République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj
Faculté des Mathématiques et de l'Informatique
Département de Recherche Opérationnelle.

Mémoire

Présentée par :

Ghanem khaled

Pour l'obtention du diplôme de :

Master

Filière : Des Mathématiques appliquées
Spécialité : Méthodes et outils pour la recherche Opérationnelle

---

Thème :

Résolution d'un problème de sac à dos bi-objectif

Soutenu publiquement **le 15/07/ 2023** devant le jury composé de

M. .Hillal touati... .... M.A.A. Président
M. Brahmi boualem M.C.B. Encadrant
M.Zauache djaafar .A.A. Examinateur

Promotion 2022/2023

Remerciements

Nous remercions "Allah" tout puissant de nous avoir donné la force, le courage et la patience pour l'élaboration de ce modeste travail.

Nous remercions notre encadreur **Dr Brahmi boualem**, Maitre de Conférence à l'université Mohamed El Bachir El Ibrahimi, Bordj Bou Arréridj pour son encadrement de qualité, sa motivation professionnelle, ses conseils et critiques constructives, ses correction, sa gentillesse et sa patience ainsi pour le temps qu'elle a cosacré à la réalisation de ce travail.

Nous remercions les membres du jury pour leur présence, pour leur lecture attentive de ce mémoire, ainsi que pour les remarques qi'ils m'adresseront lors de cette soutenance afin d'améliorer mon travail.

Nous remercions tous les enseignants de la faculté des Mathématiques et Informatique de l'université Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj. de Recherche Opérationnelle 2022-2023.

Merci à tous

Dédicaces

Je dédie ce travail à...

A celui qui m'a donné l'éducation de la connaissance et de la religion et m'a soutenu dans mon épreuve

Mon père...

À qui les nuits sont-elles restées éveillées pour être avec moi et m'inclure dans ses prières de toutes les manières heure et quand

Ma mère...

Ils sont acquittés et comblés

A ceux qui m'ont donné un amour fraternel pur et sincère, mes frères et sœurs.....

A tout ami qui exprime sincèrement une position authentique, un mot de soutien ou une prière

Invisible est apparu avec une intention pure Je dédie le fruit de cet humble effort

Khaled

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES ALGORITHMES

INTRODUCTION GÉNÉRALE :

La recherche opérationnelle (RO) est la discipline des mathématiques appliquées qui traite des questions d'utilisation optimale des ressources dans l'industrie et dans le secteur public. Depuis une dizaine d'années, le champ d'application de la RO s'est élargi à des domaines comme l'économie, la finance, le marketing et la planification d'entreprise. Plus récemment, la RO a été utilisée pour la gestion des systèmes de santé et d'éducation, pour la résolution de problèmes environnementaux et dans d'autres domaines d'intérêt public.

L'optimisation est une branche des mathématiques appliquées et de la recherche opérationnelle.
Un problème d'optimisation est défini comme étant un problème de recherche, qui consiste à explorer un espace contenant l'ensemble de toutes les solutions potentielles réalisables, dans le but de trouver la solution optimale, sinon la plus proche possible de l'optimum, permettant de minimiser ou maximiser une fonction dite objectif, les variables de décision peuvent être soit continues et on parle alors de problème continu, soit discrètes et on parle donc de problème combinatoire.

Un problème d'optimisation combinatoire (COP : Combinatorial Optimisation Problems) comprend un ensemble fini de solutions, ou chaque solution doit respecter un ensemble de contraintes relatives a la nature du problème. On associe a chaque solution une valeur, nommele valeur de l'objectif qui est evalue a l'aide

d'une fonction objective. les problèmes d'optimisation combinatoire peuvent être mono-objectifs qui consistent a optimiser une seule fonction objective, ou multi-objectifs qui optimisent plusieurs fonctions objectives.

L'optimisation combinatoire regroupe une large classe de problèmes ayant des applications dans de nombreux domaines applicatifs parmi eux on peut citer (le problème du voyageur de commerce et le problème de sac à dos).

Le problème de sac a dos : il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum. Ce problème sera décrit d'une manière plus accentué dans le première chapitre.

L'optimisation multi-objectifs est un domaine d'étude important en recherche opérationnelle, à cause de la nature multi-objectifs de la plupart des problèmes réels. Les premiers travaux menés sur les problèmes multi-objectifs furent réalisés au 19éme siècle sur des études en économie par Francis Y. Edgeworth (1845-1926) et elle a été utilisée de manière plus formelle par l'économiste italien Vilfredo Pareto (1848-1923).En effet, L'optimisation multi-objectifs permet de saisir de manière adéquate les caractéristiques essentielles des problèmes du monde réel et d'améliorer leur perception par les décideurs.

Il existe plusieurs méthode de résolution pour les problèmes d'optimisation combinatoire multi-objectif, Ces méthodes appartiennent à deux grandes familles :

— les méthodes exactes (Branch and bound, Programmation dynamique, les méthodes des coups, la méthode en deux phases...)

— les méthodes approchées (le recuit simulé, la recherche tabou, les algorithmes génétique et colonies de fourmis. . .)

La méthode en deux phases est un cadre de résolution général qui a été popularisé par Ulungu en 1993 avec comme idée centrale d'exploiter la structure spécifique des problèmes d'optimisation combinatoire pour leur résolution dans un contexte multi-objectif. Elle a depuis été appliquée sur un grand nombre de problèmes, en

se limitant toutefois au contexte bi-objectif [38]. Comme son nom l'indique, cette méthode est décomposée en deux étapes : la première consiste à trouver toutes les solutions supportées du front Pareto, puis la deuxième phase cherche parmi ces solutions les solutions Pareto non supportées.

Ce mémoire est organisé et conçu comme suit :

En commençant, par une introduction qui explique et motive le choix du sujet.

- ***Le premier chapitre :*** On présenté le problème du sac à dos et les méthodes de résoudre exactes et approchées pour ce problème.
- ***Le deuxième chapitre :*** nous essayons d'aborder quelques notions de base en optimisation et on définit le problème d'optimisation MultiuObjectif , avec ses concepts fondamentaux : (la domination, Front de Pareto et La surface de compromis, la convexité...). Puis on décrit les méthodes de résolution.
- ***Le troisième chapitre :*** est le coeur de ce travail, ils décrivent la méthode en deux phases appliquée à un problème de sac à dos bi-objectif.

Finalement, on terminera le mémoire par une conclusion.

CHAPITRE 1

LE PROBLÈME DE SAC À DOS(KNAPSACK)

1.1 Introduction

Les problème du knapsack classique (sac-à-dos) est un problème de sélection qui consiste à maximiser un critère de qualité sous une contrainte linéaire de capacité de ressource. Il doit son nom à l'analogie qui peut être faite avec le problème qui se pose au randonneur au moment de remplir son knapsack : il lui faut choisir les objets à emporter de façon à avoir un sac le plus utile possible, tout en respectant

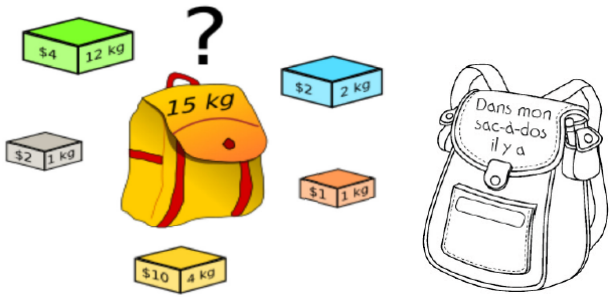


FIGURE 1.1 – Problème du sac à dos.

son volume, la question qui se pose est de savoir quels objets il doit mettre dans le sac. Il existe plusieurs variantes du problème du knapsack. Elles diffèrent dans la distribution des objets et des sacs. Dans le problème du knapsack en variables binaires, chaque objet peut être pris dans le sac au plus une fois. En revanche, dans un problème du knapsack borné, nous avons une quantité bornée de chaque type d'objet. Dans le problème du knapsack à choix multiple, les objets sont à choisir à partir de classes disjointes. S'il existe plusieurs sacs à remplir simultanément, le problème du knapsack est multidimensionnel. Notons que toutes les variantes de problème du knapsack sont NP-difficiles. Les problèmes knapsacks ont été intensivement étudiés depuis le travail de Dantzig [9] à la fin des années cinquante. Leur importance dans les applications industrielles ainsi qu'en planification financière ont aussi nécessité la recheche des méthodes de résolution efficaces. Ces problèmes sont aussi d'une grande importance théorique vu qu'ils interviennent comme sous-problèmes dans plusieurs problèmes en nombres entiers. En effet, le problème du knapsack unidimensionnel intervient, par exemple, comme sous-problème dans le problème de découpe (voir Gilmore et Gomory [4]) lors de la génération d'un pivot du Simplexe. Il intervient également dans la résolution du problème d'affectation généralisée [29]. Par ailleurs, les différentes variantes de problème du knapsack, bien qu'elles semblent très proches, ne font pas du tout appel aux mêmes méthodes de résolution (exactes ou approchées) pour une variante donnée. Dans ce chapitre, nous rappelons quelques méthodes de résolution du knapsack unidimensionnel et nous présentons une de ses variantes que nous avons étudiée ainsi que l'essentiel des approches de résolution qui lui ont été dédiées.

1.2 Historique

Ce problème fait partie des 21 problèmes NP-complets identifiés par Richard Karpen 1972. Ces 21 problèmes sont réputés comme les problèmes les plus difficiles en optimisation combinatoire. Un grand nombre d'autres problèmes NP-complets peuvent se ramené à ces 21 problèmes de base. Nous pouvons retrouver le problème du sac à dos dans de nombreux domaines :

- en cryptographie, où il fut à l'origine du premier algorithme de chiffrement asymétrique en 1976.
- dans les systèmes financiers, où l'idée est la suivante : étant donné un certain montant d'investissement dans des projets, quels projets choisir pour que le tout rapporte le plus d'argent possible.
- pour la découpe de matériaux, afin de minimiser les pertes dues aux chutes.
- dans le chargement de cargaisons (avions, camions, bateaux ...).
- ou encore, dès qu'il s'agit de préparer une valise ou un sac à dos.

1.3 Le problème de sac à dos classique (knapsack)

On considère un ensemble d'objets étiquetés de 1 à n. Chaque objet $j \in \{1,...,n\}$, dispose d'un poids w_j de valeur entière et d'un profit v_j (réel à priori). On dispose d'un sac dont le contenu ne peut excéder une capacité C entière. On désire le remplir de façon à maximiser le profit total des objets emportés, en respectant la contrainte de capacité. Plus formellement, le problème, noté (KP) peut s'écrire de la façon suivante :

$$(KP) = \begin{cases} max\ Z(x) = \sum_{j=1}^{n} v_j x_j \\ s.c.\ \sum_{j=1}^{n} w_j x_j \leq C, \\ x_j \in \{0,1\}, j \in \{1,...,n\} \end{cases}$$

On appelle une instance d'un problème du knapsack, la donnée des n profits v_j pour $j = 1,...,n$, des n poids w_j pour $j = 1,...,n$ et de la capacité C.
Le vecteur $x := (x_j, 0 \leq j \leq n) \in \{0,1\}^n$ est une solution du problème avec $x_j = 1$, si l'objet j est emporté dans le sac et $x_j = 0$ sinon. Le problème consiste donc à choisir un sous-ensemble d'objets parmi la collection d'objets initialement prévue en vue de maximiser la fonction objectif :

$$Z(x) = \sum_{j=1}^{n} v_j x_j$$

Nous formulons l'hypothèse suivante :

$$\forall j \in \{1,...,n\}, w_j \leq C \; et \; \sum_{j=1}^{n} w_j > C.$$

On dit qu'une solution est réalisable, notée $\overline{x}$, si elle vérifie la contrainte de capacité,c'est-à-dire $\sum_{j=1}^{n} w_j \overline{x}_j \leq C$. La solution est dite optimale, notée x^*, si elle est à la fois réalisable et si elle maximise la somme des valeurs profits des objets mis dans le sac. En d'autres termes, pour toute solution réalisable $\overline{x}$, on a :

$$\sum_{j=1}^{n} v_j \overline{x}_j \leq \sum_{j=1}^{n} v_j x_g^*.$$

On peut illustrer le problème KP dans un exemple pratique très simple. Supposons que n projets s'offrent à un investisseur qui dispose d'un fond de "C euros". Sachant que le profit du j^{me} projet est $p_j, (j = 1,...,n)$ et qu'investir dans ce projet coûte w_j . Alors, un investissement optimal peut être trouvé en résolvant un problème du knapsack en 0-1.

