N° d'ordre : 24/2010-M/M

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DES SCIENCES ET DE LE TECHNOLOGIE
HOUARI BOUMEDIENE
FACULTÉ DES MATHÉMATIQUES

MÉMOIRE

Présenté pour l'obtention du diplôme de MAGISTER
En : Mathématiques
Spécialité : Recherche Opérationnelle
Option : Mathématiques de Gestion

Par : **HARFOUCHE Hassiba**

Sujet

CONTRIBUTION DES MÉTAHEURISTIQUES DANS L'OPTIMISATION MULTIOBJECTIF

Soutenu publiquement, le *15/06/2010*, devant le jury composé de :

Mr A. BERRACHEDI, Professeur à l'USTHB	Président
Mr M. MOULAÏ, Professeur à l'USTHB	Directeur de mémoire
Mr H. AÏT HADDADENE, Professeur à l'USTHB	Examinateur
Mr S. BOUROUBI, Professeur, USTHB	Examinateur

À toute la 6ème promo de Management général de l'ISGP.
À mes chères parents et grands-parents.
À mes sœurs et frères.
À toute ma famille.

Remerciements

Je tiens d'abord à remercier Dieu de m'avoir guidée et menée jusqu'ici.

Je remercie le Professeur Moulai Mustapha, pour avoir bien voulu être mon directeur de mémoire et pour la confiance qu'il m'a accordées.

Je remercie Monsieur H. Berrachedi, Professeur à l'USTHB, qui m'a fait l'honneur d'accepter de présider le jury de ce mémoire.

Que Messieurs H. Ait Haddadene et S. Bouroubi, Professeurs à l'USTHB, trouvent ici l'expression de ma reconnaissance pour avoir accepté de juger ce travail en tant qu'examinateurs.

Je remercie tout particulièrement M^{elle} Ait Mahdi meryam qui m'a aidé et encouragé aux moments où j'en avais besoin. Sa gentillesse, ses conseils et sa disponibilité m'ont permis d'avancer dans le bon sens.

J'adresse mes vifs remerciements à monsieur L. BENAISSA pour ses conseils et encouragement.

Ces remerciements ne seraient pas complets sans y avoir associé ma famille. Je remercie tout d'abord mes parents pour leur soutien moral, leur présence et leur écoute. Ils ont su m'épauler et me faire confiance pendant ces années. Je remercie également mes sœurs nabila et halima ainsi que leurs maris. Mes frères Sofiane, badr dine et hamza sans oublier mes belles sœurs hassiba et fadila. Je remercie aussi ma tante ouahiba pour ses conseils précieux.

Merci à tous.

Table des matières

Table des figures

Introduction générale

De très nombreux problèmes du monde industriel sont de nature à la fois multi-crières (qualité de service, coûts, quantité à produire etc.) et de grande dimension. Durant longtemps, la plupart des travaux d'optimisation étaient dédiés à l'optimisation d'une seule fonction objectif. Or la majorité des problèmes rencontrés en pratique sont rarement mono-objectif, il y a généralement plusieurs critères souvent conflictuels à satisfaire simultanément. L'optimisation multiobjectif s'intéresse à résoudre ce type de problèmes. Ses racines remontent au 19$^{\text{ième}}$ dans les travaux en économie de Edgeworth et Pareto[1].

La notion de solution optimale disparaît pour les problèmes de nature multiobjectif au profit de la notion de *compromis*. Trouver un compromis entre les divers objectifs du problème a donné lieu à la définition de l'ensemble de solutions Pareto optimales communément appelé *front Pareto*, connu aussi sous le nom de l'ensemble des solutions non dominées.

Pour la résolution des problèmes d'optimisation, il existe de nombreuses méthodes de résolution. Ces méthodes se classent en deux catégories bien distinctes. D'une part les méthodes exactes qui cherchent à trouver de manière certaine la solution optimale et d'autre part les méthodes approchées. La majorité des applications réelles ne peuvent pas être résolues de manière pratique par des algorithmes exacts car leur résolution exacte exige souvent un temps de calcul prohibitif. Dans le cas d'instances de grande taille, le temps de calcul nécessaire pour la résolution peut devenir si important que l'algorithme développé devient inutilisable en pratique. Cette problématique se complique davantage dans le contexte multiobjectif où on doit trouver tout un ensemble de solutions qui peut lui-même avoir une taille trop large pour qu'on puisse utiliser un algorithme exact permettant d'identifier toutes les solutions du front Pareto. Dans de telles situations, les méthodes approchées ou métaheuristiques constituent une alternative indispensable et complémentaire.

Ces dernières années, on assiste à un accroissement d'intérêt porté sur l'utilisation de métaheuristiques pour la résolution de problèmes multiobjctifs. Parmi ces métaheuristiques, on trouve la famille d'algorithmes évolutionnaires multiobjectifs qui ont connu un grand succès dans l'optimisation multiobjectif. Ces méthodes qui utilisent des mécanismes spécifiques basés sur le principe d'évolution et travaillant avec une population de solutions, sont les mieux adaptées pour générer des fronts de solutions. Coello Coello, Veldhuizen et Lamont [2] ont publié un livre qui montre le spectre des applications multiobjectifs abordées par les algorithmes évolutionnaires. Ces algorithmes, ont permis l'élaboration de méthodes de résolution très performantes puisqu'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, ce qui n'est pas le cas pour la plupart des approches basées sur la transformation d'un problème multiobjectif en un problème mono-objectif qui sont sensibles au paysage du front Pareto. Les méthodes d'agrégation linéaire, par exemple, ne trouvent que les solutions supportées (i.e solutions appartenant à l'enveloppe convexe des solutions réalisables). De plus, pour trouver différentes solutions Pareto optimales, le problème doit être résolu plusieurs fois, d'où le coût associé à cette classe d'algorithmes.

À travers ce mémoire nous montrons l'apport des métaheuristiques dans le domaine d'optimisation multiobjectif de manière générale et nous nous intéressons aux algorithmes évolutionnaires multiobjectifs de manière particulière.

Dans le cadre de notre travail, nous avons adopté l'approche Pareto[1] nommée *Fast Nondominated Sorting Genetic Algorithm (NSGA-II)* pour la résolution du problème de sac à dos multiobjectif en variables binaires (*MultiObjective Knapsack Problème (01-MOKP)*) qui appartient à la classe de problèmes d'optimisation combinatoire. Cette classe rassemble des problèmes qualifiés de difficiles, car ce sont des problèmes pour lesquels on ne connaît pas d'algorithme qui fournisse la solution optimale en un temps d'exécution raisonnable. Nous nous intéressons particulièrement à l'opérateur de croisement qui fait partie du processus d'évolution de l'algorithme. Notre objectif consiste à étudier la performance (au sens de la qualité des solutions générées) de l'algorithme *NSGA-II* avec un et deux points de croisement.

[1]Elle utilise directement la notion d'optimalité Pareto dans le processus de recherche.

Organisation de la thèse

Ce mémoire est décomposé en quatre chapitres :

Au chapitre 1, nous présentons l'optimisation multiobjectif. Nous exposons quelques notions fondamentales concernant l'optimisation multiobjectif telles que la dominance, la surface de compromis. Nous décrivons aussi les principales approches de résolution pour ces problèmes.

Dans le deuxième chapitre, nous donnons une description de quelques métaheuristiques existantes dans la littérature en mettant l'accent sur les algorithmes génétiques et les mécanismes spécifiques aux algorithmes évolutionnaires dans le cadre d'optimisation multiobjectif.

Le troisième chapitre est consacré aux algorithmes évolutionnaires multiobjectifs où nous décrivons en détail les méthodes évolutionnaires les plus célèbres dans le contexte multiobjectif.

Dans le quatrième chapitre, nous décrivons le problème de sac à dos multiobjectif en variables binaires qui va servir comme exemple d'application de l'algorithme *NSGA-II*, puis nous discutons les résultats numériques issus de l'implémentation de ce dernier.

Enfin, nous finissons par une conclusion contenant une synthèse sur les différentes méthodes évolutionnaires utilisées pour la résolution de problèmes multiobjectifs ainsi qu'une analyse comparative des résultats obtenus par l'algorithme *NSGA-II* appliqué au problème *01-MOKP* avec un et deux points de croisement.

Chapitre 1

Optimisation multiobjectif

Dans ce chapitre, nous introduisons les notations et concepts fondamentaux liés aux problèmes d'optimisation multiobjectif. Puis, nous décrivons sommairement quelques méthodes de résolution existantes dans la littérature en apportant un regard critique sur chacune d'elles. Le contenu de ce chapitre est basé principalement sur les ouvrages [4],[3].

1.1 Introduction

L'optimisation multiobjectif cherche à optimiser simultanément plusieurs critères souvent contradictoires. Il ne s'agit plus dans ce cas de trouver une solution optimale mais un ensemble de solutions représentant un compromis acceptable entre les différents objectifs contradictoires et connu comme l'ensemble des solutions Pareto optimales (EPO). Le premier but dans la résolution d'un problème multiobjectif (MOP) est d'obtenir l'ensemble EPO ou bien échantillonner des solutions diversifiées dans cet ensemble. La détermination de ce dernier n'est qu'une première phase dans la résolution pratique des $MOPs$, qui nécessite dans un deuxième temps le choix d'une solution à partir de cet ensemble suivant les préférences du décideur.

1.2 Concepts de base

Problème d'optimisation multiobjectif

Un Problème multiobjectif ou multicritère peut être définit comme un problème dont on cherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectifs.
Un problème multiobjectif consiste à optimiser (maximiser ou minimiser) simultanément r fonctions réelles notées f_k, $k = \overline{1,r}$, appelées critères souvent contradictoires, sur un ensemble d'actions S. Ce problème peut être formulé mathématiquement comme suit :

$$(MOP)\begin{cases} \text{“}optimiser\text{”} & (f_1(x), f_2(x), \ldots, f_r(x)) \\ t.q. & x \in S \end{cases}$$

où $S = \left\{x \in \mathbb{R}^n \mid g_j(x) \leq 0,\ j = \overline{1,m}\right\}$; f_k et g_j sont des fonctions à valeurs réelles du vecteur de décision $x \in \mathbb{R}^n$, $\forall\, k = \overline{1,r}$, $\forall\, j = \overline{1,m}$.

Remarque

Le symbole “ ” signifie qu'il n'est généralement pas possible de trouver dans S une action qui optimise simultanément les r critères. Il est remarquable que de cette façon, un problème multicritère est correctement formulé par rapport à la réalité concernée par le problème de décision. Il faut donc déterminer une action $x^* \in S$ telle que, en regard des actions de S, le vecteur $f(x^*) = (f_1(x^*), \ldots, f_r(x^*))$ est bon, acceptable selon les préférences du décideur, à condition de se doter d'un cadre décisionnel donnant une

signification à cette notion. L'action x^* est appelée souvent solution de meilleur compromis.

Dans le cadre de l'optimisation multiobjectif, le plus souvent le décideur raisonne plutôt en termes d'évaluation d'une solution sur chaque critère et se place naturellement dans l'espace des critères. L'ensemble Y représente les points réalisables dans l'espace des critères et $y = (y_1, \ldots, y_r)$ avec $y_i = f_i(x)$ représente un point de l'espace des critères. On impose une relation d'ordre partiel sur cet ensemble de points, appelée *relation de dominance*.

Sans perte de généralité, nous supposerons par la suite que nous considérons des problèmes de minimisation sauf indication contraire.

Dominance

Définition 1.1. Soient deux vecteurs critères $y, y' \in Y$. On dit que y domine y' si et seulement si $y \leq y'$ et $y \neq y'$ (i.e. $y_i \leq y'_i$ pour tout $i = \overline{1,r}$ avec au moins une inégalité stricte).
Si y domine y', alors y est au moins aussi bon que y' sur tous les critères et meilleur que lui sur au moins un critère.

Dominance forte

Définition 1.2. Soient deux vecteurs critères $y, y' \in Y$. On dit que y domine fortement y' si et seulement si $y < y'$ (i.e. $y_i < y'_i$, pour tout $i = \overline{1,r}$).
Si y domine fortement y', alors y est meilleur que y' sur tous les critères.

Efficacité

Définition 1.3. Une solution $x^* \in S$ est une solution efficace s'il n'existe pas de $x \in S$ telle que $F(x)$ domine $F(x^*)$.

Un point est efficace si son image par F est un vecteur critère non dominé. Le terme efficacité est aussi connu sous Pareto optimalité ou non infériorité.

Efficacité faible

Définition 1.4. Une solution $x^* \in S$ est une solution faiblement efficace s'il n'existe pas de $x \in S$ telle que $F(x) < F(x^*)$.

Une solution est faiblement efficace si son vecteur critère n'est pas fortement dominé.

Efficacité forte

Définition 1.5. Une solution $x^* \in S$ est une solution fortement efficace s'il n'existe pas de $x \in S$ telle que $x \neq x^*$ et $F(x) \leq F(x^*)$.

Une solution x est fortement efficace s'il n'existe pas une autre solution telle que le vecteur critère, qui lui est associé, soit aussi bon que celui de x.

Remarquons que l'efficacité forte implique l'efficacité qui implique à son tour l'efficacité faible.

1.3 Surface de compromis

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

1.3.1 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 objectifs, elles sont équivalentes dans l'espace objectif, 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 :

- 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.
- Au contraire, lorsque la frontière Pareto comporte peu de solutions, afin d'avoir une bonne représentation de l'ensemble des solutions non dominées, il sera intéressant de rechercher les solutions de même valeur.

Nous parlerons alors de recherche du **front minimal**, dans le premier cas, et du **front maximal complet**, dans le second.

Représentation de la surface de compromis

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

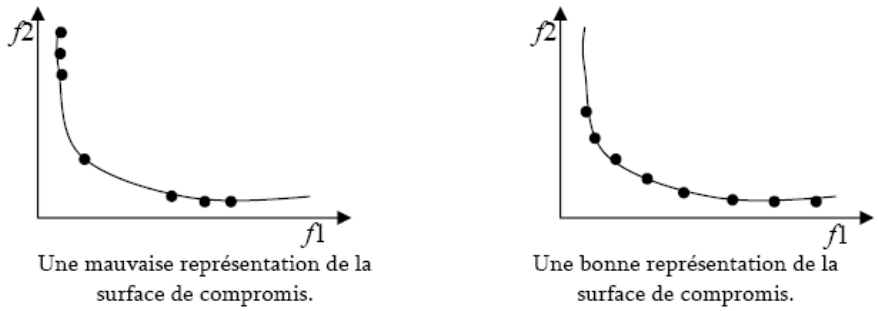

FIG. 1.1 – Représentation de la surface de compromis

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.

1.4 Points particuliers

Le point idéal

Le vecteur défini par :

$$y^* = \left(\min_{x \in S} f_1(x), \ldots, \min_{x \in S} f_r(x) \right) \in \mathbb{R}^r$$

constitue le point idéal ; en général $y^* \notin Y$.
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.

Le point nadir

Les coordonnées de ce point correspondent aux pires valeurs obtenues par chaque fonction objectif lorsque l'on restreint l'espace des solutions à la surface de compromis. Il existe plusieurs approches pour estimer le point nadir, la plus simple consiste à utiliser une matrice carrée de dimension r appelée matrice des gains et donnée par :

$$G = \begin{pmatrix} f_1(\overline{x}_1) & f_2(\overline{x}_1) & \cdots & f_r(\overline{x}_1) \\ f_1(\overline{x}_2) & f_2(\overline{x}_2) & \cdots & f_r(\overline{x}_2) \\ \vdots & \vdots & \vdots & \cdots \\ f_1(\overline{x}_r) & f_2(\overline{x}_r) & \cdots & f_r(\overline{x}_r) \end{pmatrix}$$

où $\overline{x}_i$ est la solution optimale obtenue en minimisant le critère f_i sur S, $\forall\, i = \overline{1, r}$.

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

$$\widetilde{y}_j = \max_{i=\overline{1,r}} G_{ij},\ j = \overline{1, r}.$$

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.

