \end{cases}$$

Ce problème est donc un problème d'optimisation monoobjectif, la fonction objectif à minimiser est le scalaire λ.

Cette approche est efficace et facile à implémenter, cependant elle est sensible au choix des paramètres ω et $\mathcal{B}$ (qui doivent être bien choisis par l'utilisateur), un choix arbitraire de ceux-ci peut conduire à des résultats non cohérents [3].

1.6.2.2. Approche par traitement séparé des objectifs (Approche non-Pareto)

Cette approche consiste à réaliser la recherche en traitant les différents objectifs séparément. Dans les algorithmes génétiques, la prise en compte des différents objectifs apparaît au niveau de la phase de sélection (sélection parallèle, sélection lexicographique).

a- sélection parallèle (VEGA: Vector Evaluated Genetic Algorithm**)**

Cette approche à été proposé par Schaffer [17] qui l'a utilisé dans son algorithme génétique VEGA considéré comme le premier algorithme évolutionnaire dédié aux problèmes multiobjectifs.

Le principe de cette approche permet de traiter un problème d'optimisation multiobjectif sans avoir à agréger les fonctions objectifs en une seule [17] et n'utilisent pas non plus la notion de dominance au sens de Pareto. Elles s'appuient sur des techniques qui traitent séparément les différents objectifs d'un problème.

b) L'ordonnancement lexicographique

Cette méthode consiste à considérer les fonctions objectifs les unes après les autres et à minimiser à chaque fois un problème monoobjectif, en complétant au fur et à mesure l'ensemble des contraintes [17].

Cette méthode procède en k étapes :

Soient les fonctions objectifs f_i avec i=1,..., k, supposons un ordre tel que f_1 plus important que f_2, f_2 plus important que f_3,…..,f_{k-1} plus important que f_k : il faut :

*** Etape 1 :**

Minimiser $f_1(x)$

Avec $x \in S_r$

on note f_1^* la solution de ce problème.

*** Etape2 :**

Minimiser $f_2(x)$

$f_1(x) = f_1^*$

Avec $x \in Sr$

on note f_2^* la solution de ce problème.

...

*** Etape k :**

Minimiser $f_k(x)$

$f_1(x) = f_1^*$, $f_2(x) = f_2^*$,…, $f_{k-1}(x) = f_{k-1}^*$

Avec $x \in Sr$

La procédure est donc répétée jusqu'à ce que tous les objectifs soient traités. La solution obtenue à l'étape k sera la solution du problème.

L'inconvénient de cette méthode est qu'elle requiert un choix de la séquence des objectifs à minimiser. Ainsi, deux choix différents génèrent deux solutions distinctes.

1.6.2.3. Approche Pareto

Cette approche s'appuie directement sur la notion d'optimalité Pareto. La notion de dominance est alors utilisée comme moyen de comparaison et d'ordonnancement des solutions. Le principal avantage de cette approche est qu'elle soit capable de générer un ensemble de solutions Pareto appartenant aux parties concaves de la frontière Pareto. C'est ce type d'approche que nous allons considérer par la suite. Dans les algorithmes génétiques la prise en considération

du caractère dominant ou dominé des solutions apparaît dans la phase de sélection.

En effet, il existe deux générations des approches Pareto :

-La première génération est caractérisée par l'utilisation :

- d'un mécanisme de sélection Pareto basé sur la technique de ranking [51], utilisant la relation de dominance pour affecter des rangs aux individus de la population, faisant apparaître la notion de Front.
- du "sharing" (niching) qui pour éviter qu'un grand nombre d'individus se concentre autour d'un même point, la valeur d'adaptation est pénalisée en fonction du nombre d'individus au voisinage du regroupement [13].

- la deuxième génération est composée généralement des approches élitistes manipulant une population secondaire externe.

Parmi les approches représentatives de cette génération, on peut citer :

- SPEA et SPEA-2 (Strength Pareto Evolutionary Algorithm)
- NSGA-II (Non dominated Sorting Genetic Algorithm)
- PAES (Pareto Archived Evolution Strategy)
- PESA et PESA-II (Pareto Envelope-based Selection Algorithm)

Une description détaillée de ces approches sera présentée dans le chapitre 2 section 2.

CHAPITRE 2
LES METHAHEURISTIQUES
POUR
L'OPTIMISATION MULTIOBJECTIF

2.1 Introduction

Tout comme pour l'optimisation monoobjectif, on peut distinguer deux grandes familles de méthodes de résolution pour traiter un problème multiobjectif : les méthodes exactes et les méthodes approchées.

On remarque tout d'abord qu'il y a très peu de travaux sur les méthodes exactes [78] dans le contexte de la résolution d'optimisation multiobjectif NP-difficiles sans doute, à cause de la grande difficulté de ce type de problème. Dans ce travail nous nous intéressons à la résolution approchée des problèmes d'optimisation multiobjectif NP-difficiles, notamment par des métaheuristiques. Ainsi, nous présentons brièvement dans ce chapitre deux types de métaheuristiques les plus connues : la recherche locale et les algorithmes évolutionnaires.

Pour une présentation plus élaborée des métaheuristiques on définit les métaheuristiques qui suivent à partir des références suivantes [18], [19], [20] et [21]. Les métaheuristiques ont été appliquées avec succès sur un grand nombre de problèmes académiques et réels : problème d'affectation quadratique, coloriage de graphe, voyageur de commerce.

2.1.1. Heuristiques et Métaheuristiques

Du fait que tous les algorithmes exactes sous-entendent une énumération explicite ou implicite de toutes les solutions, de tels algorithmes deviennent alors inefficaces dès que la taille des problèmes augmente de façon considérable.

En outre, les problèmes de type POMO sont NP-difficiles [22] et [23] dans le sens où il n'existe pas (jusqu'à présent!) d'algorithme(s) exact(s) capable(s) de les résoudre en un temps polynomial en fonction de leur taille.

Dans l'absence de définitions précises et normalisées pour les concepts d'heuristique et de métaheuristiques, on peut se contenter des définitions suivantes:

- Une heuristique est une méthode approchée conçue pour un problème particulier dans le but de fournir des solutions de bonne qualité durant un temps de calcul raisonnable.
- Une métaheuristique est un schéma de calcul heuristique, général et adaptable à un ensemble de problèmes différents.

Ainsi, les métaheuristiques sont des stratégies heuristiques génériques dont l'adaptation à un problème donné génère un ensemble d'heuristiques spécifiques pour ce problème. La plupart des méthodes métaheuristiques ont été inspirées de phénomènes naturels (l'éthologie comme les colonies de fourmis, physiques comme le recuit simulé et biologie comme les algorithmes évolutionnaires) [21]. D'autre part, on peut partager les métaheuristiques en deux grandes classes: les métaheuristiques à solution unique : évoluant avec une seule solution et celles à solution multiple : évoluant avec plusieurs solutions. Une classification non exhaustive des méthodes métaheuristiques est illustrée dans la figure ci-dessous.

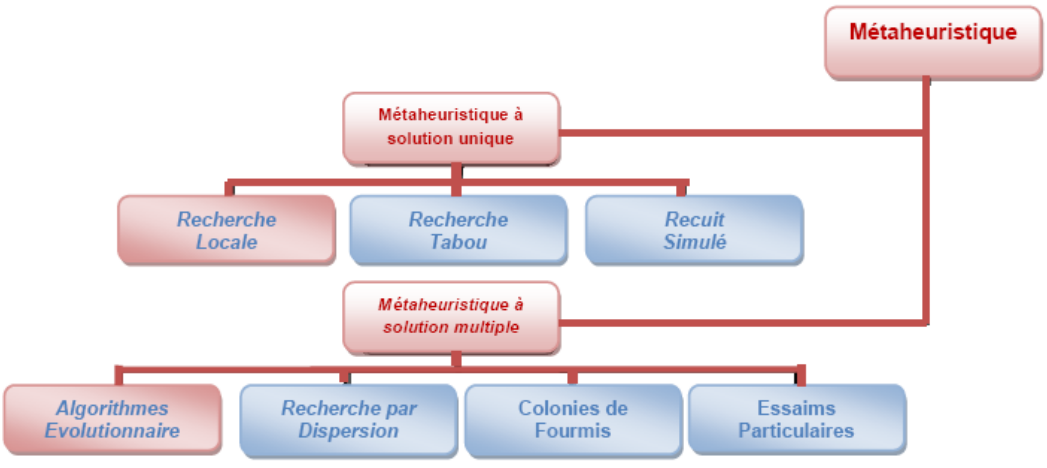


Figure 2.1 Classification des métaheuristiques.

2.1.2. Métaheuristique à solution unique

2.1.2.1. Méthodes de recherche locale

La première classe de métaheuristiques présentées regroupe les méthodes utilisant les principes de la recherche locale. Ces méthodes résolvent le problème

d'optimisation de manière itérative. Elles font évoluer la configuration courante en la remplaçant par une autre issue de son voisinage, ce changement de configuration est couramment appelée un mouvement

Dans le cas d'un problème de type POMO, on peut utiliser, par exemple, soit une fonction d'utilité, soit la relation de dominance pour évaluer les solutions voisines. La figure 2.2 montre le schéma d'évolution d'une recherche locale simple.

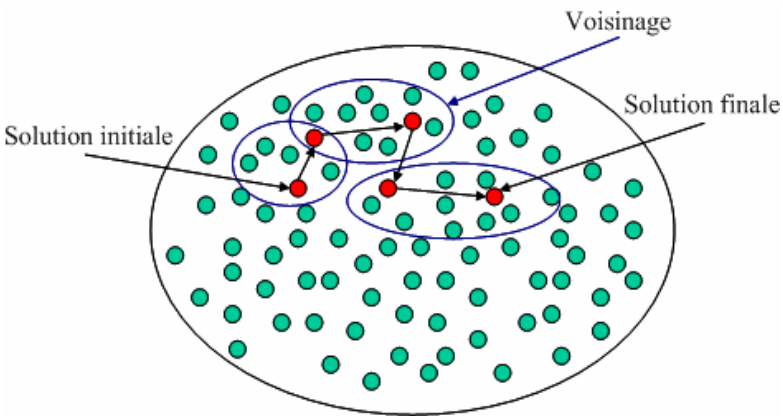


Figure 2.2 Schéma d'évolution d'une recherche locale simple [2].

Le problème majeur de la méthode de recherche locale simple réside dans le fait qu'elle s'arrête sur le premier optimum local trouvé. Cependant, cet optimum local est souvent loin de faire partie ou même d'être proche du front Pareto. Pour remédier à ce problème, on peut citer les deux techniques suivantes. La première technique couramment utilisée est celle de la relance. Elle consiste à recommencer une nouvelle recherche à partir d'une autre solution initiale appartenant à une autre région de l'espace de recherche. Cette technique peut être réitérée plusieurs fois dans l'espérance d'atteindre différentes zones prometteuses de l'espace de recherche. La deuxième technique est celle du chemin aléatoire. Elle consiste à effectuer, de temps en temps, un mouvement aléatoire pour diversifier la recherche et ainsi espérer se rapprocher du front Pareto [2].

2.1.2.2. Recherche Tabou

La recherche Tabou (Tabu Search) est introduite par Glover [24] et [25]. Cette méthode fait aussi usage de la structure de voisinage pour évoluer d'une solution à une autre. Par ailleurs, la recherche tabou utilise d'autres techniques plus sophistiquées pour palier aux problèmes posés par la recherche locale. En

fait, en examinant le voisinage de la solution courante, la recherche tabou retient toujours la meilleure solution voisine même si celle-ci est plus mauvaise que la solution courante. Cependant, cette stratégie peut entraîner des cycles (on peut reboucler indéfiniment sur des solutions déjà visitées auparavant). Pour éviter ce problème de cyclage, on introduit une sorte de mémoire appelée liste Tabou et qui sert à mémoriser les dernières solutions visitées dans le but d'interdire leur visite pendant les prochaines itérations. Concrètement, si L : est la taille de la liste tabou, alors la dernière solution retenue est introduite dans cette liste en remplaçant la solution tabou la plus ancienne et elle sera ainsi interdite pendant L les prochaines itérations. De cette façon, on évite tous les cycles de longueur inférieure à L, L : étant déterminé en fonction du problème.

La mémorisation de configurations entières serait trop coûteuse en temps de calcul et en place mémoire. Il est préférable de mémoriser des caractéristiques des configurations au lieu de configurations entières. Plus précisément, il est possible de mémoriser uniquement un attribut de la configuration [2].

Il existe aussi d'autres techniques permettant d'améliorer les performances de la méthode tabou, en particulier, **l'intensification et la diversification**. L'intensification permet de se focaliser sur certaines zones de l'espace de recherche en apprenant des propriétés favorables, par exemple les propriétés communes souvent rencontrées dans les meilleures configurations visitées. La diversification a un objectif inverse de l'intensification.

En effet, elle cherche à diriger la recherche vers des zones inexplorées, en modifiant par exemple la fonction d'évaluation. L'intensification et la diversification jouent donc des rôles complémentaires. Dans certains cas, il est également judicieux d'intensifier les recherches sur des zones qui semblent être prometteuses. D'autre part, la recherche tabou peut nécessiter une longue période à explorer une zone particulière de l'espace de recherche. Alors pour stopper une telle recherche coûteuse et diversifier la recherche sur une autre zone, on peut ajouter un mécanisme de diversification. La diversification est d'autant plus importante que lorsqu'il s'agit de résoudre des problèmes de type Multiobjectifs du fait que, dans un tel cas, il ne s'agit pas d'une seule solution optimale mais d'un ensemble de solutions bien réparties sur le front Pareto [12].

La recherche tabou a fait l'objet de bien des développements ces dernières années. Elle est utilisée pour traiter un grand nombre de problèmes d'optimisation

: coloriage de graphe, sac à dos, sac à dos multidimensionnel. La majorité des développements apportés ainsi qu'un large éventail d'applications de la recherche Tabou, se retrouvent dans [25]**.**

2.1.2.3. Recuit simulé

Le recuit simulé (Simulated Annealing (SA)) [26] s'inspire du processus de recuit physique. Le recuit simulé peut retenir une solution voisine de qualité moins bonne que celle de la solution courante et d'une notre manière Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût. . Mais, l'acceptation d'une telle solution se fait de manière stochastique et non pas déterministe. Pour ce faire, la méthode introduit une *probabilité d'acceptation* qui dépend de la qualité de la solution et d'un paramètre T appelé température. Le paramètre T (simulant le degré de température dans le processus du recuit métallurgique) est systématiquement ajusté pendant la recherche. Pour être clair, nous essayons de décrire le fonctionnement de l'algorithme du recuit simulé de la façon la plus générale et la plus simple possible [2].

Plaçons-nous dans le contexte d'un problème de type POMO. Désignons par *F* la fonction évaluant la qualité d'une solution donnée (elle correspond souvent à une agrégation linéaire des fonctions objectifs) et par *A* l'archive des solutions potentiellement non-dominées. Alors, le fonctionnement général d'un algorithme de recuit simulé multiobjectif est le suivant :

- Choisir aléatoirement une solution initiale x.
- Calculer une solution voisine de x; soit y.
- Calculer l'écart de qualité entre y et x:

$$\Delta F = \mathrm{F}(y) - F(x)$$

- Si $\Delta F \geq 0$, alors remplacer *x* par *y*. Sinon, pour accepter (ou ne pas accepter) *y*, utiliser la probabilité d'acceptation suivante:
- $P = e^{\frac{\Delta F}{T}}$
- Diminuer la température T au fur et a mesure du déroulement de l'algorithme.
- Mettre à jour $\mathbf{A}$ vis-à-vis de la nouvelle solution retenue.

- Répéter les étapes de 2 à 6 tant que la température *T* n'a pas atteint un seuil minimal fixé à l'avance.

Dans l'algorithme que nous présentons, le nombre d'itérations devant être atteint pour effectuer un changement de palier est fixe. Dans la pratique, les algorithmes de recuit simulé font varier ce paramètre en fonction de la température actuelle T. En effet, une diminution rapide de *T* peut entraîner une convergence prématurée de l'algorithme tandis qu'une diminution lente peut altérer la convergence de l'algorithme vers des solutions de bonne qualité. Voici quelques suggestions qui ont été proposées pour le réglage du paramètre T. La température T est souvent diminuée par paliers (diminuée de manière périodique en fixant un nombre d'itérations) au lieu d'être diminuée à chaque itération. La température initiale doit être suffisamment élevée afin de donner une chance équilibrée à toutes les solutions visitées durant le premier palier. On peut aussi envisager d'accentuer la diminution au fur et à mesure qu'on s'approche du seuil de température représentant le critère d'arrêt de l'algorithme. La combinaison de ces techniques permet de diversifier la recherche au début de l'algorithme en espérant atteindre plusieurs zones prometteuses qui vont être exploitées intensivement vers la fin de l'algorithme [2].

Une présentation détaillée sur les différentes variantes de l'algorithme de recuit simulé multiobjectif a été proposées sur différents problèmes de type POMO. Parmi ces variantes, on trouve l'algorithme PASA [27], l'algorithme PSA [28] et l'algorithme MOSA [29].

2.1.3. Métaheuristiques à base de population

2.1.3.1 Recherche par dispersion

La recherche par dispersion [24] est une métaheuristique dont le fonctionnement est relativement proche de celui des algorithmes évolutionnaires. Toutefois, Elle propose d'utiliser des conceptions stratégiques là où les algorithmes évolutionnaires utilisent des techniques stochastiques. Cette méthode se base essentiellement sur trois opérateurs dits de **dispersion, de** recombinaison et d'optimisation. Elle démarre par la génération d'une population initiale, de préférence, constituée de solutions de bonne qualité et bien dispersées dans l'espace de recherche. Puis, un autre ensemble de solutions, appelé ensemble de référence**,** est construit en sélectionnant les meilleures solutions (en

termes de qualité ainsi qu'en termes de répartition) de la population courante. Les éléments de l'ensemble de référence sont alors recombinés avant la phase d'optimisation. A noter que l'algorithme de recherche par dispersion utilise généralement un opérateur de recombinaison basé sur des combinaisons linéaires au lieu d'utiliser un opérateur de recombinaison stochastique comme c'est généralement le cas pour les algorithmes évolutionnaires. Ensuite, l'algorithme applique une technique d'optimisation (en général une recherche locale) sur les solutions générées par l'opérateur de recombinaison. Enfin, la population courante ainsi que l'ensemble de référence sont mis à jour à partir des solutions obtenues à l'issue de la phase d'optimisation. Ce procédé est réitéré tant que le critère d'arrêt (un nombre maximum d'itérations) n'est pas satisfait [2].

Quelques variantes de l'algorithme de recherche par dispersion peuvent être rencontrées dans la littérature. La variante la plus originale est la méthode du chemin de liaison [24].

La métaheuristique de recherche par dispersion est relativement peu utilisée dans la littérature, sans doute, à cause du fait qu'elle exige une connaissance approfondie de la structure du problème étudié pour pouvoir générer ,dès le départ, des solutions bonnes et représentatives de l'espace de recherche. Néanmoins, cette méthode a été adaptée pour résoudre des problèmes de type POMO [30] [31].

2.1.3.2. Colonies de fourmis

L'optimisation par colonies de fourmis (Ant Colony Optimization ou ACO) [32, 33] est inspirée du comportement des fourmis lors de la recherche de la nourriture. On peut résumer un tel comportement comme suit. En se déplaçant du nid à la source de nourriture et vice-versa, les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qui ont un rôle de marqueurs de chemin. En effet, les fourmis ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Or, comme les pistes de phéromones s'évaporent avec le temps, alors généralement la quantité de phéromones déposée sur le plus court chemin est davantage renforcée en comparaison avec des chemins qui sont plus longs (à condition qu'une partie, non

tout à fait négligeable, de fourmis ait initialement choisi le plus court chemin). Ainsi, il y a une grande chance pour que le plus court chemin soit choisi à terme par la majorité des fourmis [2].

Dans les travaux utilisant l'optimisation par colonie de fourmis (OCF) dans un contexte multiobjectifs :

- [34] ont présenté une approche qui favorise l'échange d'information entre des familles d'agents travaillant sur des objectifs différents pour la conception d'un réseau de distribution d'eau. Leur approche vise à générer l'ensemble Pareto optimal. [35] ont développé un OCF biobjectifs.
- Iredi, D. Merkle et M. Middendorf [36] proposent, pour leur part, une approche biobjectifs de l'OCF basée sur la coopération de plusieurs colonies pour un problème d'ordonnancement de machine unique avec réglages dépendant de la séquence. Chacune des colonies possède deux matrices de trace de phéromone utilisées de manière différente par chacune des fourmis.
- En 2006, [37] ont propose un algorithme nommé P-ACO (Pareto Ant Colony Optimization) dédie à la résolution du problème d'allocation de portefeuilles. Cet algorithme présente de bons résultats, notamment face au recuit simulé.

2.1.3.3. Essaims particulaires

L'optimisation par essaims particulaires (Particle Swarm Optimization ou PSO) [38] est issue d'une analogie avec les comportements collectifs de déplacements d'animaux. En effet, chez certains groupes d'animaux comme les bancs de poissons, on peut observer des dynamiques de déplacements relativement complexes alors que les individus eux-mêmes n'ont accès qu'à des informations limitées (la position et la vitesse de leurs plus proches voisins). Les concepteurs de la méthode d'optimisation par essaims particulaires se sont inspirés de ces comportements tout en les enrichissant de quelques hypothèses de la socio-psychologie. En fait, la méthode en elle-même met en jeu des groupes de particules (appelés essaims particulaires) sous forme de vecteurs se déplaçant dans l'espace de recherche. Ainsi, chaque particule est caractérisée par sa *position* et un vecteur de changement de position (appelé vitesse). A chaque

itération, la particule fait un nouveau déplacement. La socio-psychologie suggère que pendant son déplacement, un individu est influencé par son comportement passé ainsi que par celui de ses voisins. Ainsi, pendant la mise à jour de la position de chaque particule, on tient compte de la direction de son mouvement, sa vitesse, sa meilleure position précédente et la meilleure position de ses voisines (voir Figure 2.3). Ici, la mémoire n'est structurée qu'au niveau local (entre particules voisines). En d'autres termes, à chaque itération, chaque particule n'évolue qu'en fonction de ses proches voisines et non pas selon l'état global du système comme c'est le cas pour les algorithmes de colonies de fourmis. En résumé plus formel, supposons que l'espace de recherche est de dimension n. La position courante d'une particule dans cet espace est notée x. Sa vitesse courante est notée v. La meilleure position trouvée jusqu'ici par cette particule est notée p. La meilleure position parmi celles trouvées jusqu'ici par ses proches voisines est notée g. La composante k de l'un de ces vecteurs est indiquée par l'indice k. Avec ces notations, les équations de déplacement (pour chaque composante k) d'une particule sont les suivantes:

$$v_k \leftarrow c_1 v_k + c_2(p_k - x_k) + c_3(g_k - x_k) \qquad (1)$$

$$x_k \leftarrow x_k + v_k \qquad (2)$$

Dans l'équation (1), les coefficients c_i (pour i = 1, 2, 3) sont des paramètres de l'algorithme. Leur choix ne doit pas être fait de manière indépendante l'un de l'autre. De plus, le choix de ces coefficients peut être fait de manière stochastique à partir d'un ensemble de valeurs qu'on peut générer expérimentalement via des tests préliminaires. Le réglage de ces paramètres constitue ainsi la difficulté principale posée par un algorithme de type PSO.

Néanmoins, on peut trouver dans la littérature quelques techniques visant à surmonter cette difficulté.

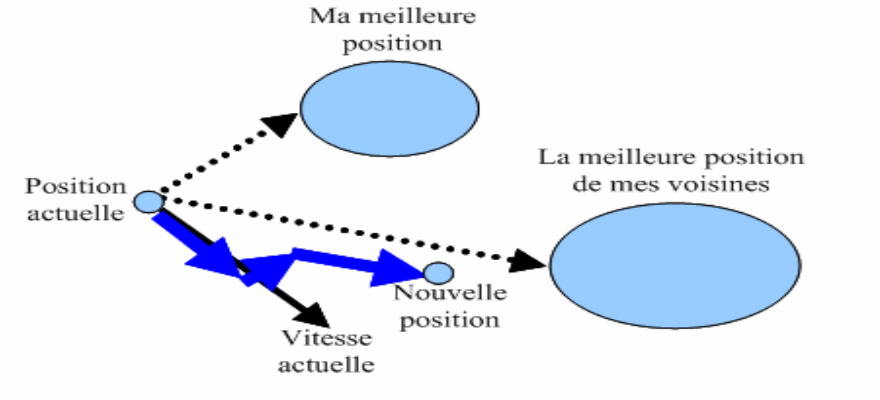


Figure 2.3 Principe de mise à jour de la position d'une particule.

