N° d'ORDRE :11/2011-M/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET
DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
≪ HOUARI BOUMEDIENE ≫
FACULTE DE MATHEMATIQUES

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

EN : Mathématiques

Spécialité : Recherche Opérationnelle

Par : Selma ZOUAOUI

Thème

Approche multicritère pour le problème de sac à dos

Soutenu publiquement le 23/10/2011 , devant le jury composé de :

M^{r} M. MOULAÏ	Professeur à l'USTHB	Président.
M^{r} A. KHELLADI	Professeur à l'USTHB	Directeur de memoire.
M^{r} M.E. CHERGUI	Maitre de conférence/A à l'USTHB	Examinateur.
M^{r} A. MEZGHICHE	Maitre de conférence/B à l'USTHB	Invitée.

J*e tiens tout d'abord à remercier le Professeur M^r^ A. KHELLADI pour l'honneur qu'il ma fait en acceptant de m'encadrer. Ses conseils précieux ont permis une bonne orientation dans la réalisation de ce modeste travail.*

J*e teins également à remercier M^r^ M. MOULAI d'avoir accepté de présider le jury de ce mémoire.*

J*e remercie M^r^ M.A. CHERGUI et M^r^ A. MEZGHICHE d'avoir accepter de faire partie du jury et consacrer leurs temps à la lecture et à la correction de ce mémoire.*

M*es remerciements les plus vifs vont tout particulièrement à mes parents, mes soeurs et mes frères.*

E*nfin, merci à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.*

Table des matières

Table des matières 1
Liste des figures 5
Liste des tables 6
Notations 7

Introduction générale **8**

1 *Etat de l'art de l'optimisation multiobjectif* **12**
1.1 Introduction 12
1.2 Définition 13
1.3 Problème d'optimisation combinatoire multiobjectif 13
1.4 Concepts de base 14
1.4.1 Dominance de Pareto 14
1.4.2 Efficacité(ou Pareto optimale) 15
1.4.3 Optimalité lexicographique 16
1.4.4 Points particuliers 17
1.4.5 Caractérisation du front Pareto 19
1.4.6 Convexité 22
1.5 Complexité et difficulté de l'optimisation combinatoire multiobjectif 22
1.6 Choix de la méthode d'aide à la décision 23
1.7 Méthodes classiques de résolution 24
1.7.1 Méthodes de résolution exacte 24
1.7.2 Méta-heuristiques multiobjectif 32
1.7.3 Méthodes hybrides de résolution des problèmes d'OCMO 37
1.8 Conclusion 39

2 *Problèmes de type sac à dos* **40**
2.1 Introduction . . . 40
2.2 Le problème de sac à dos (knapsack problem (KP)) . . . 41
2.2.1 Variantes autour du problème . . . 42
2.2.2 Méthodes de résolution . . . 45
2.3 Conclusion . . . 51

3 Pré-traitement et règles de réduction pour le sac à dos multiobjectif **52**
3.1 Introduction . . . 52
3.2 Pré-traitement pour le sac à dos mono objectif . . . 53
3.2.1 Réduction de la taille des instances . . . 53
3.3 Pré-traitement et règles de réduction pour le sac à dos multiobjectif . . . 54
3.3.1 Variables régulières et données du problème . . . 54
3.3.2 Bornes sur la cardinalité des solutions efficaces . . . 56
3.3.3 Propriétés des variables régulières . . . 57
3.4 Expérimentations numériques . . . 58
3.4.1 Instances du problème de sac à dos . . . 59
3.4.2 Résultats expérimentaux . . . 60
3.5 Conclusion . . . 63

4 *La méthode en deux phases pour le problème de sac à dos bi-objectif unidimensionnel en variables binaires* **66**
4.1 Introduction . . . 66
4.2 La méthode en deux phases pour le problème de sac à dos bi-objectif unidimensionnel en variables binaires . . . 67
4.3 Application de la procédure de réduction dans la méthode en deux phases . 73
4.4 Implémentation et résultats . . . 75
4.5 Discussion . . . 77
4.6 Conclusion . . . 79