1.4 Variantes autour du problème

Il existe de nombreuses variantes du problème de sac à dos, selon le domaine des variables (valeurs binaires, entières ou réelles), le nombre de contraintes (unidimensionnel, bi-dimensionnel ou multi-dimensionnel), le nombre de profits associés à chaque objet (mono-objectif ou multi-objectif), le nombre de sacs, etc. Du fait de la quantité des paramètres intervenant dans la formulation, les variantes sont nombreuses. On présente dans cette section quelques unes d'entre elles.

1.4.1 Variables continues

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

Définition 1.1. ***(efficacité d'un objet)*** *On appelle efficacité d'un objet le rapport de son coût sur son poids, noté* $\boldsymbol{e_j = \frac{c_j}{w_j}}$.

Définition 1.2. ***(élément bloquant)*** *On appelle élément bloquant le premier objet ne pouvant tenir dans le sac lorsque les objets sont ajoutés par ordre décroissant d'efficacité. Son indice sera noté*

$$\boldsymbol{s = min\{k : \sum_{j=1}^{k} w_j > C\}\ pour\ les\ e_j\ tris\ en\ orde\ dcroissant}$$

La valeur optimale z^*(LKP) d'une instance est obtenue en prenant les objets dans l'ordre décroissant de leur efficacité, jusqu'à l'élément bloquant puis en ajoutant la fraction de cet élément permettant de saturer le sac.

$$\boldsymbol{z^*(LKP) = \sum_{j=1}^{s-1} v_j + (C - \sum_{j=1}^{s-1} w_j)\frac{v_s}{w_s}}$$

Le problème (LKP) est un problème relaxé de (01-KP). Il s'obtient en remplaçant la contrainte $\boldsymbol{x_j \in \{0,1\}}$, par $\boldsymbol{x_j \in [0;1], \forall j\{1,...,n\}}$.
Il est définit ainsi :

$$\boldsymbol{(LKP)} \begin{cases} \boldsymbol{Max\ z(x) = \sum_{j=1}^{n} v_j x_j \quad v_j \in N^*} \\ \boldsymbol{s.c:} \\ \boldsymbol{\sum_{j=1}^{n} w_j x_j \leq C \quad C, w_j \in N^*} \\ \boldsymbol{x_j \in [0,1], \quad \forall j \in \{1,...,n\}} \end{cases}$$

1.4.2 Sac à dos multi-dimensionnel

Le problème de sac à dos multi-dimensionnel (d-KP), est une variante du problème ayant plusieurs contraintes de capacité. Il est à la frontière entre l'optimisation combinatoire et la programmation linéaire en nombres entiers. Son champ d'application est large, ce qui a grandement contribué à sa popularité.

$$(d-KP)\begin{cases} Max\ z(x)=\sum_{j=1}^{n} v_j x_j \quad v_j \in N^* \\ s.c: \\ \sum_{j=1}^{n} w_j^i x_j \leq w_i \quad w_i, w_j \in N_{>}^{d}, i \in \{1,...,d\} \\ x_j \in \{0,1\}, \quad \forall j \in \{1,...,n\} \end{cases}$$

1.4.3 Sac à dos multiple

Le problème du sac à dos multiple (MKP) est une généralisation du problème standard du sac à dos en variable (0-1), où on essaie de remplir plusieurs sacs à dos de différentes capacités au lieu de considérer un seul sac à dos [31], Le (MKP) peut être formulé de la manière suivante :

$$(MKP)\begin{cases} Max\ z(x)=\sum_{j=1}^{n}\sum_{i=1}^{m} v_j x_{ji} \quad v_j \in N^* \\ s.c: \\ \sum_{j=1}^{n} w_j x_{ji} \leq w_i \quad i \in \{1,...,m\} \\ \sum_{i=1}^{m} x_{ji} \leq 1 \quad \forall j \in \{1,...,n\} \\ x_{ji} \in \{0,1\}, \quad \forall j \in \{1,...,n\}, i \in \{1,...,m\} \end{cases}$$

On peut citer le chargement de conteneurs sur les navires comme application du monde réel du MKP. Le problème est de choisir certains conteneurs dans un ensemble de n conteneurs à charger dans m navires de différentes capacités de chargement.

1.4.4 Sac à dos multi-objectif

Le problème du sac à dos multi-objectif (MOKP) en varibale binaire (0-1), est une variante du problème ou plusieurs objectifs sont à maximiser simultanément

$$(MOKP)\begin{cases} Max\ z_i(x) = \sum_{j=1}^{n} v_j^i x_j \quad i \in \{1,...,p\}, v_j \in N_>^p \\ s.c: \\ \sum_{j=1}^{n} w_j x_j \leq C \quad C, w_j \in N^* \\ x_j \in \{0,1\}, \quad \forall j \in \{1,...,n\} \end{cases}$$

Dans ce cas la notion d'optimalité laisse place à la notion d'efficacité, comme on la déja vu dans le chapitre 2, l'objectif est donc de trouver l'ensemble des solutions du problème non dominées.

1.4.5 Sac à dos multidimensionnel multi-objectif

Le problème de sac à dos multi-objectif multidimensionnel (MMOKP) consiste à sélectionner un sous-ensemble d'objets maximisant une fonction multi-objectifs exprimée en fonction des valeurs de profit, tout en respectant un ensemble de contraintes de type sac à dos. Le problème du (MMOKP) peut être défini plus formellement de la manière suivante :

$$(MMOKP)\begin{cases} Max\ z_i(x) = \sum_{j=1}^{n} v_j^i x_j \quad i \in \{1,...,p\}, v_j \in N_>^p \\ s.c: \\ \sum_{j=1}^{n} w_j^l x_j \leq C_l \quad l \in \{1,...,q\} C, w_j \in N^* \\ x_j \in \{0,1\}, \quad \forall j \in \{1,...,n\} \end{cases}$$

1.5 Méthodes de résolution pour Le problème de sac à dos (knapsack)

Il est difficile de développer un algorithme polynomial pour le problème du knapsack (KP) vu que ce dernier appartient à la classe NP-difficile. Malgré cette difficulté, plusieurs instances de grande taille peuvent être résolues en une fraction

de seconde. Ce résultat impressionant est la récolte de plusieurs décennies de recherche qui ont exposé les différentes propriétés du problème KP et qui ont rendu sa résolution moins difficile. Nous présentons dans les sections I.5.1 et I.5.3 quelques méthodes de résolution approchée et exacte pour le KP. Pour plus de détails, nous invitons le lecteur à voir Fayard et Plateau [13], Martello et Toth [21], [22] et Pisinger [27] où diverses approches aussi bien exactes qu'approchées sont présentées.

1.5.1 Méthodes approchées

Parmi les méthodes heuristiques utilisées pour résoudre le problème (KP), on peut citer la méthode dite gloutonne. Cette dernière consiste à construire une solution de manière incrémentale, en faisant à chaque pas, le choix qui semble localement le meilleur. Autrement dit, faire un choix localement optimal, dans l'espoir que ce choix mènera à une solution optimale globale.
Comme il existe plusieurs variantes pour ces méthodes gloutonnes, nous présentons ici une des versions. Tout d'abord, les objets sont ordonnés selon l'ordre décroissant du rapport (profit par poids), c'est-à-dire :

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq .... \geq \frac{v_j}{w_j} \geq ... \geq \frac{v_n}{w_n}$$

L'algorithme glouton que nous présentons dans l'algorithme 1 consiste donc à sélectionner à chaque étape un élément selon l'ordre précédemment défini. Si le poids de l'élément séléctionné ne dépasse pas la capacité restante (dite résiduelle) après fixation des autres éléments, alors il est mis dans le sac. Dans le cas contraire, l'élément sélectionné se situe juste après et ainsi de suite jusqu'à épuisement de tous les objets pouvant être mis dans le sac.

Algorithme 1 : Heuristique gloutonne pour le KP

Entrée : Une instance du KP.

Sortie : Une solution réalisable $\overline{x} = (\overline{x_1}, ..., \overline{x_n})$.

1. $\overline{C} := C$;
2. **pour** $i = 1$ jusqu'à n **faire**
   . **Si** $(w_i \leq \overline{C})$ **alors**
   . $\overline{x} \longleftarrow 1$;
   . $\overline{C} \longleftarrow \overline{C} - w_j$
   . **Sinon**
   . $\overline{x} \longleftarrow 0$;
   . **finsi**
   . **finpour**
3. Sortir avec une solution réalisable $\overline{x}$

1.5.2 Calcul de bornes supérieures

Calculer des bornes supérieures ou inférieures permet d'encadrer la valeur de la solution optimale pour les problèmes que l'on tente de résoudre et de réduire ainsi l'espace de recherche des solutions. Ces bornes sont ensuite, utilisées pour le développement de méthodes de résolution exacte s'appuyant sur des procédures d'énumération implicite (ou méthodes de séparation et évaluation).
Dantzig [6] proposa une méthode efficace et élégante pour résoudre la relaxation continue du problème KP et donc determina aussi une borne supérieure pour le KP. Le problème relâché est obtenu en éliminant les contraintes d'intégrité sur les variables x_j en les relâchant dans l'intervalle [0, 1]. Il s'écrit de la façon suivante :

$$R(KP) = \begin{cases} max\ Z(x) = \sum_{j=1}^{n} v_j x_j \\ s.c.\ \sum_{j=1}^{n} w_j x_j \leq C, \\ x_j \in \{0,1\}, j \in \{1,...,n\} \end{cases}$$

En supposant que les objets sont ordonnés selon l'ordre (I.1), la résolution du problème R(KP) consiste alors à remplir le knapsack objet après objet et de proche en

proche jusqu'à sa saturation. Ensuite, le premier objet, noté $x_l(1 < l \leq n)$, ne pouvant être mis en totalité dans le sac est repéré. Ce dernier est appelé l'élément critique. Il vérifie la propriété suivant :

$$\sum_{j=1}^{l-1} w_j \leq C < \sum_{j=1}^{l} w_j$$

Plus formellement, la solution $\overline{x}$ de R(KP) peut se se présenter selon la propriété de Dantzig [9] comme suit :

$$\overline{x} := \begin{cases} 1 & si \quad j = 1,...,l-1 \\ \frac{C-\sum_{j=1}^{l-1} w_j}{w_j} & si\, j = l \\ 0 & si \quad j = l+1,...,n, \end{cases}$$

et la valeur de la solution optimale de R(KP) est donnée par :

$$Z(\overline{x}) = \sum_{j=1}^{l-1} v_j + \overline{C}\frac{v_l}{w_l} \; ou \; \overline{C} = C - \sum_{j=1}^{l-1} w_j$$

Sachant que les valeurs des v_j et x_j sont entières, la partie entière inférieure de $U(\overline{x})$, notée UB_d, définit une borne supérieure pour le KP. Cette dernière, communément appelée borne de Dantzig, s'écrit de la façon suivante :

$$UB_d = \sum_{j=1}^{l-1} v_j + \lfloor \overline{C}\frac{v_l}{w_l} \rfloor$$

Une amélioration de cette borne a été proposée par Martello et Toth [23].

1.5.3 Méthodes exactes

Plusieurs approches de résolution exacte pour le problème du KP ont été élaborées. La plupart de ces méthodes sont basées sur les techniques énumératives s'appuyant, généralement, sur une méthode par séparation et évaluation (Branch , Bound). Il s'agit d'énumérer d'une manière implicite les solutions et d'en choisir la meilleure parmi toutes. Le premier algorithme par séparation et évaluation a été

introduit par Kolesar [19] pour résoudre le KP mais nécessitait un large temps de résolution. Un peu plus tard, d'autres versions de l'algorithme par séparation et évaluation ont vu le jour tels que l'algorithme de Greenberg and Hegerich [15] et l'algorithme de Horowitz and Sahni [17] que nous présentons dans ce qui suit.