Pour réaliser son prochain mouvement, chaque particule combine trois tendances : suivre sa vitesse propre, revenir vers sa meilleure performance, aller vers la meilleure performance de ses voisines.

A partir de 2002 plusieurs auteurs ont proposé simultanément plusieurs approches pour utiliser l'OEP pour l'optimisation multiobjectif. Toutes les variantes multiobjectifs des OEP sont appelées MOPSO (multiobjective particle swarm optimisation) même si les approches différent.

Le principal problème qui se pose est de savoir comment définir pbest et lbest pour chaque particule. Il faut en effet éviter que les particules ne convergent vers un seul optimum afin de trouver un front de Pareto le plus complet possible. Toutes les approches gardent une archive de toutes les particules trouvées qui ne sont pas dominées (approches élitistes). A chaque itération l'archive des particules non dominées est mise à jour et pour chaque particule il est nécessaire de définir pbest et lbest pour recalculer le vecteur vitesse [12].Pour avoir une idée détaillée sur l'MOPSO, le lecteur intéressé peut consulter les références suivantes [39], [40] et [41].

2.1.4 Autres métaheuristiques

Les métaheuristiques présentées précédemment sont les plus utilisées. Néanmoins, il existe dans la littérature d'autres métaheuristiques qui sont relativement moins utilisées. Parmi ces métaheuristiques, on peut citer les algorithmes à estimation de distribution (Estimation of Distribution Algorithms) [42] qui consistent à contrôler la probabilité de distribution des meilleurs individus dans

la population courante tout en utilisant une technique pour améliorer la qualité de ces individus. Nous pouvons citer aussi la métaheuristique des systèmes immunitaires (Immune Systems) [43], basée sur une analogie avec le fonctionnement du système immunitaire des êtres vivants [2].

D'autre part, il existe un grand nombre de métaheuristiques dites hybrides (Hybrid Metaheuristics). Une métaheuristique hybride consiste à combiner deux métaheuristiques ou plus de type différent pour en former une nouvelle. Le schéma d'hybridation la plus fréquente est celle concernant les algorithmes dits *mimétiques* (Memetic Algorithms) [44]. Ce schéma consiste à hybrider un algorithme évolutionnaire et une méthode de voisinage (une recherche locale ou une recherche Tabou) [2].

D'autres schémas d'hybridation peuvent être rencontrés dans la littérature. Dans ce sens, on peut citer le schéma consistant à hybrider une (ou plusieurs) métaheuristique(s) avec une (ou plusieurs) méthode(s) exacte(s). Les algorithmes qui résultent de ce schéma sont dits algorithmes coopératifs (Cooperative Algorithms). Récemment, les méthodes coopératives commencent à susciter un intérêt croissant lors du traitement de plusieurs problèmes de type POMO [45]. Un tel intérêt est justifié par la croissance continuelle de la puissance des ordinateurs actuels [2].

2.2 Algorithmes évolutionnaires(EAS)

2.2.1. Historique

La deuxième classe de métaheuristiques présentée dans cette thèse est celle des algorithmes évolutionnaires [46]. Le terme algorithme évolutionnaire (EA) porte sur des méthodes d'optimisation stochastiques qui simulent le processus de l'évolution naturelle. Les origines des algorithmes évolutionnaires peuvent être rendu aux années 1950, et depuis les années 1970 beaucoup de méthodologies évolutionnaires ont été proposées :

En 1859 Charles Darwin publie son livre intitulé < Origin of species >. Dans ce livre,

Darwin expose sa théorie de l'évolution des espèces sous l'influence de contraintes extérieures (environnement), les êtres vivants se sont 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 [12].

Historiquement, les premiers algorithmes évolutionnaires ont été développés depuis les années soixante. Toutefois, l'application intensive des EAs n'a commencé qu'à partir des années 90 avec l'apparition d'ordinateurs informatiques suffisamment puissants. En général, les EAs se divisent en trois classes dont la description et l'historique sont les suivants :

- Algorithmes Génétiques (GA)

Les algorithmes génétiques (Genetic Algorithms ou GAs) sont probablement les plus connus et utilisés parmi les EAs. Ils ont été développés dans les années 60 par Holland [48]. L'idée de son système était d'étudier, dans un contexte psycho-biologique, le processus d'adaptation des populations à la base des données sensorielles introduites au système grâce à des détecteurs binaires [49].

Les GAs ont été appliqués à l'optimisation paramétrique pour la première fois en 1975 par De Jong [50], qui a posé les fondements d'application de ces algorithmes. Cependant, le manque de puissance des ordinateurs à l'époque ne permettait pas leur application à des problèmes réels de grande taille. Leur application restait alors très limitée, pendant un certain temps, jusqu'à ce qu'ils aient été rendus populaires grâce à l'ouvrage de référence écrit par Goldberg [51].

- Stratégies d'évolution (ESs)

Les stratégies d'évolution (Evolution Stratégies ou ESs) ont été initialement dédiées à la résolution des problèmes d'optimisation continus. Les premiers efforts pour la mise en place des ESs ont eu lieu en 1964 dans l'université de Berlin au cours de la résolution d'un problème aérodynamique. En 1965, Rechenberg [52] a introduit l'algorithme (1+1)-ES qui fait évoluer un seul individu en utilisant un opérateur de mutation, appelé mutation Gaussienne, spécifique au cas continu. Cette stratégie a servi de base pour la transition aux stratégies (μ + λ)-ES et (μ, λ)-ES qui ont été introduites par Schwefel en 1977 (voir [53] pour plus de détails sur l'historique et les fondements des stratégies d'évolution) [2].

-Programmation génétique (GP)

La première utilisation des structures arborescentes dans un algorithme génétique a été suggérée par Cramer en 1985 [53], dans le but de faire évoluer des sous-programmes séquentiels d'un langage algorithmique simple. L'adoption de ce schéma pour définir la programmation génétique (Genetic Programming ou GP) comme une nouvelle classe des EAs a été proposée par Koza en 1992. Son objectif initial était de faire évoluer des sous-programmes du langage LISP. Grâce à l'ouvrage de Koza [54], l'utilisation de la GP s'est étendue à la résolution de nombreux types de problèmes dont les solutions peuvent être représentées par des structures arborescentes; comme l'architecture orientée-objet [55] et les représentations fonctionnelles linéaires [56].

2.2.2. Vocabulaire et principe de fonctionnement

Avant d'expliquer en détail le fonctionnement d'un algorithme évolutionnaires, nous allons présenter quelques mots de vocabulaire relatifs à la génétique. Ces mots sont souvent utilisés pour décrire un algorithme génétique :

- Individu : un élément de l'espace de recherche.
- gène : Codage associé à chaque variable.
- Population : un ensemble fini d'individus, de taille N.
- Evolution : un processus itératif de recherche d'un (ou plusieurs) individu optimal.
- Performance (fitness function ou fonction d'adaptation) : Cette fonction est déterminée en fonction du problème à résoudre. A un individu particulier, elle attribue une valeur numérique. L'évaluation de ce dernier ne dépendant pas de celle des autres individus, le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu, ce qui permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

- Croisement : l'opérateur de reproduction appliqué avec la probabilité pc et correspondant à un brassage d'information entre les individus de la population. Il consiste à échanger des composantes entre deux individus.
- Mutation : l'opérateur de modification d'un ou plusieurs composantes appliqué avec la probabilité pm dans le but d'introduire une nouvelle variabilité dans la population.
- Sélection : processus du choix des individus pour la reproduction basé sur leur Performance
- Remplacement : processus de formation d'une nouvelle population à partir des ensembles des parents et d'enfants effectué le plus souvent sur la base de leur performance.

- Principe de fonctionnement

Un EA nécessite en premier le *codage* de chaque solution du problème étudié en une chaîne de longueur finie appelée individu (ou chromosome). Le principe général d'un EA consiste à simuler l'évolution d'une population d'individus jusqu'à atteindre un critère d'arrêt. On commence par générer une population initiale d'individus représentant des solutions possibles au problème. En général, la population initiale est générée de façon aléatoire. Puis, à chaque génération, des individus de la population courante sont sélectionnés selon un critère d'évaluation. Les individus sélectionnés constituent une population dite population de parents. Ensuite, les opérateurs de recombinaison *et* de mutation sont respectivement, appliqués sur la population de parents pour générer une autre population dite population d'*enfants*. Enfin, un opérateur dit de remplacement est appliqué pour remplacer la population courante, en faisant intervenir la population d'enfants et éventuellement celle des parents. Ce processus constitue une génération d'un EA. Sa répétition se poursuit tant qu'un critère d'arrêt, fixé à l'avance, n'est pas atteint. En général, le critère d'arrêt correspond à un nombre maximum de générations

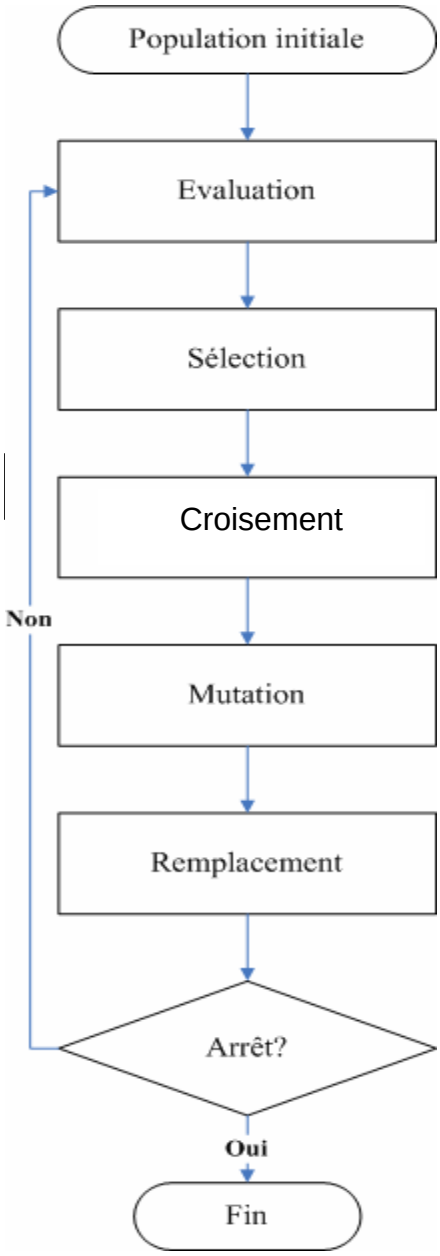


Figure 2.4 Organigramme d'un algorithme évolutionnaire [2].

2.2.3. Opérateurs d'un algorithme évolutionnaire

Trois opérateurs caractérisent les algorithmes évolutionnaires et rappellent l'origine de ces méthodes, ils vont permettre à la population d'évoluer, par la création d'individus nouveaux construits à l'aide des individus anciens. Plus précisément, on prélève, dans certains individus de la population courante, une partie de leurs caractéristiques en choisissant certaines parties des chromosomes

qui les représentent ; puis on recombine ces différentes parties pour constituer les individus de la nouvelle population.

La phase de sélection indique dans quelles configurations de la population courante on va prélever des morceaux de chromosomes ; la phase ***de croisement*** prélève ces morceaux de chromosomes et les recombine pour former les configurations de la population suivante ; la phase **de mutation** s'applique à la nouvelle population en changeant éventuellement certains gènes de certains chromosomes obtenus à la fin de la phase de croisement. Une succession des trois opérations de sélection, de croisement et de mutation constitue une génération, et les algorithmes évolutionnaire consistent donc à faire évoluer une population initiale pendant un certain nombre de générations, nombre déterminé par l'utilisateur.

2.2.3.1. Le codage

Le choix du codage est important est souvent délicat. L'objectif est bien sûr d'abord de pouvoir coder n'importe quelle solution. Mais il est souhaitable, au-delà de cette exigence, d'imaginer soit un codage tel que toute chaine de caractères représente bien une solution réalisable du problème. En pratique, sauf le codage binaire et le codage réel sont généralement employés [12].

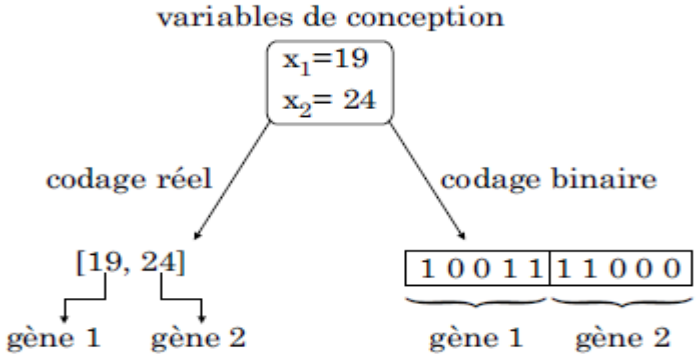


Figure 2.5 Exemple d'un codage réel et binaire [12].

2.2.3.2. Evaluation de la population

Cette étape consiste à évaluer chaque solution contenue dans la population, la mesure de performance des solutions s'appuie sur la valeur de fonctions objectifs. Chaque solution se voit attribuer une fonction d'adaptation qui traduit sa performance par rapport aux autres solutions, et qui dépend de l'estimation de

chaque objectif du problème étudie. Cette étape permet de classer les solutions, afin de déterminer les solutions qui seront sélectionnées pour construire une nouvelle population de solutions [12].

2.2.3.3. Sélection

L'étape de sélection est l'un des principaux opérateurs de recherche de l'algorithme évolutionnaires. Elle consiste à choisir des solutions de la population courante qui seront utilisées pour générer la population Enfant. En général, meilleure est une solution, plus élevé sera sa chance d'être sélectionnée. Les stratégies classiques et les plus utilisés sont la sélection par *tournoi* (tournament Selection) et la sélection à la roulette (Wheel) [12].

- La sélection à la roulette (Wheel)

C'est le plus ancien opérateur de sélection, il a été développé par Goldberg [51]. Cet opérateur consiste à attribuer à chaque individu une probabilité de sélection proportionnelle à sa *valeur* d'*adaptation* obtenue par la fonction d'adaptation F à partir de population E alors la probabilité $\bar{P}$ pour qu'un individu x soit sélectionné est donnée par la formule suivante [2]:

$$\bar{P} = \frac{F(x)}{\sum_{y \in E} F(y)}$$

C'est une sorte de roulette de casino sur laquelle sont placés tous les individus de la population en cour, la place accordée à chacun des individus étant proportionnelle à sa valeur d'adaptation. Ainsi, les individus ayant un fitness grand ont plus de chance d'être sélectionnés pour la génération suivante. Ensuite, la bille est lancée et s'arrête sur un individu (voir la figure suivant où l'individu *y* a plus de chance d'être sélectionné après un tirage à la roulette). Cela peut être simulé par l'algorithme suivant :

- On calcule la somme S de toutes les fonctions d'évaluation d'une population.
- On génère un nombre r entre 0 et S.
- On calcule ensuite une somme S' des évaluations en s'arrêtant dés que *r* est dépassé.

- Le dernier chromosome dont la fonction d'évaluation vient d'être ajoutée est sélectionné.

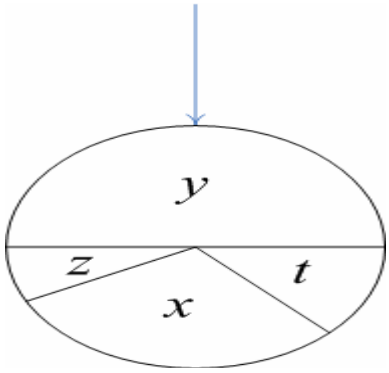


Figure 2.6 Principe de la sélection à la roulette

L'inconvénient majeur de ce type de reproduction vient du fait qu'il peut favorisé la domination d'un individu ce qui engendre une perte de diversité car les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'évolution [12].

- La sélection par tournoi

La sélection par tournoi [57] consiste à choisir aléatoirement k individus et à les confronter entre eux par le biais de la fonction d'adaptation, et de sélectionner ensuite le meilleur parmi eux. Cette opération est répétée autant de fois que nécessaire afin de sélectionner le nombre d'individus désiré. Il est ainsi possible que certains individus participent à plusieurs tournois. Si de tels individus gagnent plusieurs fois, ils seront sélectionnés plusieurs fois, ce qui favorisera la pérennité de leurs gènes. L'avantage de la sélection par tournoi en comparaison avec la sélection à la roulette réside dans le fait qu'elle est tout à fait paramétrable par k (le nombre de participants à un tournoi). En d'autres termes, la pression sélective est parfaitement contrôlable via le paramètre k. En effet, pourvu que la valeur de k ne soit pas très élevée, cette technique permet de garder la diversité génétique de la population tout en favorisant les individus les mieux adaptés. Pour k = 2, on parle de sélection par tournoi binaire [2].

2.2.3.4. Croisement

L'opérateur de croisement (ou recombinaison) permet la création de nouveaux individus selon un processus de sélection. Ainsi, les individus sélectionnés sont répartis aléatoirement en couples de parents. Puis, chaque couple de parents subit une opération de recombinaison avec une probabilité donnée (la probabilité de recombinaison, souvent supérieur à 60%) afin de générer un ou deux enfants. L'opérateur de recombinaison favorise l'exploitation de l'espace de recherche en examinant continuellement les recombinaisons exercées sur les parents. En effet, cet opérateur permet d'accumuler les gènes favorables en mélangeant systématiquement le matériel génétique des individus prometteurs. Cette accumulation permet alors de créer de nouvelles combinaisons de gènes qui peuvent se révéler potentiellement favorables. Les opérateurs de recombinaison les plus fréquents dans la littérature sont: le croisement n-point ou multi-point (Multi-Point Crossover) et le croisement uniforme (Uniform Crossover) [2].

- Le croissement n-point

Le croisement *n*-point [58] consiste à choisir d'abord n points de coupure pour chaque couple de parents, puis à échanger alternativement les fragments délimités par ces points de coupure afin d'obtenir un nouveau couple d'enfants. Les deux versions les plus utilisées sont: le croisement 1-point et le croisement 2-point [59]. Dans le croisement 1-point, un seul fragment est échangé selon un point de coupure choisi aléatoirement. La figure 2.7 (a) illustre cette opération. Quant au croisement 2-point, il consiste à échanger deux fragments selon deux points de coupure choisis aléatoirement (la figure 2.7 (b)).

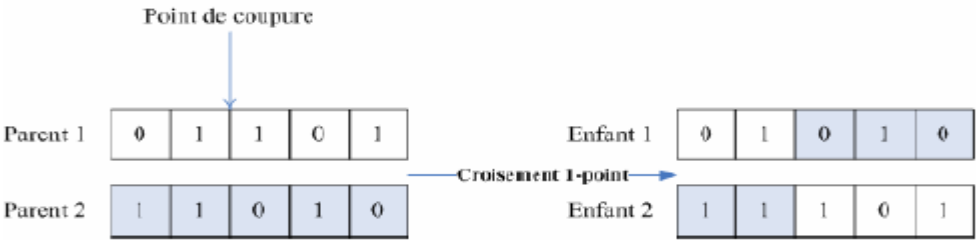


Figure 2.7 (a) Croisement 1-point.

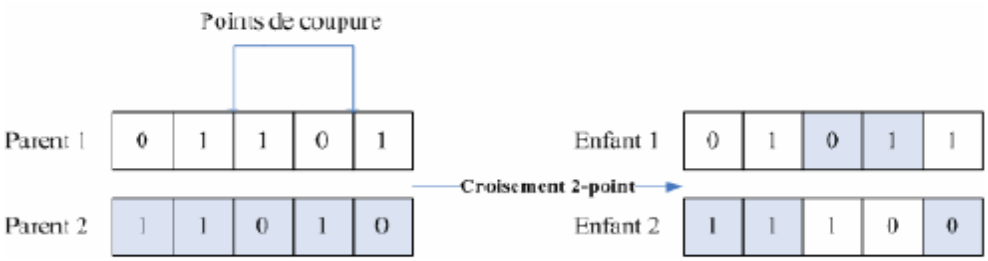


Figure 2.7 (b) Croisement 2-point.

- Le croisement uniforme

Le croisement uniforme [60] consiste à générer un enfant en échangeant chaque gène des deux individus parents avec une probabilité uniforme égale à 0.5 (Figure 2.8). Cet opérateur peut être vu comme le cas extrême du croisement multi-point dans lequel le nombre de points de coupure est déterminé aléatoirement au cours de l'opération.

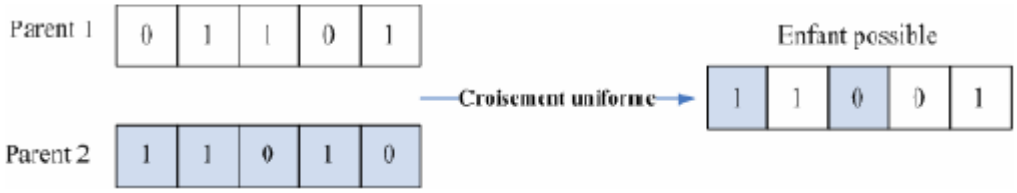


Figure 2.8 Croisement uniforme.

2.2.3.5. Mutation

L'opérateur de mutation consiste à sélectionner aléatoirement un (ou plusieurs) gène(s) dans un individu et à modifier sa (leurs) valeur(s). L'opérateur de mutation apporte aux EAs la possibilité de recouvrir la totalité de l'espace de recherche [51] [58]. Les types de mutation les plus utilisés sont: la mutation 1-bit (1-Bit Mutation) et la mutation stochastique (Bit-Flip Mutation).

- La mutation 1-bit

La mutation 1-bit inverse la valeur d'un seul bit (de 1 à 0 ou de 0 à 1) choisi au hasard avec une certaine probabilité p_m faible dans un environnement stable afin de favoriser l'accumulation des gènes favorables (obtenus par croissement) tout en permettant d'élargir l'espace des solutions évaluées. Cependant, dans un environnement dynamique

subissant une évolution rapide, il peut être judicieux d'utiliser un taux de mutation p_m relativement élevé pour permettre à la population de s'adapter rapidement à son environnement.

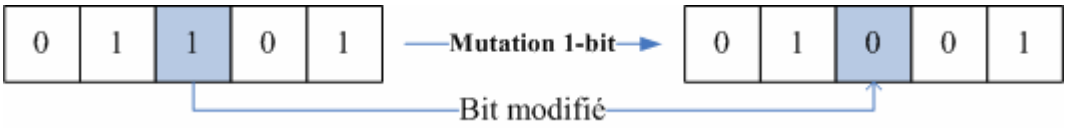


Figure 2.9 Mutation 1-bit.

- La mutation stochastique

  La mutation stochastique peut être vue comme une généralisation du principe de la mutation 1-bit. En effet, cet opérateur consiste à inverser la valeur de chaque bit (indépendamment des autres bits) avec une probabilité uniforme:

$$p_m = {}^{c}/_{l}$$

où c: est une constante positive (généralement, c =1) et l: est la longueur de la chaîne représentant un individu (le nombre de bits dans un individu). Cette technique a l'avantage de distribuer uniformément la probabilité de mutation sur tous les gènes (bits).

2.2.3.6. Remplacement

Après la phase de reproduction (croissement et mutation), on est en possession de deux populations intermédiaires: la population de parents et celle d'enfants. L'opérateur de remplacement consiste à remplacer la population courante par une nouvelle population. La population est remplacée soit par la totalité de ses descendants (Algorithme génétique générationnel) ou par une partie seulement de ceux-ci (algorithme stationnaire). Néanmoins, de génération en génération, la taille la population doit rester constante [12].