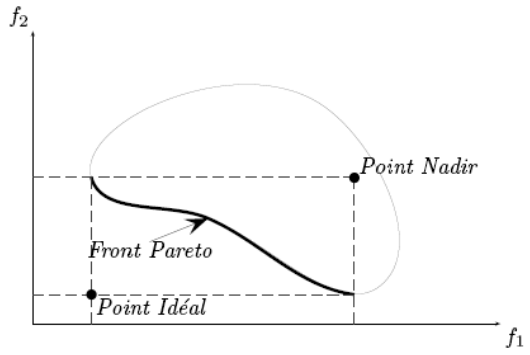

FIG. 1.2 – Représentation de la frontière Pareto et des points particuliers

1.5 Méthodes de résolution

Pour la résolution des Problèmes d'optimisation multiobjectif, des méthodes exactes, ainsi que des heuristiques ont été proposées :

- Le Branch and Bound [5].
- L'algorithme A* [6].
- La programmation dynamique [7].

Ces méthodes sont efficaces pour des problèmes de petites tailles. Pour des problèmes à plus de deux critères ou de grandes tailles, il n'existe pas de procédures exactes efficaces, étant donné les difficultés simultanées de la complexité NP-complet, et le cadre multicritères des problèmes. Pour surmonter ces difficultées plusieurs méthodes heuristiques ont été proposé. Ces méthodes ne garantissent pas de trouver de manière exacte tout l'ensemble des solutions Pareto, mais une approximation, aussi bonne que possible, de cet ensemble.

1.5.1 Classification des méthodes de résolution

Les approches de résolution d'un problème d'optimisation multiobjeif peuvent être classées selon deux points de vue : celui du décideur et celui du concepteur.
- Selon le décideur : Cette classification est définie selon le moment d'ntervention du décideur pour la prise de décision à savoir le choix d'une solution parmi l'ensemble des solutions acceptables. Nous distinguons trois classes de méthodes :

1. Les méthodes d'optimisation a priori :

   Dans cette première classe de méthodes, le décideur doit définir ses préférences entre les différents objectifs avant d'utiliser la méthode d'optimisation. Une tâche qui n'est pas toujours possible pour le décideur du fait qu'elle demande une bonne connaissance du problème et parfois même une expérience non négligeable. Ainsi une seule exécution permettra d'obtenir la solution recherchée.

2. Les méthodes d'optimisation a posteriori

   Ici, le décideur choisit la solution la plus satisfaisante parmi l'ensemble des solutions fournies par la méthode d'optimisation. Il n'est donc pas nécessaire de disposer d'une quelconque information initiale sur les préférences du décideur mais il faut en contrepartie fournir un ensemble de solutions bien réparties, ce qui peut également être difficile et requérir un temps de calcul important.

3. Les méthodes interactives

   Une méthode interactive consiste en une alternance d'étapes de calculs et d'étapes de dialogue avec le décideur. La première étape de calculs fournit une première solution. Celle-ci est présentée au décideur qui réagit en apportant des informations supplémentaires sur ses préférences (étape de dialogue). Cette information est injectée dans le modèle utilisé et permet de construire une nouvelle solution. Un grand nombre de méthodes interactives ont été proposées dans la littérature (voir [8, 9].

-Selon le concepteur : Une autre classification est proposée dans la littérature qui adopte le point de vue du concepteur. Les approches utilisées pour la résolution de MOP peuvent être classées en trois catégories en fonction de la manière adoptée pour traiter les fonctions objetifs du problème étudié.

1. Les approches scalaires : Ces approches se basent sur la transformation du problème multiobjectif en un problème mono-objectif. Pour cette catégorie, deux méthodes les plus connues sont : la méthode d'agrégation [10] et celle d'ϵ-contrainte [11]. Ces méthodes sont généralement associées à des métaheuristiques pour la résolution comme les algorithmes génétiques, le recuit simulé et la recherche tabou.
2. Approches pareto : ces approches utilisent la notion de dominance pour comparer les solutions entre elles. Une seule résolution permet d'approximer l'ensemble de la frontière Pareto. C'est ce type d'approche que nous allons considérer par la suite. D'une manière générale, les approches Pareto identifiées dans la littérature sont implémentées à l'aide d'algorithmes génétiques, l'un des premiers à discuter de l'intérêt de l'utilisation de la notion de dominance pour la recherche de solutions pareto a été Goldberg [22]. Le principal avantage de ces approches est qu'elles sont capables de générer des solutions Pareto optimales dans les portions concaves de la frontière Pareto. Une description détaillée de ces approches sera présentée dans le chapitre 3.
3. approches non-scalaires et non-Pareto : Ces méthode ne transforment pas les problème multiobjectif en problème mono-objectif 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.

1.6 Conclusion

Nous avons pésenté dans ce chapitre les principaux concepts de l'optimisation multiobjectif ainsi que les différentes approches de résolution pour les MOPs. On a présenté quelques méthodes de résolution scalaires qui consiste à transformer un problème d'optimisation multiobjectif en un ou plusieurs problèmes à un seul objectif. Ces méthodes nécessitent une bonne connaissance du problème a priori, notamment pour fixer les vecteurs poids et il faille relancer plusieurs fois les algorithmes de résolution avec des valeurs différentes pour certains paramètres pour obtenir plusieurs points distincts de la surface de compromis sans oublier qu'ils sont aussi sensibles au paysage de la frontière Pareto (convexité). 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.

Chapitre 2

Métaheuristiques pour l'optimisation multiobjectif

Nous présentons dans ce chapitre un état de l'art des métaheuristiques suivant une classification proposée, leurs mécanismes de base et les principes qui les régissent. Nous citons par la suite, l'origine des algorithmes évolutionnaires (evolutionary algorithms (EAs)) en particulier les algrithmes génétiques et leur principe de fonctionnement. Dans la dernière section, nous mettons l'accent sur les mécanismes spécifiques aux algorithmes évolutionnaires multiobjectifs (Multiobjective evolutionary algorithms (MOEAs)) et nous terminons par la présentation de quelques mesures visant l'évaluation de la performance des algorithmes d'optimisation multiobjectifs.

2.1 Introduction

En l'absence d'approches exactes garantissant des temps d'exécution raisonnables et capables de résoudre certains problèmes d'optimisation. Les techniques de recherche et particulièrement les métaheuristiques se sont illustrées comme une alternative prometteuse. La pratique nous fait observer que les décideurs semblent se satisfaire largement et généralement d'une bonne approximation de la solution optimale. Les métaheuristiques se présentent dans ce cas comme une bonne façon de garantir des bonnes solutions approchées tout en respectant des délais de réponse raisonnables.
Ce chapitre se consacre à l'étude de ces métaheuristiques, leurs mécanismes de base et les principes qui les régissent. Nous essayerons d'élaborer cette étude sur la base de critère de performance qui nous serviront de paramètre de comparaison entre les différentes Métaheuristiques.
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, etc. Le lecteur intéressé peut consulter [23] pour une présentation de quelques applications importantes.
Dans ce chapitre nous introduirons les différentes métaheuristiques rencontrées dans la littérature, classées en deux groupe : les méthodes à solution unique et les méthodes à population de solutions. Nous détaillerons plus précisément les algorithmes génétiques et nous ferons un bref état d'art sur leur utilisation dans l'optimisation multiobjectifs.

2.2 Qu'est-ce qu'une métaheuristique

Les métaheuristiques sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile. Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Ces méthodes ont des inspirations de l'éthologie comme les colonies de fourmis, de la physique comme le recuit simulé, et de la biologie comme les algorithmes évolutionnaires. Une métaheuristique est donc une méthode générale, qui nécessite quelques transformations avant de pouvoir être appliquée à la résolution d'un problème particulier.

2.3 Métaheuristique à solution unique

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 appelé un mouvement.

2.3.1 La Recherche Tabou

La recherche Tabou (Tabu Search) est introduite par Glover [24]. 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 (i.e. ; 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 les L prochaines itérations. De cette façon, on évite tous les cycles de longueur inférieure à L.

Il existe plusieurs 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. La diversification a un objectif inverse de l'intensification. En effet, elle cherche à diriger la recherche vers des zones inexplorées. 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.

À noter que ces dernières années, la recherche Tabou a été appliquée avec succès à un certain nombre de problèmes d'optimisation Combinatoire [25, 26]. Un large éventail d'applications de cette méthode, se retrouve dans [27].

2.3.2 Le recuit simulé

La méthode du recuit simulé (Simulated Annealing (SA)) est inspirée du processus de recuit métallurgique (i.e. ; processus de refroidissement des métaux). Comme pour la recherche Tabou, le recuit simulé peut retenir une solution voisine de qualité moins bonne que celle de la solution courante. 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. Une difficulté spécifique à l'algorithme du recuit simulé est l'ajustement du paramètre de la température T. Un ajustement adéquat du paramètre T s'avère nécessaire d'autant plus que ce paramètre a un effet non négligeable sur la performance de l'algorithme. 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 (i.e. ; 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. La méthode du recuit simulé en optimisation multiobjectif a d'abord été abordée sous l'angle agrégatif [12, 13]. Les deux méthodes les plus populaires sont la méthode MOSA (Multiple Objective Simulated Annealing) proposée par Ulungu et al. [14] et la méthode PASA (Pareto Archived Simulated Annealing) proposée dans [28].
L'algorithme MOSA utilise les caractéristiques du recuit simulé pour rechercher le front Pareto. Cette méthode fonctionne bien car, à haute température, le recuit simulé répartis les individus sur toute une surface.
L'algorithme PASA utilise une fonction d'agrégation des fonctions objectifs, couplée

avec un système d'archivage des solutions non-dominées.

2.4 Métaheuistiques à base de population

2.4.1 Les colonies de fourmis

L'optimisation par colonies de fourmis (Ant Colony Optimization ou ACO) [29, 30] 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.
Dans les travaux utilisant l'optimisation par colonie de fourmis (OCF) dans un contexte multiobjectifs, [31] 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 a générer l'ensemble Pareto optimal. [32] ont développé un OCF biobjectifs.
Iredi, D. Merkle et M. Middendorf [33] proposent, pour leur part, une approche biobjectifs de I'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, Doerner et al. [34] ont proposé un algorithme nommé $P - ACO$ (Pareto Ant Colony Optimization) dédié à la résolution du problème d'allocation de portefeuilles. Cet algorithme présente de bons résultats, notamment face au recuit simulé.

2.4.2 Les essaims particulaires

L'optimisation par essaims particulaires [35] (Particle Swarm Optimization ou PSO)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). On peut par exemple observer qu'un banc de poissons est capable d'éviter un prédateur en se divisant en deux groupes avant de reformer, à nouveau, le banc originel. Cela sous-entend une forme d'auto-organisation au sein du groupe. En fait, chaque individu utilise l'information locale, à laquelle il peut accéder, sur le déplacement de ses plus proches voisins pour décider de son propre déplacement. Plus précisément, des règles très simples (e.g., rester proche des autres individus, aller dans la même direction, aller à la même vitesse) suffisent pour maintenir la cohésion du groupe tout entier et pour faire émerger des comportements collectifs complexes et adaptés. 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). À 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 (C'est-à-dire, la meilleure performance en termes de la fonction d'évaluation utilisée pour mesurer la qualité des solutions) précédente et la meilleure position de ses voisines (voir fig.2.1). Ici, la mémoire n'est structurée qu'au niveau local (i.e. ; 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 :

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

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

Dans l'équation (1), les coefficients c_1, c_2 et c_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.

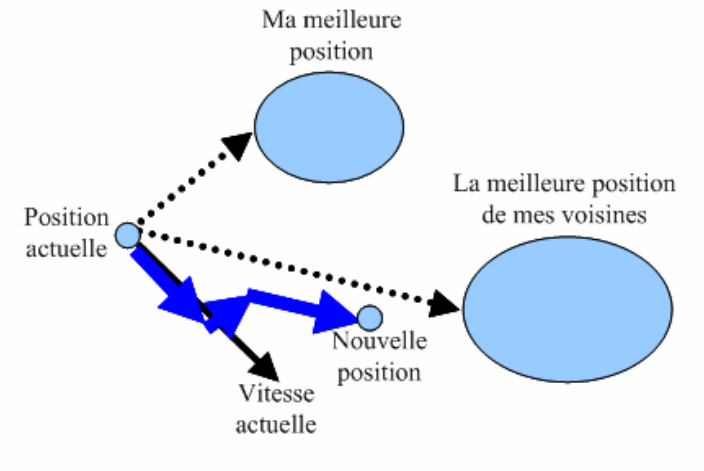

FIG. 2.1 – Principe du déplacement d'une particule.

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.

À partir de 2002 plusieurs auteurs ont proposé simultanément plusieurs approches pour utiliser l'OEP pour l'optimisation multiobjectifs. Toutes les variantes multiobjectifs des OEP sont appelées MOPSO (multiobjective particle swarm optimisation) même si les approches diffèrent. Pour avoir une idée détaillée sur l'MOPSO, le lecteur intéressé peut consulter les références suivantes [37, 38, 39].

2.4.3 Les algorithmes évolutionnaires

Historique

Le terme Algorithme évolutionnaire (EA) porte sur des méthodes d'optimisation stochastiques qui simulent le processus de l'évolution naturelle. Les origines des Algorithmes Evolutionnaires 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 des contraintes extérieurs (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.
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. Les premiers travaux sur l'évolution artificielle ont concerné les AGs, les stratégies d'évolution (SE) et la programmation évolutive :

En 1966 L. J. Fogel introduisit le concept de la programmation évolutionnaire.

En 1973 I. Rechenberg donna lieu aux stratégies d'évolution.

En 1975 J.H. Holland, professeur à l'université du Michigan, entreprit avec ses étudiants, une vaste étude qui permit de poser les fondements des AG en calquant les principes de Darwin (sélection, croisement, mutation, chromosome, gènes). Il parvint alors, à mettre au point les étapes de l'algorithme et ses principes de codage. Il esquissa aussi les grandes perspectives d'application des AGs.

En 1989 David Goldberg publie un ouvrage de vulgarisation des algorithmes génétiques.

En 1992 une autre classe d'algorithmes évolutionnaires vit le jour, la programmation génétique (PG) proposée par John Koza.
Ces différentes classes d'algorithmes évolutionnaires qui ne diffèrent que sur les détails d'implantation des opérateurs et sur les procédures de sélection et remplacement de la population sont maintenant largement utilisés pour résoudre des problèmes d'optimisation.

2.5 Les algorithmes génétiques

Les Algorithmes Génétiques (AGs) ont été introduits par Holland [40], ils représentent une famille assez riche et très intéressante d'algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique (ce qui d'ailleurs justifie son nom). Les champs d'application sont fort diversifiés. On les retrouve aussi bien en théorie des graphes qu'en compression d'images numérisées et dans beaucoup d'autres domaines. Les raisons de ce nombre d'applications sont claires. Leur principe est d'opérer une recherche stochastique sur un important espace à travers un ensemble de population solutions. Ces algorithmes sont simples et très performants dans leur recherche d'amélioration de la qualité de la solution recherchée. De plus, ils ne sont pas limités par des hypothèses contraignantes dans le domaine d'exploration. Ainsi, le mathématicien abordant le sujet n'a guère à se préoccuper de la continuité et de la différentiabilité de la fonction à optimiser. En résumé, les algorithmes génétiques diffèrent fondamentalement des autres méthodes d'optimisation selon quatre axes principaux :

1. Les AGs utilisent un codage des paramètres et non les paramètres eux-mêmes.
2. Les AGs travaillent sur une population de solutions au lieu qu'une seule.
3. Les AGs n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée ou une autre connaissance auxiliaire.
4. Les AGs utilisent des règles de transition probabilistes et non déterministes.

2.5.1 Vocabulaire et principe de fonctionnement

Avant d'expliquer en détail le fonctionnement d'un algorithme génétique, 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. À 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 p_c 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 modificatin d'un ou plusieurs composantes appliqué avec la probabilité p_m 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 :

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 dit fonction d'adaptation, ce mécanisme de sélection cherche à diriger l'exploration de l'espace des solutions en déterminant les individus ayant la plus grande probabilité d'être choisis. 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.

2.5.2 Les principales étapes d'un algrithme génétique

Afin d'expliciter le fonctionnement des algorithmes génétiques, nous présentons les différentes étapes d'un algorithme génétique simple illustré par la Figure suivante.