Conclusion générale **80**

Bibliographie **82**

Table des figures

1.1 Exemple de domminance 15
1.2 À gauche, les solutions . À droite, leur images dans l'espace des objectifs. . 17
1.3 Représentation de la frontière Pareto et des points particuliers. 19
1.4 Représentation des différents types de solutions en bi-objectif. 21
1.5 Problème P défini par y^r et y^s. 26
1.6 Un nouveau point supporté est trouvé, la dichotomie continue. 26
1.7 Interprétation graphique de la méthode 28
1.8 a 30
1.9 b 31
1.10 c 31
1.11 Classification des approches d'optimisation multi-objectif. 38

2.1 problème du sac à dos 41
2.2 Une procédure de séparation et évaluation pour(01KP) 48
2.3 Programmation dynamique pour (01KP) 50
2.4 Algorithme glouton pour(01KP) 51

3.1 Régularité des valeurs des variables dans les solutions efficaces. 55
3.2 Nombre moyen de variables fixées par les propriétés pour les instances du groupe A. 61
3.3 Nombre moyen de variables fixées par les propriétés pour les instances du groupe A. 62
3.4 Nombre moyen de variables fixées par les propriétés pour les instances du s.groupe B-1. 62

3.5 Nombre moyen de variables fixées par les propriétés pour les instances du s.groupe B-2. 63
3.6 Nombre de variables régulières par les instances bi-objectif où au moins un vecteur de coût est corrélé au vecteur des poids. Les solutions efficaces des instances du sous-groupe B-3 sont inconnues à partir de 350 variables. . . . 64
3.7 Nombre moyen de variables fixées par les propriétés pour les instances du groupe C. 64
3.8 Nombre moyen de variables fixées par les propriétés pour les instances du groupe D. 65
3.9 Nombre moyen de variables fixées par les propriétés pour les instances du groupe D. 65

4.1 Problème P défini par y^r et y^s. 68
4.2 Un nouveau point supporté est trouvé, la dichotomie continue. 69
4.3 L'espace de recherche pendant la seconde phase se décrit comme un ensemble de triangles. La ligne en pointillés représente l'enveloppe convexe de Y_N ainsi que les hypoténuses. Les angles droits des triangles se situent à l'intersection des cônes de dominance de points supportés adjacents. 70
4.4 Illustration des bornes lors de l'exploration d'un triangle dans la seconde phase. Les rayures représentent les régions ôtée de l'espace de recherche par ces bornes. Les carrés (□) représentent les performances des solutions décrivant le triangle. Le rond (∘) illustre le point nadir local utilisé pour le calcul de la borne $\underline{z}^{\lambda}$, représentée ici par une ligne en pointillés. 71
4.5 La borne inférieure $\underline{z}^{\lambda}$. 72
4.6 principe de séparation 72

Liste des tableaux

1.1 Ensembles de solutions et leurs projections dans l'espace des objectifs. . . . 20

3.1 Coefficients utilisés pour générer les instances. 60
3.2 Coefficients utilisés pour générer les instances. 60

4.1 *Table des résultats pour les instances de s. groupe* A_1. 76
4.2 *Table des résultats pour les instances de s. groupe* A_2 76
4.3 *Table des résultats pour les instances de s. groupe* A_3. 77
4.4 *Table des résultats pour les instances de s. groupe* A_4 77
4.5 *Table des résultats pour les instances de s. groupe* B_1 77
4.6 *Table des résultats pour les instances de s. groupe* B_2 78
4.7 *Table des résultats pour les instances de s. groupe* B_3. 78

Notations

Symbole	signification
$\mathbb{R}^p_>$	$\{y \in \mathbb{R}^p : y > 0\}$
$\mathbb{R}^p_\geq$	$\{y \in \mathbb{R}^p : y \geq 0\}$
$\mathbb{R}^p_{>_=}$	$\{y \in \mathbb{R}^p : y >_= 0\}$

Présenté par : Selma Zouaoui *Sous la direction de : Abdelkader KHELLADI, Professeur à l'USTHB*

Introduction générale

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

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

Généralement, les méthodes que l'on met en œuvre pour résoudre un problème d'optimisation combinatoire dépendent de la complexité de ce dernier. La théorie de la NP-complétude (Garey et Johnson [37]) fournit de précieux renseignements sur le genre de méthodes à adopter en fonction de la difficulté intrinsèque des problèmes. Il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable.