►Méthode par séparation et évaluation

La méthode par séparation et évaluation utilise deux concepts : le branchement ou la séparation du problème en sous-problèmes (représentés par des nœuds) et l'évaluation de ceux-ci à l'aide d'une relaxation (continue ou lagrangienne par exemple). L'évaluation d'un nœud consiste à borner les solutions et élaguer les nœuds inutiles, comme illustré dans la figure I.2. Notons que l'efficacité de la méthode par séparation et évaluation dépend essentiellement de la stratégie de développement qui peut être en largeur, en profondeur ou meilleur d'abord. De plus, la méthode du branchement employée et la nature de la fonction d'évaluation utilisée influencent fortement cette méthode. L'algorithme par séparation et évaluation développé par Horowitz et Sahni considère les éléments ordonnés selon l'ordre décroissant du rapport profit par poids. Ensuite, la séparation se fait sur l'élément suivant. Depuis chaque nœud de l'arborescence, et pour un élément j pris dans l'ordre prédéfini, on développe deux branches : la première branche, $x_j = \mathbf{1}$ correspondant à l'élément j mis dans le sac et la deuxième branche $x_j = \mathbf{0}$ cor

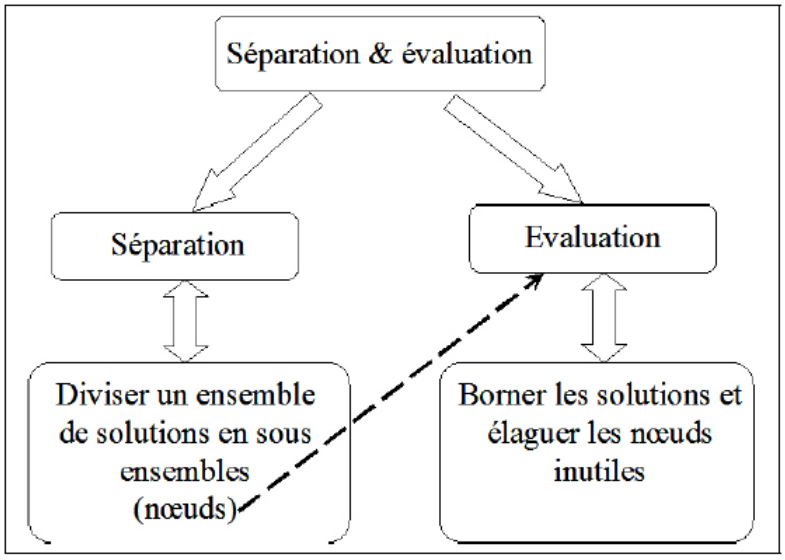


FIGURE 1.2 – Schéma de la méthode de Branch and Bound

respond à l'élément j qui n'est pas mis dans le sac. Le développement de l'arborescence

Algorithme 2 : Algorithme de branch-and-bound en profondeur pour le KP

Entrée : Une instance d'un problème du knapsack.

Sortie : Une solution optimale x^* ;

1. $j := \mathbf{1}$; Meilleure Valeur := 0 ;
2. Calcul de la borne supérieure UBd que l'on peut atteindre depuis j ;
3. **Si** (UBd$\geq$ MeilleureValeur) **alors** ;
. Développer une branche de l'arborescence en profondeur d'abord
. pour obtenir une solution réalisable $\underline{x'}$;
. $j := j + \mathbf{1}$;
. **Si** (Z($\underline{x'}$) $\geq$ MeilleureValeur) **alors**
. $\underline{x} = \underline{x'}$
. MeilleureValeur := Z($\underline{x}$') ;
. **FinSi**
. **FinSi**
4. $j := max\{l : l < j$ et $x'_l = \mathbf{1}\}$
. **Tant que** j existe **faire**
. $\underline{x'_j} := \mathbf{0}$;
. Aller à 2 ; . **Fin Tant que**
5. $\underline{x'} := \underline{x}$

se fait en profondeur, en privilégiant les branches pour lesquelles les éléments sont mis dans le sac. La fonction d'évaluation utilisée est la borne de Dantzig UB_d présentée dans la section I.5.2 Un élément n'est sélectionné que si son poids ne dépasse pas la capacité restante ou encore disponible du sac. Ainsi, à chaque développement d'une branche complète de l'arborescence, la solution obtenue reste réalisable. Le calcul des bornes de Dantzig se fait uniquement au niveau des nœuds développés par une branche correspondant à un élément j non sélectionné, c'est-à-dire correspondant à $x_j = \mathbf{0}$. Pour les nœuds développés par une branche correspondant à $x_j = \mathbf{1}$, la valeur de la borne de Dantzig n'est autre que celle du nœud père. Celle-ci est comparée à la valeur de la meilleure solution obtenue jusque là. Si

la valeur de la borne est inférieure à la valeur de cette meilleure solution, alors il est évident qu'on ne pourra jamais atteindre une meilleure solution à partir de ce nœud. Donc, la troncature au niveau de cette branche courante est possible. L'algorithme 2 représente les principales étapes de cet algorithme. Notons qu'il existe d'autres méthodes de résolution exacte pour les problèmes de type knapsack. En particulier, pour le problème du knapsack classique (KP), on trouve la programmation dynamique (Elkihel [2], Kellerer [18]) ainsi que les méthodes hybrides (voir, par exemple Boyor [5], Plateau [26]).

CHAPITRE 2

PROBLÈMES D'OPTIMISATION MULTI-OBJECTIF

Avant de parler d'optimisation multi-objectifs, il faut définir d'abord le domaine d'optimisation. Les problèmes d'optimisation cherchent le meilleur rendement possible. Selon la nature de l'espace de recherche, des contraintes et des objectifs, on définit plusieurs types d'optimisation.

2.1 Problème d'optimisation

Un problème dans le point de vue informatique (Computational Problem) veut dire comment faire relier un ensemble de données par un ensemble de résultats, où les données vont subir un ensemble d'opérations dont on les appelle le traitement. Donc il est conçu comme une relation $\mathbf{\Pi} \subseteq \boldsymbol{\ell x S}$ entre les entrées ou instances, et les sorties ou solutions. Pour qu'une instance d'un problème soit compréhensible par un ordinateur numérique, elle doit être décrite comme une séquence finie de symboles d'un ensemble fini arbitraire appelé alphabet. De même, la solution d'une instance d'un tel problème est émise dans ce format. Généralement, l est infini alors

que S peut être fini. [28]

L'optimisation est un nom féminin qui définit une action du verbe optimiser. C'est la démarche consistant à rendre optimal le fonctionnement d'un système : Donner à quelque chose (à une machine, à une entreprise, etc.) le rendement optimal en créant les conditions les plus favorables ou en tirant le meilleur parti possible.
L'optimisation appartient au domaine de la Recherche Opérationnelle qui représente la science de prise de décision. Un problème d'optimisation en mathématique consiste à trouver la meilleure solution possible pour un problème donné.

Définition 2.1. *Problème d'optimisation*
Soit $\boldsymbol{n} \in \mathbb{N}^*$ *un entier strictement positif,*
$\boldsymbol{X} \subset \mathbb{R}^{\boldsymbol{n}}$ *un sous-ensemble non vide*
$\boldsymbol{f : X \to R}$ *une fonction à valeur réelle.*
Un problème d'optimisation consiste à trouver la meilleure solution $\overline{\boldsymbol{x}} \in \boldsymbol{X}$ *appelée solution globale.*

La meilleure solution du problème varie selon l'objectif. On parle d'une minimisation ou une maximisation de la fonction f sur l'ensemble X. On note : $\overline{x} = \boldsymbol{min}_{x \in X} f(x)$ ou $\overline{x} = \boldsymbol{max}_{x \in X} f(x)$.

2.1.1 Un problème combinatoire

Un problème combinatoire est toute situation dont on cherche d'avoir une solution tout en respectant la présence d'un ensemble de contraintes. La solution c'est un résultat de faire combiner ces contraintes ensemble d'une manière qu'on maximise quelques uns et on minimise les autres, ces contraintes ont une caractéristique primordiale, c'est que chaque contrainte influe sur les autres soit quand on minimise sa valeur ou on la maximise, dans un autre terme on dit que les contraintes sont conflictuelles.
Par exemple, le schéma suivant présente une situation de problème combinatoire : où on veut acheter une voiture dans la mode et en même temps avec un prix raisonnable qui ne peut pas dépasser certaine limite. Si on maximise la première contrainte

(une bonne voiture) on va avoir un prix maximale, dans le contraire on va aboutir à une mauvaise voiture mais avec un prix minimale dans les limites ; on constate dans cet exemple que c'est difficile d'arranger ces deux contraintes dans nos besoins.

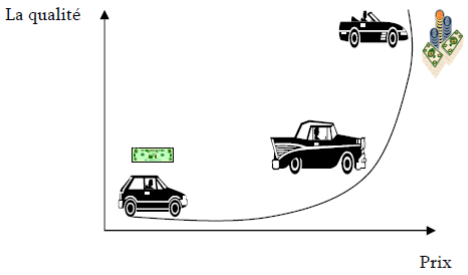


Figure 1 : une figure illustrant un problème combinatoire.

2.1.2 Classification des problèmes d'optimisation

Les problèmes d'optimisation que l'on rencontre au monde réel sont classifiés selon leurs caractéristiques. Nous distinguons ainsi les problèmes selon :

- Le nombre de variables de décision :
  - Mono-variable.
  - Multi-variables.
- Le type de variables de décision :
  - Variables continues : où les valeurs sont des nombres réels.
  - Variables discrètes : où les valeurs sont nombres entiers.
  - Variables combinatoires : où les valeurs sont des permutations sur un ensemble fini de nombre.
- Le type de la fonction-objectif :
  - Fonction linéaire.
  - Fonction non linéaire.
  - Fonction quadratique.
- La formulation du problème :
  - Problème sans contraintes.
  - Problème avec contraintes.

2.1.3 Optimisation mono ou multi-objectifs

Un problème d'optimisation peut avoir un ou plusieurs objectifs. Un problème monoobjectif est défini par une unique fonction objectif. S'il se trouve plusieurs objectifs qui génèrent une contradiction entre eux, on parle alors d'un problème multi-objectifs. Pour des raisons de simplification, un problème multi-objectif peut être reformulé avec une seule fonction objectif ou en transformant des objectifs sous forme de contraintes.

2.2 Formulation générale d'un problème d'optimisation multi-objectif

Un problème d'optimisation multi-objectifs (multicritéres) (MOP) consiste à optimiser (minimiser) simultanément plusieurs fonctions objectifs $f_1, f_2, ..., f_p,\ p \geq 2$ sur un domaine $\mathbb{R}^n$.
Mathématiquement, ce problème peut s'écrire comme suit :

$$(MOP) = \begin{cases} "min"(ou)"max"(f_1(x), f_2(x), ..., f_p(x))^T,\ p \geq 2 \\ g_j(x) \leq 0, j = 1, 2, ..., m \end{cases}$$

Tel que :

- $x = (x_1, x_2, ..., x_n)^T$ est le vecteur de n variables de décision.
- On pose $F(x) = (f_1(x), f_2(x), ..., f_p(x))^T$ est le vecteur de $p \geq 2$ fonctions objectifs à optimiser.
  Où

$$f_i : \mathbb{R}^n \to R, i = 1, 2, ..., p$$
$$x \to f_i(x)$$

- Notons par $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0\}$ l'ensemble de décision .

Considérons l'application suivante :

$$F : S \rightarrow \mathbb{R}^p$$
$$x \rightarrow F(x)$$

L'image d'une solution x dans l'espace des critères est le point $y = (y_1, ..., y_m)$ avec $y_i = f_i(x), i = 1..m$, et Y = F(S) représente les points réalisables dans l'espace des objectifs. On impose sur cet ensemble une relation d'ordre partiel appelée relation de dominance que nous allons définir dans la section suivante.

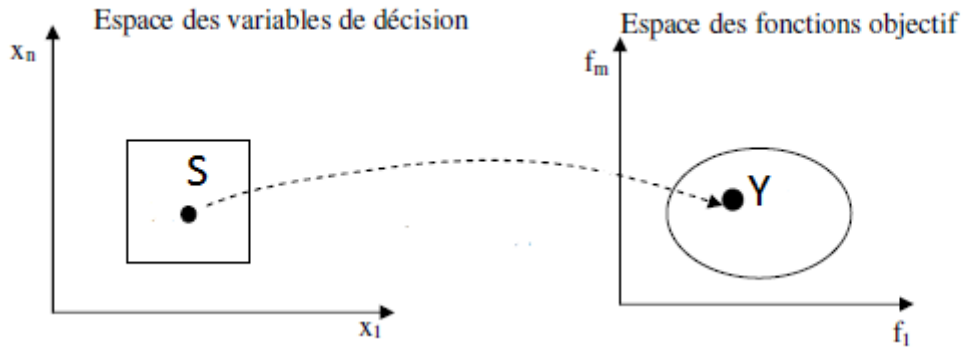


FIGURE 2.1 – Représentation d'un problème multi-objectif

La recherche de la solution optimale pour un problème d'optimisation multi-objectif soulève quelques réflexions par rapport à la notion même de l'optimalité. En effet, il est impossible de trouver une solution optimale unique pour un problème multi-objectif, car il n'y a aucune combinaison des variables de décisions qui minimise (ou maximise) toutes les composantes du vecteur F simultanément. Les problèmes multi-objectif ont en général un ensemble de solutions optimales dont les valeurs des fonctions sont en fait les meilleurs compromis possibles dans l'espace des fonctions objectif. Il faut donc utiliser une autre définition de la "meilleure solution", afin de déterminer exactement quelle solution peut être considérée meilleure par rapport à une autre. Le concept de "l'optimalité de Pareto" (Pareto, 1896) est ainsi utilisé pour établir une hiérarchie entre les solutions d'un problème multi-objectif en

vue de déterminer si une solution appartient réellement à l'ensemble des meilleurs compromis.