Avant de terminer cette section dans laquelle nous avons abordé les différentes étapes d'un algorithme génétique, on doit noter que les opérateurs reproduction dépendent de plusieurs paramètres qui sont fixée à l'avance et dont dépend fortement la performance de l'algorithme. Les paramètres de base sont les suivants :

- Taille de la population : C'est le nombre d'individus de la population.
  Une taille trop petite peut entrainer une convergence prématurée de l'algorithme à cause de la perte de la diversité génétique au sein de la population au fil des générations, et notamment lorsque l'algorithme ne contient pas de mécanisme de diversification. Si, au contraire, la taille est trop grande, alors l'algorithme peut demander un temps d'exécution très long à cause de l'évaluation itérative de tous les individus de la population. Une population de taille modérée (en fonction de l'instance du problème étudie) est donc préférable [61].

- Probabilité de recombinaison : c'est la probabilité pour qu'un couple d'individus parents subisse l'opérateur de recombinaison. En général, la probabilité de recombinaison est élevée (p_c Є [0.7, 0.9]) dans l'espérance de générer plusieurs individus enfants éventuellement différents de leurs parents mais ayant les bonnes caractéristiques communes de ceux-ci.
- Probabilité de mutation : elle peut être définie de deux farçons. Soit la probabilité pour qu'un individu subit l'opérateur de mutation, soit la probabilité pour qu'un glène d'un individu (c'est une composante de la chaîne représentant cet individu) subit une mutation. La probabilité de mutation est en général très faible, inférieure à 0,1 dans la plupart du temps. En effet, une probabilité de mutation élevée peut modifier radicalement les meilleurs individus en entraînant, éventuellement, une dégradation de leur qualité.

- Nombre maximum de générations : ce paramètre correspond au critère d'arrêt usuel pour les EAs. Il doit être fixé convenablement. Si ce nombre est trop petit, les résultats peuvent être médiocres pour un algorithme standard qui n'utilise pas de mécanismes l'accélération de la convergence. Par contre, un nombre de générations très grand peut nécessiter un temps de calcul très élevée sans garantir l'amélioration des résultats.

2.2.4. Algorithmes évolutionnaires multiobjectifs (MOEAs)

Les algorithmes évolutionnaires sont très bien adaptés au traitement d'un problème d'optimisation multiobjectif. En témoigne le nombre important d'articles qui ont été publiés sur ce sujet. De plus, ce domaine est très dynamique et ne cesse de se développer. Les MOEAs ont été largement utilisés pour la résolution Pareto de POMO, étant donné qu'ils travaillent sur une population de solutions [12]. Deux objectifs doivent être pris en compte dans la résolution d'un POMO :

- Converger vers la frontière Pareto : la plupart des travaux de recherche sur l'application des Algorithmes génétiques aux POMO se sont concentrés sur l'étape de sélection. Dans cette étape, des méthodes de ranking sont appliques 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 [12].

- Diversifiées les solutions Pareto optimales : il s'agit d'assurer une diversification des solutions sur la frontière de Pareto à l'aide des méthodes de maintien de la diversité comme la technique de niches écologiques qui regroupent les solutions les plus voisines. Cette procédure permet de stabiliser des sous-populations multiples le long de la frontière Pareto [12].

Les algorithmes évolutionnaire multiobjectifs offrent, par le biais de la notion de population, des mécanismes pertinents sont exploités pour l'approximation de la frontière Pareto. Ces mécanismes sont présentés dans la section suivante :

2.2.4.1. Mécanismes de préservation de la diversité

Pour les algorithmes évolutionnaires vus précédemment, l'opérateur de sélection duplique, de génération en génération, les individus les mieux adaptés et conduit inévitablement à un phénomène de dérive génétique : à terme, tous les

individus de la population sont identiques et si l'algorithme a converge prématurément vers une solution, les possibilités d'évolution s'en trouvent réduites, car des opérateurs tels que les croisements sont alors incapables de faire apparaître de nouvelles solutions. En effet, l'usage d'une technique appropriée de préservation de la diversité permet non seulement d'éviter une convergence prématurée vers des optima locaux, mais aussi d'identifier plusieurs bonnes solutions représentatives de l'espace de recherche [12].

1. Fonction de partage (fitness sharing)

La technique de partage consiste à modifier la valeur de coût d'un individu (calculée uniquement à partir de la fonction objectif du problème). C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection. Cette technique, introduite dans [13], est largement utilisée aujourd'hui. Pour éviter qu'un trop grand nombre d'individus ne se concentrent autour d'un même point, il faut pénaliser la valeur d'adaptation en fonction du nombre d'individus au voisinage du regroupement : plus les individus sont regroupés, plus leur valeur d'adaptation est faible, et des individus proches les uns des autres doivent partager leur valeur d'adaptation. Dans la pratique, on estime ce taux de concentration en ouvrant un domaine autour d'un individu, puis on calcule les distances entre les individus contenus dans ce domaine. Pour déterminer les bornes du domaine ouvert autour de l'individu choisi, on définit une distance maximale, appelée ρ_{sh} au delà de laquelle les individus ne seront plus considères comme faisant parti du domaine ouvert.la fonction d'adaptation de chaque individu (i) est dégradée par un compteur de niche m_i, calculée pour ce même individu. La nouvelle fonction de partage calculée shared fitness f^*est obtenue en divisant la fonction d'adaptation f de l'individu par le compteur de niche.

$$.f_i^* = \frac{f_i}{m_i}$$

Le compteur de niche $m_i$donne une estimation du nombre d'individus qui se trouvent autour d'un individu (i)

$$m_i = \sum_{j \in P} sh(d[i,j])$$

où $d[i,j]$ est la distance entre l'individu (i) et (j) et Sh est la fonction de partage, elle est calculée de sorte que les individus appartenant à la même niche (appartenant au même voisinage) partagent la même valeur. Il existe plusieurs fonctions de partage.

La fonction de partage classique suggérée dans [13] est :

$$sh(d[i,j]) = \begin{cases} 1 - \left(\frac{d[i,j]}{\rho_{sh}}\right)^{\alpha} & si \quad d[i,j] < \rho_{sh} \\ 0 & sinon \end{cases}$$

où α et ρ_{sh} sont des constantes. La constante ρ_{sh} spécifie le seuil de dissimilarité (taille des niches). La constante α est généralement initialisée à 2, permet de réguler la forme de la fonction de partage.

La fonction de partage sh dépend de la distance entre deux individus de la population. Elle retourne 1 si les deux individus sont identiques, 0 s'ils sont différents (à partir d'un seuil donné).

Le problème qui peut être posé lors de l'application de la technique du partage est comment ajuster correctement le paramètre ρ_{sh}. En effet, une valeur trop élevée de ρ_{sh} peut rendre la technique inutile puisque la plupart des individus vont alors appartenir à la même niche. Si, au contraire, la valeur de ρ_{sh} est trop faible, alors il y a un risque pour que des individus isolés mais ayant probablement une faible valeur de f seront mieux évalués que des individus appartenant à des niches denses mais ayant probablement une bonne valeur de Dans ce cas, l'algorithme peut ne pas converger vers des solutions de bonne qualité. On verra ultérieurement un autre mécanisme de diversification appelé *crowding* (Crowding) qui a un principe proche de celui du partage mais qui évite le paramètre ρ_{sh} en introduisant une technique alternative [2].

2. Réinitialisation

La réinitialisation est une technique de diversification dont le principe est très simple. C'est, sans doute, pour cette raison qu'elle demeure largement utilisée par la grande majorité des métaheuristiques dont les EAs. Cette technique consiste à réinitialiser, de temps en temps (périodiquement), la population (ou

une partie de la population), par exemple de façon aléatoire. En effet, en introduisant de manière régulière certains individus générés aléatoirement, on peut explorer de nouvelles zones de l'espace de recherche.

2.2.4.2. Un mécanisme de sélection

Un des principaux avantages des algorithmes évolutionnaires pour l'optimisation multiobjectif est qu'ils permettent non seulement la mise en œuvre d'approches non Pareto (agrégation des objectifs,…), mais aussi l'implémentation d'approches Pareto. En effet, certains mécanismes de sélection implémentent la relation de dominance réalisant ainsi une sélection Pareto [65].

La sélection Pareto utilise la relation de dominance pour affecter des rangs aux individus de la population, faisant apparaître la notion de front. Citons par exemple la technique de ranking qui a été initialement proposée par Goldberg [51] et implémentée dans l'algorithme NSGA (Non Dominated Sorting Genetic Algorithm) [14].

Initialement, tous les individus non dominés de la population reçoivent le rang 1 et sont retirées temporairement de la population. Puis, les nouveaux individus non dominés reçoivent le rang 2 avant d'être à leur tour retirée. Le processus s'itère tant qu'il reste des individus dans la population. La valeur d'adaptation de chaque individu correspond à son rang dans la population. Ainsi, l'évaluation d'un individu ne dépend pas uniquement de lui même, mais aussi de la population (figure ci-dessous) [65].

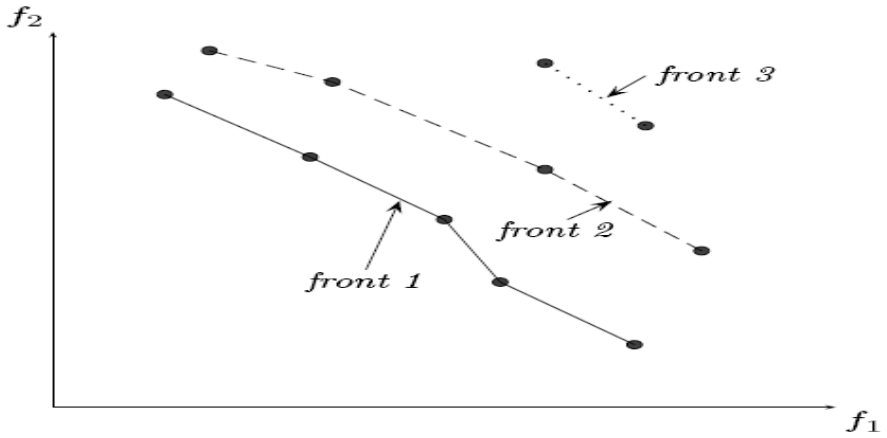


Figure 2.10 Ranking basé sur la sélection Pareto [65].

2.2.4.3. L'élitisme

L'élitisme permet de conserver les meilleurs individus dans les générations futures. Une des premières implémentations de ce mécanisme dans un algorithme génétique est présentée dans [50]. L'élitisme est introduit pour conserver les bonnes solutions lors du passage de la génération courante à la prochaine génération. Conserver ces solutions pour les générations futures permet d'améliorer les performances des algorithmes sur certains problèmes. Réaliser un algorithme élitiste dans le cadre des problèmes multiobjectifs est plus difficile que pour les problèmes à un objectif. En effet, la meilleure solution n'est plus un unique individu, mais tout un ensemble dont la taille peut aller jusqu'à dépasser la taille maximale de la population [65].

Les approches récentes [11,63] tendent à utiliser une population externe d'individus dans laquelle est stocké le meilleur ensemble des points non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et les individus stockés continus à pouvoir être choisis par l'opérateur de sélection. Ils peuvent ainsi se reproduire et transmettre leurs caractéristiques aux générations suivantes.
Actuellement, les algorithmes élitistes obtiennent de meilleurs résultats sur un grand nombre de problèmes multiobjectifs [64] et [65].

2.2.5. Quelques algorithmes évolutionnaires multiobjectifs (MOEAs)

Dans cette section, on va présenter quelques algorithmes évolutionnaires multiobjectifs de l'état de l'art pour avoir une idée claire sur les différentes techniques utilisées par de tels algorithmes. Au cours de cette présentation, nous essayons de représenter toutes les catégories issues de la classification donnée précédemment tout en mettant l'accent sur certains MOEAs performants et fréquemment utilisés pour le traitement de problèmes de type POMO [2].

2.2.5.1. Algorithme basés sur la dominance de Pareto

1-MOGA (Multiple Objective Genetic Algorithm)

Fonseca et Fleming [66] ont été les premiers à proposer l'algorithme qui utilise la notion de dominance directement pour évaluer la performance des individus. Ils proposent de déterminer pour chaque solution i, le nombre n_i de solutions la dominant. Le rang r_i d'une solution x_i à une génération t est donné par :

$$r_i(x_i, t) = n_i + 1$$

Ainsi le rang d'une solution non dominée est égal à 1. La fonction de fitness dépend du rang de chaque solution et le rang maximal ne peut pas être plus grand que la taille de la population N.

La fonction de fitness dépend du rang de chaque solution. Pour des solutions de même rang, la fonction Fitness est identique. Cependant, cette procédure ne permet pas d'avoir une diversité des solutions de la frontière Pareto. Fonseca et Fleming, proposent l'introduction d'une fonction permettant de construire des niches ou des classes de solutions de même rang. Le remplissage de ces niches s'appuie sur la distance entre les solutions de même rang. Ainsi si la distance euclidienne entre deux solutions de même rang est inférieure à une valeur donnée notée σ_{share}, les deux solutions appartiendront à la même niche. Pour cette méthode la distance est définie dans l'espace phénotypique (Espace des objectifs). Soit deux solution de même rang *i* et, la distance est donnée comme suit :

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^i - f_k^j}{f_k^{max} - f_k^{min}} \right)^2}$$

avec M représente le nombre d'objectifs considérés dans le problème f_k^i, la valeur de la fonction objectif *k* et f_k^{max}, f_k^{min} , les valeurs maximale et minimale de la fonction objectif *k* pour la population considérée. Pour les solutions de même rang, la fonction de partage $Sh(d_{ij})$ est définie par :

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\beta} \\ 0 \quad sinon \end{cases}$$

Le compteur de niche m_i traduit le nombre de solutions de même rang dont la distance les séparant est inférieure à σ_{share} avec σ_{share} β =*1*. Soit n_{ri} le nombre de solutions de même rang.

$$m_i = \sum_{j=1}^{n_{ri}} sh(d_{ij})$$

Afin d'assurer une distribution des solutions sur la frontière de Pareto, la valeur du Fitness F est corrigée à l'aide du compteur de niche. La nouvelle fonction de fitness, $F_i^{'}$ tel que :

$$F_i^{'} = \frac{F_i}{m_i}$$

Les valeurs de la performance partagée $F^{'}$ ne sont plus les mêmes pour les individus du même rang ; les solutions situées dans des régions plus éparses ayant une meilleure performance. Cela se traduit par une forte pression de sélection pour telles solutions.

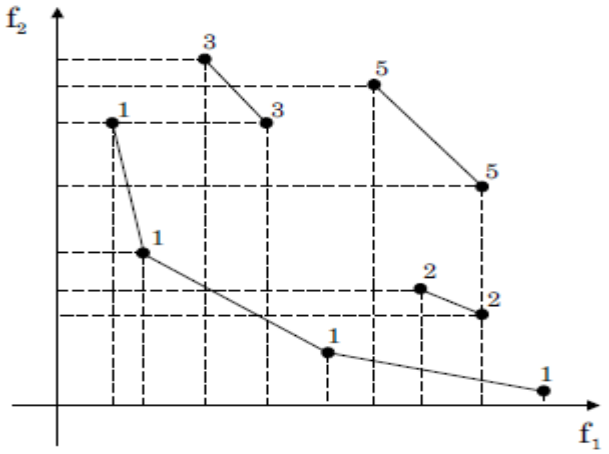


Figure 2.11 Le mécanisme de sélection de MOGA.

Discussion de MOGA

La façon de calculer la performance est simple dans MOGA. Mais bien que le concept de dominance soit utilisé pour calculer le rang d'une solution, différentes solutions appartenant au même front Pareto (à part le premier) n'ont pas nécessairement la même performance. Ceci peut impliquer quelques biais indésirables vers certaines solutions dans l'espace de recherche. En particulier, l'algorithme peut être sensible à l'allure de la frontière de Pareto ou à la densité des solutions sur cette frontière [67].

La procédure du partage n'assure pas que les solutions d'un rang inférieur autant une moins bonne valeur de performance partagéeF'que les solutions d'un rang meilleur. Cela peut se produire en particulier s'il y a un nuage très dense de solutions. De plus le réglage du paramètre de partage σ_{share} influence la performance de l'algorithme.

2. NSGA (Non Dominated Sorting Genetic Algorithm)

L'idée de Goldberg d'utiliser le Concept de ranking par dominance dans les AG [14] a été plus directement mise en œuvre par Srivnas et Deb en 1994 dans leur méthode NSGA- Non Dominated Sorting Genetic Algorithm**.** L'approche à double objectif convergence-diversité est atteint, d'une part, par l'utilisation d'un schéma du calcul de la performance qui préfère les solutions non-dominées et d'autre part, par l'application de la technique du partage entre les solutions du même front non-domine.

La première étape de NSGA consiste à trier la population P selon le principe de dominance. Cette procédure divise la population en un nombre de classe distinctes P_j de la façon suivante :

Tous les individus non-dominées de P appartiennent à l'ensemble P_1; ensuite, tous les éléments non dominées de $P\ /\ P1$ sont placés dans l'ensemble $P2$ etc. La procédure est terminée quand toute la population est triée :

$$P = \bigcup_{j=1}^{r} P_j$$

Notons qu'entre deux solutions de la même classe aucune ne peut être considérée meilleure de l'autre compte tenu de tous les objectifs du problème. Le nombre total de classes, noté r dans l'équation ci-dessus, dépend de la population P, on appelle les ensembles P_j fronts de dominance ou tout simplement fronts. La figure 2.12 présente un exemple de telle classification (les numéros associés à chaque point correspondent au numéro de la classe à laquelle il appartient),

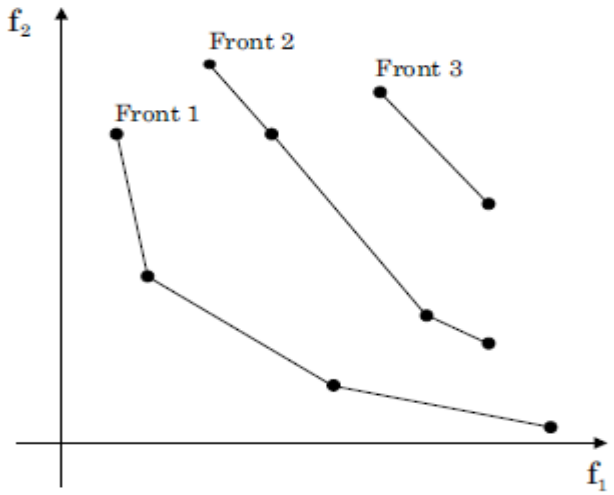


Figure 2.12 Ranking de NSGA.

Quand toute la population est triée, le front P_1 contient toutes les solutions non dominées de P. Ces solutions sont les meilleures au sens de leur proximité de la surface de Pareto du problème. La fonction de fitness F la plus grande sera donc attribuée à ces individus, elle diminué progressivement en passant d'un front à l'autre jusqu'à sa valeur la plus faible qui va être attribuée aux individus du front P_r. En pratique, toute solution i de P_1 reçoit la valeur $F_i = N$.

A chaque fois qu'on passe du front courant au front suivant, cette valeur est diminuée d'une petite quantité. Afin de préserver la diversité entre les solutions de même rang, une technique de partage est utilisée pour partager les individus ayant le même rang et la valeur de la fonction Fitness $F_i$sera alors dégradée en fonction de nombres de solutions voisines de la solution i.

Dans NSGA la procédure du partage est basée sur les distances calculées dans l'espace de décision, c'est la distance entre deux solutions i et j du même front de
taille T est donnée par l'équation :

$$d_{ij} = \sqrt{\sum_{k=1}^{T}\left(\frac{x_k^i - x_k^j}{x_k^{max} - x_k^{min}}\right)^2}$$

Ces distance est utilisée pour calculer la fonction de partage, l'expression de la fonction de partage $Sh(d_{ij})$ est donnée par l'équation avec $\beta = 2$. Le calcul du compteur de niche permet de corriger la valeur de la fonction de fitness $F_i^{'}$donnée par l'équation :

$$F_i^{'} = {}^{F_i}/_{nc_i}$$

Discussion de NSGA

L'avantage principal de NSGA c'est que les valeurs de performance sont attribuées aux individus au niveau de l'ensemble non dominée auquel ils appartiennent. Ceci permet à l'algorithme d'assurer l'avancement de la population vers la surface Pareto optimale. De plus, l'utilisation de la technique du partage assure une certaine diversité des solutions trouvées par NSGA. Cependant le choix du paramètre σ_{share} à une influence très importante sur la performance de l'algorithme [14].

3. NPGA (Niched Pareto Genetic Algorithm)

En même 1994 Horn et al. [68] ont est proposé leur méthode basée aussi sur le concept de dominance au sens de Pareto mais qui diffère des approches présentées ci-dessus dans la sélection des solutions. Dans un NPGA Les auteurs utilisent une procédure de sélection par tournoi au lieu de la sélection proportionnel. Deux solutions *i* et *j* sont tirées aléatoirement de la population P de taille N de façon aléatoire. Ensuite, toujours aléatoirement, une sous-population T de taille $t_{dom}(\ll N)$ est choisie. Chaque solution i et j est comparée avec toutes

les solutions de l'ensemble T au sens de la dominance aux deux solutions. Deux situations sont possibles :

- Si une des solutions i et j domine le sous-ensemble de solutions, alors cette solution est retenue pour la sélection et elle représente le candidat gagnant du tournoi. La figure suivant illustre ce processus de sélection dans le cas d'une minimisation de deux fonctions objectifs.
- Si une des solutions i et j dominent le sous-ensemble de solutions ou les deux solutions sont dominées par au moins une solution de T, on calcule le compteur de niche pour chacune des deux solutions. La solution dont le compteur de niche est petit est sélectionnée pour la reproduction pour générer une nouvelle population.

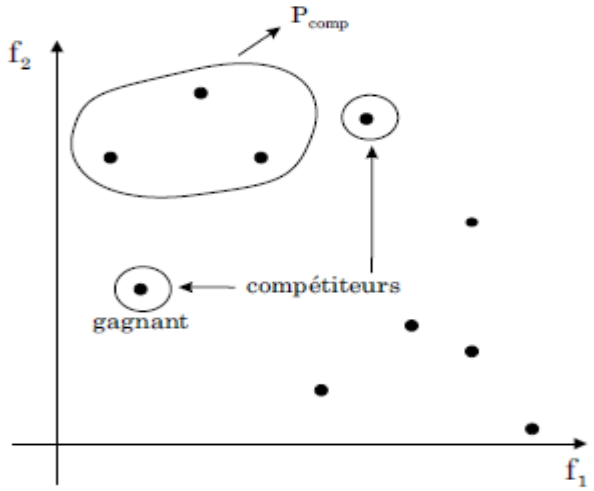


Figure 2.13 Le mécanisme de sélection de NPGA.

- Avantages de NPGA

Si le paramètre t_{dom} est beaucoup plus petit que N, alors le coût de calcul ne dépend presque pas du nombre d'objectifs, ce qui rend NPGA particulièrement intéressent pour la résolution de problème mettant en jeu un grand nombre d'objectifs [67].

- Désavantages de NPGA

La méthode nécessite la définition de deux paramètres : un paramètre de partage σ_{share} pour déterminer le vainqueur entre deux solutions équivalentes, et un paramètre t_{dom} relatif à la taille de la sous-population T.Mais le réglage de ce paramètre influence les performances de l'algorithme car si elle est trop petite, la

sélection sera trop bruitée ce qui est dangereux pour la convergence. Si, par contre, elle est trop grande, le coût de calcul va considérablement augmenter [67].

2.2.5.2. Elitistes au sens de la dominance

Ces méthodes s'affirment peu à peu comme les techniques évolutionnaires les plus efficaces pour résoudre des problèmes multiobjectifs car une amélioration notable des performances de ces algorithmes est enregistrée. Les techniques élitistes cherchent à conserver les solutions non dominées tout au long de la procédure de recherche des solutions. Ces techniques définissent un ensemble dit archive qui résume les solutions non dominées d'une génération à l'autre. L'archive est mise à jour à chaque génération, en fonction des nouvelles solutions explorées. La taille de cette archive étant généralement limitée, il est nécessaire d'éliminer certaines solutions lorsque leur nombre devient plus grand que la taille limite. Ceci est réalisé à l'aide d'une méthode de « *clustering* » qui préserve les solutions les plus représentatives et garantit une répartition uniforme des solutions sur l'ensemble du front de Pareto. Les techniques de *clustering* ne requièrent pas de connaissance a priori du rayon des différentes niches. La figure suivante illustre la structure de ce type d'algorithme lors d'une génération [12].