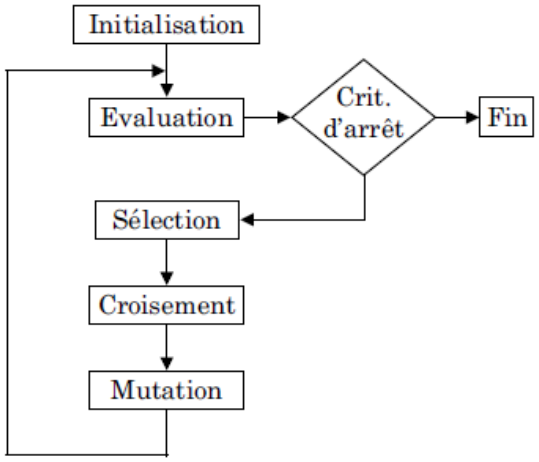

FIG. 2.2 – schéma général du fonctionnement d'un Algorithme génétique.

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 chaîne de caractères représente bien une solution réalisable du problème. En pratique, sauf le codage binaire et le codage naturel sont généralement employés.

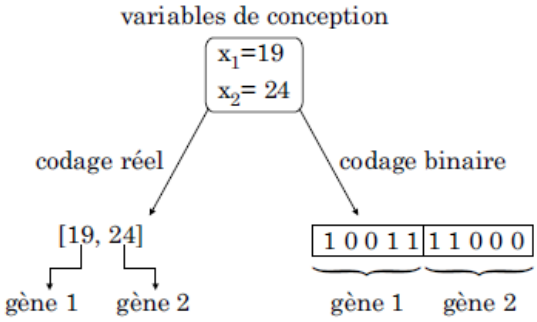

FIG. 2.3 – Exemple d'un codage réel et binaire.

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 des 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 étudié. Cette étape permet de classer les solutions, afin de

déterminer les solutions qui seront sélectionnées pour construire une nouvelle population de solutions.

3. **Sélection des individus :**
   La sélection a pour objectif d'identifier les individus qui doivent se reproduire. La sélection doit favoriser les meilleurs éléments selon le critère à optimiser. Ceci permet de donner aux meilleurs individus une probabilité plus élevée de contribuer à la génération suivante. Il existe plusieurs méthodes de sélection, les plus connues étant La sélection par tournoi (*tournament*), la sélection par roulette (*wheel*).

   La sélection par roulette (Wheel)
   Développé par Goldberg [22]. C'est le plus ancien opérateur de sélection qui consiste à attribuer à chaque individu i une probabilité de sélection proportionnelle à sa valeur d'adaptation à partir de la population E :

$$P_i = \frac{f_i}{\sum_{j \in E} f_j}$$

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

   L'inconvénient majeur de ce type de reproduction vient du fait qu'il peut favoriser 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.

   La sélection par tournoi
   La sélection par tournoi [41] 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 (i.e. ; le nombre de participants à un tournoi). Pour $k = 2$, on parle de sélection par tournoi binaire.

4. **Opération de croisement :**
   L'opérateur de croisement (recombinaison) a pour but de produire une nouvelle génération d'individus en recombinant les individus sélectionnés par l'opérateur de sélection. Ainsi, dans un premier temps, les individus sélectionnés sont répartis aléatoirement en couples de parents. Puis, chaque couple de parents subit une opération de recombinaison 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. Les opérateurs de recombinaison les plus fréquents dans la littérature sont : le croisement n-point ou multi-points (Multi-Point Crossover) et le croisement uniforme (Uniform Crossover).

   Opération de croisement n-point Le croisement n-point [42] 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 [43]. Dans le croisement 1-point, un seul fragment est échangé selon un point de coupure choisi aléatoirement. La Figure (fig.2.4) illustre cette opération. Quant au croisement 2-point, il consiste à échanger deux fragments selon deux points de coupure choisis aléatoirement (voir fig.2.5).

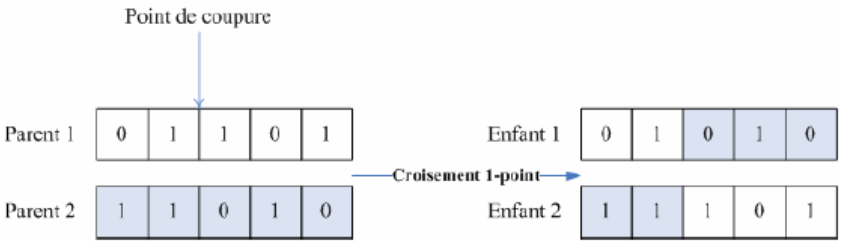

FIG. 2.4 – Procédure de croisement en 1-point.

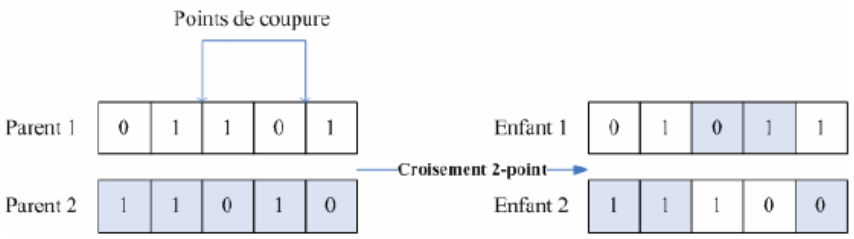

FIG. 2.5 – Procédure de croisement en 2-point.

Opération de croisement uniforme

Le croisement uniforme [44] consiste à générer un enfant en échangeant chaque gène des deux individus parents avec une probabilité uniforme égale à 0.5 (voir fig.2.6). 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.

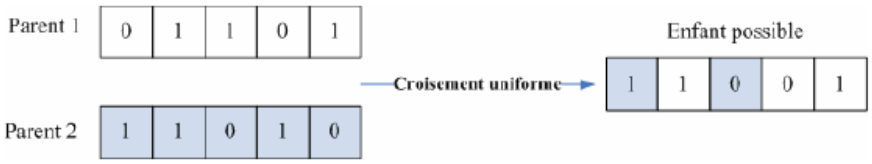

FIG. 2.6 – Procédure de croisement uniforme.

5. **Opération de mutation :**

La mutation est exécutée seulement sur un seul individu. Elle représente la modification aléatoire et de faible probabilité de la valeur d'un gène de l'individu [22], pour un codage binaire cela revient à changer un 1 en 0 et vice versa (fig.2.7). Cet opérateur introduit de la diversité dans le processus de recherche des solutions et peut aider l'AG à ne pas stagner dans un optimum local.

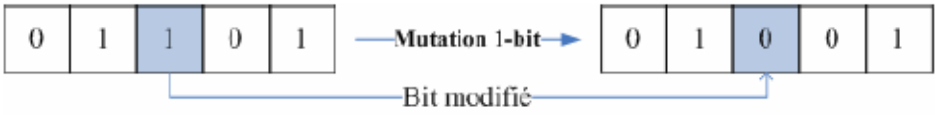

FIG. 2.7 – Opération de mutation.

6. **Remplacement :**
   Après la phase de reproduction (i.e. ; recombinaison 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.

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[1] qui sont fixé 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-à-dire le nombre d'individus de la population. Une taille trop petite peut entraîner 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 étudié) est donc préférable [45].
- **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 ($p_c \in [0.7, 0.9]$) 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 façons. Soit la probabilité pour qu'un individu subit l'opérateur de mutation, soit la probabilité pour qu'un gène d'un individu (i.e. ; 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

[1] le réglage adéquat de ces paramètres dépend fortement du problème étudié et de l'instance à traiter.

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 élevé sans garantir l'amélioration des résultats.

2.6 Les algorithmes génétiques et l'optimisation multiobjectif

Les algorithmes génétiques sont très bien adaptés au traitement d'un problème d'optimisation multiobjectifs. 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 AGs ont été largement utilisés pour la résolution Pareto de *MOP*, étant donné qu'ils travaillent sur une population de solutions. Deux objectifs doivent être pris en compte dans la résolution d'un *MOP* :

- converger vers la frontière Pareto : la plupart des travaux de recherche sur l'application des Algorithmes génétiques aux *MOP* se sont concentrés sur l'étape de sélection. Dans cette étape, des méthodes de *ranking* sont appliquées dont le rôle est d'établir un ordre (rank) entre les individus. Cet ordre dépend directement de la notion de l'optimalité Pareto. Les méthodes de ranking permettent de converger vers les solutions Pareto optimales ;
- 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 procedure permet de stabiliser des sous-populations multiples le long de la frontière Pareto.

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

2.6.1 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 convergé 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.

Techniques de Nichage

Ces techniques ont été introduites pour réduire la dérive génétique et permettre d'explorer en parallèle plusieurs solutions optimales. Elles reposent aussi sur une analogie entre les espaces de recherche en optimisation et les écosystèmes naturels. Dans la nature, chaque espèce se distingue par des caractéristiques biologiques propres et évolue de manière à occuper une niche écologique. Dans chaque niche, les ressources naturelles sont limitées et doivent être partagées entre les représentants qui la peuplent.
Par analogie en optimisation, les niches symbolisent les optima de la fonction (les régions de l'espace présentant des ressources susceptibles d'être exploitées par les différentes espèces). Les espèces sont constituées par des groupes d'individus similaires, la similarité étant caractérisée à partir d'un critère de distance, et d'un seuil de voisinage. Les méthodes de nichage visent à répartir les individus de la population sur les optima de la fonction à optimiser.

Fonction de partage (fitness sharing)

La technique du partage a été introduite par Goldberg et Richardson en 1987 [46]. elle consiste à ajuster l'adaptation des individus pour éviter qu'ils se concentrent dans la niche principale (c-à-d l'optimum global). En d'autres termes, la technique consiste à pénaliser les individus qui sont trop proches les uns des autres. L'objectif est d'éviter une grande concentration d'individus sur une petite zone de l'espace de recherche. Dans la pratique, on définit un voisinage autour de chaque individu, puis on calcule les distances entre les individus appartenant à ce voisinage. Formellement, le voisinage d'un individu est défini à partir d'un paramètre ρ_{sh} qui peut être vu comme une sorte de distance maximale à l'individu courant au-delà de laquelle les individus ne sont plus considérés comme voisins de cet individu. La fonction d'adaptation de chaque individu (i) est dégradée par un compteur de niche m_i , calculé 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

dans le voisinage de l'individu (i). Ce coefficient est calculé pour tous les individus (j) de la population courante P :

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

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 (i.e. ; appartenant au même voisinage) partagent la même valeur. Il existe plusieurs fonctions de partage. La fonction de partage classique suggérée dans [46] est :

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

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).

Crowding

Une première implémentation de cet opérateur a été réalisée par De Jong [42]. Dans la reproduction d'un nouvel individu, l'opérateur consiste à remplacer l'individu existant le plus semblable à l'individu généré. En remplaçant les individus semblables, le crowding permet de conserver les différents niches de la population tout en accélérant la convergence.

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 (e.g., 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.

2.6.2 Techniques de Ranking

Plusieurs techniques de ranking ont été utilisées dans la littérature :

Selecion Pareto

La sélection Pareto utilise la relation de dominance pour affecter des rangs aux individus de la population. Cette techniquqqe de ranking a été initialement proposée par Goldberg [22] et implémentée dans l'algorithme NSGA [47]. Initialement, tous les individus non dominés de la population reçoivent le rang 1 et sont retirés temporairement de la population. Puis, les nouveaux individus non dominés reçoivent le rang 2 avant d'être à leur tour retirés. 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.

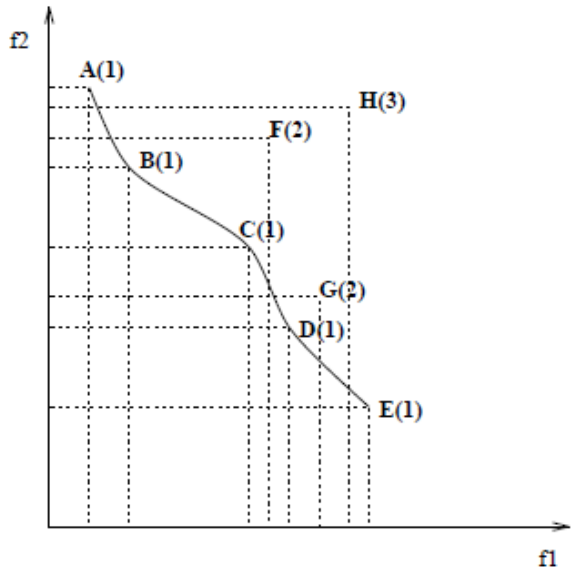

FIG. 2.8 – Ranking basé sur la selection Pareto

Non Dominated Sorting (NDS)

Dans cette méthode, le rang d'un individu est le nombre de solutions dominant l'individu plus un [48]. Considérons par exemple un individu x_i à la génération t, qui est dominé par p_t^i individus dans la population courante. Son rang dans la population est donné par :

$$rang(x_i, t) = 1 + p_t^i$$

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

prématurée.

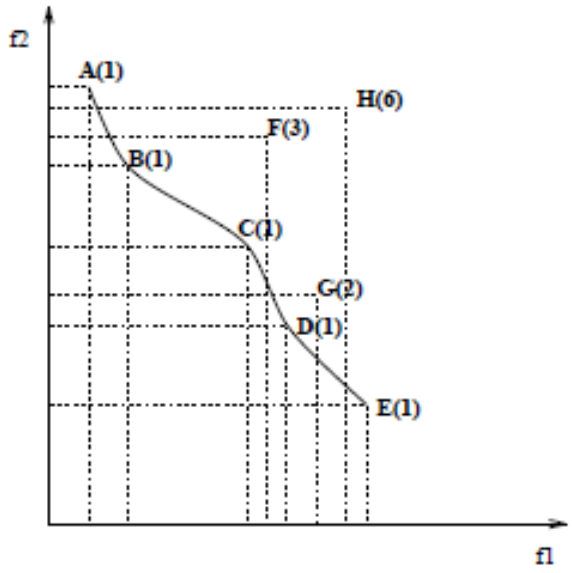

FIG. 2.9 – Ranking suivant la technique NDS

2.6.3 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 [42]. 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 diffcile 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.
Les approches récentes [49, 50] 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 participer à la phase 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 [51, 52].

2.7 Les métaheurstiques hybrides

Les métaheuristiques hybrides [53] consiste à combiner deux métaheuristique (parfois on combine plus deux métaheurisiques) de types différents pour en former une nouvelle. 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 combinatoires multiobjectifs [54].

2.8 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les métaheuristiques d'une manière générale et nous nous sommes intéressés aux algorithmes génétiques d'une manière spéciale. Nous avons présenté comment les EAs exploitent le principe de la génétique pour résoudre les problèmes d'optimisation multiobjectifs à l'aide des mécanismes qui assurent la convergence et diversité du front Pareto. Dans le chapitre suivant, nous allons présenter les algorithmes évolutionnaires multiobjectifs en général et les plus performant d'entres eux en détail.

Chapitre 3

Les algorithmes évolutionnaires multiobjectifs

Ce chapitre présente un large éventail des algorithmes évolutionnaires multiobjectifs. Au cours de cette présentation, nous essayons d'exposer certains algorithmes évolutionnaires performants et fréquemment utilisés pour la résolution de problèmes multiobjectifs et cela nous a permit, à la fin, de faire une étude comparative en essayant d'apporter une discussion à-propos des avantages et inconvénients de chacune d'elle.

3.1 Introduction

Le but des algorithmes évolutionnaires multiobjectifs (MOEAs) est double. En effet, il s'agit de trouver une bonne approximation du front Pareto aussi bien en termes de convergence (i.e. ; proximité au front Pareto) qu'en termes de diversité (i.e. ; répartition sur le front Pareto). D'un point de vue structurel, les EAs s'adaptent bien à une telle tâche. En effet, le fait d'évoluer avec une population d'individus permet aux EAs de trouver plusieurs solutions potentiellement non dominées en une seule exécution plutôt qu'en plusieurs exécutions. D'autre part, les EAs sont moins sensibles à la forme du front Pareto. Toutes ces caractéristiques encouragent l'utilisation intensive des EAs dans différents contextes d'optimisation multiobjectif.