2.3 Notions de dominance

La présence de plusieurs objectifs dans le problème d'optimisation multi-objectif modifie la terminologie employée, on ne parle plus de « d'optimum », mais de « Notion de domination »

Définition 2.2. ***(La relation de dominance)***
On dit que le vecteur $\boldsymbol{y = (y_1, y_2, ..., y_k)}$ *domine le vecteur* $\boldsymbol{z = (z_1, z_2, ..., z_k)}$ *si*

- *y est au moins aussi bon que z sur tous les objectifs et,*
- *y est strictement meilleur que z sur au moins un objectif.*

D'une manière équivalente nous avons

Définition 2.3. *La solution y domine une autre solution z, si les conditions suivantes sont vérifiées*

$$f_l(y) \leq f_l(z) \; \forall \; l \in \{1, ..., k\} \; et \; \exists \; l \; \in \{1, ..., k\} \; tq \; f_l(y) < f_l(z)$$

Si la solution y domine la solution z, nous allons écrire $\boldsymbol{y \prec z}$.

Notons que pour toute paire de solutions y et z, une et seulement une des affirmations suivantes est vraie

- y domine z.
- y est dominé par z.
- y et z sont équivalentes au sens de la dominance (appelées aussi solutions Pareto-équivalentes)

Lorsque l'on applique la définition de la dominance, on peut avoir quatre régions auxquelles on peut attribuer des niveaux de préférence.

Définition 2.4. *Soient* $\boldsymbol{y, z} \in \mathbb{R}^p$ *on dit que le vecteur y domine le vecteur z si*

— *Dominance faible* $\exists i$ *tel que*

$$i = 1,2,...,p : y(i) \leq z(i)$$

— *Dominance forte* $\exists i$

$$i = 1,2,...,p : y(i) < z(i)$$

2.3.1 Propriétés de la relation de dominance

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

— n'est pas réflexive, car une solution ne se domine pas elle même.

— n'est pas symétrique, car on n'a jamais $y \prec z$ et $z \prec y$

— est transitive, car si $y \prec z$ et $z \prec w$ alors $y \prec w$

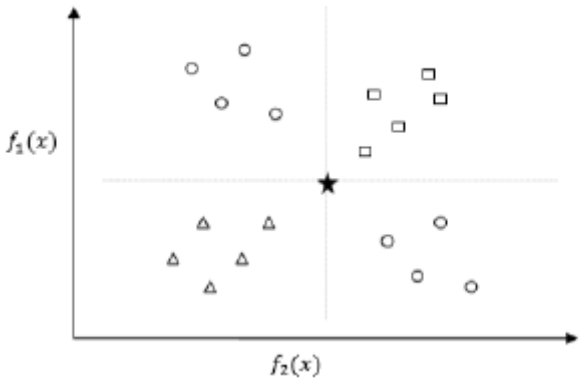


FIGURE 2.2 – Relation de dominance

Un exemple illustratif du principe de domination est donné par la figure suivante :
En observant la figure ci-dessus, on peut dire que l'étoile :

— domine chaque carré.

— dominé par chaque triangle.

— équivalente à chaque anneau au sens de la dominance.

2.3.2 Optimalité de Pareto

Soit P un ensemble de solutions candidates d'un problème d'optimisation multiobjectif. L'ensemble $\boldsymbol{P' \subseteq P}$, composé de tous les éléments de P qui ne sont dominés par aucun élément de P est dit sous-ensemble non dominé de l'ensemble P.

— Optimalité locale au sens de Pareto : Une solution y est optimale localement au sens de pareto s'il existe un réel $\delta > \mathbf{0}$ telqu'il n'y ait pas une solution z dominant y et vérifiant $\|z - y\| \leq \delta$

— Optimalité globale au sens de Pareto : Une solution y est optimale globalement au sens de pareto, ou optimale au sens de Pareto, ou encore Pareto-optimale s'il n'existe aucun point de l'espace faisable D qui la domine. L'ensemble des solutions Pareto optimale est appelé l'ensemble de Pareto ou également l'ensemble des compromis optimaux. L'image de l'ensemble de Pareto dans l'espace des critères est appelé la surface de Pareto (ou le front de Pareto pour le cas bi-objectif) ou encore la surface des compromis optimaux.

Solutions supportées / non supportées

Sur le front Pareto, deux types de solutions peuvent être différenciées :
les solutions supportées et les solutions non supportées. Les premières sont celles situées sur l'enveloppe convexe de l'ensemble des solutions (voir la figure 2.3) et peuvent donc être trouvées à l'aide d'une agrégation linéaire des objectifs [32]. Elles sont donc plus simples à obtenir que les solutions non supportées qui elles, sont plus difficile à trouver.

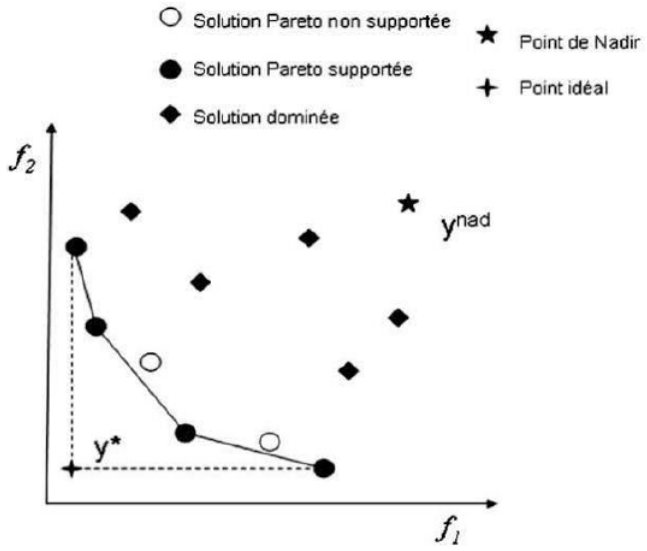


FIGURE 2.3 – Représentation des solutions supportées et non supportées, point idéal et point de Nadir

2.3.3 Efficacité

Définition 2.5.

Une solution $x^* \in S$ est une solution efficace ou pareto optimal s'il n'existe pas de $x \in S$ tel que $F(x)$ domine $F(x^*)$.

Définition 2.6. ***(Efficacité faible)***

Une solution $x^* \in S$ est dite faiblement efficace s'il n'éxiste 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.

Définition 2.7. ***(Efficacité forte)***

Une solution $x^* \in S$ est dite fortement efficace s'il n'éxiste 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'éxiste pas une autre solution telle que le vecteur critère, qui lui est associé soit aussi bon que celui x^*.

2.4 Front de Pareto et La surface de compromis

Définition 2.8. ***(Front de Pareto)***

le front de Pareto est l'ensemble généralement non convexe constitué par l'image des solutions efficaces dans l'espace des critères. En effet, le front de Pareto est l'ensemble $Y_{ND} = F(X_E)$.

La figure suivant montre le front de Pareto pour les différentes situations envisageables pour un problème deux critères. Notons que, le front de Pareto de chaque cas et la partie en gras de l'ensemble.

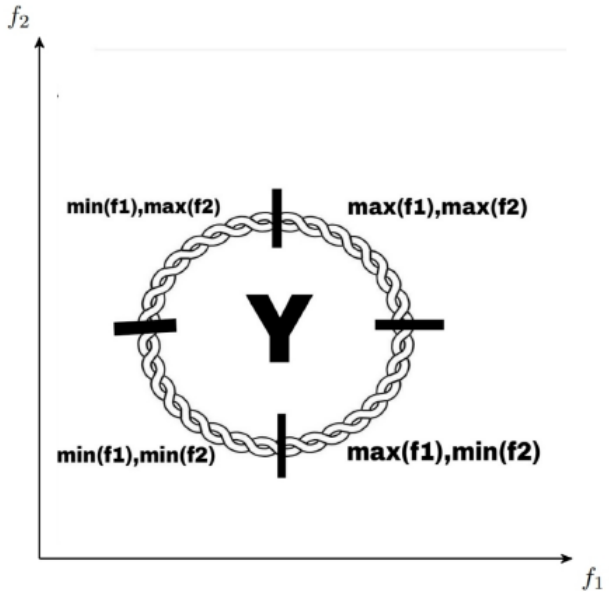


FIGURE 2.4 – Le front de Pareto

2.4.1 Les points caractéristiques

Nous présentons quelques points caractéristiques très utilisent dans l'optimisation multiobjectif notamment le point idéal, le point nadir, la matrice des gains.

Point idéal

Le vecteur idéal $y^* = (y_1^*,..,y_m^*)$ est obtenu en optimisant séparément chaque fonction objectif $f_i, i.e. y_i^* = f_i^*(x), x \in S$. Généralement ce vecteur n'appartient pas

à l'espace objectif réalisable mais il est dans certains cas utile en tant que référence, par exemple, pour normaliser les valeurs des objectifs.

Point de Nadir

A la différence du vecteur idéal qui représente les bornes inférieures de chaque objectif dans l'espace faisable, le vecteur de Nadir correspond à leurs bornes supérieures sur la surface de Pareto, et non pas dans tout l'espace faisable . Ce vecteur sert à restreindre l'espace de recherche ; il est utilisé dans certaines méthodes d'optimisation interactives.

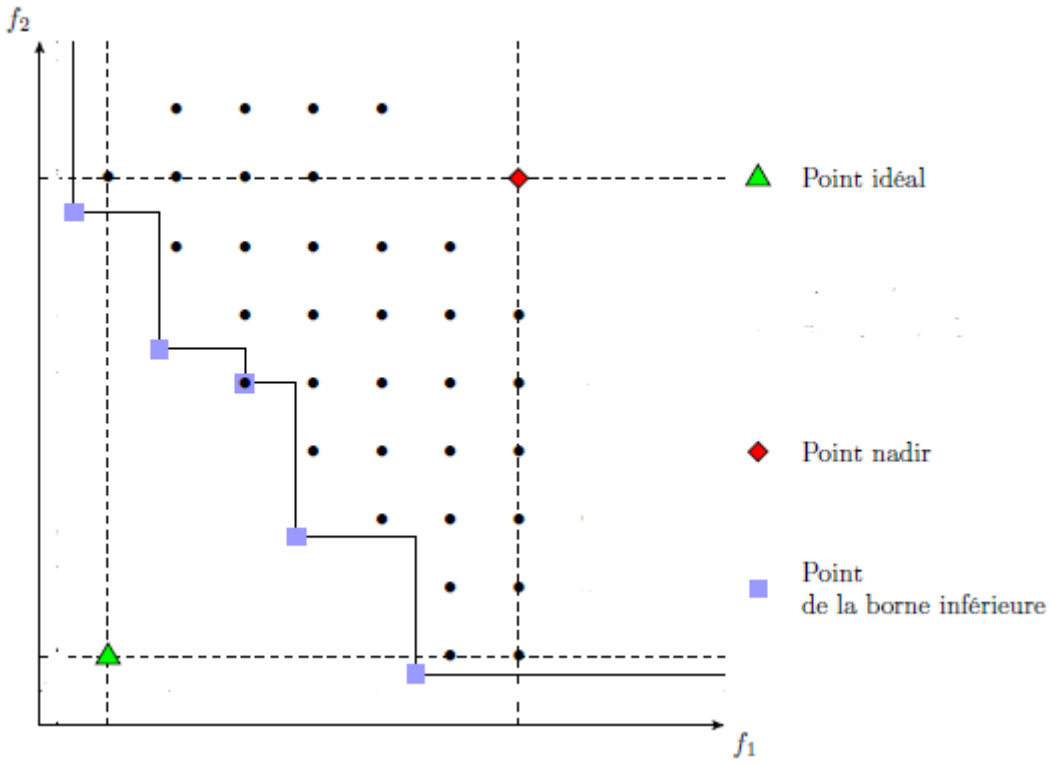


FIGURE 2.5 – Point idéal et relaxation, point nadir, borne inférieure

Matrice des gains

(Matrice des gains) soit $\widehat{x}^j$ une solution optimale du critère f_j . La matrice des gains $p \times p$ est formée des éléments $z_j^i = f_i(\widehat{x}^j)$ tels que

$$G = \begin{bmatrix} z_1^* & \cdots & z_{1j} & \cdots & z_{1j} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{i1} & \cdots & z_i^* & \cdots & z_{ik} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{p1} & \cdots & z_{kj} & \cdots & z_p^* \end{bmatrix}$$

Les coordonnées du point idéal apparaissent sur la diagonale principale de cette matrice et les coordonnées du point Nadir apparaissent sur la diagonale opposée. Lorsqu'un critère j possédé plusieurs solutions optimales, la colonne j de la matrice des gains dépendra de la solution $\widehat{x}^j$ choisie (la matrice des gains est univoquement déterminée si, pour tous les critères j, la solution $\widehat{x}^j$ est unique).