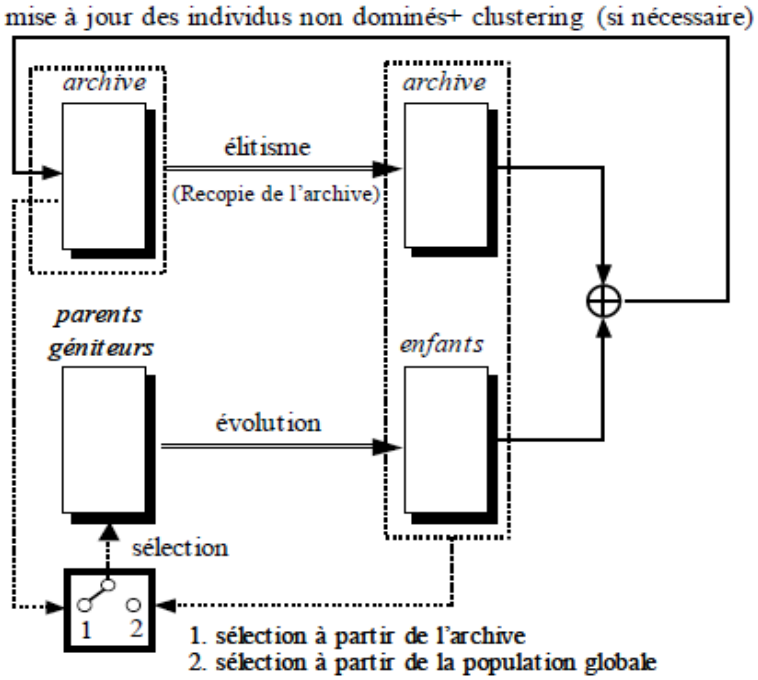


Figure.2.14 Principe général d'un algorithme élitiste.

Les différentes méthodes développées récemment diffèrent essentiellement dans la gestion de l'archive, les méthodes de clustering utilisées, l'affectation de l'adaptation et la manière dont sont sélectionnés les parents [12].

1. SPEA (Strength Pareto Evolutionary Algorithm)

Zitzler et Thiele ont proposé en 1999 un MOEA élitiste de type Pareto qu'ils ont appelé SPEA [63]. Cette approche base sur le maintien de l'élitisme dans la population, par la constitution d'un ensemble externe P' de solutions non-dominées appelé *Archive*. A chaque génération, l'ensemble est actualisé en fonction des nouvelles solutions non-dominées obtenues. Il est important de noter que SPEA non seulement préserve l'élite mais aussi la fait participer aux opérateurs génétiques (sélection, croisement et mutation) [2].

L'algorithme démarre avec une population P_0 de taille *N* générée de manière aléatoire avec une population *Archive* P'_0 *vide* .A chaque génération t, les solutions non dominées de P_t sont copiées dans la population externe P'_t et toutes les solutions dominées qui peuvent apparaitre du fait de l'ajout de nouveaux éléments à P'_t sont éliminées [67].

Quand la taille P'_t de atteint une valeur limite N' fixée à l'avance, un critère supplémentaire rentre en jeu pour sélectionner les individus élitistes qui vont être acceptés dans P'_t de sorte à ne pas dépasser la taille limite [67].

Ce critère visant la mise à jour de P'_t est basé sur le principe de *clustering* dans l'espérance de préserver une certaine diversité au sein de P'_t . L'algorithme de mise à jour de la population externe sera présenté en détail un peu plus loin dans cette section [67].

Après la mise à jour de l'ensemble P'_t Le premier pas consiste à associer une valeur d'adaptation à chaque solution de P_t à P'_t Notons que dans SPEA cette valeur sera à minimiser. La valeur d'adaptation S_i d'une solution i dans l'archive dépend du nombre de solutions n_i qu'elles dominent dans la population courante :

$$S_i = \frac{n_i}{N+1}$$

Ensuite, La fonction d'adaptation d'une solution *j* de la population courante dépend de la somme des S_i solutions qui la domine :

$$F_j = 1 + \sum_{i \in P'} S_i$$

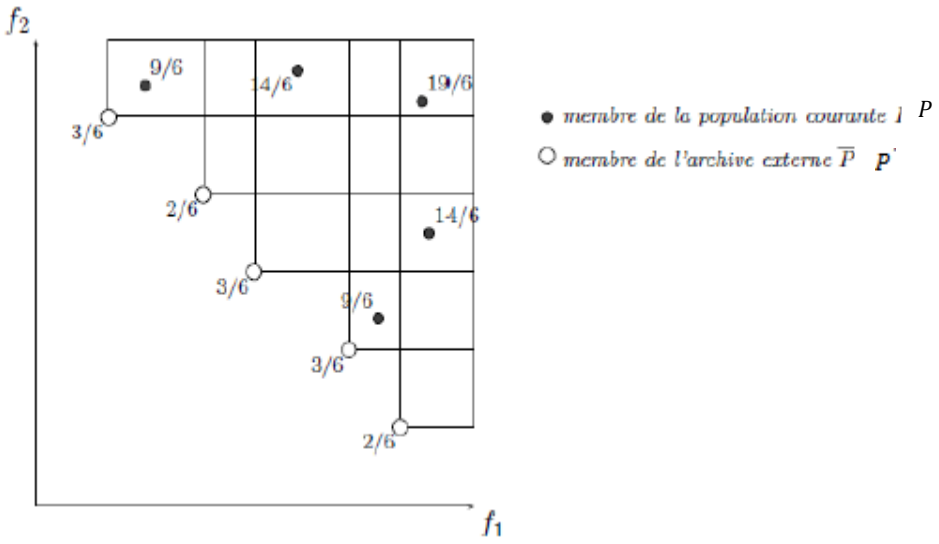


Figure 2.15 Calcul des valeurs d'adaptation en dimension deux.

Comme nous l'avons déjà dit, SPEA fait participer l'archive dans le processus génétique. La sélection par tournoi binaire est effectuée sur l'ensemble $P_t \cup P'_t$ en se basant donc sur les valeurs de S_i et F_j de façon à préférer les valeurs d'adaptation inférieures.

- Une itération de SPEA

1. Initialiser la population P_0 et créer l'archive vide P'_0 .
2. Déterminer l'ensemble des solutions non dominées $E^*(P_t)$ de la population P_t.
3. $P'_t \leftarrow P'_t \cup E^*(P_t)$.
4. Déterminer toutes les solutions non dominées $E^*(P_t)$ de l'archive actualisée P'_t et éliminer toutes les solutions dominées : $P'_t \leftarrow E^*(P_t)$.
5. Si $|P'_t| > N^*$, utiliser la technique de clustering pour réduire la taille de l'archive jusqu'à N^*, La population résultante est la population externe de la génération suivante P'_{t+1}.
6. Calculer la valeur d'adaptation des solutions de P'_{t+1} et des solutions de P_t .

7. Appliquer le tournoi binaire, le croisement et la mutation aux solutions de l'ensemble $P_{t+1} \cup P_t$ pour créer une nouvelle population P_{t+1} de taille N.

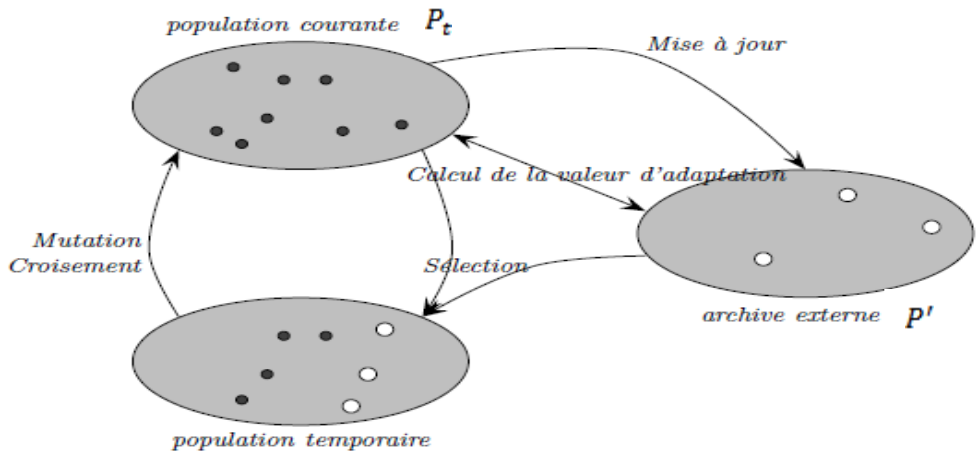


Figure 2.16 Fonctionnement général de l'algorithme SPEA.

- Procédure de clustering

Afin de réduire la taille de la population externe P'_t de N^{**} à N^*, la procédure suivante est utilisée. Au début de la procédure, chaque élément de P'_t constitue son propre cluster (groupe) donc nous avons N^{**} clusters [12]. Ensuite, les distances entre chaque paire de clusters sont calculées comme suit :

$$d_{kl} = \frac{1}{|c_k||c_l|} \sum_{i \in c_k, j \in c_l} d(i,j)$$

Les distances $d(i,j)$entre les solutions i et j peuvent être calculées dans l'espace de décision ou dans l'espace de critères. Après ce calcul, les clusters k' et l' dont la distance $d_{k'l'} = \min_{k,l \in P'_t} d_{kl}$ sont réunis pour former un cluster plus grand.

Toutes les distances d_{kl} sont alors recalculées et les deux clusters les plus proches sont de nouveau fusionnés en un seul. Le processus est répété jusqu'à ce que le nombre de clusters atteigne N^*. Dans chaque cluster, une solution est

retenue, c'est la solution dont la distance moyenne des autres solutions du même cluster est minimale [12].

-Algorithme de clustering

1. Toute solution i appartient à son cluter : $C_i = \{i\}$.

Soit $C = \{C_1, \ldots, C_{N^*}\}$ l'ensemble de tous les cluters.

2. Si $|C| \leq N^*$, passer à l'étape 5. Sinon, passer à l'étape 3.

3. Pour tout pair de cluters, calculer les distances utilisant l'équation ci-dessus. Trouver le pair $(i; j)$ qui correspond à la distance minimale.

4. Fussionner les cluters C_i et C_j . Retourner à l'étape 2.

5. Dans chaque cluster, trouver la solution dont la distance moyenne des autres solutions du même cluters est minimale et éliminer les autres solutions du cluters.

-Discussion de SPEA

Etant un MOEA élitiste, SPEA possède la faculté de trouver des solutions de bonne qualité en termes de convergence (proximité du front Pareto). Cette faculté est favorisée davantage par la façon dont SPEA évalue les solutions. En effet, la procédure d'évaluation est basée sur la relation de dominance ce qui favorise les solutions non-dominés au détriment de ceux qui sont dominés. D'autre part, la procédure de *clustering* permet de garantir une sorte de diversité au sein de l'archive finale des solutions potentiellement non-dominées. Un autre avantage d'une telle procédure c'est qu'elle ne nécessite la définition d'aucun paramètre.

Toutefois, un certain équilibre entre la taille de la population de base $\boldsymbol{P}$ et celle de la population externe est nécessaire pour une bonne performance de SPEA. En effet, si la taille $\boldsymbol{P}$ est trop grande par rapport à celle de l'archive, alors il y a risque de non-convergence puisque, dans ce cas, la pression sélective sera trop inclinée vers les solutions de l'archive et ainsi l'algorithme a moins de chances de trouver régulièrement de nouvelles solutions non- dominées. Si, au contraire, la taille de l'archive est trop petite, alors l'effet de l'élitisme sera perdu. Néanmoins, l'algorithme SPEA reste parmi les MOEAs les plus performants et les plus utilisés pour la résolution de plusieurs problèmes multiobjectifs réels (POMO) [2].

2. SPEA 2 (Strength Pareto Evolutionary Algorithm)

En 2001 Ziztler et al [68] .ont apporte des améliorations à SPEA. Ces améliorations porte essentiellement sur la définition d'une taille fixe de l'archive qui à l'état initial est constitué par les solutions non-dominés et complété par celles dominées si les solutions non-dominées ne suffisent pas pour remplir l'archive. L'autre modification porte sur le calcul de la fonction d'adaptation. Pour une solution i un calcul de distance $\sigma_i^{(k)}$avec ces voisins est effectué dans l'espace des objectifs. Les distances sont stockées dans une liste triée par ordre croissant [12].

La position *k* de la distance est donnée par *k* =$\sqrt{N+N'}$. Chaque solution sera caractérisée par une densité $D(i)$ tel que :

$$D(i) = \frac{1}{\sigma_i^{(k)} + 2}$$

La fonction d'adaptation sera calculée par l'équation suivante :

$$F_j = D(i) + \sum_{i \in P'} S_i$$

3. NSGA-II (Le Non Sorting Dominated Genetic Algorithm II)

En 2002, Deb et *al*. [15] ont proposé une version améliorée de l'algorithme NSGA. Leur algorithme, appelé NSGA-II, utilise des techniques avancées d'élitisme et de diversification basé sur la technique de partage. En pratique, NSGA-II démarre avec une population initiale (générée de façon aléatoire). Cette population initiale est triée suivant la technique du classement par dominance de manière similaire à celle utilisée dans NSGA. A l'étape de sélection des parents, NSGA-II utilise un tournoi binaire qui est basé tout d'abord sur les rangs de dominance attribués aux individus, et utilise ensuite une mesure de densité appelée distance de crowding pour comparer deux individus du même rang. En d'autres termes, si l'un des deux individus a un rang plus petit que l'autre, alors cet individu gagne le tournoi. Dans le cas où les deux individus ont le même rang, alors on compare leurs distances de crowding et celui ayant la meilleure valeur (la plus grande distance de crowding) gagne le tournoi. Le but de la technique de

crowding est de favoriser les solutions dont les voisinages sont moins denses. Le calcul des distances de crowding sera détaillé ultérieurement dans cette section. Après la sélection de la première population de parents, on lui applique les opérateurs de recombinaison et de mutation pour créer la première population d'enfants (voir en détail en chap3).

- Discussion de NSGAII

A la différence du mécanisme de préservation de la diversité basé sur la technique de partage, celui basé sur la technique de *crowding* n'exige aucun paramètre à fixé. De plus, le principe sur lequel est basée la stratégie de *crowding* favorise une distribution uniforme des solutions sur le front Pareto [15].

D'autre part, la stratégie élitiste utilisée par NSGA-II (évolution avec deux populations qui participent toutes les deux à la création de la prochaine population) favorise la convergence vers le front Pareto.

Cependant, il a été constaté à travers des expérimentations [69] qu'à partir d'une certaine génération, presque tous les individus évoluant dans NSGA-II se concentrent dans le premier front. A ce stade, seules les distances de *crowding* permettent de sélectionner les individus qui vont survivre. Par conséquent, des solutions Pareto-optimales situées dans des régions très peuplées peuvent être éliminées en laissant la place à des solutions non-dominées de la population courante mais qui ne sont pas éventuellement Pareto-optimales.

L'algorithme NSGA-II a été appliqué avec succès à de nombreuses applications dont plusieurs problèmes de type problème d'optimisation multiobjectif [70]. Ceci est dû essentiellement à sa grande capacité exploratoire. En fait, NSGA-II est probablement le MOEA le plus populaire et le plus utilisé dans la littérature.

4. SEEA (simple elitist evolutionary algorithm)

Liefooghe et al en 2010 [71] propose un nouvel algorithme évolutionnaires multiobjectif appelé SEEA (simple elitist evolutionary algorithm) est basé sur une approche élitiste très simple présenté ci-dessous.

Si l'évaluation d'une solution dans l'espace objectif ne consomme pas trop de temps de calcul, le calcul des valeurs de fitness et de diversité correspond

généralement aux étapes les plus coûteuses d'un algorithme évolutionnaires multiobjectif. Partant de ce constat, [71] proposons une méthode de recherche simple, pour laquelle aucune de ces phases n'est nécessaire, illustre par l'algorithme ci-dessous. Ainsi, une archive de solutions potentiellement Pareto optimales est maintenue et mise à jour à chaque itération de l'algorithme. L'étape de sélection consiste en une stratégie élitiste où les individus Parent sont sélectionnés dans l'archive uniquement. La population principale est donc crée à l'aide de l'application des opérateurs de variation sur les membres de l'archive choisis aléatoirement. SEEA est donc un algorithme élitiste.

- l'algorithme SEEA

1. Initialisation. Démarrer avec une population initiale P de taille N fournie en paramètre, ou la générer de façon aléatoire ; initialiser l'archive A avec les solutions non-dominées de la population *P*.
2. Evaluation. Evaluer les solutions de la population *P*.
3. Mise à jour de l'archive A. $A \leftarrow$ solutions non-dominées de $A \cup P; P \leftarrow \emptyset$
4. Condition d'arrêt. si une condition d'arrêt est satisfaite, retourner les solutions non- dominées de l'archive A. Stop.
5. Sélection pour la reproduction. Répéter jusqu'à ce que $|P| = N$ $|P| = N$: sélectionner aléatoirement une solution de l'archive A et ajouter la à la population P.
6. Variation**.** Appliquer les opérateurs de croisement et de mutation aux solutions de la population P. Aller à l'étape 2.

-Discussion de SEEA

En quelque sorte SEEA lié à d'autres algorithmes évolutionnaires élitistes tels que SPEA [72]. Néanmoins, contrairement à ces autres approches, aucune stratégie permettant de préserver la diversité ou de gérer la taille de l'archive n'est incluse ici, la taille de l'archive n'étant pas bornée. Notez que pour être appliqué à des problèmes d'optimisation où un nombre exponentiel, voir infini, de solutions non-dominées existent, un mécanisme additionnel devrait être conçu afin de limiter la taille de l'archive [73] .Un des avantages de cet algorithme est que la population (ou la taille de la population si les solutions initiales sont aléatoires) est le seul paramètres

indépendant du problème. Si les solutions non-dominées sont relativement proches les unes des autres dans l'espace de décision, et si la taille de l'archive n'est pas trop petite par rapport à celle de la population principale, [71] pensent que SEEA converge vers une bonne approximation de l'ensemble Pareto optimale en un temps d'exécution très court.

2.2.6. Comparaison des approches évolutionnaires multiobjectif

[68] comparent les approches d'optimisation multiobjectif et mettent en avant l'influence de l'introduction de l'élitisme sur la performance. Cependant la comparaison des la performance des méthodes élitistes dépendent des problèmes étudiés. Elle dépend de la méthode de comparaison de performance et la définition des paramètres pour chaque approche.

[67] présente une étude comparative d'approches métaheuristiques pour les problèmes d'optimisation multiobjectif traités dans la littérature. En général, les résultats suivent le schéma suivant :

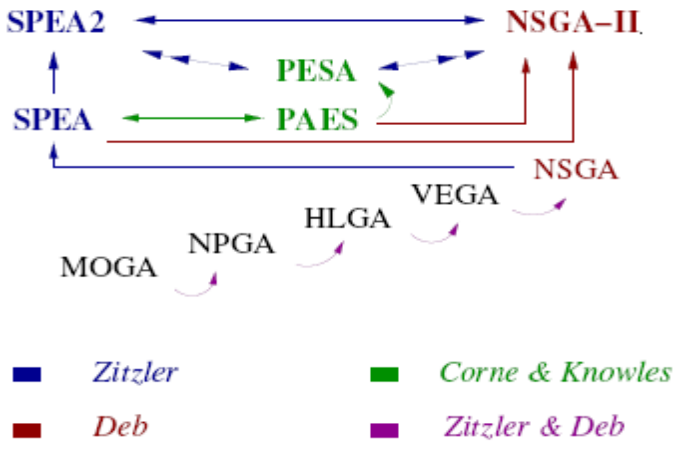


Figure 2.17 : Comparaison des performances des différents approches d'optimisation multiobjectif ; les flèches pointent vers les meilleurs. [67]

Remarque

D'après [67] sur l'ensemble des algorithmes évolutionnaires multiobjectifs examinées, est le SPEA-2 et le NSGA-II semblent être aujourd'hui des techniques de références solides, en raison des opérateurs de clustering et du crowding distance, qui garantit la diversité des solutions Pareto optimales sans nécessiter la connaissance d'un rayon de niche a priori. Ces deux méthodes ont donne lieu à

des travaux récents et nombreux, illustrant leurs originalités et leurs bonnes performances sur de nombreuses instances de problèmes.

CHAPITRE 3

RESOLUTION DU PROBLEME DE SAC A DOS MULTIOBJECTIF PAR UN ALGORITHME EVOLUTIONNAIRE

3.1.Introduction

Le problème de sac à dos multiobjectif (Multiobjective Knapsack Problem ou MOKP) est un problème représentatif d'une famille de problèmes combinatoires discrets qui appartenant à la classe des problèmes NP-difficiles. La popularité du MOKP est naturellement due à son grand intérêt tant au niveau académique qu'au niveau pratique. En effet, le MOKP permet de modéliser plusieurs applications réelles (chargement de cargaisons, répartition de budgets, allocation de ressources, ect,…).

Le problème de sac à dos contient diverses formes de problèmes, à savoir : sac à dos monoobjectif unidimensionnel, borné et non borné multidimensionnel, sac à dos multiobjectif et sac à dos multiobjectif multidimensionnel.

Dans ce chapitre on étudie un problème de sac à dos multiobjectif. On commence par donner du problème d'optimisation adapté à notre problème de sac à dos, ainsi que les méthodes de résolutions existantes dans la littérature, en utilisant l'algorithme évolutionnaire déjà cité dans le chapitre 2 (section 2.2.5.2) et trouver la solution approchée aussi on démontre l'efficacité, la mesure de performance de l'algorithme et les paramètres de génétique implémenté.

3.2. Problème de sac à dos (KP)

3.2.1. Définition du (KP)

Le problème de sac à dos est un problème combinatoire consiste à sélectionner un sous-ensemble d'objets pour remplir le sac à dos dont on dispose. En outre, cette sélection doit être faite de manière à maximiser une fonction exprimée en fonction des profits associés aux objets, tout en respectant la contrainte relative à la capacité du sac. Formellement, si n est le nombre d'objets, alors le Problème de sac à dos unidimensionnel en variable binaire 01- (KP) peut être formulé de la manière suivante :

$$(01-KP)\quad \begin{cases} \text{MaxZ}(x) = \sum_{i=1}^{n} C_i\ x_i\ , \quad C_i \in \mathbb{N} \\ telque \\ \sum_{i=1}^{n} a_i\ x_i \leq b \qquad a_i, b\ \in \mathbb{N} \\ x_i \in \{0,1\} \qquad\quad \forall i = \overline{1,n} \end{cases}$$

- $x = (x_1, \ldots, x_n)$ est un vecteur binaire de dimension n, définissant une solution possible du problème (*KP*) de sorte que:
- $x_i = 1;$ si l'objet i est sélectionné. $x_i = 0$; sinon.
- C_i est le profit apporté par l'objet i.
- $\boldsymbol{a_i}$ est le poids de l'objet i
- b est la capacité du sac à dos

3.2.2. Variantes autour du problème

Il existe de nombreuses variantes du problème de sac à dos, selon le domaine des variables (valeurs binaires, entières ou réelles), de nombre de contraintes (unidimensionnel, bidimensionnel ou multidimensionnel), le nombre de fonction objectif (mono-objectif ou multiobjectif), le nombre de sac, etc.
Le lecteur désireux de connaitre plus de détails sur ces variantes ou sur d'autres pourra se référer à [74], [75].

3.2.2.1. Variables continues

Le problème de sac à dos en variables continues (LKP) est une variante dans le quelle il est possible de ne prendre qu'une fraction des objets. Sa résolution s'appuie sur les concepts d'efficacité d'un objet et d'élément bloquant, définis ci-dessous.

Définition 1 (Efficacité d'un objet)

On appelle efficacité d'un objet i le rapport de son cout sur son poids, noté :

$$e_i = \frac{C_i}{\boldsymbol{a_i}}$$

Définition 2 (Elément bloquant)

On appelle Elément bloquant le premier objet ne pouvant tenir dans le sac lorsque les objets sont ajoutés par ordre décroissant d'efficacité.