Dans la littérature relative aux MOEAs, un certain nombre de méthodes ont été développées pour répondre aux deux buts mentionnés dans le paragraphe précédent. Le premier MOEA qui a été proposé dans la littérature s'appelle VEGA [55] est basé sur une approche non scalaire et non Pareto. Dans la catégorie des MOEAs basés sur une approche scalaire, on peut citer l'algorithme WBGA [56]. Après ces premiers travaux, une première génération de MOEAs de type Pareto (i.e. ; utilisant le classement par dominance pendant l'évaluation) a vu le jour : MOGA [57], NSGA [58], NPGA [59]. Les MOEAs les plus récents constituent ce qu'on peut appeler une seconde génération de MOEAs de type Pareto. Ces algorithmes se distinguent de ceux de la première génération par l'inclusion de populations secondaires et de techniques avancées pour assurer un bon compromis entre la convergence et la diversité. Les populations secondaires utilisées par ces méthodes contiennent généralement un nombre limité d'individus de meilleure qualité qui sont appelés individus élitistes. Les méthodes de la seconde génération sont alors appelées MOEAs élitistes de type Pareto. Parmi ces méthodes, on peut citer les suivantes : SPEA [60], SPEA-2 [61] et NSGA-II [62].

Dans la section suivante, nous allons présenter quelques MOEAs pour avoir une idée claire sur les différentes implémentations permettant la prise en compte de l'aspect multiobjectif par les algorithmes génétiques.

3.2 Vector Evaluated Genetic Algorithm (VEGA)

Le premier travail consistant à utiliser les AGs pour résoudre des *MOP* est celui de Schaffer [55]. L'algorithme développé VEGA sélectionne les individus de la population de taille (N) courante suivant chaque objectif, indépendamment des autres (sélection

parallèle). À chaque génération, la population est donc divisée en un nombre de sous-populations qui est égal au nombre d'objectifs (m). - Les m fonctions objectifs créent tour à tour une sous population contenant ($\frac{N}{M}$) individus, chaque sous-population i est sélectionnée suivant l'objectif f_i.

- Par la suite, les sous-populations sont mélangées et les opérateurs génétiques (croisement et mutation) sont appliqués à la totalité de la population. La figure fig.3.2 (a) illustre le mécanisme de sélection dans le cas de minimisation de deux fonctions objectifs.

- Le mélange des sous-populations revient à attribuer à chaque individu une fonction d'adaptation définie par :

$$f(i) = \frac{\sum_{k=1}^{m} f_k(i)}{N}$$

où $f_k(i)$ désigne la fonction d'adaptation de l'individu i par rapport à l'objectif k. Dans la mésure où Schaffer utilise un schéma de sélection par roulette la fonction d'adaptation des individus s'apparente donc à une combinaison linéaire des objectifs.

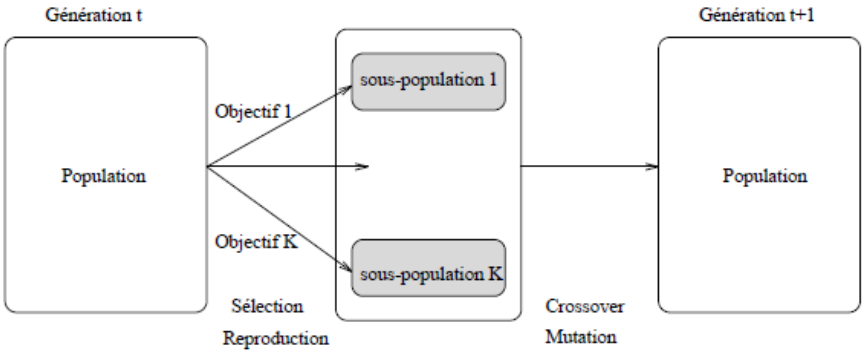

FIG. 3.1 – schema général du fonctionnement de l'algorithme VEGA.

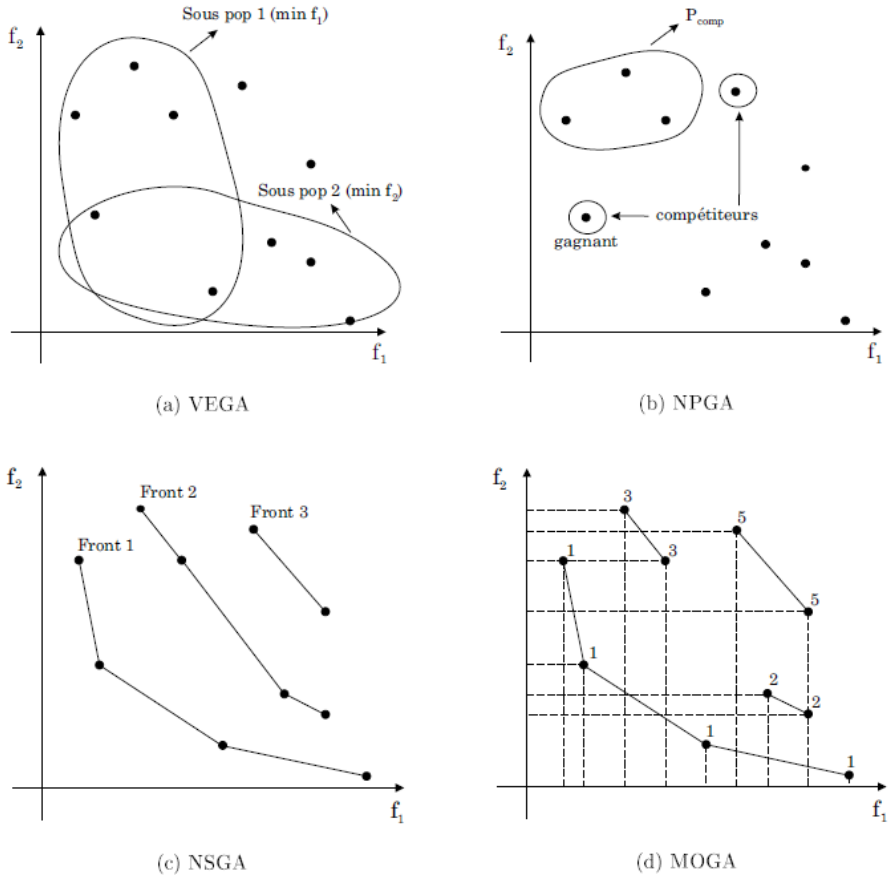

FIG. 3.2 – Les différents mécanisme de sélection.

Discussion

L'algorithme de Schaffer est basé sur une idée très simple et il est facile à mettre en oeuvre. Seuls quelques changements doivent être apportés à un EA mono-objectif . De plus, cela n'implique aucun coût de calcul supplémentaire. VEGA est particulièrement adapté aux problèmes où on cherche surtout des bonnes solutions individuelles pour chacun des différents objectifs.

La sélection de VEGA favorise les solutions qui sont bonnes pour un des objectifs. Et même si on peut espérer trouver de bons compromis à l'aide du croisement entres les champions individuels, il est peu probable que tels compromis puissent survivre assez longtemps au cours de l'évolution.

De plus, la façon d'attribuer la performance aux individus employée dans VEGA correspond à l'évaluation des individus selon une combinaison linéaire des critères (élaborée implicitement par l'algorithme) avec des poids dépendant de la distribution de la population en chaque génération. Ceci explique la tendance de la population de VEGA de

se concentrer dans les régions de la surface de Pareto, où un des critères est nettement meilleur que les autres. Par conséquent, l'algorithme VEGA est inefficace lorsque le front de pareto est non-convexe.
Les individus que Schaeffer appelle les individus milieu, parce qu'ayant une performance générale acceptable mais ne possédant aucun critère fort vont être éliminés car ils ne sont pas sélectionnés dans aucune sous-population. Cette disparition entraine la spécialisation des individus pour chaque objectif. Ce résultat est contraire au but initial de la méthode qui était de trouver un compromis entre les différents critères.

3.3 Weight-based Genetic Algorithm (WBGA)

L'algorithme WBGA a été proposé en 1992 par Hajela et Lin [56]. Dans WBGA, la fonction d'adaptation est calculée comme une agrégation linéaire des fonctions objectifs. La particularité ici c'est qu'à chaque individu, on associe un vecteur de pondération différent.
Le point très important de cette approche est la préservation de la diversité entre les vecteurs des poids. Cela peut être fait de deux façons suivantes. Soit le vecteur des poids fait partie de l'individu et la technique de nichage est appliquée, soit des sous-populations habilement choisies sont évaluées en utilisant des vecteurs des poids différents prédéfinis.

Approche basée sur les sous-populations

Cette méthode est similaire à VEGA. D'abord, un ensemble de K différents vecteurs des poids $w^k, k = 1, \cdots, K$, est choisi. Ensuite, chacun de ces vecteurs est utilisé pour calculer la performance pondérée normalisée pour tous les N membres de la population. Les N/M meilleurs d'entre eux sont groupés en une sous-population (qui est associée au vecteur w^k) pour le processus génétique postérieur. C'est-à-dire, la sélection, le croisement et la mutation sont restreints aux sous-populations. La taille de sous-population N/M pour chaque vecteur w^k est maintenue, ce qui assure qu'à la fin de ces opérations la taille de la population entière est toujours N.
Á la génération suivante encore, chaque vecteur w^k est utilisé pour trouver la sous-population correspondante des meilleurs etc.
Il y a donc K sous-populations, chacune correspondant à un vecteur de poids. Notons qu'un membre de la population peut être associé à plus d'un vecteur. En particulier, les solutions non extrémales seront incluses dans plus d'une sous-population.
Dans cet algorithme, de multiples solutions sont trouvées grâce à la préservation expli-

cite de K différents vecteurs des poids. Il n'est pas nécessaire d'introduire un opérateur de nichage supplémentaire.

Calcul de la performance dans WBGA basé sur les sous-populations :

1. Soit le compteur des solutions $k = 1$.
2. Calculer la performance Fj de chaque solution j. Choisir N/M meilleures solutions à partir des valeurs F_j. Copier ces solutions dans les sous-populations P_k.
3. Effectuer la sélection, appliquer le croisement et la mutation aux membres de P_k pour créer une nouvelle population de taille N/M.
4. Si $k < K$, incrémenter k de 1 et retourner à l'étape 2.
5. Sinon, combiner toutes les sous-population. pour créer une population $P = \cup_{k=1}^{K} P_k$. Si $|P| < N$, ajouter des solutions aléatoires pour compléter la population.

Notons qu'à l'étape 2, N solutions sont évaluées et cette étape est répété K fois, ce qui coûte $(K * N)$ évaluations. Cependant, cette approche est meilleure du point de vue de la complexité par rapport à l'approche basée sur le partage grâce à l'absence du mécanisme de nichage avec ses calculs des distances.

Discussion

Dans cet algorithme, de multiples solutions potentiellement non dominées peuvent être trouvées grâce au maintien explicite de différents vecteurs de pondération. De plus, contrairement à VEGA, WBGA peut trouver des solutions non extrémales de bonne qualité puisque de telles solutions peuvent être sélectionnées plusieurs fois pendant la construction des sous-populations (i.e. appartenir à plusieurs sous-populations) ; ce qui augmente leur chance de survivre et de s'améliorer au fil des générations. Toutefois, étant basé sur une approche d'agrégation linéaire, l'algorithme WBGA est incapable de trouver des solutions appartenant à des concavités (e.g. les solutions non-supportées). En outre, il est souvent difficile d'effectuer un choix judicieux d'un ensemble de vecteurs de pondération de sorte à assurer une distribution uniforme des solutions sur le front Pareto. Un tel choix exige en effet une connaissance approfondie de la structure du problème à résoudre. Les algorithmes VEGA et WBGA sont habituellement cités dans les travaux consacrés à la comparaison de la performance de différents MOEAs. Comparés aux approches basées sur la dominance, ces algorithmes (surtout VEGA) ne peuvent pas être considérés comme inférieurs aux méthodes décrites dans la section suivante, tant qu'il s'agit des problèmes pour lesquels le front de Pareto est convexe. Compte

tenu de son faible coût de calcul et de la simplicité de sa mise en oeuvre, VEGA peut être parfois plus approprié à l'usage que certains algorithmes basés sur le principe de dominance.

3.4 Algorithmes basés sur la dominance de Pareto

3.4.1 Multiple Objective Genetic Algorithm (MOGA)

L'une des premières applications de la notion de dominance pour le calcul de la performance des solutions a été proposée par Fonseca et Fleming [57]. Ils proposent de déterminer pour chaque solution i, le nombre de solutions n_i la dominant. Le rang r_i d'une solution x_i à une génération t est donné par

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

Le rang de chaque solution non dominées est donc égal à 1 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 mê me rang, la fonction fitness est identique. Cependant, cette procédure ne permet pas d'avoir une diversité des solution 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 inférieur 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 j , 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. Et f_k 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 est définie par :

$$sh\left(d_{ij}\right) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\beta} & si\ dij \leq \sigma_{share}, \\ O & sinon. \end{cases}$$

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 $\beta = 1$, Soit n_{r_i} le nombre de solutions de rang r_i.

$$m_i = \sum_{j=1}^{n_{r_i}} sh\left(d_{ij}\right),$$

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 meilleure performance. Cela se traduit par une forte pression de sélection pour telles solutions.
Cette méthode permet d'obtenir des solutions Pareto optimale de bonne qualité.
Cependant, les résultats dépendent de la valeur du paramètre de partage σ_{share} qui détermine la taille des niches et donc assure la diversité des solutions dans la population.

Discussion

Comme l'indique Coello [63] l'algorithme est connu pour sa facilité d'implémentation et la simplicité du calcul de la performance. Mais, d'autre part, 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.

La procédure du partage n'assure pas que les solutions d'un rang inférieur autont une moins bonne valeur de performance partagée F' 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.

3.4.2 Nondominated Sorting Genetic Algorithm (NSGA)

En 1994, Srinivas et Deb ont proposé l'algorithme NSGA [58] dans lequel l'idée du classement par dominance est implémentée pour la première fois. Cet algorithme tente d'atteindre le double objectif (convergence/diversité) en favorisant les solutions non

dominées (via la technique du classement par dominance) et en exploitant la technique du partage pour pouvoir distinguer entre les solutions ayant le même rang de dominance. 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 dominés de P appartiennent à l'ensemble P_1 ; ensuite, tous les éléments non dominés de P/P_1 sont placés dans l'ensemble P_2 etc. La procédure est terminée quand toute la population est triée :

$$P = \cup_{j=1}^{r} Pj$$

Le nombre total de classes, noté r dans l'équation ci-dessus, dépend de la population P. La figure fig.3.2 (c) présente un exemple de telle classification (les numéros associés à chaque point correspondent au numéro de la classe à laquelle il appartient).
Par convention, on appelle les ensembles P_j fronts de dominance ou tout simplement fronts. Notons que entre deux individus appartenant au même front, aucun ne peut être considéré comme meilleur que l'autre.

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 valeur de performance F la plus grande sera donc attribuée à ces individus, elle diminu 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, tout solution i de P_1 reçoit la valeur $F_i = N$.

À 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 solution de même rang, une technique de partage est utilisée pour partager les individus ayant le même rang. C'est-à-dire que, parmi ces individus, ceux appartenant aux régions moins denses sont considérés comme meilleurs que ceux appartenant aux régions plus denses. La valeur de performance 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-à-dire, 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 distances sont utilisées pour calculer la fonction de partage $Sh(dij)$. Le calcul du compteur de niche permet de dégradée la performance initial F_i de façon à augmenter la pression de sélection dans les régions les plus éparses.

$$F_i' = F_i / nc_i$$

Discussion

L'avantage principal de NSGA c'est que les valeurs de performance sont attribuées aux individus au niveau de l'ensemble non dominé 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 diversite des solutions trouvees par NSGA. Cependant le choix du paramètre σ_{share} a une influence très importante sur la performance de l'algorithme.