2.4.2 Fonctions scalarisantes

Le choix d'une solution efficace parmi toutes les solutions réalisables exige une certaine connaissance de la structure de préférence. Cette information est obtenue directement ou indirectement et peut parfois se traduire en terme de paramètre de préférence. Les plus courantes sont :

1) Les poids $\lambda_k, k = 1, 2, ..., p$ qui reflètent l'importance relative de chaque critère.
2) Le point de référence qui est défini par des niveaux de réservation (valeurs souhaitables) sur chaque critère .
3) Le point de réservation qui est défini par des niveaux (valeurs non souhaitables) sur chaque critère.

La fonction scalarisante est défine par $s(z, \lambda) : \mathbb{R} \times \mathbf{\Lambda} \rightarrow \mathbb{R}$ où $\mathbf{\Lambda}$ est l'ensemble des paramètres. Les fonctions scalarisantes les plus courantes sont les suivantes :

Fonctions linéaires :

$$s_1(z,\lambda) = \sum_{k=1}^{p} \lambda_k z_k \; avec \sum_{k=1}^{p} \lambda_k = 1, \lambda_k > 0, \forall k \; et \; s_2(z,\lambda) = \sum_{k=1}^{p} \lambda_k |z_k - \bar{z}_k|$$

Norme L_p pondérée :

$$s_3(z,\lambda) = \left[\sum_{k=1}^{p} \lambda_k |z_k - \bar{z}_k|^q \right]^{\frac{1}{q}}$$

Norme L_∞ de Tchebycheff pondérée :

$$s_4(z,\lambda) = max_{1 \leq k \leq p} \{\lambda_k |z_k - \bar{z}_k|\}$$

Norme Tchebycheff pondérée augmentée :

$$s_5(z,\lambda) = max_{1 \leq k \leq p} \{\lambda_k |z_k - \bar{z}_k|\} + \rho \sum_{k=1}^{p} \lambda_k |z_k - \bar{z}_k|; \rho > 0.$$

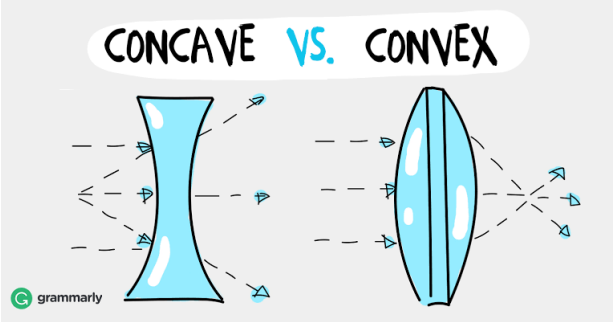
Ici $\bar{z}_k$ désigne une valeur de référence de z_k.

Les fonctions scalarisantes ne peuvent engendrer que des vecteurs non dominés. Dans ce cas, elles caractérisent complètement l'ensemble des solutions efficaces.

2.5 Convexité

Comme nous le verrons plus tard, certaines méthodes d'optimisation multiobjectif nécessitent de travailler sur un espace Y des valeurs de F qui soit convexe

Définition 2.9. *Un ensemble Y est convexe si, pour n'importe quels deux points distincts de cet ensemble, le segment qui relie ces deux points est contenu dans l'ensemble S.*



2.6 Les Méthodes de résolution des problèmes multi-objectives

La résolution d'un problème multi-objectif menant à la détermination d'un ensemble de solutions Pareto, il est nécessaire de faire intervenir l'humain à travers un décideur, pour le choix final de la solution à garder. Ainsi, avant de se lancer dans la résolution d'un problème multi-objectif, il faut se poser la question du type de méthode d'optimisation à utiliser. En effet, on peut répartir les méthodes de résolution de problèmes multi-objectif en trois familles, en fonction du moment où intervient le décideur.

Ainsi nous pouvons trouver les familles suivantes :

— ***Les méthodes d'optimisation a priori :*** dans ce cas, le compromis que l'on désire faire entre les objectifs à été défini avant l'exécution de la méthode. Ainsi une seule exécution permettra d'obtenir la solution recherchée. Cette approche est donc rapide, mais il faut cependant prendre en compte le temps de modélisation du compromis et la possibilité pour le décideur de ne pas être satisfait de la solution trouvée et de relancer la recherche avec un autre compromis.

— ***Les méthodes d'optimisation progressives :*** ici, le décideur intervient dans le processus de recherche de solutions en répondant à différentes questions afin d'orienter la recherche. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche.

— ***Les méthodes d'optimisation a posteriori :*** dans cette troisième famille de méthodes, on cherche à fournir au décideur un ensemble de bonne solutions bien réparties. Il peut ensuite, au regard de l'en- semble des solutions, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contre partie fournir un ensemble de solutions bien réparties, ce qui peut également être difficile et requérir un temps de calcul important (mais ne nécessite pas la présence du décideur).

2.6.1 Les méthodes exactes

Les méthodes exactes cherchent à trouver de manière certaine la solution optimale en examinant de manière explicite ou implicite la totalité de l'espace de recherche. Elles ont l'avantage de garantir la solution optimale néanmoins le temps de calcul nécessaire pour atteindre cette solution peut devenir très excessif en fonction de la taille du problème (explosion combinatoire) et le nombre d'objectifs à optimiser. Ce qui limite l'utilisation de ce type de méthode aux problèmes bi-objectifs de petites tailles. Les références existantes et qui présentent la plupart des méthodes exactes sont Ulungu [30] et Ehrgott [11].
Ces méthodes sont basées principalement sur les procédures de Branch and Bound, Cut ou Price et la programmation dynamique. Une approche particulière pour l'optimisation multi-objectif est la programmation par but.

► ***La recherche dichotomique :***

La recherche dichotomique offre un schéma d'application de l'agrégation linéaire permettant d'obtenir les solutions non supportées. Cette méthode consiste à explorer de façon dichotomique des intervalles de front de plus en plus petits. Tout d'abord les solutions extrêmes sont recherchées. Puis une recherche est menée entre ces solutions r et s suivant une direction perpendiculaire à la droite (r,s). En interdisant de ré-obtenir les solutions r et s et en éliminant les solutions dominées par ces solutions, cette recherche trouve la meilleure solution Pareto relativement à cette

direction de recherche, solution qui peut alors être non supportée. Cette nouvelle solution crée deux nouveaux intervalles qu'il faut explorer de la même façon. Cette méthode, dédiée au bi-objectif, est intéressante mais nécessite de l'ordre de $\mathbf{2^n}$ recherches, si n est le nombre de solutions du front Pareto.

► ***Méthode en deux-phases :***

La méthode deux-phases a initialement été proposée par Ulungu et Teghem pour la résolution d'un problème d'affectation bi-objectif. Comme son nom l'indique, cette méthode est décomposé en deux étapes : la première consiste à trouver toutes les solutions supportées du front Pareto, puis la deuxième phase cherche entre ces solutions les solutions Pareto non supportées. Cette méthode travaille donc essentiellement dans l'espace objectif.

1). Première phase

L'objectif de la première phase est d'obtenir l'ensemble des solutions Pareto supportées. Comme nous l'avons vu précédemment, ces solutions ont l'avantage d'être relativement faciles à trouver puisqu'elles optimisent une certaine combinaison linéaire des objectifs. Ainsi, durant la première phase de la méthode, les deux solutions extrêmes (solutions optimisant chacune un des deux objectifs) sont recherchées (voir figure 2.6.a). Puis, de façon récursive, dès que deux solutions supportées r et s sont trouvées, la méthode recherche d'éventuelles autres solutions supportées entre r et s, à l'aide de combinaisons linéaires bien choisies des objectifs (voir figure 2.6.b et 2.6.c). A la fin de la première phase l'ensemble des solutions supportées est donc trouvé (voir figure 2.6.d). Cette première phase rappelle la méthode dichotomique, mais ici seules les solutions supportées sont recherchées. Pour cela, lors de l'exploration entre deux solutions, on s'autorise à retrouver l'une de ces deux solutions, lorsqu'il n'existe pas d'autre solutions supportées dans l'intervalle.

2). Deuxième phase

La deuxième en phase consiste alors en la recherche des solutions non supportées appartenant au front Pareto. Ces solutions ne peuvent être obtenues par combinai-

sons d'objectifs. Alors on utilise les solutions supportées trouvées pour réduire l'espace de recherche puisque les solutions Pareto non supportées restantes sont forcément dans les triangles rectangles basés sur deux solutions supportées consécutives (voir figure 2.6.e). Ainsi, une recherche de type deuxième phase est exécutée entre chaque couple de solutions supportées adjacentes (voir figure 2.6.f et 2.6.g). La méthode de recherche au sein de ces triangles dépend du problème étudie. A la fin de la deuxième phase, toutes les solutions Pareto sont trouvées.

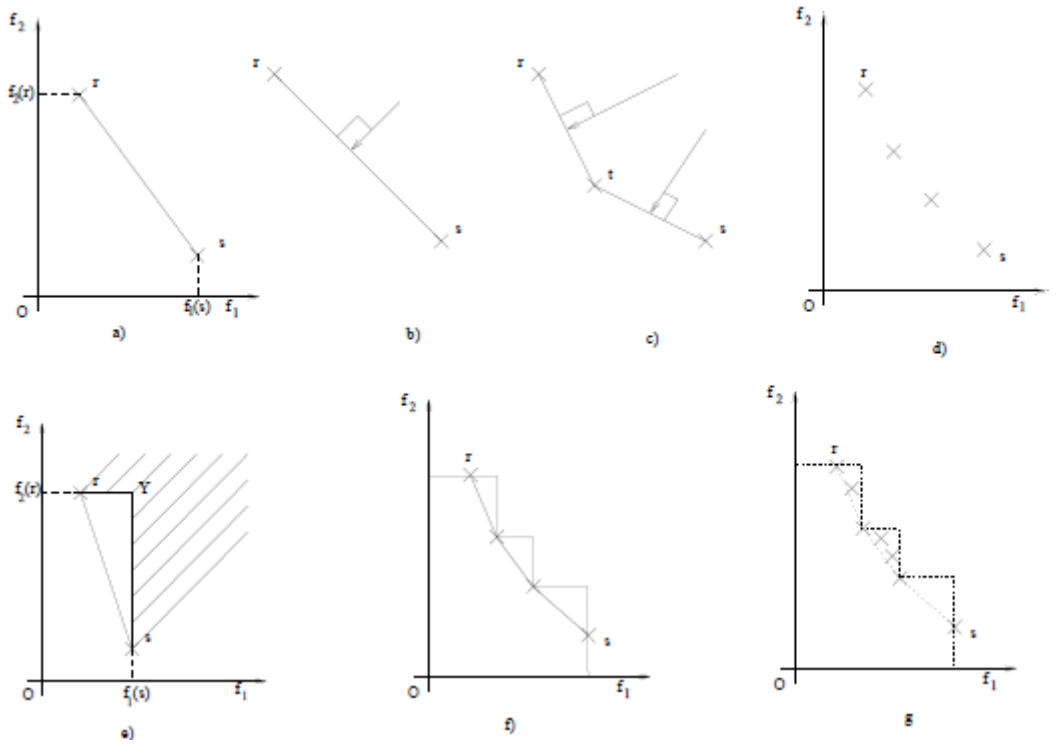


FIGURE 2.6 – Illustration des différentes étapes de la méthode deux phases.

► ***Méthode des sommes pondérées :***

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

pour obtenir une nouvelle et unique fonction objectif. Un problème multi-objectif se transforme alors de la manière suivante :

$$min \sum_{i=1}^{m} \lambda_i f_i(x) \; m \geqslant 2$$

où les poids $\lambda_i \geqslant 0$ sont tels que : $\sum_{i=1}^{m} \lambda_i = 1$

Les résultats obtenus avec de telles méthodes dépendent fortement des paramètres choisis pour le vecteur de poids. Les poids λ_i doivent également être choisis en fonction des préférences associées aux objectifs, ce qui est une tache délicate. Une approche généralement utilisée consiste à répéter l'exécution de l'algorithme avec des vecteurs poids différents. Cette méthode est très efficace du point de vue algorithmique, mais son principal inconvénient est qu'elle ne permet pas de trouver les solutions enfermées dans des concavités (surface de compromis non-convexe).