Son indice sera noté :

$$s = \min\left\{k : \sum_{i=1}^{k} a_i > b\right\}$$

En suivant dans l'ordre décroisant des e_j la solution optimale du (LKP) sera détaillé dans [76].

3.2.2.2. Sac à dos à variables bornées

Le problème défini précédemment peut être généralisé en assignant pour chaque objet i une borne w_j: ($w_j \leq \frac{b}{a_j}$), nous obtenons donc le problème de sac à dos à variables bornées dont la formulation mathématique est le suivante :

$$\begin{cases} \text{MaxZ}(x) = \sum_{i=1}^{n} C_i\, x_i \\ telque \\ \sum_{i=1}^{n} a_i\, x_i \leq b \\ 0 \leq x_i \leq w_i \qquad i = \overline{1,n} \\ x_i entier, \qquad i = \overline{1,n} \end{cases}$$

Dans le cas $w_j = +\infty$ le problème s'appelle sac à dos à variables non bornées.

3.2.2.3. Sas à dos multidimensionnel

Le problème de sac à dos multidimensionnel (d-KP), est une variante du problème ayant plusieurs contraintes de capacité. Il peut être considéré comme un programme linéaire en nombre entiers (ILP) classique sous la seule restriction que les coefficients soit positif et les variables binaires.

$$(d-KP)\begin{cases} \text{Max Z (x)} = \sum_{i=1}^{n} C_i\, x_i \\ telque \\ \sum_{i=1}^{n} a_i^j\, x_i \leq b_j \quad a_i^j, b_j \in \mathbb{N}, j = \overline{1,m} \\ x_i \in \{0,1\} \qquad \forall i = \overline{1,n} \end{cases}$$

IL est largement utilisé dans l'évaluation de métaheuristique [80], en particulier pour la recherche tabou et les algorithmes génétiques. Cette variante est intrinsèquement difficile. Ainsi, les méthodes de résolution exacte actuelles se limitent à des instances de quelques centaines de variables pour une dizaine de contraintes.

Ainsi, la meilleure solution pour résoudre une instance de d-KP aujourd'hui consiste en l'utilisation d'heuristique [81] ou de métaheuristique [82], [83][84].

3.2.2.4. Sas à dos multiple

Le problème de sac à dos multiple (MKP) consiste à répartir un ensemble d'objets dans plusieurs sacs à dos de capacité différente en respectant les contraintes de capacité de chaque sac en maximisant le profit total. Une application possible du problème (MKP) est le chargement de cargos (Eilon et Christofides).

$$(MKP)\begin{cases} \text{Max Z (x)} = \sum_{i=1}^{n}\sum_{j=1}^{m} C_i\, x_{ij} \quad C_i \in \mathbb{N} \\ telque \\ \sum_{i=1}^{n} a_i\, x_{ij} \leq b_j \quad a_i, b_j \in \mathbb{N}, j = \overline{1,m} \\ \sum_{i=1}^{n} x_{ij} \leq 1 \qquad \forall i = \overline{1,n} \\ x_i \in \{0,1\} \qquad \forall i = \overline{1,n}, j = \overline{1,m} \end{cases}$$

3.2.2.5. Sac à dos multiobjectif

Le problème de sac à dos multiobjectif (01-MOKP), est une variante du problème (KP) où plusieurs objectifs sont à maximiser simultanément.

$$(01-MOKP)\quad \begin{cases} \mathrm{MaxZ}(\mathrm{x}) = \left(\sum_{i=1}^{n} C_i^j \; x_i\right) \quad j = \overline{1,m} \\ tel\ que \\ \sum_{i=1}^{n} a_i \; x_i \leq b \quad a_i, b \in \mathbb{N}^* \\ x_i \in \{0,1\} \quad \forall i = \overline{1,n} \end{cases}$$

3.2.3. Méthodes de résolutions existantes

3.2.3.1. Méthodes de résolutions exactes

Plusieurs méthodes exactes ont été proposées et sont capables de résoudre plusieurs classes d'instances problème de sac à dos multiobjectif (01-MOKP) deux et trois objectifs. Le lecteur intéressé trouvera dans la récente thèse de J. Jorge [83] un état de l'art sur ces méthodes, une description algorithmique ainsi qu'une analyse comparative des propositions les plus performantes.

Dans ce sens, on peut citer la méthode à deux phases qui a été adaptée au le problème de sac à dos multiobjectif (MOKP) [85] On peut citer également le travail de Gandibleux et Fréville [86], basé sur une hybridation entre une méthode exacte de réduction de l'espace de recherche et une recherche tabou, il n'existe pas dans la littérature d'algorithmes exacts destinés à le résoudre. Ceci est dû, sans aucun doute, à son caractère à la fois NP-difficile et fortement combinatoire. En effet, dans la pratique, on rencontre souvent des instances de MOKP dites difficiles dans le sens où les meilleures combinaisons d'objets en termes de profits ne peuvent pas satisfaire toutes les contraintes de capacités.

3.2.3.2 Méthodes de résolutions approchées

Nonobstant les avancées significatives de ces méthodes exactes, les méthodes approchées, fondées notamment sur les métaheuristiques, restent une réponse opérationnelle lorsque les premières se heurtent à des difficultés face à certaines instances. On parle dès lors de métaheuristiques multiobjectif (MOMH) [83], où les algorithmes évolutionnaires multiobjectifs (MOEAs) occupent une place importante. Par ailleurs, le problème de sac à dos multiobjectif fait souvent office de benchmark privilégié quand il s'agit de montrer l'efficacité d'une nouvelle MOMH.

La plupart des MOEAs ont été conçue pour être génériques et répondre à des problèmes d'optimisation non-linéaires en variables continues. De ce fait, les (MOEAs) ne correspondent généralement pas dans leur définition originale aux besoins des problèmes d'optimisation multiobjectif (POMO). Par exemple :

- Ishibushi et Murata ont montré dès 1995 l'intérêt d'adjoindre une recherche locale à un (MOEAs), introduisant à cette occasion les méthodes hybrides [87].
- Gandibleux et al. [88] ont souligné l'impact positif dans un MOEAs d'utiliser des solutions initiales construites de qualité, alors que traditionnellement des solutions aléatoires simplement réalisables sont utilisées. Elles se comportent comme des attracteurs pour les autres solutions de la population.
- Delorme et al. [89] ont proposé un couplage GRASP-SPEA pour le problème de set packing multiobjectif, GRASP étant utilisé pour élaborer une population initiale.
- Gandibleux et al. [90] ont introduit le path-relinking dans un contexte multiobjectif comme composant d'une MOMH qui se couple avantageusement avec des bornes heuristiques locales [91].

Fort de ces résultats, un schéma méthodologique en trois étapes d'une MOMH pour l'optimisation discrète a été discuté en 2007 par Gandibleux et Chamayou [92].

3.4. Adaptation d'une méthode évolutionnaire au problème de sac à dos multiobjectif

Dans cette partie, on applique l'algorithme évolutionnaire NSGAII (Fast Non Dominated Sorting Genetic Algorithm) qui traite le problème de sac à dos multiobjectif. L'utilisation de cet algorithme est choisie par rapport à la facilité de son utilisation et la rapidité de son exécution. On définie la méthode NSGAII ainsi les étapes correspondantes ce cas.

3.4.1. Définition de la méthode NSGA-II

NSGA-II (Fast Nondominated Sorting Genetic Algorithm) [15] est un algorithme qui utilise des techniques avancées d'élitisme et de diversification en intégrant un opérateur de sélection, basée sur un calcul de la distance crowding, très différent de celui de NSGA pour la préservation de la diversité et pour gérer

l'élitisme, l'algorithme assure qu'à chaque nouvelle génération, les meilleures solutions rencontrées soient NSGAII obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de K. Deb, ce qui fait de cet algorithme un des plus utilisées aujourd'hui .

3.4.2. Les étapes de la méthode NSGA-II au problème de sac à dos

Avant de commence les étapes de fonctionnement de NSGAII On note que NSGAII évolue en utilisant deux populations de même taille une population P dite population initiale de parents et une population Q dite population d'enfants et on peut résume les étapes de fonctionnement de la méthode NSGAII dans la l'organigramme (figure3.1) suivant:

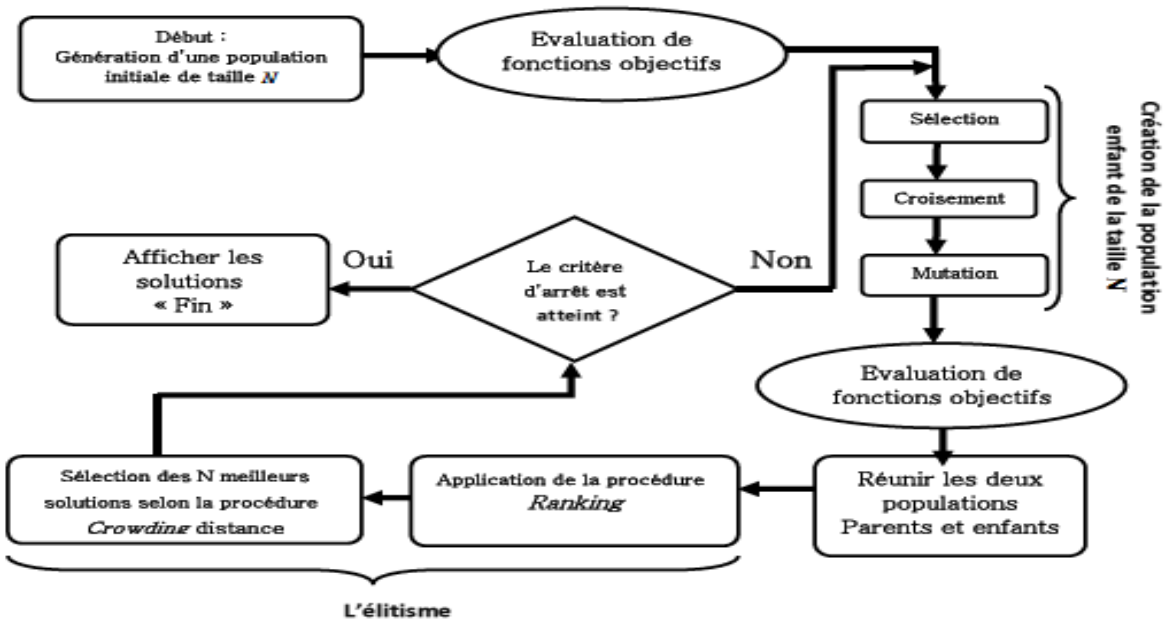


Figure 3.1 Organigramme de la méthode NSGAII.

1. Initialement, Au début générée de façon aléatoire une population initiale P_0. Cette population initiale P_0 est triée suivant la technique du classement par dominance de manière similaire à celle utilisée dans NSGA, tous d'abord en applique l'étape de sélection pour crée une population parent P_0 de taille N et on vérifier si elle set réalisable avec la procédure de l'évaluation de fonctions objectifs.

2. Ensuite, les parents sélectionnés subissent des opérations de croissement et de mutation pour créer la population d'enfants Q et on vérifier aussi si elles réalisable. dans la section suivant on détaillé sur ces trois opérations.

a-Sélection

A l'étape de sélection des parents, NSGA-II utilise un tournoi binaire qui est basé tout d'abord sur les rangs de dominance attribués aux individus, et utilise ensuite une mesure de densité appelée distance de crowding pour comparer deux individus du même rang. En d'autres termes, si l'un des deux individus a un rang plus petit que l'autre, alors cet individu gagne le tournoi. Dans le cas où les deux individus ont le même rang, alors on compare leurs distances de crowding et celui ayant la meilleure valeur (la plus grande distance de crowding) gagne le tournoi. Le but de la technique de crowding est de favoriser les solutions dont les voisinages sont moins denses. Le calcul des distances de crowding sera détaillé ultérieurement dans la section plus loin

Une population Q_0 de taille N est obtenue par l'application des opérateurs de croisement et de mutation aux parents sélectionnés lors du tournoi.

b-croissement

A l'étape de croissement, NSGAII utilise l'opérateur de croisement 1-point et le croisement 2-point, Dans le croisement 1-point consiste à choisir d'abord un points de coupure(un seul fragment est échangé selon un point de coupure) choisi aléatoirement pour chaque couple de parents, puis à échanger alternativement les fragments délimités par ce point de coupure afin d'obtenir un nouveau couple d'enfants et dans le croisement 2-point, il consiste à échanger deux fragments selon deux points de coupure choisis aléatoirement pour chaque couple de parents, puis à échanger alternativement les fragments délimités par ces points de coupure afin d'obtenir un nouveau couple d'enfants, dans nos travaux on utilise le croissent de 2-points.

c-Mutation

NSGAII utilise L'opérateur de mutation *de* 1-bit consiste à sélectionner aléatoirement un (ou plusieurs) gène(s) dans un individu et à modifier sa (leurs) valeur(s) par l'inverse la valeur d'un seul bit (de 1 à 0 ou de 0 à 1) choisi au hasard

avec une probabilité faible fixé p_m=0.6 dans un environnement stable afin de favoriser l'accumulation des gènes favorables (obtenus par croissement) tout en permettant d'élargir l'espace des solutions évaluées.

3. Ces deux populations sont regroupées en une autre population R de taille 2N, où N est la taille commune à P et Q Puis, la population R est triée selon le principe du classement par dominance d'après la figure suivant :

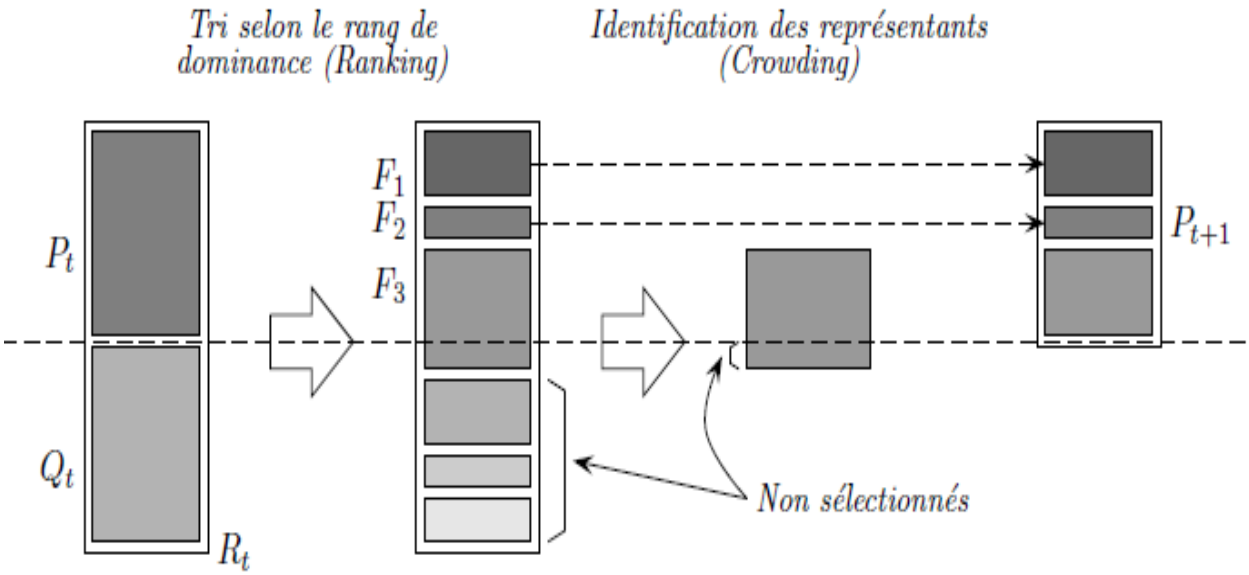


Figure 3.2 Schéma représentant les étapes de l'algorithme NSGAII.

4. La nouvelle population de parents est de taille N sera constituée à partir des meilleurs fronts de la population R. Mais, toutes les solutions ne peuvent pas rentrer dans la nouvelle population de parents dont la taille est N. Dans ce cas pour sélectionner les solutions de R qui vont survivre à la génération suivante on utilise un mécanisme de priorité qui favorise d'abord les meilleurs fronts.

La procédure de classification en fonts Pareto «Technique de Ranking»

1. Début R= P ∪ Q
2. Initialiser le compteur i des fronts Pareto à 1 ;
3. Répéter jusqu'à R= Ø
4. Trouver F_i, l'ensemble des solutions non dominées correspondants au front de Pareto i ;
5. Supprimer de R les solutions non dominées trouvées appartenant au front F_i;

Fin Répéter
Fin.

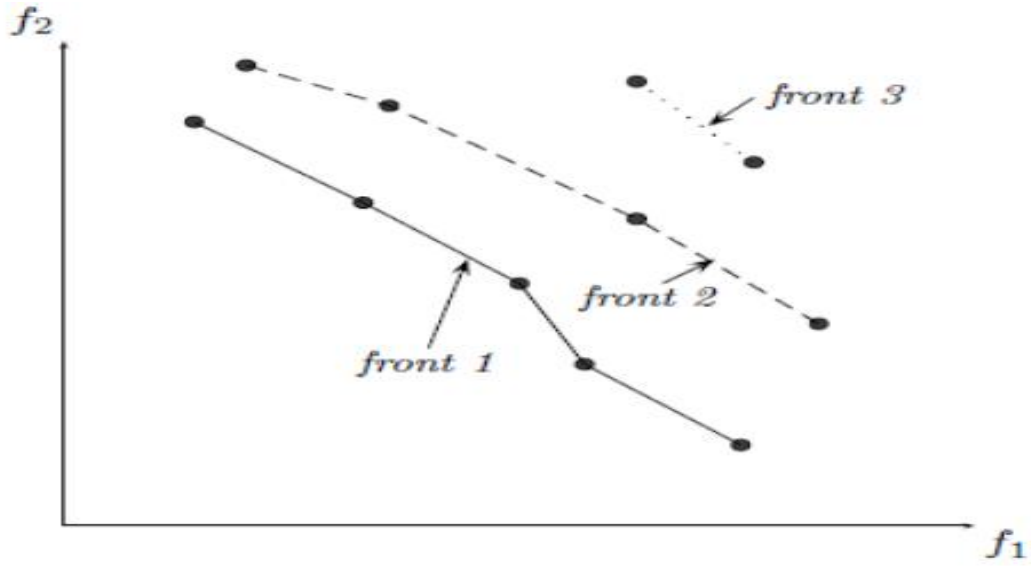


Figure 3.3 Classification des solutions en plusieurs fronts Pareto.

5. Pour sélectionner les solutions de *R* qui vont survivre à la génération suivante on utilise un mécanisme de priorité qui favorise d'abord les meilleurs fronts (les fronts qui représentent les premiers rangs) et qui fait intervenir ensuite la technique de *crowding* au cas où la sélection doit être faite entre des individus ayant le même rang.

La procédure de calcul des distances de crowding

1- Initialisation soit l =$|F|$; l : le nombre de solution dans le front F et soit M : le nombre d'objectifs

Pour i=1 jusqu'à l faire

$$d_i = 0$$

2- calculer la valeur de la fonction objectif f_i

Soit I^m : le vecteur d'indices obtenu en ordonnant les solutions de F selon l'ordre croissant des valeurs de l'objectif f_m

Soit I_i^m : l'indice de la solution i dans la liste ordonnée selon l'objectif f_m

Pour m = 1 jusqu'à M faire

-Trouver le vecteur d'indices I^m= Trier (f_m, <)

3- Associer des distances de crowding très grandes aux individus résidant sur les extrémités de I_i^m :

$$d_1^m = d_l^m = \infty$$

Pour i = 2 jusqu'à l -1 faire :

$$d_i = d_i + \frac{f_m^{I_i^m+1} - f_m^{I_i^m-1}}{f_m^{max} - f_m^{min}}$$

4- si i =m, la procédure est terminée. Sinon, incrémenter le compteur d'objectif i ← (i+1) et retourner à l'étape 2.

Géométriquement, d_3 la distance de crowding associée à une solution x_3 correspond au semi-périmètre du cuboïde dont les sommets sont les deux voisins les plus proches de x_3.

$$d_3 = \frac{f_1^{(4)} - f_1^{(2)}}{f_1^{max} - f_1^{min}} + \frac{f_2^{(2)} - f_2^{(4)}}{f_2^{max} - f_2^{min}}$$

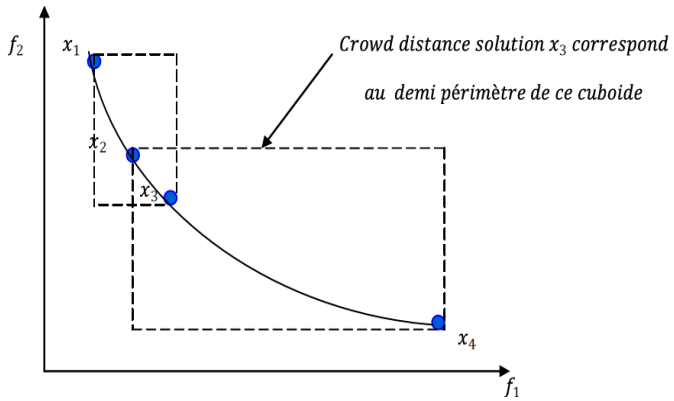


Figure 3.4 Illustration du calcul de crowding distance dans NSGAII

On peut résume les étapes de la méthode NSGAII dans L'algorithme suivant :

Algorithme de NSGAII

1–Début

Création des populations P_0 et Q_0 de taille N. illustration du calcul de crowding distance dans NSGAII.

2–Tant que critère d'arrêt non rencontrée faire

- Poser $R_t = P_t \cup Q_t$;
- Trouver les différents fronts de non dominance F_i de la population R_t

3–Poser $P_{t+1} = \emptyset$ et i=0 ;

Tant que $|P_{t+1}| + F_i < N$

$P_{t+1} = P_{t+1} \cup F_i$

i =i+1

Fin Tant que

4 –Trier les solutions du front F_i selon l'ordre décroissant de leurs crowding distance.

5 – Ajouter à la population P_{t+1} les $(N-|P_{t+1}|)$ premières solutions du front F_i.

6 – Application des opérateurs génétiques pour la génération d'une nouvelle population Q_{t+1} de taille N.

Fin Tant que

Fin.

3.5. Heuristique de génération de population réalisable pour problème de sac à dos multiobjectifs

On commence par générer aléatoirement N vecteurs à n dimensions de valeurs binaires (1 si l'objet est retenu, 0 sinon) formant la population pop. Afin d'étudier la réalisabilité des solutions générées, on vérifie pour chaque solution si le volume d'objets retenus n'excède pas la capacité du sac. Dans le cas contraire, on retire l'objet j dont le rapport :

$$\sum_{i=1}^{m} C_j^i / a_j \ \ est\ plus\ faible$$

Ce procède est répété jusqu'à l'obtention d'une population réalisable. Enfin on évalue chaque solution de cette dernière suivant les m critères du problème 01-MOKP (on résume cette procède dans l'organigramme suivant).