3.4.3 Niched Pareto Genetic Algorithm(NPGA)

Horn et al. [59] proposent une approche basée sur la dominance au sens de Pareto mais qui diffère des approches présentées ci-dessus dans la sélection des solutions. Les auteurs définissent une procédure de sélection par tournoi. Deux solutions i et j sont tirées aléatoirement de la population de taille N.
Une sous-population P_{comp} de taille n_{comp} est choisie aléatoirement. Chaque solution de la sous-population est comparée aux deux solutions i et j. Deux situations sont possibles :

- Si l'une des deux solutions domine le sous-ensemble de solutions, alors cette solution est retenue pour la sélection et elle représente le candidat gagnant du tournoi. La figure fig.3.2 (b) illustre ce processus de sélection dans le cas d'une minimisation de deux fonctions objectifs.

- Si les deux solutions dominent le sous-ensemble de solutions ou les deux solutions sont dominées par au moins une solution de P_{comp}, on calcule le compteur de niche pour chacune des deux solutions.

La solution dont le compteur de niche est petit gagne le tournoi. Cette étape permet de sélectionner les parents qui feront l'objet de reproduction pour générer une nouvelle population.

Discussion

Cet algorithme combine astucieusement l'étape de sélection par tournoi et la notion de dominance de pareto qui ne s'applique ici qu'à une partie de la population P_{comp}.
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 n_{comp} relatif à la taille de la sous-population P_{comp}.
Si le paramètre n_{comp} 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. 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.

3.5 Les méthodes élitistes

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. D'une génération à l'autre les solutions non dominées sont conservées dans une archive associée à la population courante. 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.

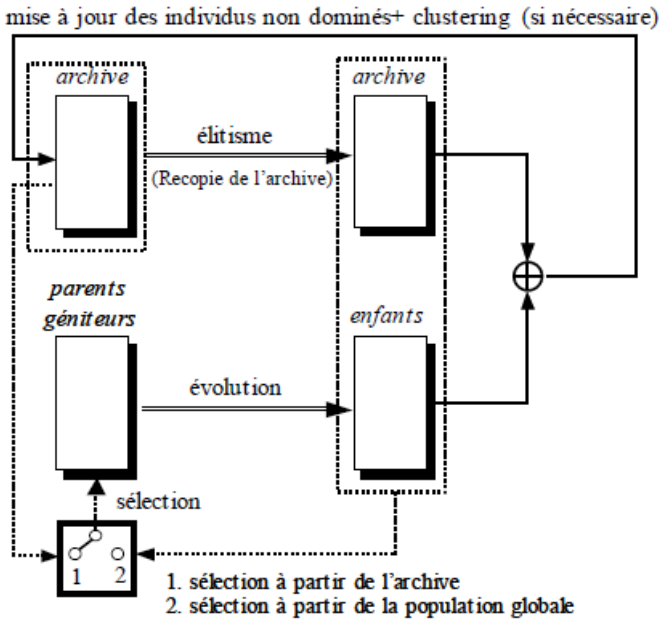

FIG. 3.3 – Principe général d'un algorithme élitiste.

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.

3.5.1 Strength Pareto Evolutionary Algorithm (SPEA)

En 1999, Zitzler et Thiele ont proposé une méthode évolutionnaire élitiste basée sur la notion de dominance pareto [60]. L'élitisme est introduit par le maintien explicite d'une population externe $\bar{P}$ appelée aussi archive. À 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).

L'algorithme démarre avec une population P_0 de taille N générée de manière aléatoire avec une population Archive $\bar{P}_0$ vide.

À chaque génération t, les solutions non dominées de P_t sont copiées dans la population externe $\bar{P}_t$ et toutes les solutions dominées qui peuvent apparaître du fait de l'ajout de nouveaux éléments à $\bar{P}_t$ sont éliminées.
Quand la taille de $\bar{P}_t$ atteint une valeur limite N^* fixée à l'avance, un critère supplémentaire rentre en jeu pour sélectionner les solutions qui vont être rajouter à $\bar{P}_t$ de sorte à ne pas dépasser la taille limite.
Ce critère, issu de la préoccupation de la préservation de la diversité entre les membres de $\bar{P}_t$, est basé sur le principe de *clusering* (voir section 3.4).

Calcul de la valeur d'adaptation

Après la mise à jour de l'ensemble $\bar{P}_t$. Le premier pas consiste à associer une valeur d'adaptation à chaque solution de P_t et $\bar{P}_t$. 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'elle dominent dans la population courante :

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

Ensuite, la valeur d'adaptation d'une solution j de la population courante dépend de la somme des S_i des solutions de l'archive qui la domine :

$$F_j = 1 + \sum_{i \in \bar{p}_t \wedge i \leq j} S_i$$

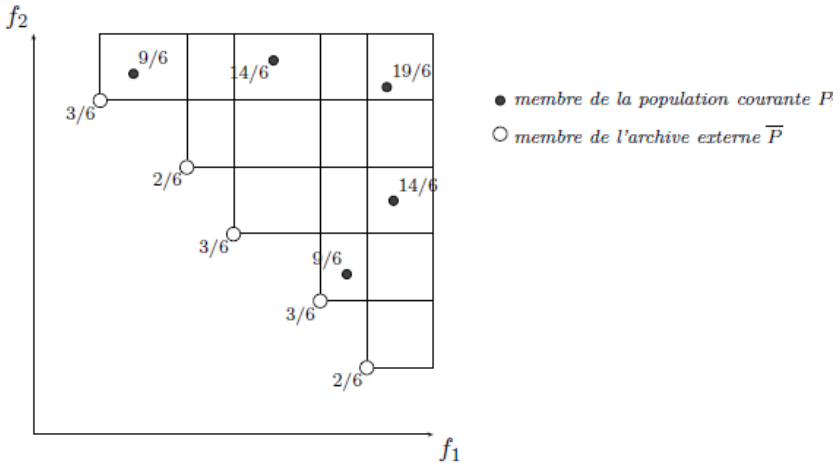

FIG. 3.4 – Calcul des valeurs d'adaptation en dimension deux.

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 $P_t U \bar{P}_t$ 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 $\bar{P}_0 = \emptyset$
2. Déterminer l'ensemble des solutions non dominées $E^*(P_t)$ de la population P_t.
3. $\bar{P}_t \leftarrow \bar{P}_t \cup E^*(P_t)$.
4. Déterminer toutes les solutions non dominées $E^*(\bar{P}_t)$ de l'archive actualisée $\bar{P}_t$ et éliminer toutes les solutions dominées : $\bar{P}_t \leftarrow E^*(\bar{P}_t)$.
5. Si $|\bar{P}_t| > N^*$, 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 $\bar{P}_{t+1}$.
6. Calculer la valeur d'adaptation des solutions de $\bar{P}_{t+1}$ 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.

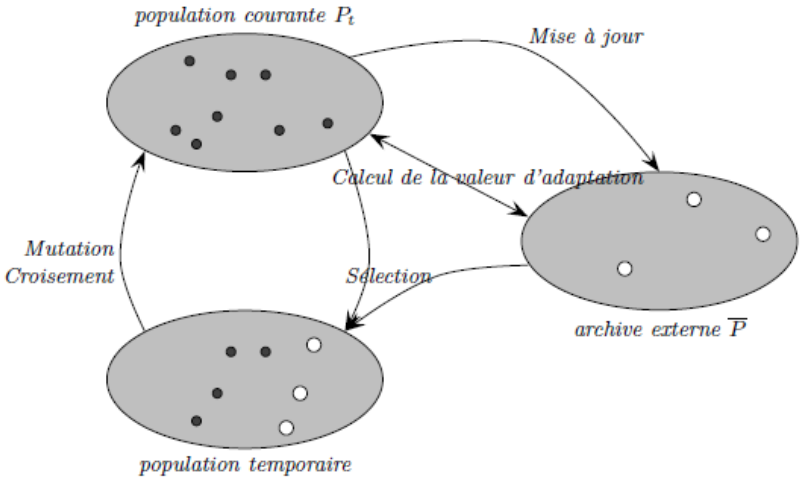

FIG. 3.5 – Fonctionnement général de l'algorithme SPEA.

Procédure de clustering

Afin de réduire la taille de la population externe $\bar{P}_t$ de N^{**} à N^*, la procédure suivante est utilisée. Au début de la procédure, chaque élément de $\bar{P}_t$) constitue son propre cluster (groupe) donc nous avons N^{**} clusteurs. 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 $d_{k'l'} = min_{k,l \in \bar{P}_t} d_{kl}$ 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 atteint N^*. Dans chaque cluster, une solution est retenue, c'est la solution dont la distance moyenne des autres solutions du même cluster est minimale.

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.

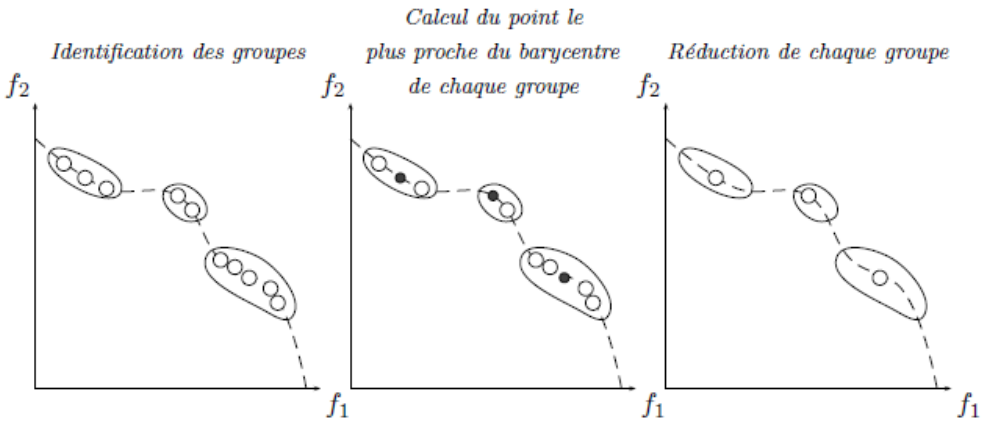

FIG. 3.6 – Illustration du clustering en dimension deux.

Discussion

Ètant un MOEA élitiste, SPEA possède la faculté de trouver des solutions de bonne qualité en termes de convergence (i.e. ; 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 dominées au détriment de ceux qui sont dominées. 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 méthode c'est qu'elle ne nécessite la définition d'aucun paramètre. Toutefois, un certain équilibre entre la taille de la population de base P et celle de la population externe est nécessaire pour une bonne performance de SPEA. En effet, si la taille 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.

3.5.2 Strength Pareto Evolutionary Algorithm II (SPEA-2)

En 2001, Zitzler et al. [61] ont proposé une version améliorée de SPEA. Ces améliorations porte essentiellement sur la définition d'une taille fixe de l'archive qui à l'état initial est constitué par les solution nondominés 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. 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 :

$$Fj = D(i) + \sum_{i \in p'} Si$$

La reproduction des solutions s'effectue seulement dans l'ensemble Archive.

3.5.3 Fast Nondominated Sorting Genetic Algorithm (NSGA-II)

[62] présente une deuxième version de l'algorithme NSGA. L'auteur tient compte des insuffisances de NSGA qui concernent la complexité algorithmique, la non prise en compte de l'élitisme et la nécessité de déterminer un paramètre de partage. Contrairement à NSGA, la nouvelle version permet de réduire la complexité algorithmique et considère une approche élitiste dans la recherche des solutions. A la génération initiale une population de solutions de taille N est constituée aléatoirement. Les procédures de croisement et de mutation sont appliquées, les solutions sont choisies en utilisant un tournoi basé sur la fonction d'adaptation calculée à partir du classement des solution à l'aide d'une procédure de classement *Ranking* qui détermine un rang pour chaque solution en fonction de son appartenance à un front Pareto. Ainsi une population enfants de taille N est crée. A la prochaine génération, les enfants obtenus à la génération précédente sont mélangés avec la population initiale, On obtient alors une population de $2N$ solutions qu'il faudra ranger en fronts de non-domination. Le but de cette procédure est de garder les solutions non dominées. Les N meilleures solutions constitueront la nouvelle population de solutions potentielles pour la génération encours. Cette procédure vise à assurer une convergence rapide de l'algorithme en sauvegardant les solutions non dominées à chaque itération. Pour assurer la diversité des solutions, un calcul de distance basé sur une procédure de *Crowding distance* permettant de choisir entre des

solutions de même rang.Pour plus de détails, une présentation complète de cette Algorithme est donnée dans le chapitre suivant (voir section 4.2.2).

Discussion

À la différence du mécanisme de diversification basé sur la technique de partage, celui basé sur la technique de crowding distance n'exige aucun paramètre à fixer à priori. De plus, le principe sur lequel est basée la technique de crowding disatnce favorise une distribution uniforme des solutions sur le front Pareto [64]. Notons que rien n'empêche de calculer les distances dans l'espace de décision si cela est considéré plus adapté au problème traité. D'autre part, la stratégie élitiste adoptée par NSGA-II (i.e. ; évolution avec deux populations qui participent toutes les deux à la création de la prochaine population) favorise la convergence vers le front Pareto.

À travers des expérimentations [65], il a été constaté qu'à partir d'une certaine génération, presque tous les solutions évoluant dans NSGA-II se concentrent dans le premier front. A ce stade, seules les distances de crowding permettent de sélectionner les solutions qui vont survivre. Par conséquent, des solutions Pareto-optimales situées dans des régions très peuplées peuvent être éliminées.

L'algorithme NSGA-II a été appliqué avec succès à de nombreuses applications dont plusieurs problèmes [66]. 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.

3.6 Comparaison des approches évolutionnaires multiobjectifs

Plusieurs travaux de classification et de comparaison des MOEAs ont été publiés jusqu'à aujourd'hui [60, 63, 67]

Les travaux comparant les MOEAs ne laissent pas de doute sur le fait que l'introduction de l'élitisme a mené à des performancess nettement supérieures. Par contre, les comparaisons de dernières méthodes élitistes entre elles ne permettent pas de faire un choix sans hésitation. Zitzler et al. [61] ont montré que la performance finale de SPEA-2 et de NSGA-II semble être meilleure sur certains problèmes tests au sens de la métriqueutilisée dans leur étude.

Pour les algorithmes VEGA et WBGA sont habituellement cités dans les travaux

consacrés à la comparaison de la performance de différents MOEA. Comparés aux approches basées sur la dominance, ces algorithmes (surtout VEGA) ne peuvent pas être considérés comme inférieurs à ces apporoches, tant qu'il s'agit des problèmes pour lesquels le front de Pareto est convexe. Compte tenu de son faible coût de calcul et de la simplicité de sa mise en oeuvre, VEGA peut être parfois plus approprié à l'usage que certains algorithmes basés sur le principe de dominance.

Dans la pratique, le choix de l'utilisateur parmi ces méthodes est plutôt dicté par des particularités de leur mise en oeuvre, et par nécessité du choix a priori de certains paramètres importantes. Par exemple, pour la résolution du problème du sac à dos multiobjectif présenté dans la partie implémentation, c'est la méthode NSGA-II qui a été utilisée. L'avantage de NSGA-II consiste en absence de paramètres liés à la mesure de la diversité, et à la taille de l'archive.

3.7 Conclusion

Nous avons présenté dans ce chapitre quelques méthodes Evolutionnaires pour l'optimisation multiobjectif. Ces méthodes ont montré leur efficacité pour trouver des solutions approchées satisfaisantes pour un grand nombre de problèmes.
Parmi l'ensemble des algorithmes évolutionnaires multiobjectifs examinés, 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 donné lieu à des travaux récents et nombreux, illustrant leurs originalités et leurs bonnes performances sur de nombreuses instances de problèmes.

Chapitre 4

NSGA-II appliquée au problème de sac à dos multiobjectif

Dans ce dernier chapitre, nous nous intéressons à la résolution du problème de sac à dos multiobjectif en variables binaires (MultiObjective Knapsack Problem (01-MOKP)) par l'algorithme évolutionnaire NSGA-II. Nous présentons tout d'abord le problème 01-MOKP, puis nous exposons en détail la méthode NSGA-II ainsi que les résultats expérimentaux obtenus avec une étude comparative commentée.