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

– Louer un appartement bien situé et d'un prix raisonnable

– Etablir un planning pour les vacances satisfaisant toute la famille.
– Acheter une voiture.
– Choisir entre plusieurs itinéraires.

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

Une autre approche consistait en l'optimisation d'une agrégation linéaire des objectifs, chacun pouvant avoir un poids représentant son importance. Lorsque l'on se trouve dans un réel contexte multiobjectif, il n'est pas toujours possible de trouver un ordre d'importance sur les critères. Il est alors nécessaire de rechercher les solutions de meilleur compromis entre les objectifs. Si cette notion de compromis sera définie plus précisément dans le chapitre 1, il est facile de voir que dans ce contexte la solution recherchée n'est pas une unique solution mais un ensemble de solutions représentant les différents compromis possibles. Ainsi l'optimisation multiobjectif s'intéresse aux particularités liées à l'existence de ces différentes solutions optimales.

En particulier, les méthodes de résolution devront être dédiées à ce type de problèmes qui sont la plupart du temps NP-difficiles. De même, la comparaison de solutions produites par différents algorithmes n'est plus une chose facile en multiobjectif car il faut alors comparer différents ensembles de solutions.

Qu'ils comportent un seul ou plusieurs objectifs, les problèmes d'optimisation sont en général difficiles à résoudre. De plus, le temps de calcul nécessaire à leur résolution peut devenir si important que l'algorithme développé devient inutilisable en pratique. Il n'existe pas d'algorithme générique capable de résoudre toutes les instances de tous les problèmes efficacement.

Un grand nombre de méthodes ont été développées pour tenter d'apporter une réponse satisfaisante à ces problèmes. Parmi celles-ci, nous distinguons les méthodes dédiées à un problème et les méthodes plus génériques pouvant s'appliquer à un ensemble de problèmes. Parmi ces méthodes génériques, nous distinguons deux grandes classes de

méthodes : les méthodes exactes et les méthodes approchées.

Les méthodes exactes examinent, souvent de manière implicite, la totalité de l'espace de recherche. Ainsi, elles ont l'avantage de produire une solution optimale lorsqu'aucune contrainte de temps n'est donnée.

Néanmoins, le temps de calcul nécessaire pour atteindre une solution optimale peut devenir vite prohibitif, et ce, malgré les diverses techniques et heuristiques qui ont été développées pour accélérer l'énumération des solutions. Dans de telles situations (et dans beaucoup d'autres), les méthodes approchées constituent une alternative indispensable et complémentaire. Le but d'une telle méthode n'est plus de fournir une solution optimale au problème donné. Elle cherche avant tout à produire une solution de meilleure qualité possible avec un temps de calcul raisonnable.

En général, une méthode approchée examine seulement une partie de l'espace de recherche. Le choix des solutions examinées est souvent dicté par des heuristiques. À partir de deux méthodes de résolution différentes, il est possible de concevoir des méthodes hybrides dans le but de fournir des résultats supérieurs aux deux méthodes qui les composent.

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

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

Dans ce mémoire nous nous intéressons au problème de sac à dos multiobjectif unidimensionnel en variables binaires cas : biobjectif. Nous étudions un algorithme de résolution exacte en s'appuyant sur une procédure de séparation et évaluation ("branch-and-bound") intitulée méthode en deux phases. Ainsi, nous présentons une procédure de

pré-traitement avant la résolution du problème.