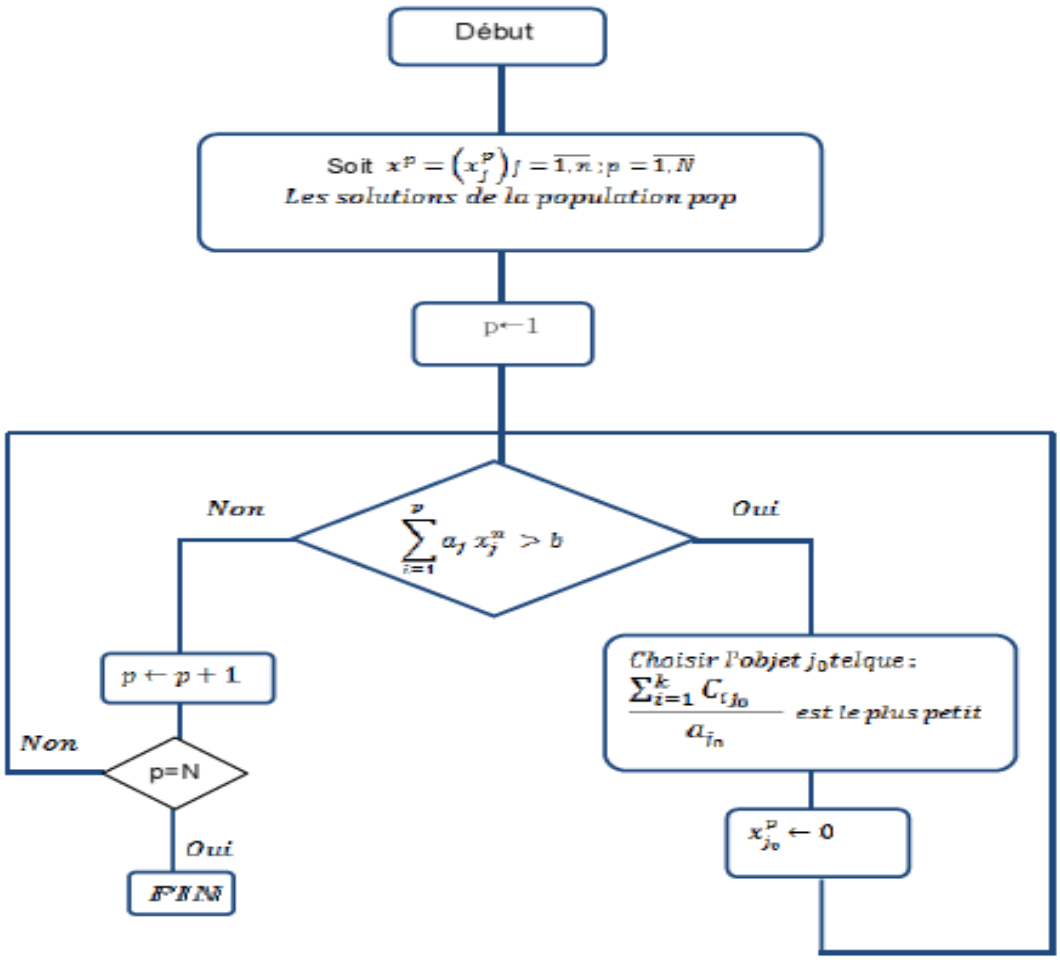


Figure 3.5 Organigramme de l'algorithme de population réalisable pour (01-MOKP)

3.6 Evaluation des performances

L'évaluation de la performance d'une métaheuristique à partir d'une présentation graphique est souvent imprécise et de plus est limitée à deux /trois objectifs. Au-delà de trois objectifs, il apparait nécessaire de définir des mesures quantitatives pour évaluer les performances des métaheuristiques développées pour l'optimisation multiobjectif. Peu de travaux présentés dans la littérature [93,94] effectuent une évaluation quantitative des méthodes proposées. Cela vient du fait que la plus part de ces dernières ont été proposées pour des problèmes de petite taille (sac-à-dos bi-critère, ordonnancement flow-shop b à deux fonctions objectifs, etc.). Nous avons recensé dans la littérature un certain nombre de mesures de performance selon qu'elles soient basées sur l'approximation par rapport à la frontière efficace exacte ou quelles soient basées sur les préférences du décideur. Dans le cadre de notre travail nous allons présenter la métrique qui mesure la qualité d'une approximation du front Pareto par rapport à l'ensemble des solutions Pareto optimal noté : *E*.

3.6.1. Absolute Efficiency (AE)

La première mesure (AE) permet de calculer la proportion des solutions non dominées de Pareto de l'ensemble approximé noté: Ê [29].

$$AE = \frac{|\hat{E} \cap E|}{|E|}$$

Cette mesure a été utilisée dans plusieurs travaux, où l'ensemble est trouvé par une méthode énumérative [69], [79], [80], Branch and Bound [29], etc.

3.6.2. Distance entre E et Ê

Une solution appartenant à Ê sans être optimale Pareto n'est pas nécessairement une mauvaise solution. Il est donc intéressant de calculer la distance entre l'ensemble E et Ê [28]. Plus petite est la distance, meilleure est la qualité de l'algorithme.

Soit d(x,y) une distance entre deux solutions dans l'espace objectif :

$$d(x,y) = \sum_{i=1}^{n} \lambda_i |f_i(x) - f_i(y)|$$

où $\lambda_i$est un poids qui permet de normaliser les différents critères La distance.

- La distance entre une solution $y \in E$ et une solution de Ê:

$$d(\hat{E}, y) = \min d(x, y), x \in \hat{E}$$

- La plus mauvaise distance entre les ensembles Ê et E :

$$WD = max d(\hat{E}, y) \quad , y \in E$$

- La distance moyenne entre les ensembles Ê et E

$$MD = \frac{\sum_{y \in E} d(\hat{E}, y)}{|E|}$$

- Une mesure l'uniformité de l'ensemble Ê :

$$Div = \frac{WD}{MD}$$

Ces mesures ont été utilisées dans [29] pour évaluer les performances d'un recuit simulé dans la résolution du problème du sac à dos multiobjectif.

3.7. Motivations

3.7.1. Paramètres de l'algorithme génétique implémenté

L'opérateur de croisement est souvent considéré comme l'opérateur principal dans les processus de recherche de plusieurs algorithmes évolutionnaires [15] car c'est l'opérateur qui a une influence directe sur l'exploration de l'espace de recherche et sur la création de nouvelles solutions à partir des solutions existantes. Rappelons que le croisement permet de produire deux nouvelles solutions, appelées enfants, à partir de deux solutions, appelées parents, Comme nous l'avons mentionné dans le chapitre 2 (section2.2.3.4). Le croisement 1-point et 2-point [59] sont les deux opérateurs de recombinaison les plus utilisés dans la littérature. Par conséquent, pour la résolution de notre problème 01-MOKP, nous avons opté pour l'implémentation de la méthode NSGA-II avec croisement de deux points qui est bien claire par rapport à un point.

Pour notre étude, nous avons fixé les paramètres de l'algorithme génétique (AG) comme suit :

- La probabilité de croisement P_c = 0:9 ;
- La probabilité de mutation P_m = 0:05 ;
- Le critère d'arrêt est fixé à 150 générations.
- La taille de la population varie selon la taille de l'instance ;

3.8. Conclusion

L'objectif de cette partie est mise en œuvre de multiples procèdes à savoir l'application d'un algorithme évolutionnaire : NSGAII au problème de sac à dos multiobjectif et l'élaboration de l'instance et l'efficacité de l'algorithme ainsi que la détermination de la distance séparant la solution potentiellement dominée et la solution non dominée, les résultats obtenus sont très satisfaisant car le maximum du minimum de la distance entre les deux solutions est minimale ce qui explique la performance de l'algorithme utilisée.

CHAPITRE 4

IMPLEMENTATION ET RESULTATS

4.1. Introduction

Ce chapitre est consacré à l'implémentation des résultats obtenus en utilisant un langage Matlab 7.0 soit par programmation, soit par l'utilisation de fonctions prédéfinies.

4.2. MATLAB 7.0

Matlab 7.0 est à la fois un langage de programmation et un environnement de développement, développé et commercialisé par la société américaine : Math Works [95].

Matlab est utilisé dans les domaines de recherche et de l'industrie pour le calcul numérique mais aussi dans les phases de développement de projets. C'est un outil très pratique et vaste d'utilisation. Il est doté de nombreuses boites à outils (Toolbox) ; incluant les possibilités données par d'autre langages de programmation comme C++ ou Java [95].

4.2.1. Principe d'utilisation

Pour la détermination les paramètres suivant :

1. CPU : le temps moyen d'exécution (en secondes).
2. Nbr.SPND : nombres de solutions potentiellement non dominées.
3. Nbr.SND : nombres de solutions non dominée.
4. Taux sol: mesure qui permet de calculer la proportion des solutions non dominées de Pareto de l'ensemble approximé.
5. d (SPND, SND) : distance entre la solution potentiellement non dominées et solutions non dominée.
6. Min (d (SPND, SND)) : la distance minimum de chaque solution potentiellement non dominée par à rapport à la solution non dominées.
7. WD=Max (Min (d (SPND, SND))) : La plus mauvaise distance entre les deux solutions
8. MD : la distance moyenne entre les deux solutions

9. Div =WD/MD : Une mesure l'uniformité des solutions potentiellements non dominées.

En effet ; en premier lieu on introduit les données du problème de sac à dos multiobjectif 01-MOKP :

$$"Max" \{Cx \setminus Ax \leq b \ , x \in \{0,1\}\}$$

Où A, C et x sont des matrices de taille respective 1× N, M × N et N× 1.

Pour calculer les instances i, ont été construites grâce au générateur aléatoire décrit ci-dessous:

Algorithme1 Génération aléatoire d'instances pour un problème multiobjectif

Entrées :

N : nombre de variables

M : nombre de critères

Sorties : les matrices A, C et b formant le problème multiobjectif

A = randint (1, n [1, 99]) ;

C = randint (m, n,[1,99])

B =$\sum_{j=1}^{n} A_j$.

Telle que randint (M ; N ; [x ; y]) : est une fonction prédéfinie de Matlab qui renvoie une matrice à M lignes et N colonnes dont les éléments sont des entiers générés aléatoirement entre x et y.

De la même manière pour calculer la distance entre la solution potentiellement non dominées SPND et la solution non dominées SND, on introduit les données qui sont obtenus dans programme élaborée précédent.

Algorithme 2 distances entre SPND et SND

Entrées :

SPND : solution potentiellement non dominée donnée par NSGAII.

SND: solution non dominée donnée par la méthode exacte.

-Calculer:

1- d (SPND, SND);

2- Min (d (SPND, SND));

3- WD= Max (Min (d (SPND, SND)));

4- MD= Min (d (SPND, SND)) /N;

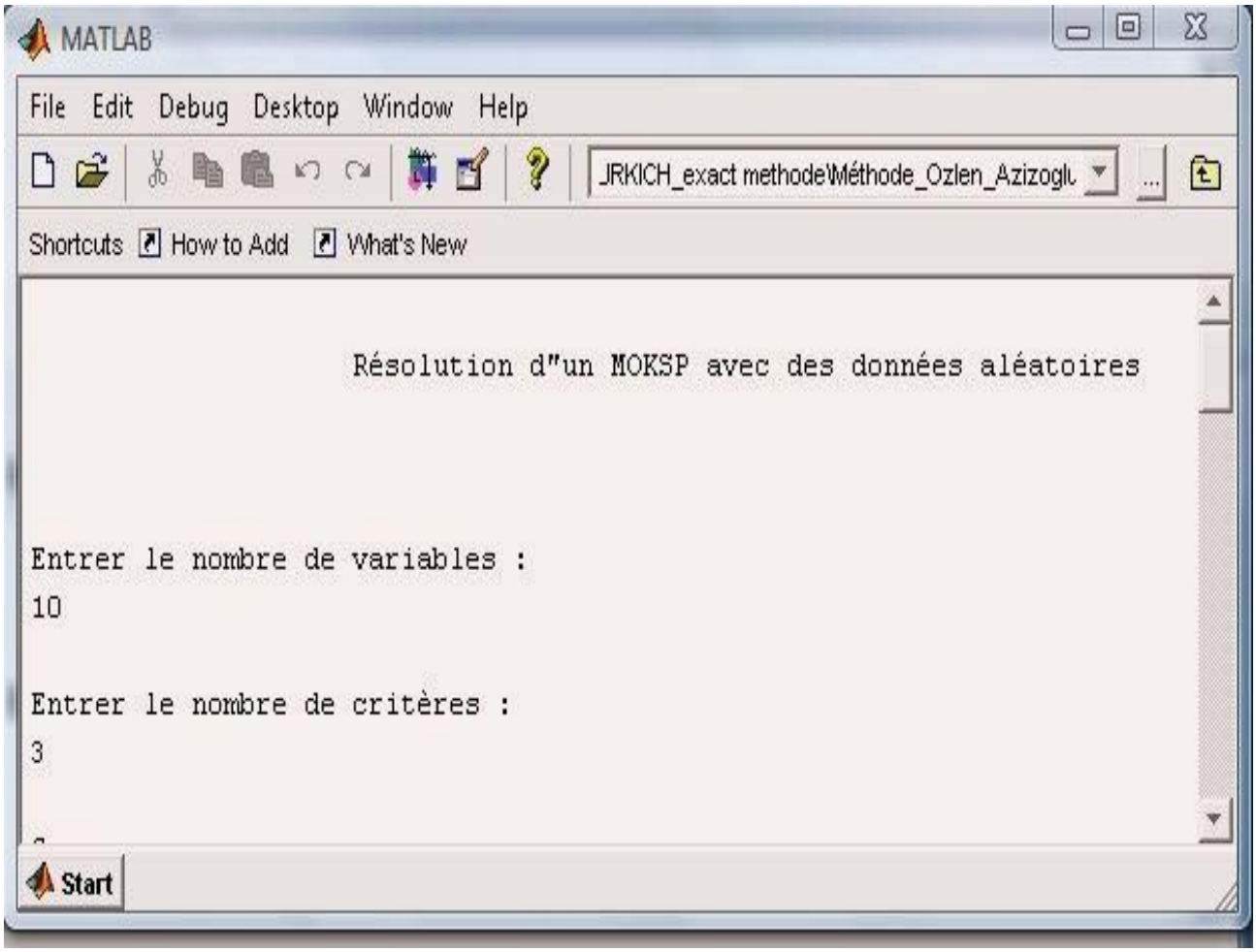
5- DIV = WD/MD.

4.2.2. Exemple d'application :

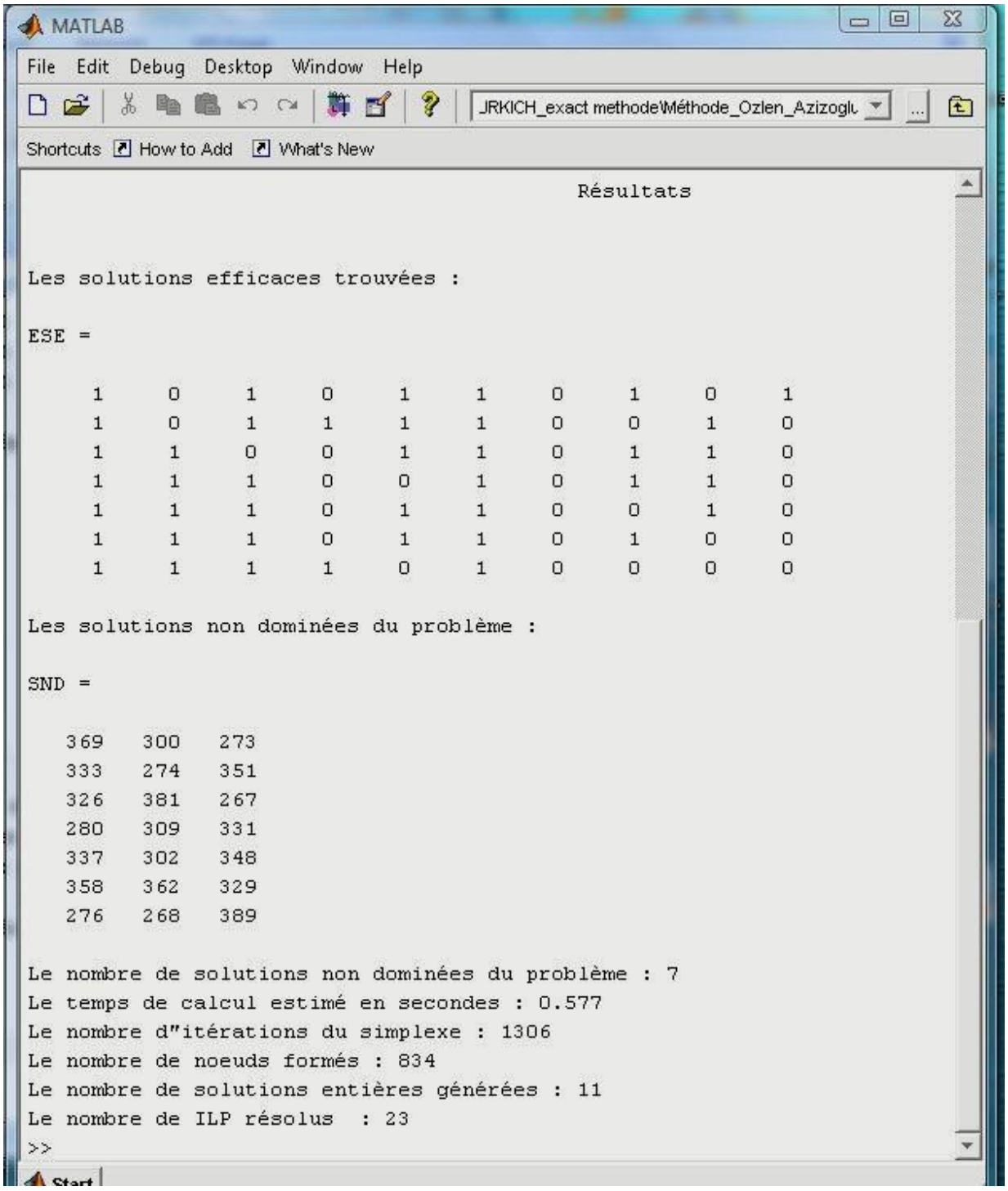
On donne un exemple de problème de sac à dos multiobjectif avec 10 variables et 3 critères tel que : les variables représentent les objets qui rentrent dans le sac à dos et les critères représentent le nombre de la fonction objectif alors le problème de sac à dos multiobjectif (01-MOKP) peut être formulé de la manière suivante :

$$\begin{cases} MaxZ(x) = \left(\sum_{i=1}^{10} C_i^j x_i\right) & j = 1,2,3. \\ \sum_{i=1}^{10} a_i x_i \leq b & a_i, b \in \mathbb{N}^* \\ x_i \in \{0,1\} & \forall\, i = \overline{1,10} \end{cases}$$

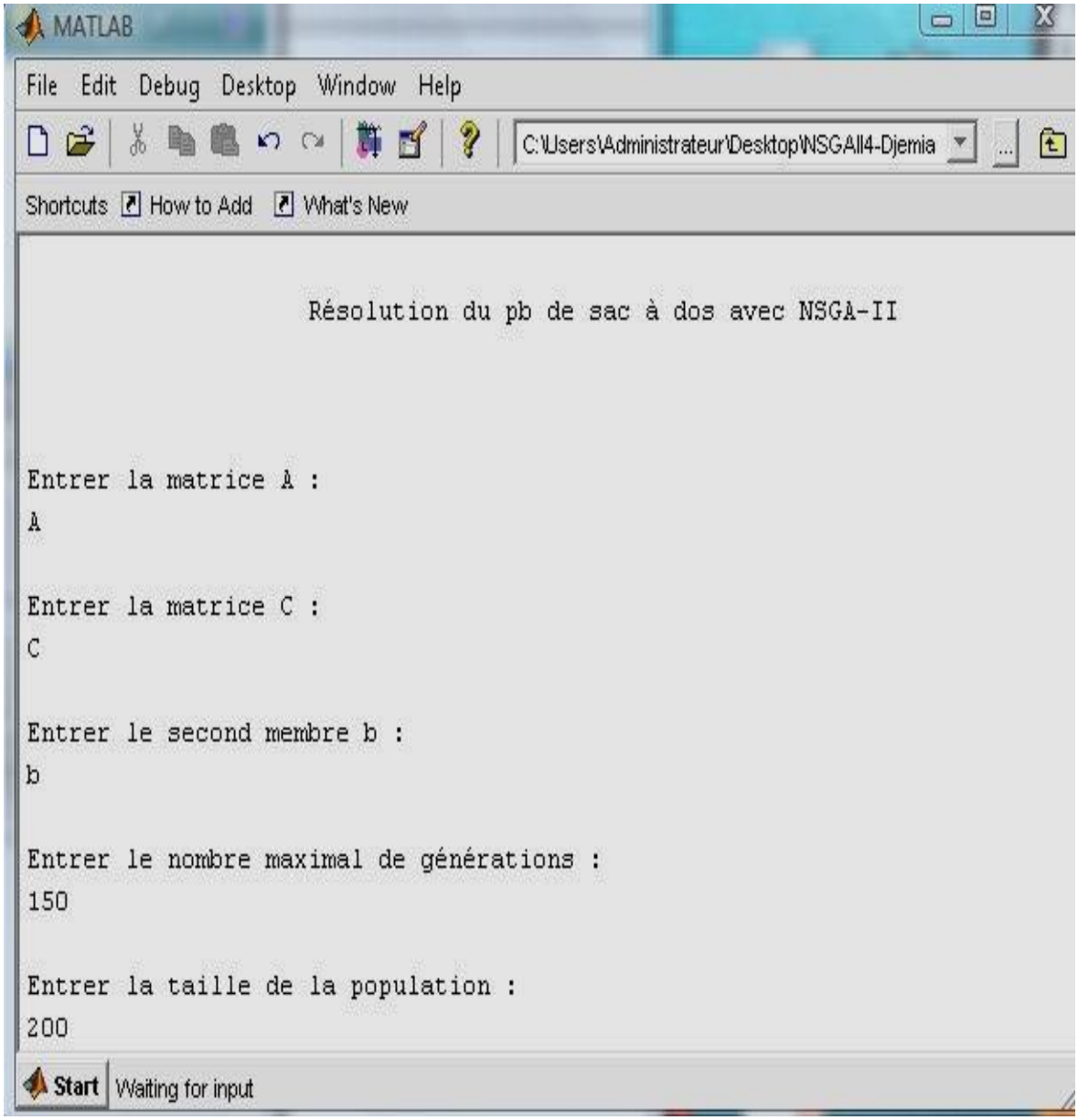
1-- On commence de faire rentrée le nombre de variables et le nombre de critères avec l'algorithme1 (avec des données aléatoires) comme le montre la figure ci-dessous :



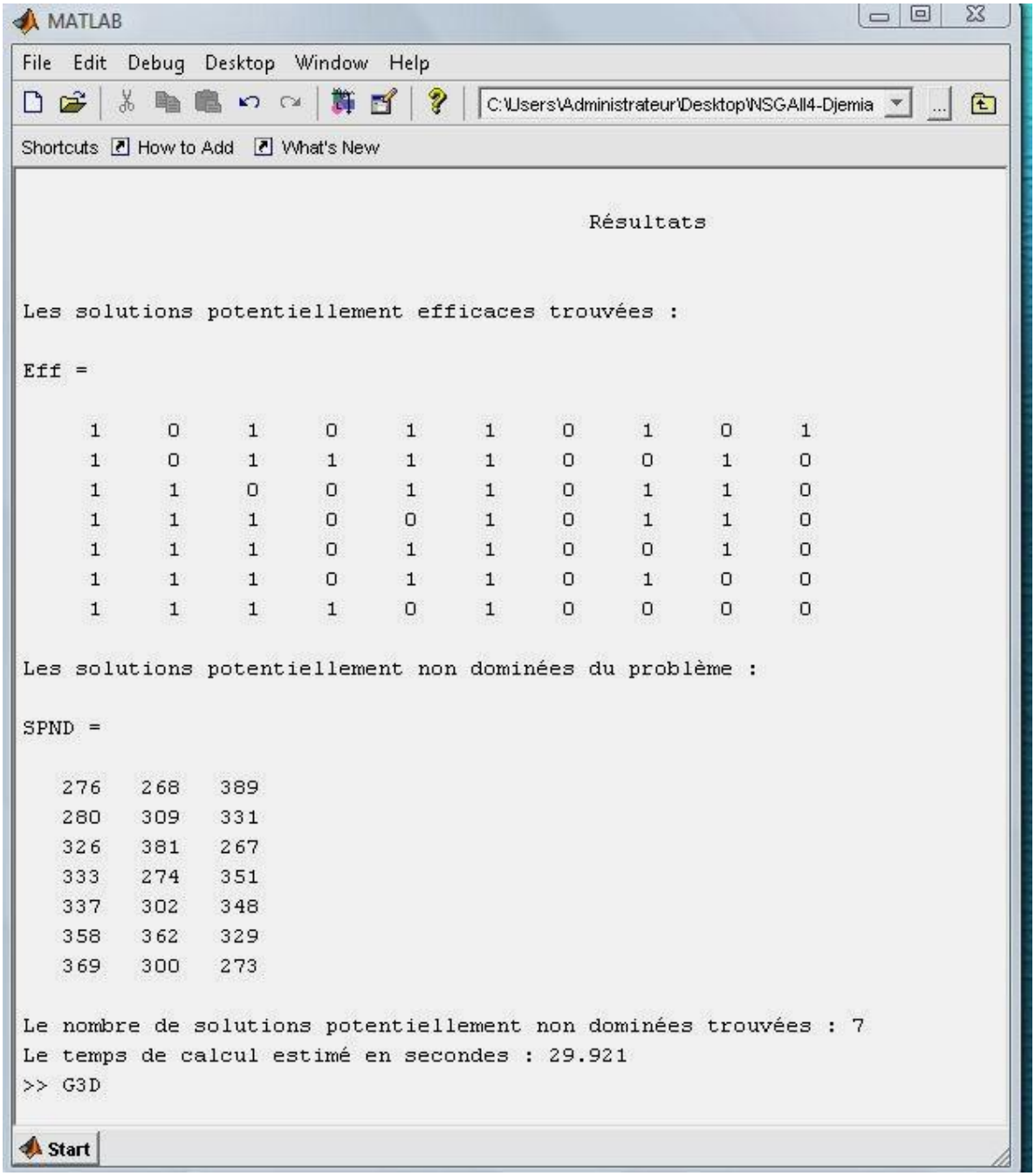
2-Après avoir rentré les données on construit la matrice A, C et le vecteur b d'une façon aléatoire et on obtient la solution efficace, non dominée et le temps estimé en secondes, on obtient la deuxième configuration :