► ***La méthode de programmation par but, (Goal programming) :***

Cette méthode est également appelée target vector optimisation, le décideur fixe un but G_i à atteindre pour chaque fonction objectif f_i. Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est modifie de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre :

$$min \sum_{i=1}^{k} |f_i(x) - G_i|$$

G_i represente le but a atteindre pour le i_eme objectif. Cette méthode est facile à mettre en oeuvre mais la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode. dans des concavités (surface de compromis non-convexe).

► ***Méthode ϵ contrainte :***

Cette méthode permet de transformer le problème d'optimisation multi- objectif en un problème mono-objectif. La méthode consiste à convertir $m-1$ des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant [8].
La démarche est la suivante :

— Nous choisissons un critère à optimiser prioritairement.

— Nous transformons le problème conservant l'objectif prioritaire et nous transformons les autres objectifs en des contraintes d'inégalité.

Le problème peut être reformulé de la manière suivante :

$$\begin{cases} Min\ f_i(x) \\ tq: \\ f_1(x) \leq \epsilon_1 \\ . \\ . \\ . \\ f_{i-1}(x) \leq \epsilon_{i-1} \\ f_{i+1}(x) \leq \epsilon_{i+1} \\ . \\ . \\ f_m(x) \leq \epsilon_m \\ et\ que\ g(x) \leq 0 \\ avec\ x \in \mathbb{R}^n, f(x) \in \mathbb{R}^m, g(x) \in \mathbb{R}^q \end{cases}$$

L'approche par ϵ-contrainte doit aussi être appliquée plusieurs fois en faisant varier le vecteur ϵ pour trouver un ensemble de points Pareto optimaux. Cette approche a l'avantage de ne pas être trompée par les problèmes non convexes. Ainsi la figure 2.7 illustre, en dimension 2, le cas où un point $(\epsilon; f_{1min})$, de la partie non convexe, est trouvé. La figure 2.7 montre aussi comment cette approche procède. En transformant des fonctions objectifs en contraintes, elle diminue la zone réalisable par

paliers. Ensuite, le processus d'optimisation trouve le point optimal sur l'objectif restant. L'inconvénient de cette approche réside dans le fait qu'il faille lancer un grand nombre de fois le processus de résolution. De plus, pour obtenir des points intéressants et bien répartis sur la surface de compromis, le vecteur ϵ doit être choisi judicieusement.

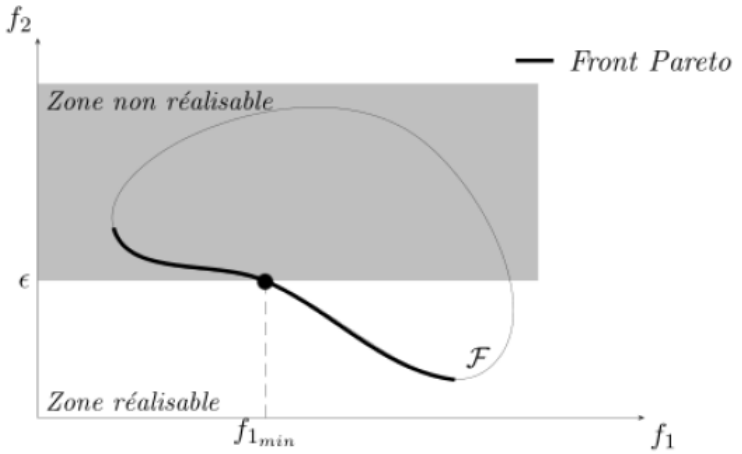


FIGURE 2.7 – Interprétation graphique de la méthode ϵ contrainte.

2.6.2 Les méthodes approchées

Méthodes souvent inspirées de mécanismes d'optimisation rencontrés dans la nature. Elles sont utilisées pour les problèmes où on ne connaît pas d'algorithmes de résolution en temps polynomial et pour lesquels on espère trouver une solution approchée de l'optimum global. Elles cherchent à produire une solution de meilleure qualité possible dictée par des heuristiques avec un temps de calcul raisonnable en examinant seulement une partie de l'espace de recherche. Dans ce cas l'optimalité de la solution n'est pas garanti ni l'écart avec la valeur optimal. Parmi ces heuristiques, on trouve les métaheuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes.

Plusieurs classifications des métaheuristiques ont été proposées, la plupart distinguent globalement deux catégories : celles se basant sur une solution unique et celles se basant sur une population de solution [20].

► *Métaheuristiques :*

Les métaheuristiques les plus utilisées

Reconnues depuis de nombreuses années pour leur efficacité, les métaheuristiques sont une famille de méthodes stochastiques qui consistent à la résolution des problèmes d'optimisation. Elles exploitent généralement des processus aléatoires dans l'exploration de l'espace de recherche pour faire face à l'explosion combinatoire engendrée par l'utilisation de méthodes exactes.

Leur particularité réside dans le fait qu'elles sont adaptables à un grand nombre de problèmes sans changements majeurs dans leurs algorithmes, d'où le qualificatif "méta". L'un des avantages de celles-ci est leur capacité à optimiser un problème à partir d'une quantité minimale d'information, cependant elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Les métaheuristiques sont des méthodes qui ont un comportement itératif, c'est-à-dire que le même schéma est reproduit un certain nombre de fois au cours de l'optimisation, et elles sont directes, dans le sens où elles ne font pas appel au calcul du gradient de la fonction. L'utilisateur est certes, demandeur de méthodes rapides et efficaces, mais il est aussi demandeur de méthodes simples d'utilisation. Un enjeu majeur des métaheuristiques est donc de faciliter le choix des méthodes et de simplifier leurs réglages, afin de les adapter au mieux aux problèmes posés. Les métaheuristiques sont en évolution permanente. De nombreuses méthodes sont proposées chaque année pour améliorer la résolution des problèmes les plus complexes. Du fait de cette activité permanente, un grand nombre de classes de métaheuristiques existe actuellement. Les méthodes les plus courantes sont le recuit simulé, la recherche tabou, les algorithmes de colonies de fourmis. La section suivante est dédiée à la présentation de ces méthodes [16].

1). Méthode de colonies de fourmis

Le principe de la méthode provient analogiquement avec les comportements collectifs des insectes, les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques,

appelées phéromones, sur le sol et le suivi de pistes observés dans les colonies de fourmis et construisent ainsi une solution à un problème en tenant compte de leur expérience collective. Au fait, elles adoptent pour la recherche de la solution la notion du plus court chemin.

D'une manière simplifiée, les fourmis commencent par se déplacer au hasard. Puis, lorsqu'elles trouvent de la nourriture, elles retournent vers leur colonie, en marquant leur chemin à l'aide de phéromone. Si d'autres fourmis rencontrent ce chemin, il y a de fortes chances qu'elles arrêtent leurs déplacements aléatoires et qu'elles rejoignent le chemin marqué, en renforçant le marquage à leur retour, s'il mène bien vers la nourriture. Par conséquent, le chemin le plus court sera davantage parcouru, et donc plus renforcé et plus attractif. Par conséquent, le nombre de fourmis suivant cette trajectoire augmente. Au fil du temps, la quantité de phéromones déposée sur le plus long chemin diminue et finit par disparaître. Toutes les fourmis suivent alors le chemin le plus court.

L'algorithme de colonies de fourmis a été principalement utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. Récemment, son emploi s'est généralisé à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification, ou encore le traitement d'image [16].

Algorithme 3 : Algorithme de colonie de fourmis [16]

1. Initialisation

Initialiser les pistes de phéromone

2. Construction de la solution

Pour chaque fourmi répéter

Construction de la solution en utilisant les pistes de phéromone

3. Mise à jour des pistes de phéromone

Jusqu'à atteindre la condition d'arrêt

2). Méthode de recuit simulé

L'origine de la méthode du recuit simulé provient de la métallurgie, où, pour atteindre les états de basse énergie d'un solide on chauffe celui-ci jusqu'à des températures très élevées, après on le laisse refroidir lentement [16].

Lorsque le solide est à une température, chaque particule possède une très grande énergie et peut effectuer de grands déplacements aléatoires dans la matière. Au fur à mesure que la température est abaissée, chaque particule perd de l'énergie et sa capacité de déplacement se réduit. Les différents états transitoires de refroidissement permettent d'obtenir des matériaux très homogènes et de bonne qualité. Ce processus est appelé le recuit. Cette méthode repose sur l'algorithme de Metropolis en 1953 [24]. Cet algorithme nous permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée, et de conserver les états les plus probables pour de très basses températures.

Algorithme 4 : Recuit Simulé [7, 25]

1. Générer aléatoirement une configuration initiale $s = s_0$ dont correspond l'énergie initiale $E = E_0$,
2. Initialiser la température $T = T_0$ en fonction du schéma de refroidissement,
3. Générer d'une manière aléatoire une configuration voisine s_0 de la configuration actuelle s, en déplaçant au hasard un atome quelconque,
4. Calculer la variation de l'énergie $\Delta E = f(s') - f(s)$

   Si $\Delta E < 0$, alors la nouvelle solution améliore la fonction objectif et diminue l'énergie, donc elle est acceptée.

   Sinon, elle sera acceptée avec une probabilité égale à $e^{\Delta E/T}$
5. Répéter 3. et 4. jusqu'à ce que la configuration optimale soit atteinte.
6. Décroitre la température et répéter jusqu'à ce que le système se solidifie.

3). Recherche Tabou

La recherche tabou est une méthode de recherche locale introduite dans les années 80 par Fred Glover. Elle est basée sur des mécanismes inspirés de la mémoire humaine. Elle se distingue des méthodes de recherche locale simples par l'utilisation d'une mémoire appelée liste tabou de taille k, qui enregistre les k dernières solutions visitées vers lesquelles il est interdit de se déplacer (d'où le nom attribué à la méthode par Glover). Cette liste est utilisée lors de déplacements dans le voisinage dans le but de favoriser une large exploration de l'espace des solutions et d'éviter d'y retourner trop rapidement à des solutions déjà visitées. En effet, à partir d'une

configuration courante s, on choisit une solution $s' \in N(s)$ en dehors des éléments de cette liste tabou T, même si elle dégrade la fonction objectif f, puis on ajoute s' à T. Quand le nombre k est atteint, chaque nouvelle solution sélectionnée remplace la plus ancienne dans la liste [1, 6, 10].

Algorithme 5 : Recherche Tabou [10]

1. Générer une solution s,
2. Choisir une nouvelle solution s' qui minimise $f(s')$ dans le voisinage de s et qui ne figure pas dans la liste tabou T,
3. Si $f(s') \leq f(s)$ alors :

$s \longleftarrow s'$,

4. Poser $s = s'$ et mettre à jour T,
5. Répéter 2. 3. et 4. jusqu'à ce que le critère d'arrêt soit atteint.

Le critère d'arrêt peut être le nombre maximal d'itération sans amélioration de la solution s ou le temps limite de fonctionnement de l'algorithme qui sont fixés à l'avance.

4). Algorithmes Génétiques

Les algorithmes génétiques ont été largement utilisés dans la communauté multi-objectif. Ils sont très appropriés pour résoudre des (PMO) grâce à l'utilisation d'une population de solutions. Ils peuvent chercher plusieurs solutions Pareto-optimales dans la même exécution. Les algorithmes génétiques (AG_s) ont été introduits par Holland comme un modèle de méthode adaptative. Ils s'appuient sur un codage de l'information sous forme de chaines binaires de longueur fixe et d'un ensemble d'opérateurs génétiques : la sélection, la mutation, le croisement. Un individu sous ce codage, appelé un chromosome, représente une configuration du problème.
Des exemples d'algorithmes évolutionnaires sont donnés par VEGA (Vector Evaluated Genetic Algorithm , MOGA (Multiple Objective Genetic Algorithm), (NSGA) (non dominated sorting genetic algorithm), SPEA (Strength Pareto evolutionary algorithm) ou encore M-PAES (memetic Pareto archived evolution strategy algorithm).