Organisation du document

Ce mémoire est structuré de la manière suivante :

•Dans le premier chapitre, nous présentons l'optimisation multiobjectif. Nous donnons 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.

•Le deuxième chapitre est consacré aux problème de sac à dos ou Knapsack problem,nous présentons quelques variantes de ce problème ainsi que des méthodes de résolution.

•Dans Le troisième chapitre, nous étudions les procédures de réduction de la taille d'une instance. Nous commençons par le cas de sac à dos mono objectif ensuite, nous passons aux problèmes d'optimisation multiobjectif

•Le quatrième chapitre contient l'adaptation de la procédure de réduction et la méthode en deux phases dans le problème de sac à dos biobjectif.

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

1

Etat de l'art de l'optimisation multiobjectif

1.1 Introduction

Les problèmes d'optimisation combinatoire issus des problématiques réelles sont la plupart du temps de nature multiobjectif car plusieurs critères d'évaluation souvent contradictoires sont à considérer simultanément. Optimiser un tel problème relève donc de l'optimisation combinatoire multiobjectif.
L'optimisation multiobjectif possède ses racines dans les travaux en économie de Edgeworth [20] et Pareto [62]. Elle a ainsi été initialement utilisée en économie et dans les sciences du management, puis graduellement dans les sciences pour l'ingénieur.
Ce chapitre à pour objectif de présenter le contexte de l'optimisation multiobjectif, ses principales définitions et surtout les problématiques liées à ce domaine.

1.2 Définition

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.

1.3 Problème d'optimisation combinatoire multiobjectif

Un problème d'optimisation combinatoire multiobjectif (PMO) (multiobjective combinatorial optimization problem) peut être défini par :

Définition 1.3.1.

$$(MOP)\begin{cases} \text{"}opt\text{"}\ [z_1(x), \cdots, z_r(x)] \\ x \in X \end{cases}$$

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

D'après cette définition, il est clair que l'optimum n'est plus une simple valeur comme pour les problèmes à un objectif, mais un ensemble de points, appelé l'ensemble des meilleurs compromis ou le front Pareto.

Remarque 1.3.1. Le symbole " " signifie qu'il n'est généralement pas possible de trouver dans X 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 $x^* \in X$ telle que,en regard des actions de X, le vecteur $z(x^*) = (z_1^*, \cdots, z_k^*)$ est bon, acceptable selon les préférences du décideur,voir optimal, 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 $Y = z(X)$ représente les points réalisables dans l'espace des critères et $y = (y_1, \cdots, y_r)$ avec
$y_i = z_i(x)$ 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.

1.4 Concepts de base

1.4.1 Dominance de Pareto

Définition 1.4.1. Soient deux vecteurs critères y,$y' \in z(X)$ .On dit que y domine y' si et seulement si $y \leq y'$ et $y \neq y'$ (i.e. $y_i \leq y'_i \forall i = 1, \cdots, r$ et $y < y'$ pour au moins un indice i).

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.
Illustrons maintenant cette relation par un exemple en dimension 2 dans la figure 1.1 .
Sur cette figure, Y (l'espace réalisable dans l'espace des objectifs) est l'image de X. Ainsi, chaque point y_i est l'image de x_i par z : $y^i = z(x^i)$. Prenons le pointy^1 comme point de référence.
Nous pouvons distinguer trois zones :

- **La zone de préférence** :
  est la zone contenant les points dominés par y^1,

- **La zone de dominance** :
  est la zone contenant les points dominant y^1,

- **La zone d'incompatibilité** :
  contient les points incomparables avec y^1.