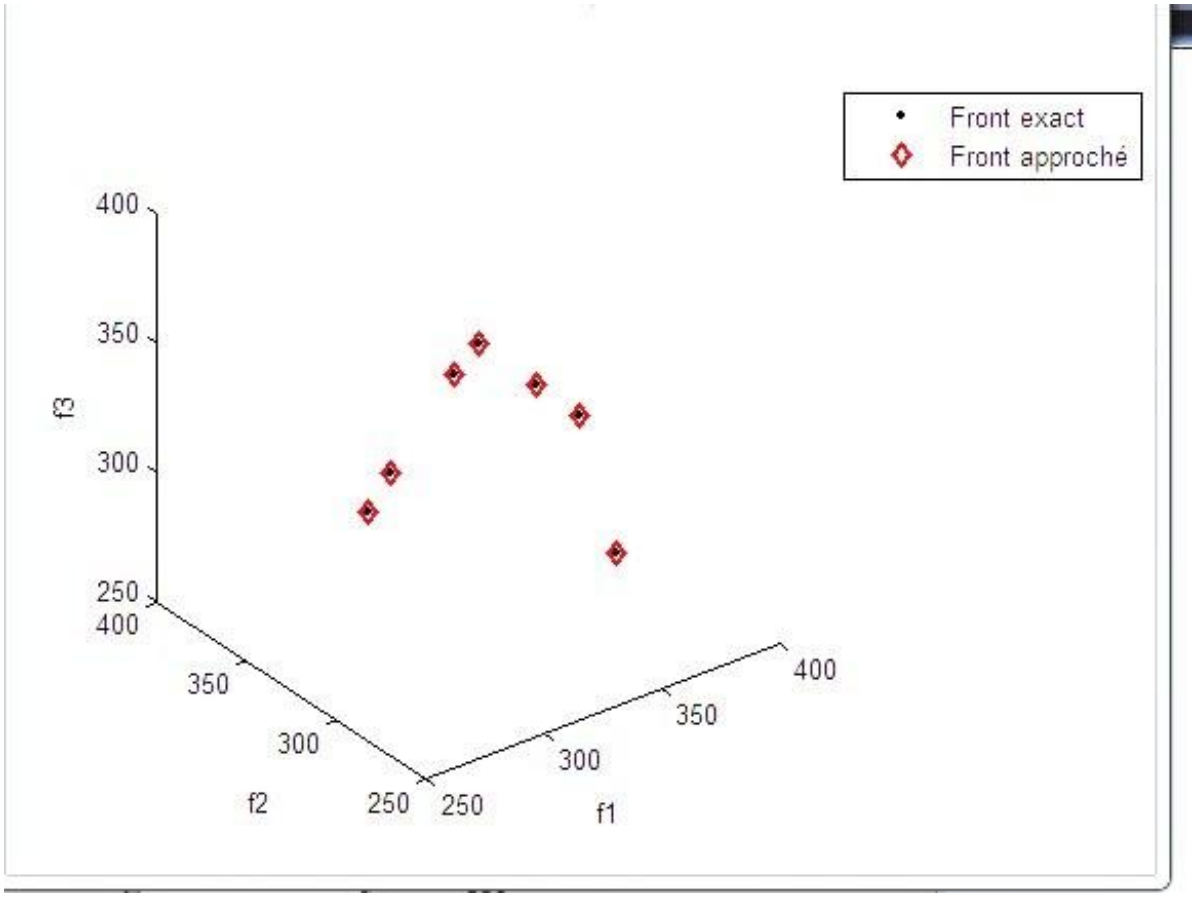
3- Maintenant on fait rentré les mêmes variables de l'étape 1 avec un algorithme NSGAII on obtient la figure suivant :



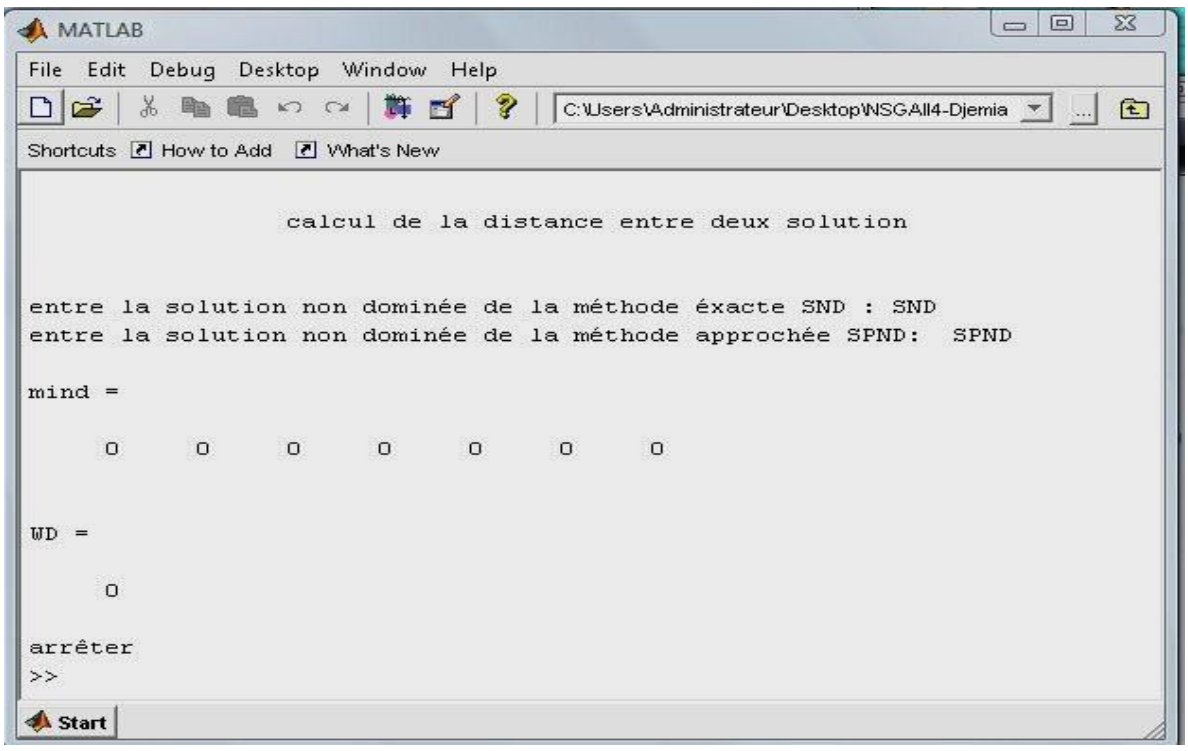
4- On clique sur DEBUG qui se trouve dans la boite à outil du Work puis sur Run on obtient la solution efficace, la solution potentiellement non dominée et le temps estimé en secondes d'après la configuration suivant :



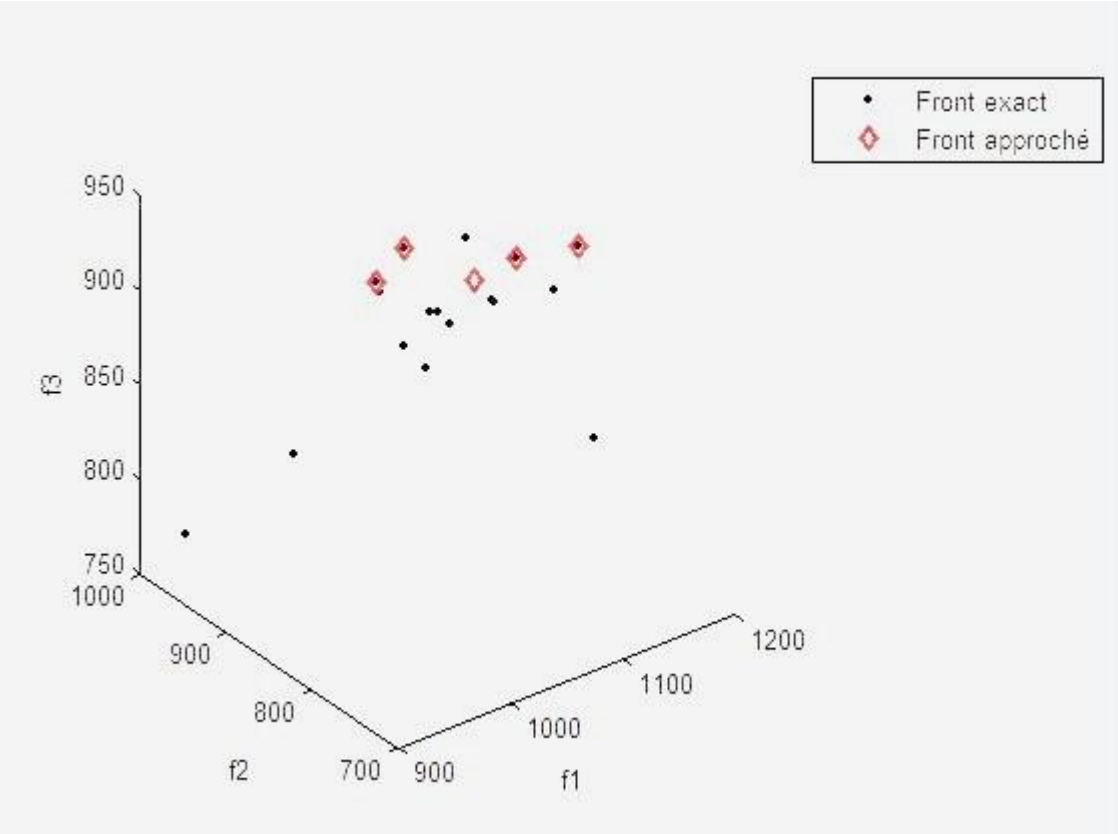
5- Après cette exécution on calcule la mesure de solution potentiellement non dominée par rapport à la solution non dominée qui sera représentée par le graphe donnant le front exacte et le front approchée (figure ci-dessous) :

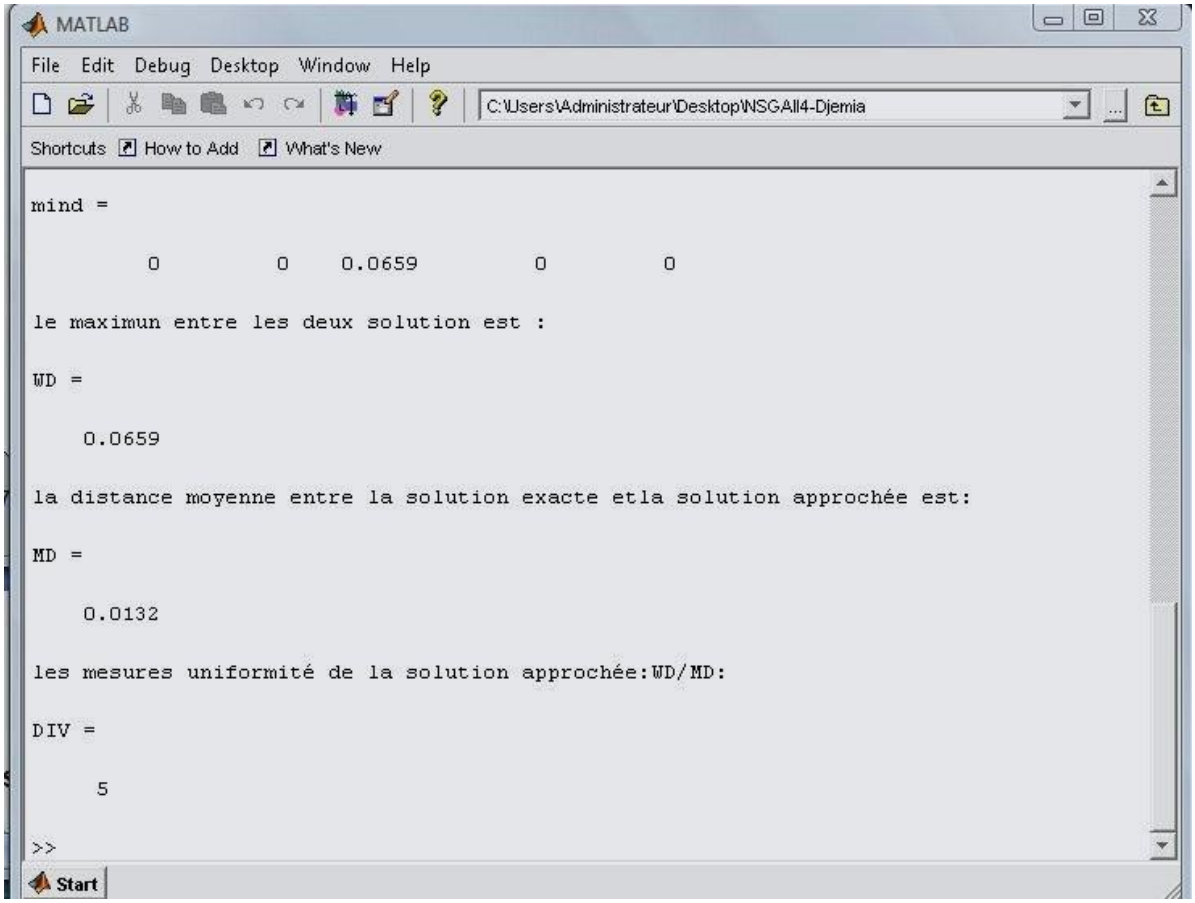


On remarque d'après cet exemple toutes les solutions approchées sont confondues sur la solution exacte ce qui montre que la distance entre les deux solutions est nul. La figure suivante le montre bien :



6- dans le cas où le nombre de solution approchée est différent de le nombre de la solution exacte (par exemple on à 17 solutions non dominée (exactes) et 5 solutions potentiellement non dominée (approchées)) et après le calcule de la mesure de solution potentiellement non dominée par rapport à la solution non dominée on obtient les résultats suivants :





4.3. Présentation des résultats

Dans les tableaux qui suivent, nous illustrons pour des instances et la distance de taille différentes, l'apport de la méthode NSGA-II dans l'optimisation multiobjectifs pour le cas du problème 01-MOKP en termes de gain en temps de calcul réalise et la qualité des solutions trouvées.

Tableau 4.1 : Valeurs moyennes du temps CPU (en secondes) pour NSGA-II [croissements 2-points] et la méthode exacte d'Ozlen et al.[94].

Instance du MOKP		Nombre de génération	Taille de population	CPU de la méthode exacte en secondes	CPU de la méthode NSGAII en secondes
m	n				
2	20	120	150	0.187	21.45
	30	120	150	0.468	25.678
	40	120	200	1.607	27.566
	60	120	150	5.569	31.527
	70	100	150	26.723	14.29
	80	120	150	49.468	37.861
	100	120	150	174.767	66.269
3	20	120	200	20.342	57.346
	40	120	200	119.275	53.164
	60	120	150	874.474	42.588
	80	100	200	6845.389	25.303
	100	100	150	56934.058	5.242
4	20	100	150	Plus de deux jours	3.026
	40	100	150	Plus de deux jours	3.007

Tableau 4.2 : Représente les métriques de problème 01-MOKP.

n	m	Nbr.SND	Nbr.SPND	M	MD	WD	Div=WD/MD
2	20	12	8	7/12	0.0016	0.0125	8
	30	18	15	13/18	0.0030	0.0406	13.404
	40	44	15	4/44	0.0079	0.0190	2.4107
	60	37	12	6/37	0.0048	0.0147	3.0638
	80	103	21	8/103	0.0021	0.0092	4.4453
	100	192	40	16/19	0.0020	0.0077	3.8710
3	20	120	88	82/120	0.0016	0.0400	25.010
	40	180	58	53/180	0.0011	0.0107	17.795
	60	463	108	37/463	0.0107	0.0211	1.9776
	80	2147	63	15/2147	0.0082	0.0340	4.1397

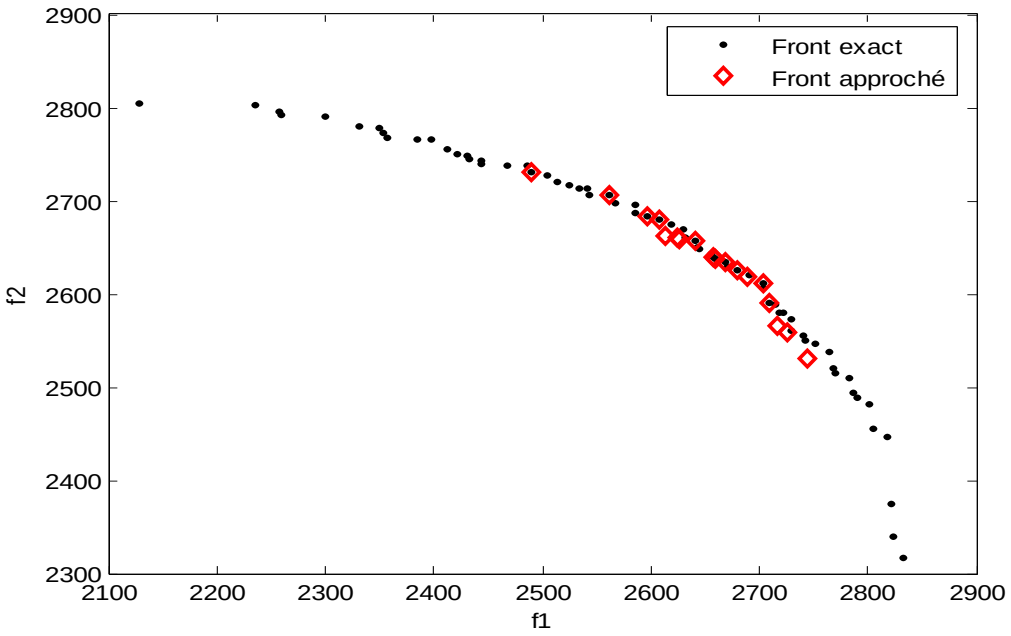


Figure 4.1 Front Pareto de 01-MOKP avec 70 variables (m=2, n=70)

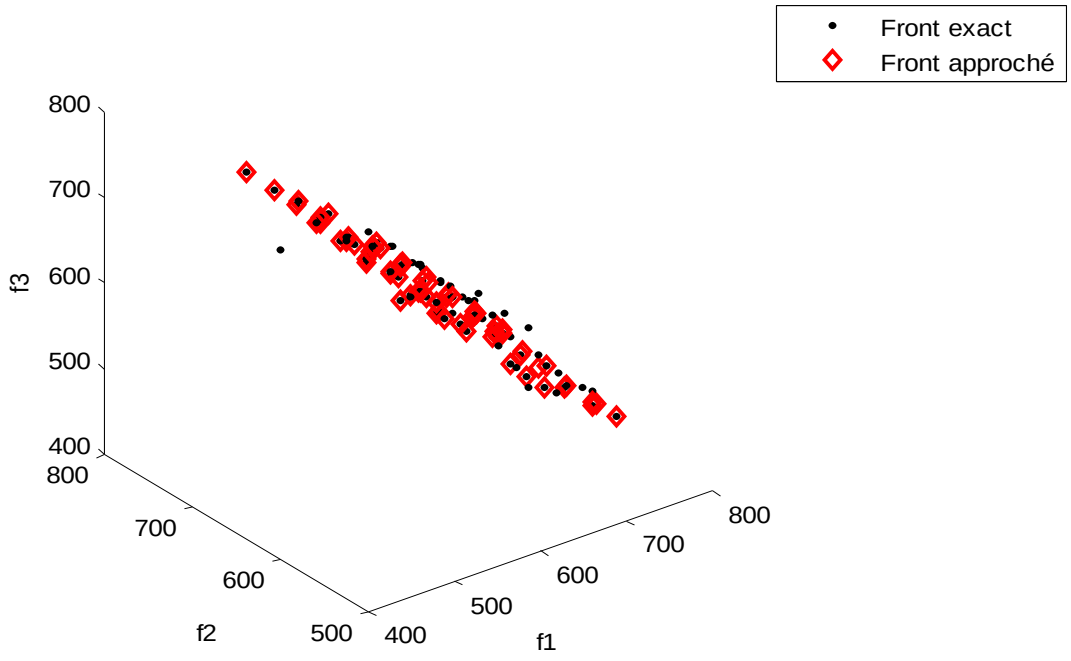


Figure 4.2 Front Pareto de 01-MOKP avec 20 variables (m=3, n=20)

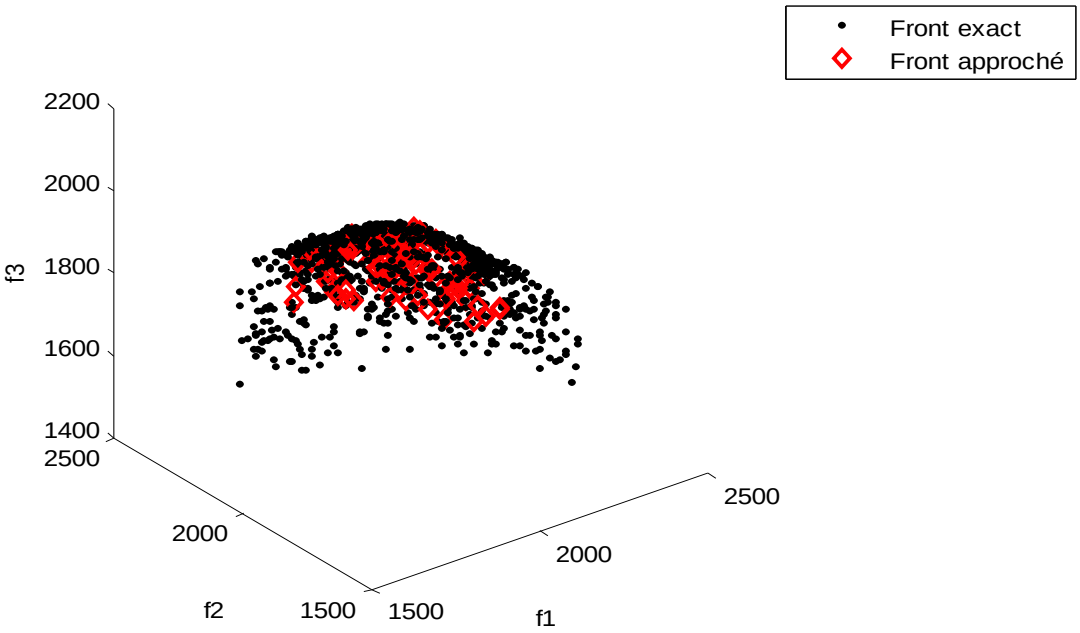


Figure 4.3 Front Pareto de 01-MOKP avec 60 variables (m=3, n=60)

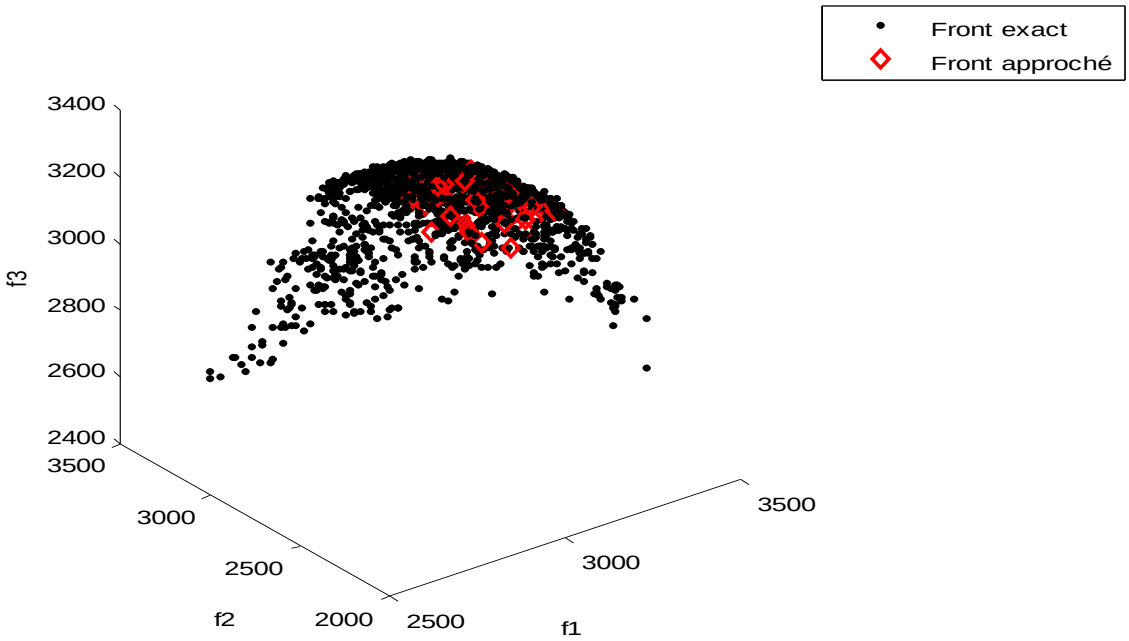


Figure 4.4 Front Pareto de 01-MOKP avec 80 variables (m=3, n=80)

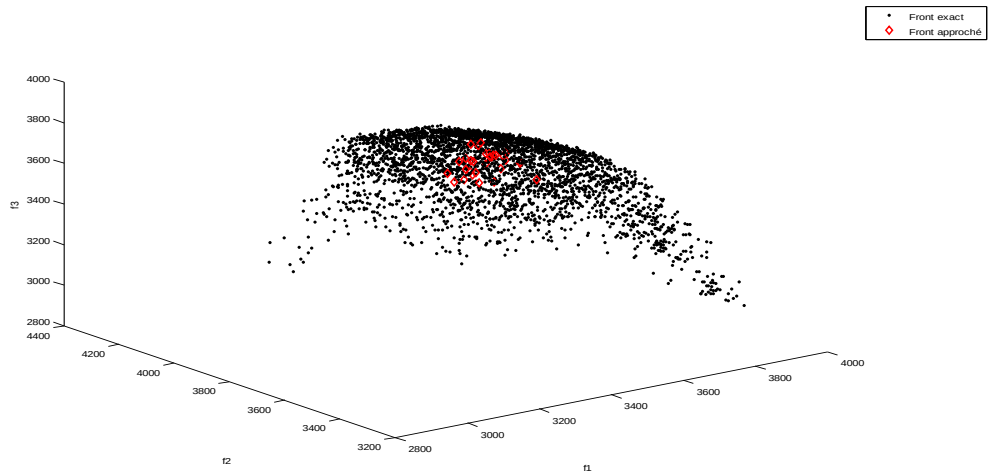


Figure 4. 5 Front Pareto de 01-MOKP avec 100 variables (m=3, n=100

.