4.1 Présentation du problème de sac à dos multiobjectif

Il existe beaucoup de problèmes combinatoires réputés. Le problème de sac à dos multiobjectif en variables binaires *01-MOKP* est l'un des plus étudiés dans la communauté multiobjectif. Sa popularité est naturellement due à son grand intérêt tant au niveau académique qu'au niveau pratique. En effet, il permet de modéliser plusieurs applications réelles (e.g., chargement de cargaisons, répartition de budgets, allocation de ressources,etc.) et peut être rencontré comme sous-problème dans d'autres applications.

4.1.1 Définition du problème *01-MOKP*

Le problème de sac à dos multiobjectif en variables binaires 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 multiobjectif 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 *01-MOKP* peut être formulé de la manière suivante :

$$(01-MOKP)\begin{cases} Max \quad f_i(x) = \sum_{j=1}^{n} C_j^i x_j \quad i = \overline{1,m} \\ t.q. \\ \qquad \sum_{j=1}^{n} a_j x_j \leq b; \\ \qquad x_j \in \{0,1\} \qquad\qquad j = \overline{1,n} \end{cases}$$

où :

– $x = (x_1, \ldots, x_n)$ est un vecteur binaire de dimension n, définissant une solution possible du problème *01-MOKP* de sorte que :

$$x_j = \begin{cases} 1 & si \;\; l'\, objet \;\; j \;\; est \;\; choisi; \\ 0 & sinon \end{cases}$$

– C_j^i est le profit de l'objet j selon le i$^{\text{ème}}$ critère.
– a_j est le poids de l'objet j.
– b est la capacité du sac à dos.

4.2 NSGA-II appliqué au problème de sac à dos multiobjectif

Dans cette section, nous décrivons l'algorithme NSGA-II appliqué au problème de sac à dos multiobjectif ainsi que les paramètres de l'algorithme génétique utilisé.

4.2.1 Fast Nondominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II [62] est un algorithme qui utilise des techniques avancées d'élitisme et de diversification en intégrant un opérateur de sélection, basé 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 meilleurs solutions rencontrées soient conservées. NSGA-II obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de K. Deb, ce qui fait de cet algorithme un des plus utilisés aujourd'hui.
Initialement, l'algorithme démarre avec une population P_0 générée aléatoirement, cette population P_0 est triée suivant la technique du classement par front de non-dominance de manière similaire à celle utilisée dans NSGA (voir section 3.4.2).
À l'étape de sélection des parents, NSGA-II utilise tout d'abord une sélecion par tournoi binaire basé sur les rangs de non-dominance pour les solutions de rang différent, et utilise ensuite une mesure de densité appelée *crowding distance* (cette mesure sera détaillée ultérieurement dans cette section) pour comparer deux solutions de même rang. En d'autres termes, si l'une des deux solutions a un rang plus petit que l'autre, alors cette solution gagne le tournoi. Dans le cas où les deux solutions ont le même rang, alors on compare leurs distances et celle ayant la plus grande distance gagne le tournoi.

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.

Crowded tournament selection

Le crowded tournament sert à guide le processus de la sélection avec une répartition uniforme des solutions. Une solution i a deux attributs :

· Le rang de (i_{rank}),

· Crowded distance ($i_{distance}$)

Soit i et j deux soltions, la solution i remportera le tournoi si :

1. $i_{rank} < j_{rank}$

   ou

2. ($i_{rank} = j_{rank}$) et ($i_{distance} > j_{distance}$)

L'algorithme évolue en utilisant deux populations de même taille N : une population P dite population de parents et une population Q dite population d'enfants. En fait, à chaque génération, les solutions qui participent à la reproduction sont sélectionnés à partir de P pour subir les opérations de croisement et de mutation afin de créer la population d'enfants Q.

Ces deux populations sont regroupées en une seule population R de taille $2N$. Puis, cette population R est triée selon le principe du classement par front de non-dominance. Par la suite, une nouvelle population P_{t+1} de taille N sera constituée à partir des meilleurs fronts de la population R. Pour y arriver, les solutions des fronts seront incluses dans la population jusqu'à ce que la taille devienne supérieure ou égale à N. Si, suite à l'ajout d'un front la taille de la population sera supérieure à N, alors les solutions de ce front seront triées par ordre décroissant de leurs *crowding distance* et inclure $N - |P_{t+1}|$ solutions ayant les valeurs de distance les plus grandes dans la population P_{t+1}. L'opérateur de sélection, croisement et mutation seront alors appliqués sur P_{t+1} pour créer la nouvelle population Q_{t+1}.

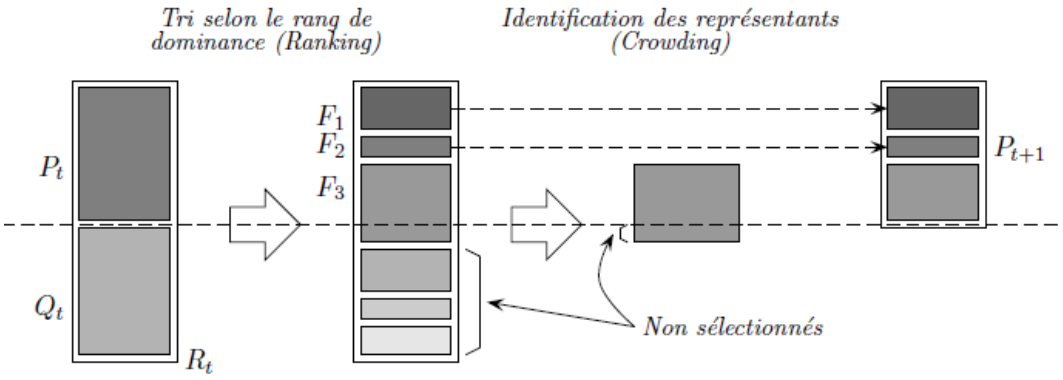

FIG. 4.1 – Schéma représentant les étapes de l'algorthme NSGA-II.

Il est important de noter que le tri de la population R_t en fronts de non-dominance et le remplissage de la population P_{t+1} peuvent être effectuer simultanément. Chaque fois qu'un nouveau front est trouvé, on vérifie s'il peut rentrer dans P_{t+1} entièrement. Si ce n'est pas le cas, le processus du ranking s'arrête.

Technique de Ranking

La technique de ranking, dont l'idée, suggérée dans les travaux de Goldberg en 1989 est basée sur le principe de l'optimalité Pareto, fut reprise et implémentée dans l'algorithme NSGA en 1994. Cette technique permet de classer toute la population (parents et enfants) en plusieurs fronts de Pareto. Le pseudo code de cette procédure de classification est représenté par l'algorithme suivant.

Pseudo-code de la procédure de classification en fonts Pareto

Dédut
$R = P \cup Q$
Initialiser le compteur i des fronts Pareto à 1 ;
Répéter jusqu'à $R = \emptyset$
- Trouver F_i, l'ensemble des solutions non dominées correspondants au front de Pareto i ;
- Supprimer de R les solutions non dominées trouvées appartenant au front F_i ;
- i = i + 1 ;

Fin Répéter
Fin

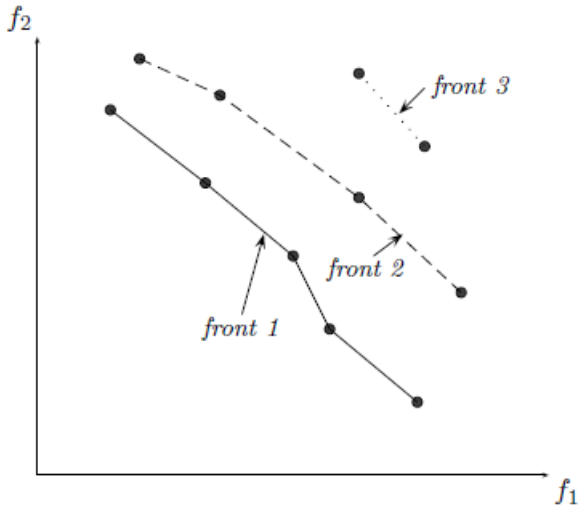

FIG. 4.2 – Classification des solutions en plusieurs fronts Pareto

Pseudo-code de l'algorithme NSGA-II :

Dédut

Création des populations P_0 et Q_0 de taille N.

Tant que critère d'arrêt non rencontré faire

- Poser $R_t = P_t \cup Q_t$;
- Trouver les différents fronts de non-dominance F_i de la population R_t ;
- Poser $P_{t+1} = \emptyset$ et $i = 0$;
- Tant que $|P_{t+1}| + |F_i| < N$ faire

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

  $i = i + 1$

  Fin Tantque
- Trier les solutions du front F_i selon l'ordre décroissant de leurs crowding distance ;
- Ajouter à la population P_{t+1} les $(N - |P_{t+1}|)$ premières solutions du front F_i ;
- Application des opérateurs génétiques pour la génération d'une nouvelle population Q_{t+1} de taille N.

Fin Tantque

Fin

La procédure Crowding distance

Données :

- l : le nombre de solutions dans le front F.
- M : le nombre d'objectifs.
- I^m : le vecteur d'ndices obtenu en ordonnant les solutions de F selon l'ordre croissant des valeurs de l'objectif f_m.
- I_i^m : l'indice de la solution i dans la liste ordonnée selon l'objectif f_m.

Pseudo-code de la procédure crowding distance appliqué au front F :

Début

1. Pour $i = 1$ jusqu'à l faire

   $D_i = 0$.

   Fin pour

2. Pour $m = 1$ jusqu'à M faire

   - Trouver le vecteur d'indices $I^m = \text{Trier}(f_m, <)$
   - $D_1^m = D_l^m = \infty$
   - Pour $i = 2$ jusqu'à $l - 1$ faire

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

     Fin pour

   Fin pour

Fin

La méthode NSGA-II utilise la procédure Crowding distance pour préserver la diversité des solutions. Sachant que la quantité D_i mesure la densité des solutions présentes autour d'une solution i. L'assignation de cette valeur aura comme effet de diminuer les chances de survie d'une solution i présente dans une région où plusieurs autres solutions sont concentrées. Concrètement, comme le montre la figure suivante, le fait de trier le vecteur d'indice I^m et d'assigner une distance très grande aux premières et dernières solutions, pour chaque fonction objectif, permet de donner une priorité aux solutions extrêmes. Géométriquement, la distance de crowding associée à la solution Z_3 correspond au demi-périmètre du cuboïde dont les sommets sont les deux voisins les plus proches de Z_3 (voir fig.4.3).

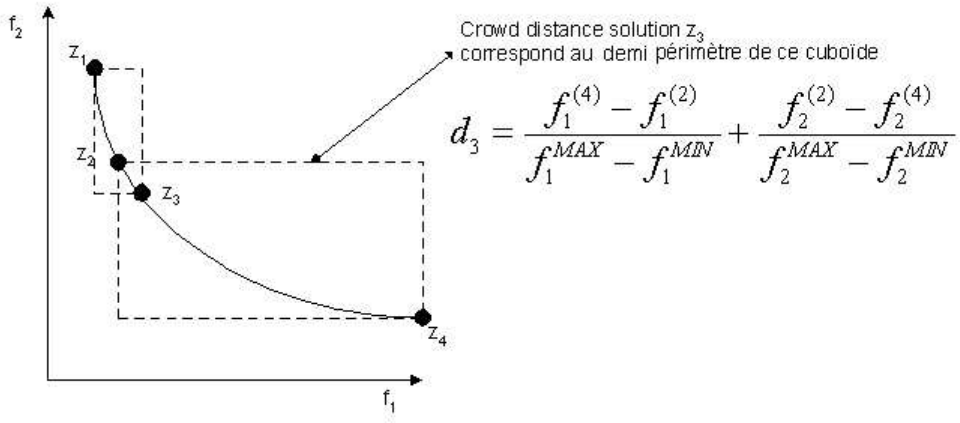

FIG. 4.3 – Illustration du calcul de crowding distance dans NSGA-II.

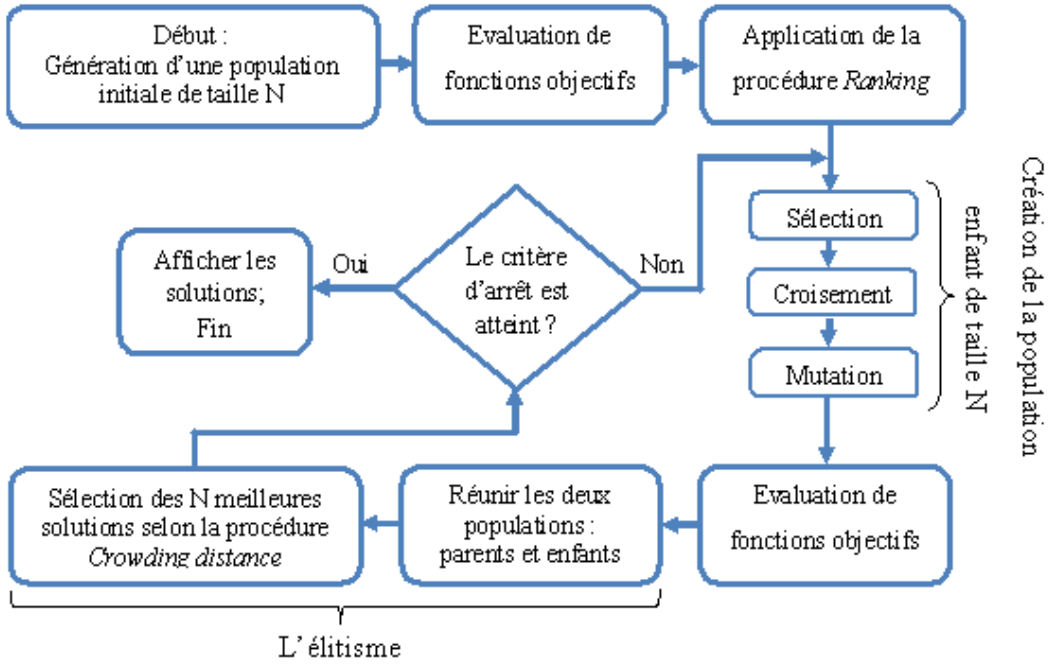

FIG. 4.4 – Organigramme de la méthode NSGA-II.

4.2.2 Heuristique de génération de population réalisable pour le 01-MOKP

On commence par générer aléatoirement N vecteurs à n dimension 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 le plus faible. Ce procédé 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*.

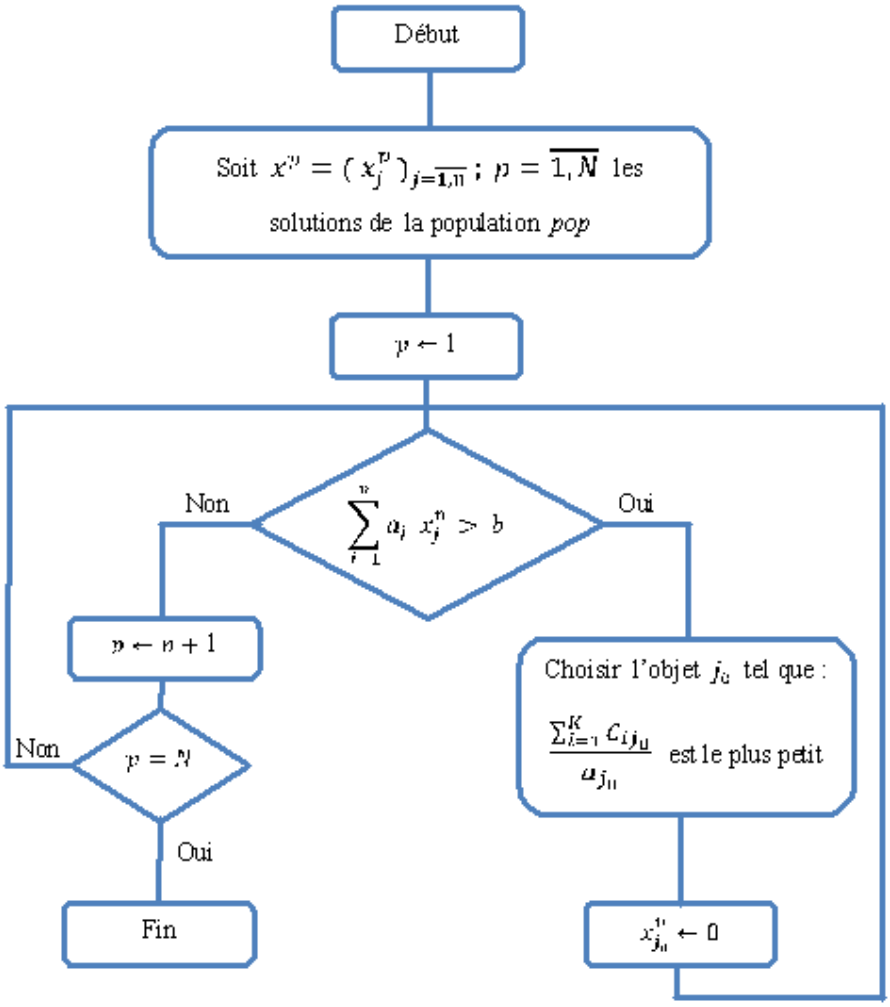

FIG. 4.5 – Organigramme de l'algorithme de génération de population réalisable pour le 01-MOKP.