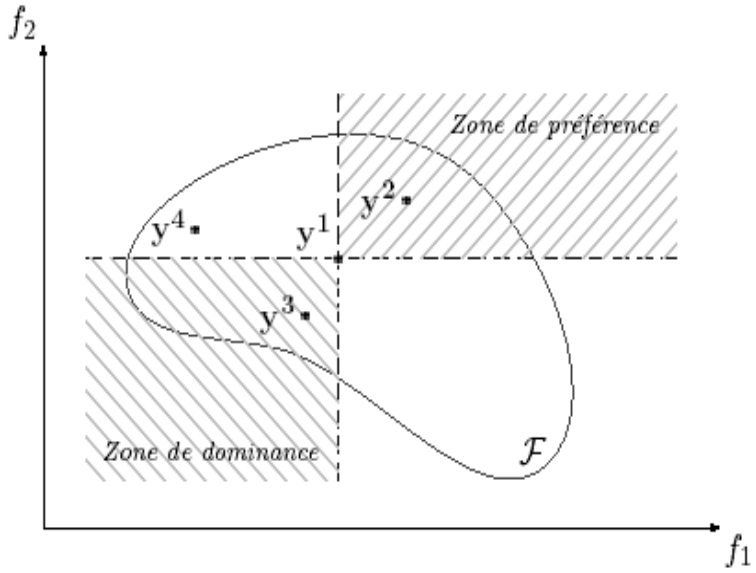

FIG. 1.1 – Exemple de domminance

Ainsi, il est clair que y^2 est dominé par y^1 (y^1 est préféré à y^2), que y^3 domine y^1 (y^3 est préféré à y^1), et que y^4 est non dominé (incomparable) avec y^1 .

Dominance forte

Définition 1.4.2. Soient deux vecteurs critères y,$y' \in Y$. On dit que y domine fortement y' si et seulement si $y < y'$ (i.e. $y_i < y'_i$; $\forall i = 1, \cdots, r$)

Si y domine fortement y' , alors y est meilleur que y' sur tous les critères.

1.4.2 Efficacité(ou Pareto optimale)

Définition 1.4.3. Une solution $x^* \in X$ est une solution efficace s'il n'existe pas de $x \in X$ tel que $z(x)$ domine $z(x^*)$.

Efficacité faible

Définition 1.4.4. Une solution $x^* \in X$ est une solution faiblement efficace s'il n'existe pas de $x \in X$telle que $z(x) < z(x^*)$. Une solution est faiblement efficace si son vecteur critère n'est pas fortement dominé.

Efficacité forte

Définition 1.4.5. Une solution $x^* \in X$ est une solution fortement efficace s'il n'existe pas de $x \in X$telle que $x \neq x^*$et $z(x) \leq z(x^*)$.

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.

1.4.3 Optimalité lexicographique

Cette définition est considérée dans le cas de maximisation.

Définition 1.4.6. (Ordre lexicographique) Soient y^1 et $y^2 \in \mathbb{R}^p$, alors y^1 est optimale au sens lexicographique ie $y^1 >_{lex} y^2$ s'il existe $l \in \{1, \cdots, r\}$ tel que pour tout $k < l$,$y_k^1 = y_k^2$ et $y_l^1 > y_l^2$.
Si $y^1 >_{lex} y^2$ et $y^1 = y^2$ alors on note $y^1 >_{=lex} y^2$

Exemple 1. *Considerons les deux points A et B de* $\mathbb{R}^6$

$$A = (1, 2, 3, 9, 4, 9)$$
$$B = (1, 2, 3, 4, 5, 6)$$

Pour ces deux points, nous avons $A >_{lex} B$ *car jusqu'à la troisième position ,nous avons* $A_i = B_i i = 1, 2, 3$ *et, pour la quatrième position, nous avons* $9 > 4$.
On conclut donc que la solution A domine lexicographiquempent la solution B.