Nous pouvons remarquer que le temps de résolution est consommé dans la méthode exacte en grand taille est supérieur au temps de résolution NSGAII et même d'après la représentation graphique obtenue des deux fronts (Pareto optimal et approchée). nous constatons que la méthode NSGA-II appliquée au problème de 01-MOKP génère une approximation du front Pareto de très bonne qualité du point de vue convergence des solutions potentiellement non dominées par rapport le front optimal.

4.4. Conclusion

Dans ce chapitre, nous avons présenté la méthode évolutionnaire NSGAII appliquée au problème de sac-à-dos multiobjectif en variables binaires.

Son intérêt réside via une étude empirique comparative, montre la qualité de l'approximation du front Pareto obtenu par l'algorithme NSGAII ainsi que l'influence de l'opérateur de croisement sur la performance d'un algorithme évolutionnaire multiobjectifs (MOEAs) en termes de convergence. D'après les résultats comparatifs que nous avons mentionnées précédemment, on conclut en comparant avec une méthode de résolution exacte [1], la méthode NSGA-II adapté au 01-MOKP permet d'obtenir un ensemble de solutions potentiellement non dominées de bonne qualité durant un temps d'exécution raisonnable et la distance entre les deux solutions tend vers zéro. Elle contribue à la résolution de

plusieurs problèmes NP-difficiles par exemple : problème de sac à dos multidimensionnelle multiobjectif [2] et donne de très bons résultats.

Conclusion

Dans la littérature relative à des problèmes d'optimisation multiobjectif, il ne s'agit plus de rechercher une solution optimale, mais un ensemble de solutions Pareto optimales qui se distinguent par les différents compromis réalisés entre les différentes fonctions objectifs considérées.

Ces dernières années les métaheuristiques, notamment les algorithmes évolutionnaires, ont permis l'élaboration de méthodes de résolution très performantes. Elles permettent de déterminer en une seule exécution une approximation de l'intégralité du front Pareto, et ceci même si les problèmes sont non convexes.

L'utilisation des algorithmes évolutionnaires « AEs » dans la résolution des problèmes multiobjectifs dépassent la simple recherche d'une seule bonne solution, il s'agit dans ce cas là de mettre en considération deux critères, la convergence de l'ensemble des solutions potentiellement non dominées vers le front Pareto et la diversification des solutions trouvées le long du front. Pour assurer ces deux critères, les algorithmes évolutionnaires multiobjectifs (MOEAs) font appel à des mécanismes spécifiques de recherche de solutions (le ranking**,** l'élitisme, le sharing et le crowding etc…). Les stratégies de diversification notamment le crowding sont des mécanismes déterminant de l'efficacité de la recherche.

L'adaptation de l'algorithme évolutionnaire (NSGAII) au problème de sac à dos multiobjectif en variables binaires est réalisée avec succès. Son implémentation met en évidence la convergence du front Pareto donnée par NSGAII vers celui donnée par la méthode exacte d'Ozlen et al. [94]. Ceci est rendu possible à l'aide du calcul de la distance de Tchebycheff séparant ses deux fronts.

Perspectives

L'implémentation de la méthode NSGAII au problème de sac à dos multiobjectif permet d'ouvrir plusieurs perspectives. Dans ce sens, nous envisageons d'examiner l'effet de certains paramètres et composants d'un algorithme évolutionnaire multiobjectif : le nombre de génération, la taille de population, la

probabilité de recombinaison, la probabilité de mutation, l'opérateur de sélection etc,… sur la performance du croissement 2-point. Afin d'avoir une idée claire sur l'intérêt générique du croissement 2-point, on peut également l'appliquer à d'autres problèmes de type problème d'optimisation multiobjectif (POMO) tout en élaborant des versions spécifiques à ces problèmes puisque de telles versions permettront probablement une meilleure performance que lorsqu'on applique seulement la version générale. Aussi, l'algorithme NSGAII, peut faire l'objet d'une étude comparative par rapport à un autre algorithme évolutionnaire, on peut citer l'exemple de l'algorithme évolutionnaire SPEA2 et enfin l'algorithme NSGAII, peut réaliser un schéma d'hybridation avec d'autres métaheuristiques de la littérature connues par leur bonne performance sur le problème de sac à dos multiobjectif.

REFERENCES

1. Chergui, M.E-A.," Contribution à la programmation non linéaire multicritère ", Thèse de Doctorat d'état en Recherche Opérationnelle USTHB, (octobre 2010).

2. Naimi, M., "Amélioration de la performance des algorithmes évolutionnaires multiobjectifs: application au problème de sac à dos multidimensionnel multiobjectif", Thèse de Doctorat d'état en Recherche Opérationnelle Casablanca, (Juillet 2008).

3. Mahdi, K., "L'optimisation multiobjectif et l'informatique quantique", Mémoire de Magistère en Information and Computation Mentouri-Constantine, (2006).

4. Moulai, M. "Optimisation multicritère fractionnaire linéaire en nombres entiers". Thèse de Doctorat d'état en Recherche Opérationnelle USTHB (octobre 2002).

5. Collette, Y. and Siarry, P. "Optimisation multiobjectif ". Eyrolles, (2002).

6. Talbi, E., "Métaheuristiques pour l'optimisation combinatoire multiobjectif : Etat de l'art ", Lille : LIFL, (2001).

7. Stewart, B.S., White, C.C., "Multiobjective A*, journal of association for computing machinery", vol 38, N°4, page 775, (1991).

8. Carraway, R., Morin, T., and Moskowitz, H., "Generalized dynamic programming for multicriteria optimization ", european journal of operational research, vol 44, (1990).

9. Haimes, Y., Ladson, L., and Wismer, D.,"On a bicriterion formulation of the problems of integrated system identification and system optimization". IEEE Transactions on System, Man and Cybernetics, 1, 276-297, (1971).

10. Serafini, P., "Simulated annealing for multiple objective optimization problems", Proceedings of the 9th International Conference on Multiple Criteria Decision Making, (1992), (pp. 87-96). Taipei.

11. Coello Coello, C.A., “An update survey of GA.-based multiobjective optimization techniques ", ACM Compt survey (2000), 32(2): 109-143.

12. Harfouch, H., " Contribution des métaheuristiques dans l’optimisation multiobjectif ", Mémoire de magistère, en Recherche Opérationnelle USTHB (Juin 2010).

13. Goldberg, D. E., and Richardson, J., "Genetic algorithms with sharing for multimodal function optimization", Proceedings of the First International Conference on Genetic Algorithms and their Applications, (1987), (pp. 41-49).

14. Srinivas, N., and Deb, K., "Multiobjective optimization using non dominated sorting in genetic algorithms". Evolutionary computation, (1994), 2 (3), 221-248.

15. Deb, K., "Multi-objective optimization using evolutionary algorithms". John Wiley and sons, (2001).

16. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Trans Evol Comput (2002), 6(2):182-97.

17. Schaffer, D., "Multiple objective optimization with vector evaluated genetic algorithm", Proceedings of the First International Conference on Genetic Algorithms, (1985), pp. 93-100.

18. Reeves, C.R., "Modern heuristic techniques for combinatorial problems", McGraw Hill, (1995) cite page 22.

19. Hao, J. K., and al., "Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes", Revue d'Intelligence Artificielle, 13(2):283-324, (1999) cite page 22.

20. Pirlot, M., and Teghem, C. J., "Optimisation approchée en recherche opérationnelle", (Traité IC2, Série Informatique et systèmes d'information).

Hermès Sciences, (2002), cite page 22.

21. Dréo, J., Pétrowski, A., Siarry, P. and Taillard, E. "Métaheuristiques pour l'optimisation difficile", Paris: Eyrolles. (2003).

22. Papadimitriou, C. H., and Steiglitz, K., "Combinatorial optimization: algorithms and complexity", Prentice-Hall, (1982).

23. Ehrgott, M., "Multicriteria optimization". Springer (2005).

24. Glover, F., "Future paths for integer programming and links to artificial intelligence", computers and operations research, (1986), 13 (5), 533-549.

25. Glover, F., and Laguna, M., "Tabu search. Kluwer academic publishers" (1997).

26. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization By Simulated Annealing", Journal of Science, 220 (4598), (1983), 671-680.

27. Engrand, P., "Approche d'optimisation multiobjectif basée sur l'adoucissement simulé, et son application à la gestion des combustibles nucléaires", Proceedings of the International Conference on Nuclear Engineering, (1997), (pp. 1-8).

28. Czyzak, P. and Jaszkiewicz, A., "A Pareto simulated annealing: a metaheuristic technique for multiple objective combinatorial optimizations", Journal of Multicriteria Decision Analysis, (1998). 7, 34-47.

29. Ulungu, E., Teghem, J., Fortemps, P., and Tuyttens, D., "MOSA method: a tool for solving multiobjective combinatorial optimization problems", Journal of Multicriteria Decision Analysis, 8 (4), 221-236.volume 1, (1998), pages 369-377.

30. Rao, A. R., and Arvind, N., "A scatter search algorithm for stacking sequence optimisation of laminate composites", Composite Structures, (2005), 70 (4), 383-402

31. Beausoleil, R. P., "MOSS: Multiobjective Scatter Search: applied to non-linear multiple criteria optimization", European Journal of Operational Research, (2006), 169 (2), 426-449.

32. Dorigo, M., and Stutzle, T., "Ant colony optimization", MIT Press, (2004).

33. Dorigo, M., and Caro, G. D., "The ant colony optimization metaheuristic", New Ideas in Optimization, (1999), 11-32.

34. Mariano, C. E., and Morales, E. M., "MOAQ an ant-Q algorithm for multiple objective optimization problems", Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 1999). Orlando (USA).

35. Gambardella, L., Taillard, M E., and Agazzi, G., "A multiple ant colony system for vehicule routing problems", with time windows; Corne. D., Dorigo. M. and F. Glover, editors, New ideas in Optimization, McGraw-Hill, (1999), page 63, 76.

36. Iredi, S., Merkle, D., and Middendorf, M., "Bi-criterion optimization with multi-colony ant algorithms". First International Conference on Evolutionary Multiobjective Optimization (2001), pp. 359-37.

37. Doerner. K and al., "Pareto ant colony optimization with ILP preprocessing in multiobjective project portofolio selection", European Journal of Operational Research, vol. 171, (2006), pp. 830-841.

3**8**. Kennedy, J., Eberhart, R., and Shi, Y., "Swarm intelligence. Morgan Kaufmann", (2001).

39. Coello, C. A., Pulido, G. T., and Lechuga, M. S., "Handling multiple objectives with particle swarm optimization", IEEE Transactions on Evolutionary Computation, (2004), 8 (3), 256-279.

40. Baumgartner, U., Magele, C., and Renhart, W., "Pareto optimality and particle swarm optimization", IEEE Transactions on Magnetics, (2004), 40

(2), 1172-1175.

41. Ho, S.J., Ku, W.Y., Jou, J.W., and Hung, M. H.,"Intelligent particle swarm optimization in multi-objective problems", Lecture Notes in Artificial Intelligence, (2006), 3918, 790-800.

42. Larranaga, P., and Lozano, J. A., "Estimation of distribution algorithms: new tools for evolutionary computation", Kluwer Academic Publishers (2001).

43. Coello, C. A. and Cruz, N.," Solving multiobjective optimization problems using an artificial immune system", Genetic Programming and Evolvable Machines, 6 (2), (2005), 163-190.

44. Moscato, P., "On evolution, search, optimization, genetic algorithms and martial arts: towards Memetic algorithms", Technical Report, California Institute of Technology (1989).

45. Basseur, M., "Conception d'algorithmes coopératifs pour l'optimisation multiobjectif ", application aux problèmes d'ordonnancement de type flowshop. Thèse de Doctorat, Université des Sciences et Technologies de Lille, (2005).

46. Back, T. Fogel, D.B., "Michalewicz, Z. and Baeck T. Handbook of Evolutionary Computation", Institute of Physics Publishing and Oxford University Press, (1997).

47. Darwin, C., "The origin of species by means of natural selection", Mentor Reprint New York, (1859).

48. Holland, J.H., "Outline for a logical theory of adaptive systems", Journal of the Association Computing Machinery3, (1962).

49. Holland, J.H., "Adaptation in natural and artificial systems", Ann Arbor, MI: MIT Press, (1975).

50. De Jong, K. A., "An analysis of the behavior of a class of genetic adaptive systems", Ph. D. Thesis, University of Michigan, (1975).

51. Goldberg, D.E., "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley, (1989).

52. Rechenberg, I., "Cybernetic solution path of an experimental problem", Royal Aircraft Establishment. Farnborough (UK): (1965), Library Translation.

53. Schwefel, H. P., "Numerical optimization of computing models", Chicester (UK): (1981), Wiley.

54. Koza, J. R., "Genetic Programming: on the programming of computers by means of natural evolution", Massachussets: MIT Press (1992).

55. Teller, A., and Veloso, M., "PADO: a new learning architecture for object recognition", Symbolic Visual Learning, (1996), 81-116.

56. Nordin, P., "A compiling genetic programming system that directly manipulates the machine code", Advances in Genetic Programming, (1994), 311-332.

57. De Jong, K. A., and Sarma, J., "On decentralizing selection algorithms", Proceedings of the 6th International Conference on Genetic Algorithms (1995), pp. 17-23.

58. De Jong, K. A., "An analysis of the behavior of a class of genetic adaptive systems", Ph. D. Thesis, University of Michigan, (1975).

59. De Jong, K. A., and Spears, W. M., “An analysis of multi-point crossover", Foundations of Genetic Algorithms, (1991), 301-315.

60. Syswerda, G.,"Uniform crossover in genetic algorithms", Proceedings of the International Conference on Genetic Algorithms, (1989), (pp. 2-9). Morgan

Kaufmann.

61. Goldberg, D. E., Deb, K., and Clark, J. H., "Genetic algorithms: noise and the sizing of populations", Complex Systems, (1992), 6 (4), 333-362.

62. Barichard, V., and Hao, J. H., “Genetic Tabu search for the multiobjective knapsack problem”. Tsinghua Science and Technology, (2003), 8 (1), 8-13.

63. Zitzler, E., and Thiele, L. “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach”. IEEE Transactions on Evolutionary Computation, (1999), 3, 257-271.

64. Zitzler, E., Deb, K., and Thiele, L., “Comparison of multiobjective evolutionary algorithms: empirical results”, Evolutionary Computation Journal, 8(2), (2000), 125-148.

65. Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., “A fast and elitist multiobjective genetic algorithm for multi-objective optimization: NSGA-II” , In Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI), (2000), pages 849,858.

66. Fonseca, C., and Fleming, P., “Genetic algorithms for multi-objective optimization: formulation, discussion and generalization", In Proceedings of The Fifth International Conference on Genetic Algorithms, (1993), pages 416-423 cite page 65, 66.

67. Roudenko, O., "Application des Algorithmes Evolutionnaires aux Problèmes d’Optimisation Multi-objectif avec Contraintes", Thèse de Doctorat. Ecole Polytechnique, (2004), Paris, France.

68. Zitzler, E., Laumanns, M. and Thiele, L., "SPEA2: improving the strength Pareto evolutionary algorithm", Technical Report, Swiss Federal Institute of Technology, (2001).

69. Ishibuchi, H., Narukawa, K., Tsukamoto, N., and Nojima, Y., “An Empirical Study On Similarity-Based Mating for Evolutionary Multiobjective

Combinatorial Optimization". European Journal of Operational Research, (2008), 188, 57-75.

70. Coello, C. A., Pulido, G. T. and Lechuga, M. S., "Handling Multiple Objectives With Particle Swarm Optimization", IEEE Transactions on Evolutionary Computation, (2004), 8 (3), 256-279.

71. Liefooghe, A., Jordan, L. and Talbi, E.G., "Metaheuristics and Cooperative Apporaches for the bi-objective Ring Star problem", Computers and Operations Research, (2010), 37(6):1033-1044.

72. Corne, D. W., Knowles, J. D., and Oates, M. J., "The Pareto envelope-based selection algorithm for multiobjective optimization", Proceedings of the 6th International Conference on Parallel Problems Solving from Nature, (2000), (pp. 839-848).

73. Knowles, J and Corne, D., "Bounded Pareto archiving: theory and practice", Methaheuristics for multiobjective optimisation, volume 535, (2004), Berlin, Germany.

74. Ehrgott, M.,"A Discussion Of Scalarization Techniques For Multiple Objective Integer Programming", Annals of Operations Research, 147(1) : 343- 360, (2006).

75. Wilbaut, C., "Heuristiques Hybrides Pour La Résolution De Problèmes En Variables 0-1 Mixtes", Thèse de Doctorat, Université de Valenciennes et du Hainaut Cambrésis, (Septembre 2006).

76. Martello, S. and Toth, P., "Knapsack Problems: Algorithms and Computer Implementations", John Wiley sons, (1990).

77. Horn. J, Nafpliotis.N, and Goldberg, D.E., "A Niched Pareto Genetic Algorithm for Multiobjective Optimization", In: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, 27-29 June, (1994).

78. Chergui, M., and Moulai, M., "An exact method for a discrete multiobjective linear fractional optimization", Journal of Applied Mathematics and Decision Sciences, vol. (2008), Article ID 760191, 12 pages, doi: 10.1155.2008.76019.

79. Zhou, G. and Gen, M., “Genetic Algorithm Approach on Multi-criteria Minimum Spanning Tree Problem”, European Journal of Operational Research, (1999), 114:141–152.

80. AARTS, E., RÉDS, J. and LENSTRA, K., “Local Search in Combinatorial Optimization”, Princeton University Press, (2003).

81. Fleszar, K. and Hindi, K. S., "Fast, effective heuristics for the 0-1 multi-dimensional knapsack problem", Computers Operations Research, (2009), 36(5):1602–1607.

82. Cappanera, P. and Trubian, M. “A local-search-based heuristic for the demand constrained multidimensional knapsack problem", INFORMS Journal on Computing, (2005), 17(1):82–98.

83. Jorge, J., "Nouvelles propositions pour la résolution exacte du sac a dos multiobjectif unidimensionnel en variables binaires". Thèse de Doctorat, Université de Nantes, (Mai 2010).

84. Meunier, H., "Algorithmes évolutionnaires parallèles pour l’optimisation multiobjectif de réseaux de télécommunications mobiles", PhD Thesis, Thèse pour obtenir le grade de docteur. Université de Lille, (2002).

85. Visée, M., Teghem, J., Pirlot, M. and Ulungu, E. L., “Two phases method and branch and bound procedures to solve the bi-objective knapsack problem”, Journal of Global Optimization, 12 (2), (1998), 139-155.

86. Gandibleux, X. and Fréville, A. "Tabu search procedure for solving the 0/1 multiobjective knapsack problem: the two objectives case". Journal of Heuristics, 6, (2000), 361-383.

87. Ehrgott, M. and Gandibleux, X. “Hybrid Metaheuristics for multi-objective combinatorial optimization”, in this chapter we give an overview over approximation methods in multi-objective, (2008).

88. Gandibleux, X, Morita, H., and Katoh, N., “The Supported Solutions Used as Genetic Information in a Population Heuristic”, In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, Evolutionary Multi-Criterion Optimization, vol. 1993 of LNCS, pp. 429–442. Springer, (2001).

89. Delorme, X., Gandibleux, X., and Degoutin, F., “Evolutionary, Constructive and Hybrid Procedures for the bi-Objective Set Packing Problem”, EJOR. 204 (2), 206-217, (2010).

90. Gandibleux, X., Morita, H., and Katoh, N,” Evolutionary Operators Based On Elite Solutions for biobjective Combinatorial Optimization". In C. Coello Coello and G. Lamont, editors, Applications of Multi-Objective Evolutionary Algorithms, (2004), ISBN 981-256-106-4.

91. Pasia, J., Gandibleux, X., Doerner, K., and Hartl, R., “Local Search Guided by Path Relinking and Heuristic Bounds”, In Sh. Obayashi, K. Deb, C. Poloni, T. Hiroyasu and T. Murata Editors, Evolutionary Multi-Criterion Optimization, vol. 4403 of LNCS, pp. 501-515, Springer, (2007).

92. Gandibleux, X., and Chamayou, C., “Potential Efficient Solutions of a bi-objective Telecommunication Network Expansion Planning Problem", MIC2007: 7th Metaheuristics International Conference. Montreal, Canada, June 25–29, (2007).

93. Knowles, J.D., and Corne, D.W., “On Metrics for Comparing Non-Dominated Sets”, In Congress on Evolutionnary Computation, IEEE Press, (2002), pages 711-716.

94. Özlen, M., and Azizoğlu, M., "Multi-objective integer programming: a general approach for, generating all non-dominated solutions". European Journal of

Operational Research, Vol. 199, pp.25.35, (2009).

95. Biran, A., and Breiner, M., "Matlab pour l'ingenieur ", Version 6 et 7 Person Education, (2004).