4.3 Mesure de Performance

Vu la difficulté de juger la qualité des approximations des fronts Pareto optimaux obtenus par les méta, un ensemble de mesures, appelées aussi métriques, ont été proposées dans la littérature [70, 71] pour estimer la performance des métaheuristiques appliquées aux problèmes d'optimisation multiobjectif. 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 EPO.

Absolute Efficiency

Cette mesure permet de calculer la proportion des solutions non dominées de l'ensemble approximé, noté EPA [3, 72] :

$$AE = \frac{|EPA \cap EPO|}{|EPO|}$$

4.4 Paramètres de l'algorithme génétique implémenté

Pour notre étude, nous avons fixé les paramètres de l'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 100 générations.
- La taille de la population varie selon la taille de l'instance ;

L'opérateur de croisement est souvent considéré comme l'opérateur principal dans le processus de recherche 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 de solutions existantes. Rappelons que le croisement permet de produire deux nouvelles solutions, appelées *enfants*, à partir de deux solutions, appelées *parents*, voir section(2.5.1). Le croisement en 1-point et 2-point [43] 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 un et deux points de croisement.

4.5 Implémentation et résultats comparatifs

En complément de notre étude théorique, nous avons programmé l'algorithme NSGA-II adapté au problème *01-MOKP* en utilisant le logiciel Matlab version 7 et effectué

des expérimentations sur des instances du problème *01-MOKP* :

$$\text{“}Max\text{”}\ \{Cx \mid Ax \leq b,\ x \in \{0,1\}\}$$

où A, C et x sont des matrices de taille respective $1 \times n$, $m \times n$ et $n \times 1$.
Ces instances ont été construites grâce au générateur aléatoire décrit ci-dessous :

Algorithme 1 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

$A \leftarrow$ **randint**$(1, n\,[1, 99])$

$C \leftarrow$ **randint**$(m, n, [1, 99])$

$b \leftarrow \sum_{i=1}^{n} A_i/2$

randint$(m, n, [v, w])$: 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 v et w.

Notons que la machine sur laquelle nous avons fait tourner les tests est dotée d'un processeur Intel Pentium 4 cadencé à 2.76 GHz avec 2 Go de RAM.

4.5.1 Résultats obtenus

Dans les tableaux qui suivent, nous illustrons pour des instances 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 réalisé et qualité de solutions trouvées. Dans la littérature il existe plusieurs méthodes exactes dédiées à la résolution de problèmes multiobjecifs en nombres entiers [69, 73, 74]. Dans notre étude comparative nous avons utilisé la méthode de Özlen et Azizoglu pour l'obtention du Front Pareto Optimal [74]
Soit :

- CPU : temps moyen d'exécution (en secondes).
- Nb-SPND : nombres de solutions potentiellement non dominées.
- Nb-SND : nombres de solutions non dominées.
- MOILP : méthode exacte en nombres entiers utilisée pour la résolution du problème *01-MOKP*.

Instances			NSGA-II	
m	n	CPU (MOILP)	CPU (1-point)	CPU (2-point)
2	20	0.05	2.4	2.2
	40	0.6	2.8	1.7
	60	4.5	5.3	5.1
	80	32.6	18.1	27.2
3	20	4.18	4.07	3.9
	40	105.6	53.5	60.3
	60	200.67	64.19	65.7
	80	5428	268	276
4	20	206	4.8	5.6
	25	468	5.3	6.2
	35	846	149	156.8
	40	2207	164	177.2

Instances			NSGA-II		Métrique (AE)	
m	n	Nb-SND (MOILP)	Nb-SPND (1-point)	Nb-SPND (2-point)	AE (1-point)	AE (2-point)
2	20	8	3	6	3/8	6/8
	40	24	4	5	0/24	3/24
	60	51	10	9	5/51	5/51
	80	132	28	35	5/132	9/132
3	20	66	45	38	34/66	33/66
	40	277	104	128	91/277	112/277
	60	269	98	108	47/269	95/269
	80	2013	279	312	163/2013	211/2013
4	20	129	46	66	43/129	63/129
	25	178	64	70	47/178	62/178
	35	275	145	169	130/275	162/275
	40	480	162	219	146/480	99/480

À partir des résultats obtenus relatifs à la mesure d'évaluation de performance AE, on conclut que les deux versions de l'algorithme NSGA-II ont donné de résultats acéptables en termes de qualité des solutions potentiellement non dominées obtenus. De plus, changer l'opérateur de recombinaison de 1-point à 2-point de croisement a montré sur la majorité des instances une amélioration de la convergence des solutions vers le

front Pareto.

Quelques représentations graphiques des résultats réalisés

Le meilleur moyen d'évaluer une approximation du front Pareto, lorsque le problème ne comporte pas plus de trois critères, reste encore l'évaluation graphique. C'est pourquoi la représentation graphique des fronts Pareto optimaux (sur les problèmes le permettant) est considérée comme une information importante lors de l'évaluation des résultats.

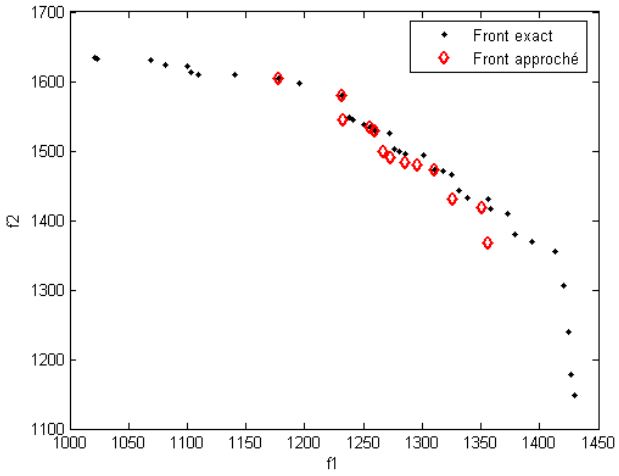

FIG. 4.6 – Front Pareto de *01-MOKP* avec 40 variables (1-point de croisement)

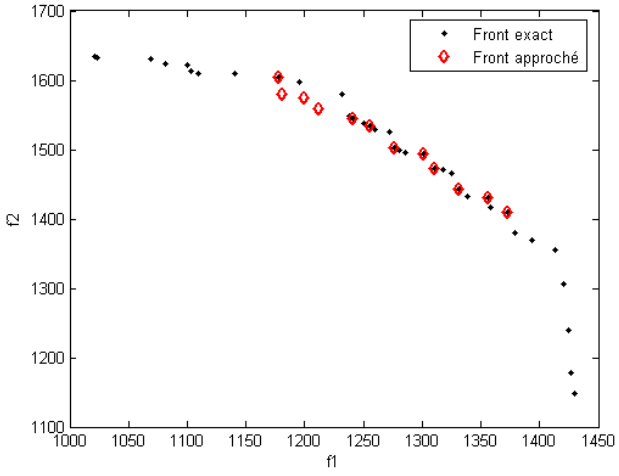

FIG. 4.7 – Front Pareto de *01-MOKP* avec 40 variables (2-point de croisement)

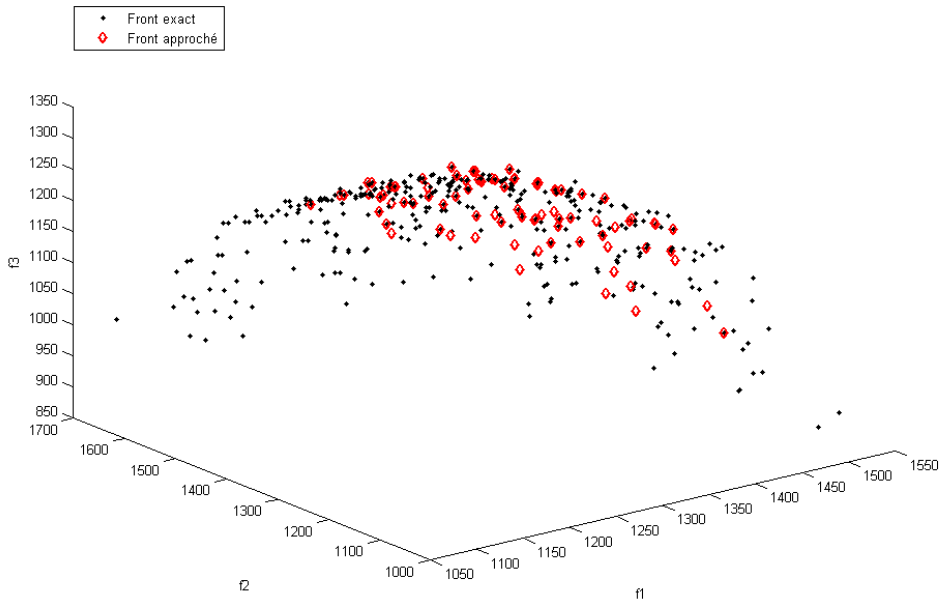

FIG. 4.8 – Front Pareto de *01-MOKP* avec 40 variables (1-point de croisement)

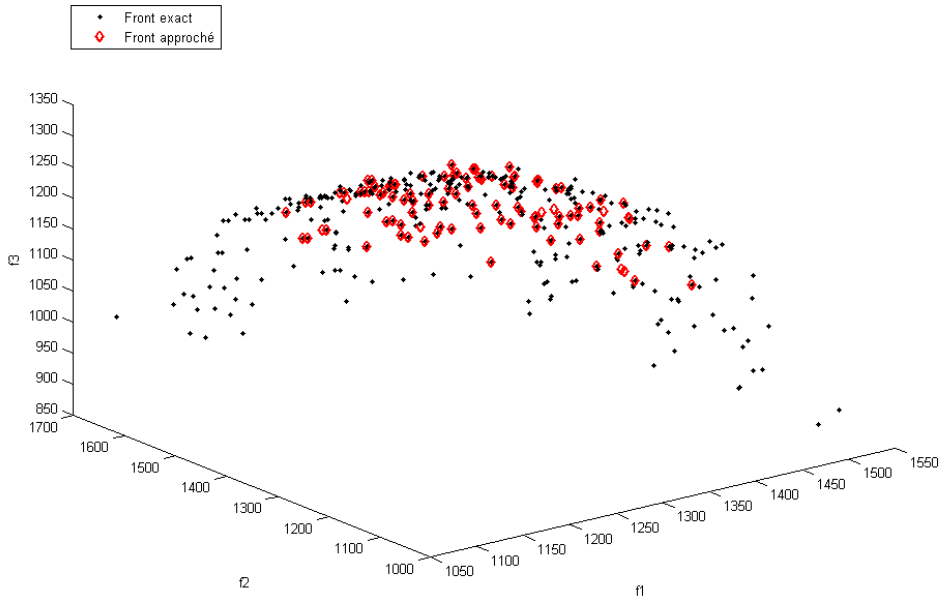

FIG. 4.9 – Front Pareto de *01-MOKP* avec 40 variables (2-point de croisement)

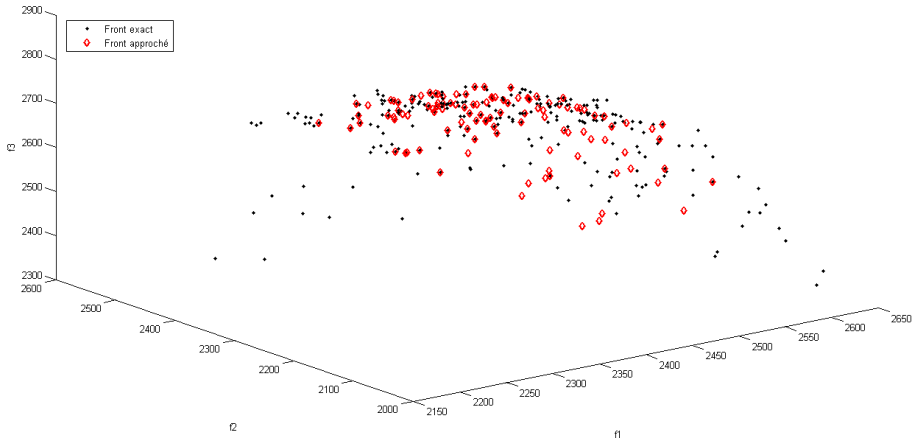

FIG. 4.10 – Front Pareto de *01-MOKP* avec 60 variables (1-point de croisement)

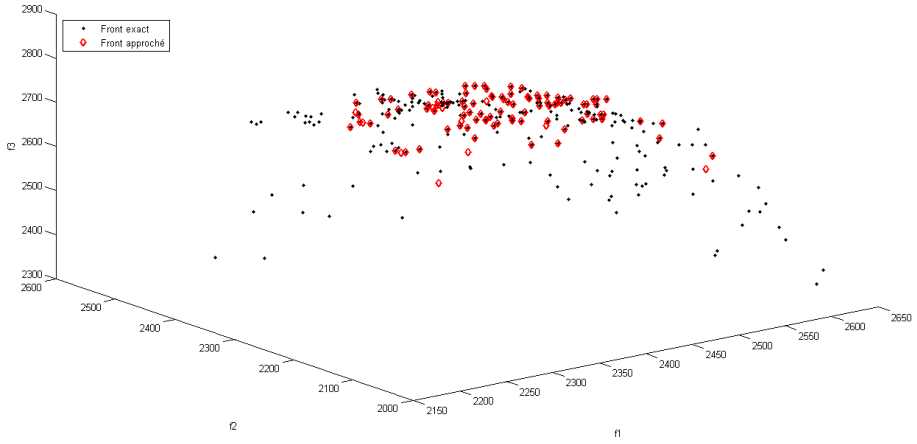

FIG. 4.11 – Front Pareto de *01-MOKP* avec 60 variables (2-point de croisement)

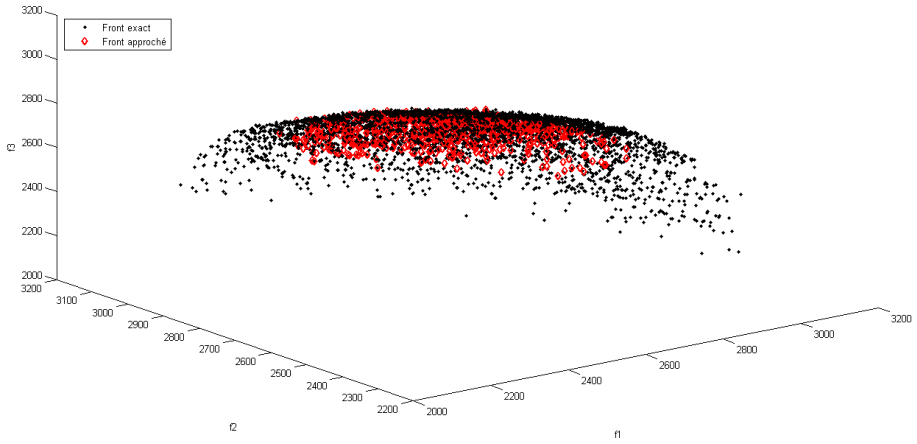

FIG. 4.12 – Front Pareto de *01-MOKP* avec 80 variables (1-point de croisement)

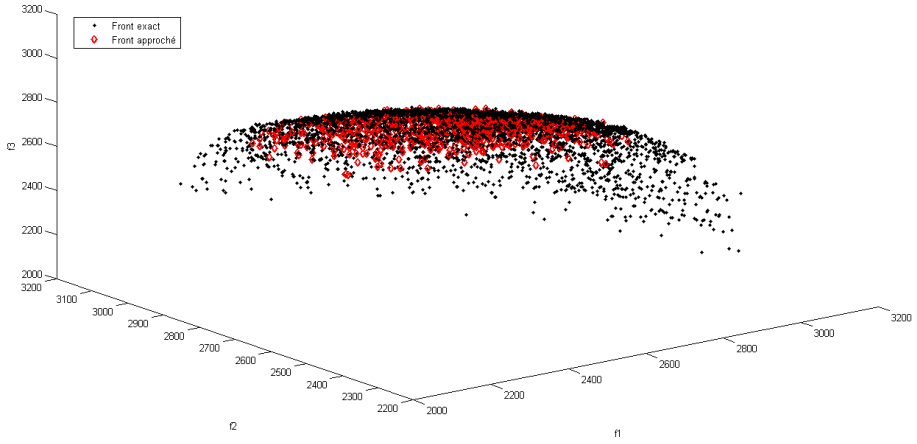

FIG. 4.13 – Front Pareto de *01-MOKP* avec 80 variables (2-point de croisement)

D'après les différentes représentations graphiques obtenus des deux fronts (Pareto optimal et approché), nous constatons que la méthode NSGA-II appliquée au problème de *01-MOKP* génère une approximation du front Pareto de qualité satisfaisante du point de vue convergence des solutions potentiellement non dominées vers le front optimal. Ce résultat est dû à l'utilisation du principe de l'optimalité Pareto dans la procédure de *ranking*, à la technique d'élitisme qui permet de conserver les meilleures solutions rencontrées durant le processus de recherche et au mécanisme de préservation de la diversité *Crowding distance* qui assure une bonne distribution des solutions potentiellement non dominées sur le front Pareto.