Exemple 2. *Soit le problème d'optimisation bi-objectif suivant :*

$$\begin{cases} \max z_1(x) = 2x_1 + 5x_2 + 9x_3 + 8x_4 + 6x_5 \\ \max z_2(x) = 8x_1 + 6x_2 + 2x_3 + 5x_4 + 8x_5 \\ s.c \\ 8x_1 + 7x_2 + 5x_3 + 4x_4 + 2x_5 \leq 17 \\ x_1 + x_2 + x_3 + x_4 + x_5 = 3 \\ x_i \in \{0, 1\} \qquad \forall i \in \{1, \cdots, 5\} \end{cases}$$

	$(x_1, x_2, x_3, x_4, x_5)$
x^1	$(0,0,1,1,1)$
x^2	$(0,1,0,1,1)$
x^3	$(0,1,1,0,1)$
x^4	$(0,1,1,1,0)$
x^5	$(1,0,0,1,1)$
x^6	$(1,0,1,0,1)$
x^7	$(1,0,1,1,0)$
x^8	$(1,1,0,0,1)$

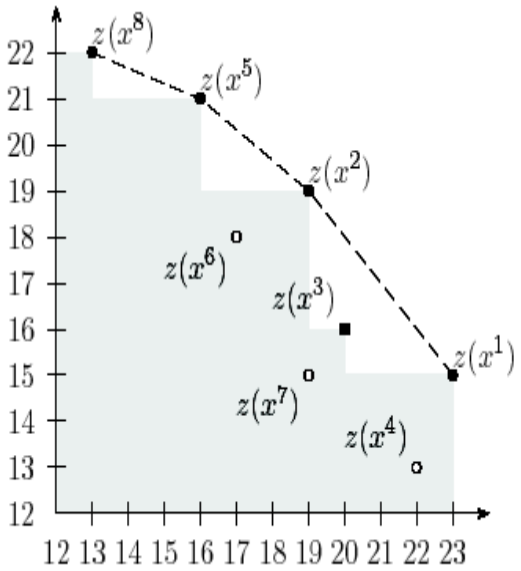

FIG. 1.2 – À gauche, les solutions . À droite, leur images dans l'espace des objectifs.

Ce problème accepte 8 solutions réalisables, énumérées dans la partie à gauche de la figure 1.2. La partie à droite représente leurs images dans l'espace des objectifs.
Seules les solutions x_1, x_2, x_3, x_5 et x_8 sont efficaces, les autres solutions sont toutes dominées.
x_1 et x_8 sont aussi lexicographiquement optimales.

Définition 1.4.7. (Optimalité lexicographique) Cette définition est considérée dans le cas de maximisation.
Soit π une permutation de $\{1, \cdots, r\}$.
Une solution admissible x est appelée lexicographiquement optimale par rapport à π si pour tout $x' \in X, z_\pi(x) >= lex z_\pi(x')$, où $z_\pi(x) = (z_{\pi_1}(x), \cdots, z_{\pi_r}(x))$

1.4.4 Points particuliers

En vue d'avoir certains points de références permettant de discuter de l'intérêt des solutions trouvées, des points particuliers ont été définis dans l'espace objectif.
Ces points peuvent représenter des solutions réalisables ou non.

- le **point idéal** y^I est le point qui a comme valeur pour chaque objectif la valeur optimale de l'objectif considéré.

$$y^I = (y_1^I, \cdots, y_r^I), \text{où } y_i^I = \min_{x \in X} Z_i(x), \forall i \in \{1, \cdots, r\}$$

Ce point ne correspond pas à une solution réalisable car si c'était le cas, cela sousentendrait que les objectifs ne sont pas contradictoires et qu'une solution optimisant un objectif, optimise simultanément tous les autres, ce qui ramenerait le problème à un problème ayant une seule solution Pareto optimale.

– le **point utopique** y^U est défini comme suit :

$$y^U = y^I + \epsilon, \text{ où } \epsilon > (0, \cdots, 0)$$

Il est clair, de par sa définition, que ce point n'est pas réalisable

– le **point Nadir** y^N :

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 :

$$G = \begin{pmatrix} z_1(\overline{x}_1) & z_2(\overline{x}_1) & \dots & z_r(\overline{x}_1) \\ z_1(\overline{x}_2) & z_2(\overline{x}_2) & \dots & z_r(\overline{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ z_1(\overline{x}_r) & z_2(\overline{x}_r) & \dots & z_r(\overline{x}_r) \end{pmatrix}$$

où x_i est la solution optimale obtenue en minimisant le critère z_i sur X, $\forall i = 1; r$.