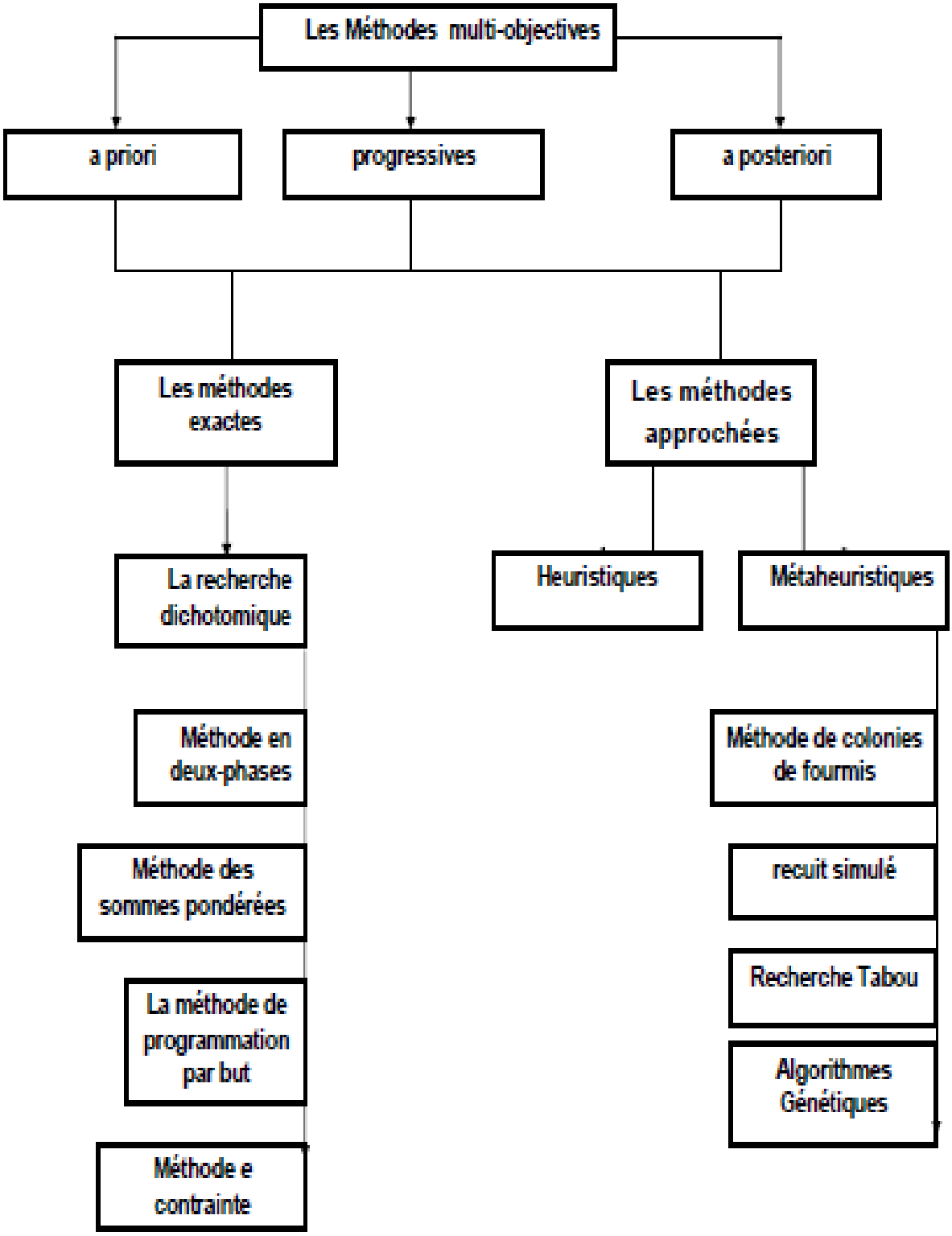


FIGURE 2.8 – Classification des méthodes de résolution multi-objectif

CHAPITRE 3

RÉSOLUTION DU PROBLÈME DE SAC À DOS BI-OBJECTIF

Ce chapitre est consacré à la résolution approchée du problème du sac à dos bi-objectif. Nous proposons à ce propos la méthode en deux phases proposée par Ulungo et Teghem [30] ,pour la résolution exacte des problèmes d'optimisation combinatoire multi-objectif .

3.1 La méthode en deux phases

La méthode en deux phases est un schéma général de résolution pour les problèmes d'optimisation combinatoire bi-objectif. La phase 1 varie très peu en général et la phase 2 doit être spécifique au problème.

Soit le problème de sac à dos bi-objectif (bi-KP) :

$$(bi(KP)) = \begin{cases} max\ z_i(x) = \sum_{j=1}^{n} v_j^i x_j,\ \ i \in \{1,2\},\ v_j \in \mathbb{N}^p \\ tq: \\ \sum_{j=1}^{n} w_j x_j \leq C, \quad C, w_j \in \mathbb{N}^* \\ x_j \in \{0,1\}, j \in \{1,...,n\} \end{cases}$$

3.1.1 Phase 1 : détermination des solutions supportées

La première phase est une variante de la méthode dichotomique de Aneja et Nair [34]. Elle consiste à calculer les solutions efficaces supportées X_{SE_M} [35].
On note S l'ensemble des solutions efficaces supportées initialisé par les deux solutions lexicographique optimales correspondantes à $(z_1(x), z_2(x))$ et $(z_2(x), z_1(x))$ respectivement.
Le calcule des ses solutions revient à l'optimisation à un seul objectif, nous résolvons donc les deux problème mono-objectif p_1 et p_2 pour $i = 1$ et $i = 2$:

$$(bi(KP)) = \begin{cases} max\ z_i(x) \\ tq: \\ \sum_{j=1}^{n} w_j x_j \leq C, \quad C, w_j \in \mathbb{N}^* \\ x_j \in \{0,1\}, j \in \{1,...,n\} \end{cases}$$

Les solutions trouvées x^1 et x^2 sont les solutions optimales des problèmes p_1 et p_2 respectivement .
Après la génération des deux solutions lexicographique optimales, nous appliquons la recherche dichotomique .
La recherche dichotomique est initialisée par les deux solutions lexicographique optimales x^1 et x^2 qu'on notera x^r et x^s respectivement.
Par la suite on considère deux points consécutifs y^r et y^s tel que $y^r = (z_1(x^r), z_2(x^r))$ et $y^s = (z_1(x^s), z_2(x^s))$ et le segment reliant ces deux points est noté $\overline{y^r y^s}$.
On associe alors un problème (p_λ) dont la pondération :
$\lambda = (y_2^r - y_2^s, y_1^s - y_1^r)$, un point y^t est obtenu lors de la résolution de (p_λ) et deux

cas sont alors à considérer :

— Si $y^t \cap \overline{y^r y^s} = \emptyset$ alors y^t n'est pas sur le segmant reliant les deux points y^r et y^s (voir figure 3.1.b), ce qui signifie que deux nouveaux intervalles qu'il faut explorer sont crée, nous avons donc deux problèmes pondéré à résoudre.

— Sinon, y^t se trouve sur le segment, donc aucun nouveau problème pondéré n'est généré, par conséquent la recherche dichotomique est terminé (voir figure 3.1.a).

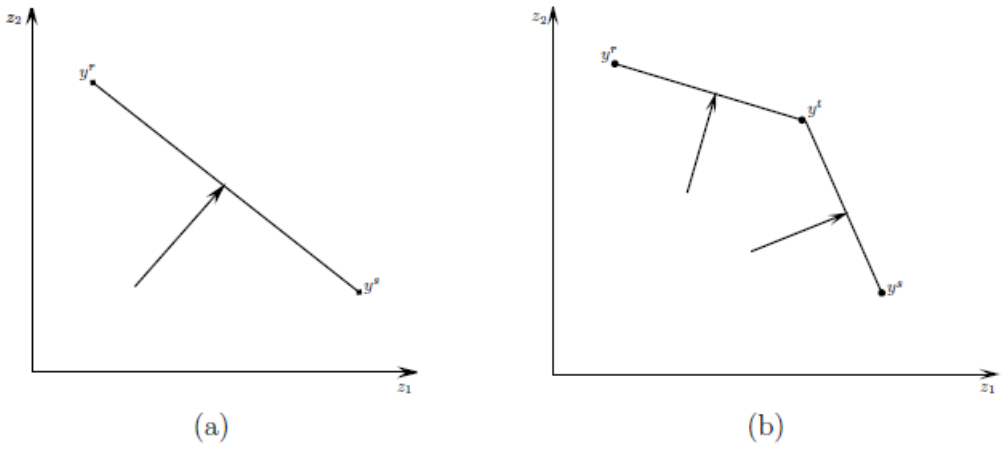


FIGURE 3.1 – Les deux cas de la recherche dichotomique.

Algorithme 6 : Phase 1

Entrée : C, w_j, v_j.

Sortie : Une solution supportées y^r, y^s, y^t.

1. **Début**
2. Résoudre p_1, p_2 , obtenir x^r, x^s ;
3. $y^r = (z_1(x^r), z^2(x^r))$, $y^s = (z_1(x^s), z_2(x^s))$;
4. $S \longleftarrow \{y^r, y^s\}$;
5. Choisir 2 point supportée adjacent y^r, y^s ;
6. Résoudre p_λ avec $\lambda = (y_2^r - y_2^s, y_1^s - y_1^r)$, obtenir y^t ;
7. Si $\lambda \cdot y^t \neq \lambda \cdot y^r$, $S \longleftarrow S \cup \{y^t\}$;
8. S'il reste des adjacences non traiter revenir à (5) ;
9. Si $\lambda \cdot y^t = \lambda \cdot y^r$ et les adjacences sont toutes traitées aller à phase 2 ;
10. **Fin**

3.1.2 Phase 2 : détermination des solutions non supportées

Lors de la deuxième phase, on détermine les solutions efficaces non supportées. Les solutions obtenues dans la première phase sont utilisées pour diminuer l'espace de recherche où se trouvent potentiellement d'autres solutions efficaces. L'espace de recherche est ainsi découpé en plusieurs zones de recherches. Chaque zone de recherche est un triangle délimité par deux solutions efficaces adjacentes, qui contient tous les points qui ne sont dominés par aucunes des deux solutions [36].

Donc, chaque couple (y^r, y^s) de points consécutifs est utilisé pour définir un triangle rectangle dont ils forment l'hypoténuse qu'on note $\triangle(y^r, y^s)$.
L'autre sommet de ce triangle est (y_1^r, y_2^s) et est appelé point nadir local de y^r et y^s . Ensuite, chaque triangle est visité un à un à la recherche de nouvelles solutions. L'exploration de chaque triangle se fait au moyen d'un algorithme d'énumération, nous utiliserons l'algorithme de séparation et évolution (Branch and Bound).
La partie évaluation consiste à définir les bornes qui décrivent chaque triangle.

1 Les bornes $\underline{z}_1, \underline{z}_2$:
Comme le triangle $\triangle(y^r, y^s)$ est définit par les deux points y^r et y^s alors $\underline{z}_1 = y_1^r$ est une borne inférieur par rapport au premier objectif et $\underline{z}_2 = y_2^s$ pour le second.

2 La borne inférieur pour (p_λ) :
La borne inférieure du problème (P_λ) est donnée par $\underline{z}^\lambda = \lambda y^N$, où y^N est le point nadir local défini par $y^N = (y_1^r, y_2^s)$.
Cette borne est mise à jour au fur et à mesure que de nouvelles solutions potentiellement efficaces sont trouvées dans le triangle. Soit $\{y^1, ..., y^m\}$ les points réalisables potentiellement non dominés connus dans le triangle, tels que $y_1^j < y_1^{j+1}$ pour tout $j \in \{1, ..., m-1\}$. En renommant $y^0 = y^r, y^{m+1} = y^s$, la valeur de cette borne inférieure est $\underline{z}^\lambda = min_{j \in \{0,...,m-1\}}(\lambda_1 y_1^j + \lambda_2 y_2^{j+1})$.
L'évaluation des noeuds se fait en calculant séparément une borne supérieure sur chacun des objectifs z_1, z_2 et z_λ. Si une de ces bornes est inférieure ou égale à la borne inférieure correspondante, alors le noeud est fermé [37].

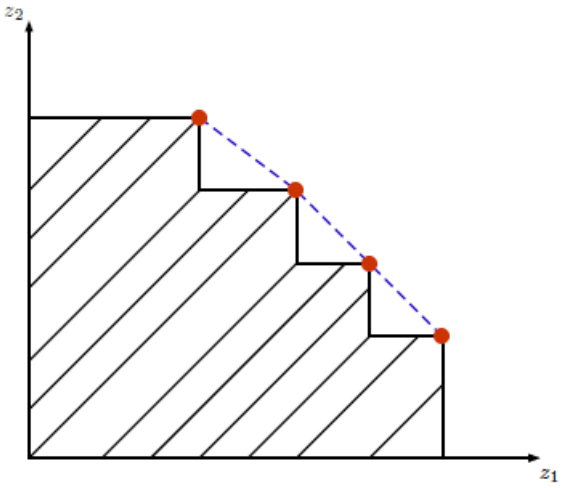


FIGURE 3.2 – Représentation des triangles.