4.6 Conclusion

L'objectif principal de ce chapitre été de montrer, via une étude empirique comparative, la qualité de l'approximation du front pareto obtenu par l'algorithme NSGA-II ainsi que l'influence de l'opérateur de croisement sur la performance d'un MOEA en termes de convergence. D'après les résultats comparatifs mentionnés précédemment, on conclut qu'en comparant avec une méthode de résolution exacte, la méthode NSGA-II adaptée au *01-MOKP* permet d'obtenir un ensemble de solutions potentiellement non dominées de bonne qualité durant un temps d'exécution raisonable. Au stade actuel de la recherche, elle est considérée comme une référence incontournable dans la communauté des métaheuristiques pour l'optimisation multiobjetif.

Conclusion générale et perspectives

De nombreux problèmes rencontrés dans la pratique nécessitent la prise en charge de plusieurs objectifs qui sont souvent conflictuels. Pour ce type de problèmes, 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 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 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. La diversification permet de mieux échantillonner la frontière Pareto.

Nous avons commencé ce travail par introduire les notions de base relatives à l'optimisation multiobjectif, ce qui a fait l'objet du premier chapitre de ce mémoire.
Après une étude détaillée,dans le deuxième chapitre, de quelques métaheuristiques existantes d'optimisation multiobjectif en l'occurrence, les algorithmes évolutionnaires multiobjectifs ; Nous avons implémenté et appliqué l'algorithme NSGA-II au problème de sac à dos multiobjectif en variables binaires. Les résultats obtenus par cette méthode ont été exposés, illustrés sur plusieurs instances. À cet effet, nous avons programmé en utilisant le logiciel Matlab version 7, toutes les procédures nécessaires intervenant dans la conception de la méthode NSGAII, entre autres, les techniques de Ranking et

la procédure du Crowding distance ainsi que les différents opérateurs de l'algorithme génétique (sélection, croisement et mutation).

D'après les résultats expérimentaux obtenus nous constatons que NSGAII appliqué au problème *01-MOKP* a montré de résultats satisfaisants en termes de convergence et de diversité pour les deux versions de l'algorithme (un et deux points de croisement) avec une amélioration de convergence enregistrée pour le croisement en deux points. Cela confirme que cet opérateur a une influence directe sur l'exploration de l'espace de recherche et sur la création de nouvelles solutions permettant d'arriver à des zones du front non encore atteintes.

Pour nos recherches futures et dans la continuité du travail réalisé dans ce mémoire, nous espérons :

- Examiner l'effet d'autres opérateurs (opérateur de sélection, le choix de la population initiale, opérateur de mutation, etc.), sur la performance des algorithmes évolutionnaires multiobjectifs.
- Faire coopérer quelques métaheuristiques entre elles ou avec des méthodes exactes afin d'allier aux mieux les qualités de chacune d'elle.

Bibliographie

[1] Pareto, V. Cours d'Economie Politique. Rouge, Lausanne, Switzerland, 1986.

[2] Coello, C. A., Veldhuizen, D. A., & Lamont, G. B. (2002). Evolutionary algorithms for solving multiobjective problems. Kluwer Academic Publishers.

[3] Collette Y., Siarry P., Optimisation Multiobjectif ; Septembre 2002 ;Edition Eyrolles, 75240 Paris Cedex 05, France ; Pages $17 - 80$.

[4] Steuer R,Multiple Criteria Optimization : Theory, Computation and Applications, John Wiley & Sons, New-York (1985).

[5] Sen, T., Raiszadeh, M., and Dileepan, P. (1988). A branch and bound approach to the bicriterion scheduling problem involving total flowtime and range of lateness. Management Science,34(2) 254-260.

[6] Stewart, B. and White, C. (1991). Multiobjective A*. Journal of the ACM, 38(4) : $775 - 814$.Surry, P., Radcliffe, N., and Boyd, I. (1995). A multi-objective approach to constraint optimisation of gas supply networks : The COMOGA method. In Fogarty, T., editor, Evolutionary Computing, AISB Workshop, LNCS, pages $166 - 180$, Sheffield, U.K. Springer–Verlag.

[7] White, D. (1982). The set of efficient solutions for multiple-objectives shortest path problems.Computers and Operations Research, 9 :101-107.

[8] Vincke Ph,Analysis of Multicriteria Decision Aid in Europe, European Journal of Operational Research 25 (1986), pp. 160-168.

[9] Geoffrion A.M., Proper efficiency and the theory of vector maximization. Journal of Mathematical Analysis and Applications 22, pages 618-630, 1968.

[10] Multiple objective decision making - methods and applications. In Lectures Notes in Economics and Mathematical Systems, volume 164. Springer-Verlag, Berlin.

[11] 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 :296-297, 1971.

[12] Serafini, P. 1992. Simulated annealing for multiple objective optimization problems. In Tenth Int. Conf. on Multiple Criteria Decision Making, pages 87,96.

[13] Friesz, T., Anandalingam, G.,Mehta, N., Nam, K., Shah, S., and Tobin, R. (1993). The multiobjective equilibrium network design problem revisited : A simulated annealing approach. European Journal of Operational Research, 65 :44-57.

[14] Ulungu, E. L., Teghem, J., Fortemps, P. H., & Tuyttens, D. (1999). MOSA method : a tool for solving multiobjective combinatorial optimization problems. Journal of Multicriteria Decision Analysis , 8 (4), 221-236.

[15] Dahl, G., Jornsten, K., and Lokketangen, A. (1995). A tabu search approach to the channel minimization problem. In Liu, G., Phua, K.-H.,Ma, J., Xu, J., Gu, F., and He, C., editors, Optimization Techniques and Applications, ICOTA'95, volume 1, pages 369-377, Chengdu, China. World Scientific.

[16] Syswerda, G. and Palmucci, J. (1991). The application of genetic algorithms to resource scheduling. In Belew, R. and Booker, L., editors, Fourth Int. Conf. on Genetic Algorithms ICGA'4, pages 502-508, San Mateo, California. Morgan Kaufmann Pub.

[17] Jakob, W., Gorges-Schleuter, M., and Blume, C. (1992). Application of genetic algorithms to task planning and learning. In Manner, R. and Manderick, B., editors, Parallel Problem Solving from Nature PPSN'2, LNCS, pages 291-300, Amsterdam. North-Holland.

[18] Yang, X. and Gen, M. (1994). Evolution program for bicriteria transportation problem. In Gen, M. and Kobayashi, T., editors, 16th Int. Conf. on Computers and Industrial Engineering, pages 451-454, Ashikaga, Japan.

[19] Hertz, A., Jaumard, B., Ribeiro, C., and Filho,W. F. (1994). A multi-criteria tabu search approach to cell formation problems in group technology with multiple objectives. RAIRO Recherche Opérationnelle, Operations Research, 28(3) :303-328.

[20] Veldhuizen, D. V., Sandlin, B., Marmelstein, R., Lamont, G., and Terzuoli, A. (1997). Finding improved wire-antenna geometries with genetic algorithms. In Chawdhry, P., Roy, R., and Pant, P., editors, Soft Computing in Engineering Design and Manufacturing, pages 231-240, London. Springer Verlag.

[21] Ritzel, B., Eheart, J., and Ranjithan, S. (1994). Using genetic algorithms to solve a multiple objective groundwater pollution problem. Water Resources Research, 30(5) :1589-1603.

[22] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley.

[23] Coello Coello C.A. and Lamont G.B. Applications of Multi-Objective Evolutionary Algorithms. World Scietific Publishing co., 2004.

[24] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers and Operations Research , 13 (5), 533-549.

[25] Tabu search procedure for solving the 0/1 multiobjective knapsack problem : the two objectives case. Journal of Heuristics , 6, 361-383.

[26] Armentano, V. A., & Claudio, J. E. (2004). An application of a multiobjective tabu search algorithm to a bicriteria flowshop problem. Journal of Heuristics , 10 (5), 463-481.

[27] Glover, F., & Laguna, M. (1997). Tabu search. Kluwer Academic Publishers.

[28] P. Engrand. (1997). A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management, Proceedings of the 5th International Conference on Nuclear Engineering, pp. 416-423, 1997, American Society Of Mechanical Engineers.

[29] Dorigo & Stützle, 2004 Ant colony optimization. MIT Press.

[30] Dorigo, M., & Caro, G. D. (1999). The ant colony optimization metaheuristic. New Ideas in Optimization , 11-32.

[31] Mariano, C. E., & Morales, E. M. (1999). A Multiple Objective Ant Algorithm for the Design of Water Distribution Irrigation Networks ; Technical Report HC 9904 ; Mexico Institut.

[32] Gambardella L. M., Taillard E. et Agazzi G., A multiple ant colony system for vehicule routing problems with time windows ; D. Corne. M.Dorigo et F. Glover Editors ; New ideas in optimization ; McGraw-Hill, page 63,76.

[33] Iredi S., Merkle D. & Middendorf M., Bi-Criterion Optimizalion with MultiColony Am Algorithms ; First International Conference (FMO'Ol) ; Zurich,Springer Verlag, page 359-372.

[34] Doerner K. et al.(2006). Pareto Ant Colony Optimization with ILP preprocessing in multiobjective project portofolio selection, European Journal of Operational Research, vol. 171, pp. 830-841, 2006.

[35] Kennedy J. and Eberhart R., Particle swarm optimization. Neural Networks,1995.

[36] Clerc M. and Kennedy J., The particle swarm explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, 2002.

[37] Coello C. and Lechunga M. MOPSO : A proposal for multiple objective particle swarm optimization, 2002. Computational Intelligence, Hawaii, May 12-17, 2002. IEEE Press.

[38] Ho, Ku, Jou, & Hung, (2006), Intelligent particle swarm optimization in multi-objective problems. Lecture Notes in Artificial Intelligence , 3918, 790-800.

[39] Baumgartner, Magele, & Renhart, (2004), Pareto optimality and particle swarm optimization. IEEE Transactions on Magnetics , 40 (2), 1172-1175.

[40] Holand, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor, MI : MIT Press.

[41] De Jong, K. A., & Sarma, J. (1995). On decentralizing selection algorithms. Proceedings of the 6th International Conference on Genetic Algorithms (pp. 17-23). Morgan Kaufmann.

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

[43] De Jong, K. A., & Spears, W. M. (1991). An analysis of multi-point crossover. Foundations of Genetic Algorithms , $301 - 315$.

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

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

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

[47] Srinivas, N. and Deb, K. 1995. Multiobjective optimisation using non-dominated sorting in genetic algorithms. Evolutionary Computation, p : 221,248.

[48] Fonseca, C. and Fleming, P. 1995. Multiobjective genetic algorithms made easy : Selection, sharing and mating restrictions. In IEEE Int. Conf. on Genetic Algorithms in Engineering Systems : Innovations and Applications, pages 45,52, Sheffield, UK.

[49] Coello Coello C.A., An updated survey of GA-based multiobjective optimization techniques. Technical report, Lania-RD-98-08, Xalapa, Veracruz, Mexico, 1998.

[50] Zitzler E. and Thiele L., Multiobjective evolutionary algorithms : a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation,3, 257-271, 1999.

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

[52] 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), pages 849,858, 2000.

[53] Talbi, E.G. Une taxinomie des métaheuristiques hybrides. Dans ROADEF'2000, 2000.

[54] 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.

[55] Schaffer JD., Multiple objective optimization with vector evaluated genetic algorithms. In : Proceedings of the international conference on genetic algorithm and their applications, 1985.

[56] Hajela P., lin C-y., Genetic search strategies in multicriterion optimal design. Struct Optimization 1992 ;4(2) :99-107.

[57] Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization : formilation, discussion and generalization. Proceedings of the 5th International Conference on Genetic Algorithms (pp. 416-423). Morgan Kaufmann.

[58] Srinivas, N., & Deb, K. (1994). Multiobjective optimization using non dominated sorting in genetic algorithms. Evolutionary Computation , 2 (3), 221-248.

[59] Horn J., Nafpliotis N., Goldberg DE. 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. Orlando, FL, USA :IEEE ; 1994.

[60] Zitzler E, Thiele L. Multiobjective evolutionary algorithms : a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 1999 ;3(4) :257-71.

[61] Zitzler E., Laumanns M., Thiele L. SPEA2 : improving the strength Pareto evolutionary algorithm. Swiss Federal Institute Techonology : Zurich, Switzerland ; 2001.

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

[63] Coello Coello C.A., An updated survey of GA-based multiobjective optimization techniques. ACM Compt surv 2000 ; 32(2) : 109-143.

[64] Deb K., Multi-objective optimization using evolutionary algorithms. John Wiley and sons, 2001.

[65] Ishibuchi, H., Narukawa, K., Tsukamoto, N., & Nojima, Y. (2008). An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization. European Journal of Operational Research , 188, 57-75.

[66] Coello Coello C.A. and Lamont G.B., Applications of Multi-Objective Evolutionary Algorithms. World Scietific Publishing co., 2004.

[67] Veldhuizen, D. A., et Lamont, G. B. (2000). Multiobjective evolutionary algorithms : Analyzing the state of the art. *Evolutionary Computation*, 8(2) :127-147, 2000.

[68] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Knapsack problems. Springer.

[69] Mohamed El-Amine Chergui and Mustapha Moulaï, An Exact Method for a Discrete Multiobjective Linear Fractional Optimization, Journal of Applied Mathematics and Decision Sciences, vol. 2008, Article ID 760191, 12 pages, 2008. doi :10.1155.2008.760191

[70] Knowles J.D. and Corne D.W., On metrics for comparing non-dominated sets. In Congress on Evolutionnary Computation, IEEE Press, pages 711-716, 2002.

[71] Hansen M.P. and Jaszkiewicz A., Evaluating the quality of approximations of non dominated set. Tech. Rep., Institute of Mathematical modeling, Tech. Univ of Denmark,1998. IMM Tech. Rep. IMM-REP-1998-7.

[72] An improved MOSA method for solving multi-objective combinatorial optimization problems.

[73] J. Sylva and A. Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. European Journal of Operational Research 158, pages 46-55, 2004.

[74] Özlen Melih and Meral Azizoglu, Multi-Objective Integer Programming : A General Approach for Generating all Non-Dominated Solutions, European Journal of Operational Research, 199, 25-35, 2009.