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

$$y_j^N = \max_{i=\overline{1,r}} G_{ij}, \forall j \in \{1, \cdots, r\}$$

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.

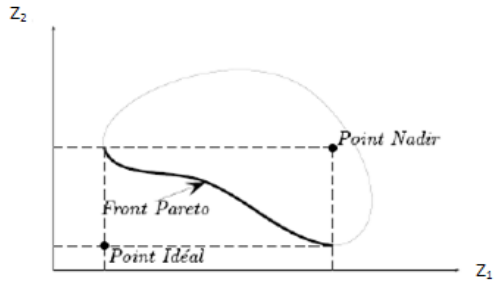

FIG. 1.3 – Représentation de la frontière Pareto et des points particuliers.

1.4.5 Caractérisation du front Pareto

Ensemble Pareto complet minimal / Ensemble Pareto complet maximal

Nous rappelons que la solution que nous cherchons pour un problème d'optimisation multiobjectif n'est pas unique, mais un ensemble de points, que nous avons appelé à la section 1.3 ensemble de compromis .
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.
Résoudre un problème multi-objectif consiste à trouver les solutions Pareto optimales, et par conséquent, le front Pareto.

Définition 1.4.8. [46] (Équivalence)
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 décisionnel.
Un ensemble complet X_E est un ensemble de solutions efficaces tel que toute solution $x \in X/X_E$ est soit équivalente, soit dominée par une solution $x_0 \in X_E$.
Un ensemble complet X_{E_m} sans solution équivalente est dit **minimal**, tandis que l'ensemble complet X_{E_M}contenant toutes les solutions équivalentes est dit **maximal**.

Remarque 1.4.1. – Pour chaque point $y \in Y_N$ il existe au moins une solution $x \in X_E$ telle que $z(x) = y$;

Notation	Signification	Image
X	ensemble des solutions réalisables	Y
X_E	ensemble complet de solutions efficaces	Y_N
X_{E_m}	ensemble complet minimal de solutions efficaces	Y_N
X_{E_M}	ensemble complet maximal de solutions efficaces	Y_N
X_{SE}	ensemble complet de solutions supportées	Y_{SN}
X_{SE_m}	ensemble complet minimal de solutions supportées	Y_{SN}
X_{SE_m}	ensemble complet minimal de solutions supportées	Y_{SN}
X_{SE1_M}	ensemble complet maximal de solutions supportées extrêmes	Y_{SN}
X_{SE_1}	ensemble complet de solutions supportées extrêmes	Y_{SN_1}
X_{SE_2}	ensemble complet de solutions supportées non extrêmes	Y_{SN_2}
X_{NE}	ensemble complet de solutions non supportées	Y_{NN}
X_{NE_m}	ensemble complet minimal de solutions non supportées	Y_{NN}
X_{NE_M}	ensemble complet maximal de solutions non supportées	Y_{NN}

TAB. 1.1 – Ensembles de solutions et leurs projections dans l'espace des objectifs.

– tout ensemble complet contient un ensemble complet minimal ;

– toute solution $x \in X/X_{E_M}$ est dominée.

La table 1.1 récapitule les notations concernant les différents ensembles de solutions.

Solutions supportées / Solutions non supportées

Le front Pareto contient deux types de solutions efficaces :

a -les solutions supportées :

leurs images dans Y se trouvent sur l'enveloppe convexe de l'ensemble des solutions (voir figure 1.4) et peuvent donc être trouvées à l'aide d'une agrégation linéaire des objectifs [38].

b -les solutions non supportées :

elles sont pareto optimales mais leurs images dans Y ne sont pas situées sur l'enveloppe convexe de Y.