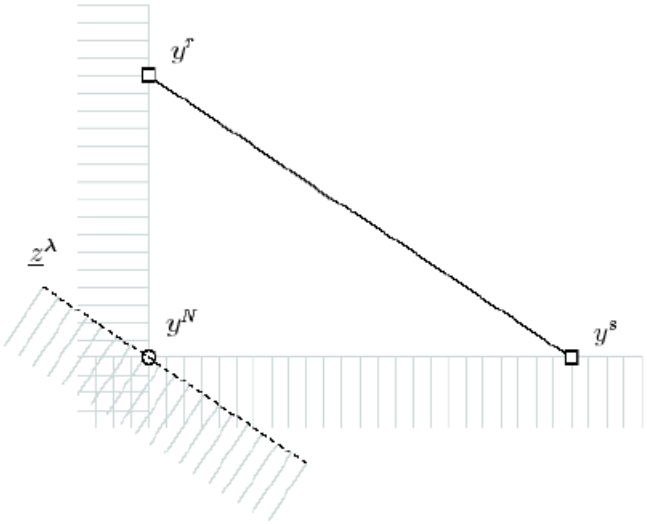


FIGURE 3.3 – Représentation des bornes du triangle.

Dans la partie séparation on suit les étapes suivantes :

1). Le noeud S_0 de l'arbre est initialisé par l'une des solutions x^r où x^s , supposons x^s.

2). Soit $\{i/x_i^s = 1\}$ l'ensemble des indices des variables égale à 1 dans cette solution.

3). On crée alors q sous noeud $S_1, ..., S_q$ qui sont caractérisés par 1 jusqu'à q variables fixes.

4). Toute les solutions non supportée obtenue sont affectées a la liste S ,qui au début est vide.

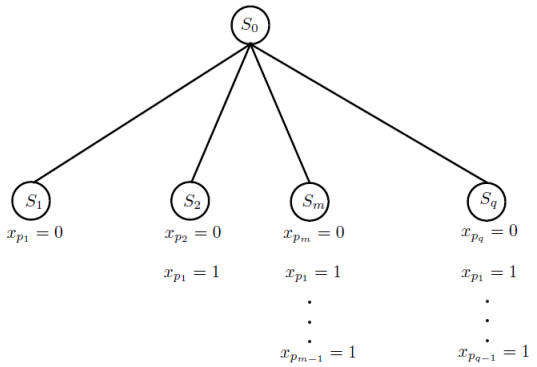


FIGURE 3.4 – Principe de séparation.

Algorithme 7 : Phase 2

Entrée : x^r.

Sortie :L'ensemble de solutions supportées et non supportées S.

1. **Début**
2. Initialiser $x = x^r$;
3. Calculer $\underline{z_1} = y_1^r$, $\underline{z_2} = y_2^s$, $\underline{z}^\lambda = \lambda \cdot (y_1^r, y_2^s)$;
4. **Tant que** $z_1(x) \geq \underline{z}$ et $z_2(x) \geq \underline{z_2}$ et $\underline{z}^\lambda(x) \geq \underline{z}$ **faire** ;
5. Déterminer l'ensemble $J^1 = \{i / x_i^r = 1\} = \{p_1, ..., p_q\}$;
6. **Si** $J_1 = \emptyset$ **terminer** ;
7. **Sinon**
8. **Pour** j allant de p_1 à p_q **tel que** $j \in J^1$ **faire** ;
9. $x_j = 0$;
10. $x_i = 1$, $i \in J^1$ et $i < j$ et $i > p_1$ résoudre (p_λ) ;
11. **Si** p_λ admet une solution x alors
12. Calculer z_1, z_2, z_λ ;
13. $S = S \cup \{(z_1(x), z_2(x))\} = \{y_1, ..., y^m\}$;
14. $y^0 = y^r, y^{m+1} = y^s$;
15. $\underline{z}^\lambda = min_{j \in \{0,...,m\}} \lambda_1 y_1^j + \lambda_2 y_2^{i+1}$, aller en 3
16. **Fin si**
17. **Fin pour**
18. **Fin si**
19. **Fin tant que**
20. **Fin**

CONCLUSION

Dans ce mémoire, Nous nous sommes intéressés à la résolution exacte de sac à dos bi-objectif unidimensionnel en variables binaires. Ce dernier étant un problème classique de l'optimisation combinatoire, présent en tant que sous problème dans de nombreux problèmes d'optimisation. Notre travail porte sur l'application de la méthode en deux phases pour sa résolution.

Pour cela, nous avons présenté le problème de sac à dos et ses différentes variantes en premier lieu, puis nous avons donné les outils de bases de l'optimisation multi- objectif et dans la dernière partie nous avons présenté la méthode en deux phases a été appliquée pour la résolution du problème de sac à dos bi-objectif unidimensionnel en variables binaires.
Finalement, nous avons appliqué la méthode sur un exemple numérique.

BIBLIOGRAPHIE

[1] Akli. M. Problème de tournées de véhicules avec contraintes et fenêtre de temps. Thèse de magister, Université de USTHB, 2013.

[2] Bandyopadhyay, S. Mukherjee.A, An algorithm for many-Objective optimization with reduced objective computations : A study in differential evolution. IEEE Transactions on Evolutionary Computation, 2014. .

[3] Basseur, M. (2014). "Analyse et conception de recherches locales génériques pour l'optimisation combinatoire à un ou plusieurs objectifs. Mémoire d'habilitation à diriger des recherches : Informatique. Université d'Angers. France. 180p.

[4] Belhoul., L. (2014). Résolution de problèmes d'optimisation combinatoire mono et multi-objectifs par énumération ordonnée. Thèse de doctorat : Analyse et Modélisation de Systèmes pour l'Aide à la Décision. Université Paris- Dauphine.

[5] BOYER. V. Contribution à la programmation en nombre entier. PhD thesis, L'Institut National des Sciences Appliquées de Toulouse, 2007.

[6] Boussaid.I. Perfectionnement de métaheuristiques pour l'optimisation continue. PhD thesis, Paris-est Créteil et USTHB, 2013.

[7] Bruyneel, M. P. Duysinx, and C. Fleury. A family of mma approximations for structural optimization. Struct Multidisc Optim, 24 :263-276, 2002.

[8] Collette, Yann et Siarry, P. Optimisation multiobjectif. Eyrolles, 2002.

[9] Dantzig. G.B. Discrete variable extremum problems. Operations research, 5 :266-277, 1957.

[10] Douiri, S. S. Elbernoussi, and H. Lakhbab. Cours des méthodes de résolution exactes heuristiques et métaheuristiques. Technical report, 2009.

[11] Ehrgott, M. Multicriteria optimization. Springer, 2005. OCLC :605688994.

[12] Elkihel. M. Programmation dynamique et rotations de contraintes pour les probl èmes d'optimisation entière. PhD thesis, Université des Sciences et Techniques de Lille, 1984.

[13] Fayard D. and G. Plateau. An algorithm for the solution of the 0-1 knapsack problem. Computing, 28 :269-287, 1982.

[14] Gilmore, P. C. and R.E. Gomory. A linear programming approach to the cutting stock problem. Operations Research, 9 :849-858, 1961.

[15] Greenberg H. and R. L. Hegerich. A branch search algorithm for the knapsack problem. Management Science, 16 :327-332, 1970.

[16] H. Hachimi. H. Hybridation d'algorithmes métaheuristiques en optimisation globale et leurs applications. PhD thesis, Université Mohammed V - Agdal, Rabat, 2013.

[17] Horowitz and S. Sahni. Computiong partitions with applications to the knapsack problem. Journal of ACM, 21 :277-292, 1974.

[18] Kellerer, H. U. Pferschy and D. Pisinger. Knapsack Problems. Springer Verlag, 2004.

[19] Kolesar. P. J. A branch and bound algorithm for the knapsack problem. Management Science, 13 :723-735, 1967.

[20] Mahdi S.. Optimisation multiobjectif par un nouveau schéma de coopération méta/exacte. Master's thesis, Université de Mentouri de Constantine.

[21] Martello S. and P. Toth. Algorithms for knapsack problems. Annals of Discrete Mathematics, 31 :70-79, 1987.

[22] Martello S. and P. Toth. A new algorithm for the 0-1 knapsack problem. Management Science, 34 :633-644, 1988.

[23] Martello S.and P. Toth. Knapsack problems : Algorithms and computer implementations. Wiley, Chichester, England, 1990.

[24] Metropolis, N.M. Rosenbluth, A. Teller, and E. Teller. Optimization by simulated annealing, equation of state calculation by fast computing machines. J. of Chemical Physic, 21 :1087-1092, 1953.

[25] Pascoal,M. M.E. Captivo, and J. Climaco. A note on new variant of murty's ranking assignments algorithm. 4OR, 1 :243-255, 2003.

[26] Plateau G. et M. Elkihel. A hybrid method for the 0-1 knapsack problem. Methods of Operations Research, 49 :277-293, 1985.

[27] Pisinger, D. An exact algorithm for large multiple knapsack problems. Journal of Operational Research, 114 :528-541, 1999.

[28] Rémy-Robert, Alexandre Joseph ; Systèmes Interactifs d'Aide à l'Élaboration de Plannings de Travail de Personnel Contraintes, Aide à la Décision, Représentation Combinatoire des Préférences, Équité et Résolution par Décomposition Arborescente et par Consistance ; 07 Novembre 2003 ; Université Joseph Fourier-Grenoble1, Science et Géographie ; Page 28.

[29] SAVELSBERGH. M. A branch-and-price algorithm for the generalized assignment problem. Operations Research, 45(6) :831-841, 1997.

[30] Ulungu, E. L. Optimisation combinatoire multicritére : détermination de l'ensemble des solutions efficaces et méthodes interactives.

[31] Lalami, M. E. Contribution à la résolution de problèmes d'optimisation combinatoire : méthodes heuristiques et parallèles.

[32] M. Geoffrion, A. Proper efficiency et the theory of vector maxi-mization.

[33] Resende, M. G., and Pardalos, P. M. Handbook of applied optimization. Oxford University Press, 2002.

[34] Aneja, Yash P. et Nair, K. P. Bicriteria transportation problem. 73-78.

[35] Olivier, S. Optimisation multi-objectifs : un tutoriel.

[36] Thomas, V. Caractérisation des solutions efficaces et algorithmes d'énumération exacts pour l'optimisation multiobjectif en variables mixtes binaires, 2013.

[37] Jorge, J. Nouvelles propositions pour la résolution exacte du sac à dos multiobjectif unidimensionnel en variables binaires, 2010.

[38] E. Ulungu and J. Teghem. An efficient procedure to solvebi-objective combinatoirial optimization problem. Foundations of Computing and Decision Sciences, 20 :149-165, 1995.

- ملخص:

في هذا العمل، قمنا بحل مشكلة التحسين التوافقي بطريقة دقيقة وهي طريقة ذات الطورين في حالة : حقيبة الظهر ثنائية الهدف. تذكرنا أولاً مشكلة حقيبة الظهر والطرق الدقيقة والتقريبية لحل هذه المشكلة ، بنفس الطريقة قدمنا بعض المفاهيم الأساسية في التحسين وعرفنا مشكلة التحسين متعدد الأهداف ، بمفاهيمه الأساسية: (الهيمنة ، باريتو الجبهة وتسوية السطح ، التحدب ...). ثم وصف طرق الحل.
أخيراً ، وصف طريقة الجملتين المطبقة على مشكلتنا.

- الكلمات المفتاحية: تحسين متعدد الاغراض ، حقيبة الظهر ، طريقة الطورين

- **Résumé :**

Dans ce travail, on s'est intéressé à la résolution d'un problème d'optimisation combinatoire avec une méthode éxacte qui est la méthode en deux phase cas : sac à dos bi-objectif . On a rappelé d'abord le problème du sac à dos et les méthodes de résoudre exactes et approchées pour ce problème, de la même façon, on a donné quelques notions de base en optimisation et on définit le problème d'optimisation MultiuObjectif , avec ses concepts fondamentaux : (la domination, Front de Pareto et La surface de compromis, la convexité...). Puis on décrit les méthodes de résolution.
Enfin, nous décrivons la méthode en deux phases appliquées à nôtre problème.

- **Mots clés : optimisation multi-objectif, sac à dos, méthode en deux phases.**

- **Abstract:**

In this work, we are aiming to solve combinatorial optimsation issue, the case of bi-creteria knapsack problem . We first recalled the problem of the backpack and the exact and approximate methods of solving for this problem, in the same way, we gave some basic concepts in optimization and we denied the problem of multi-objective optimization, with its fundamental concepts : (Domination, Pareto forehead and compromise surface, convexity...). Then we describe the resolution methods.
Lastely, to archive our mentioned aim and in order to describe the methode in two phases.

- **Keywords : Multi-objective optimization, Knapsack, method two phases.**