Aucune de ces solutions ne peut être obtenue par l'optimisation d'une agrégation

linéaire des objectifs.

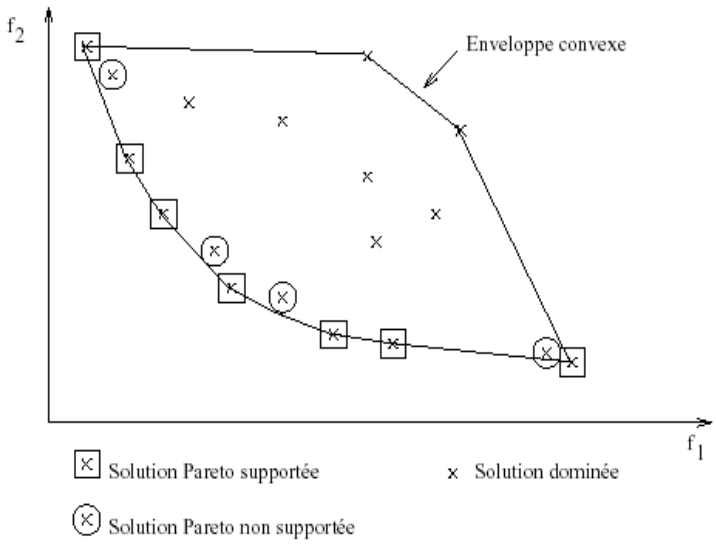

FIG. 1.4 – Représentation des différents types de solutions en bi-objectif.

Théorème 1.4.1. *(Somme pondérée [38])*
Soit P un MOP avec p objectifs.Soit $\lambda \in \mathbb{R}^p_{\geq}$.
On construit le problème mono-objectif P_λ :

$$p_\lambda \begin{cases} \max z^\lambda(x) = \lambda \cdot z(x) \\ s.c \\ Ax \leq b \\ x \in \{0,1\}^n \end{cases}$$

où l'opérateur $\cdot$ représente le produit scalaire.
Si x^* est une solution optimale de p_λ, alors on a les énoncés suivants :

1. *Si $\lambda \in \mathbb{R}^p_{\geq}$, alors x^* est faiblement efficace ;*

2. *Si* $\lambda \in \mathbb{R}^p_>$, *alors* x^* *est efficace ;*

3. *Si* $\lambda \in \mathbb{R}^p_\geq$ *et* x^* *est l'unique solution optimale de* p_λ, *alors* x^* *est efficace .*

L'utilisation d'une somme pondérée ne permet de trouver que les solutions efficaces dont les images se situent sur la frontière de l'enveloppe convexe de Y, notée conv(Y).
Les autres solutions efficaces ne peuvent être obtenues quel que soit le poids λ.

1.4.6 Convexité

Définition 1.4.9. Un ensemble A est convexe, si et seulement si l'équivalence suivante est vérifiée :
$x \in A \wedge x' \in A \Longleftrightarrow Segment(x, x') \subset A$

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

1.5 Complexité et difficulté de l'optimisation combinatoire multiobjectif

Les problèmes d'optimisation combinatoire multiobjectif (OCMO)sont très complexes. En effet, la difficulté des problèmes d'optimisation due au phénomène d'explosion combinatoire s'additione de nouvelles difficulté propre aux problèmes d'optimisation multiobjectif. Un problème d'OCMO est un problème Non Polynomial(NP) si la détermination d'un point non dominé ne peut être faite en temps polynomial pour toutes les instances du problème. Serafini (1987) [68]a proposé la définition suivante de la NP-complétudepour les problèmes OCMO :

Définition 1.5.1. Pour chaque point z existe -t-il une solution $x \in X$tel que :$F(x) \preceq z$?

D'après la définition 1.5.1, un problème D'OCMO serait NP si les problèmes d'optimisation combinatoire mono objectif qui lui correspondent sont NP.Serafini démontre aussi comment certains problème d'optimisation mono objectif "facile" deviennent NP-complet dans le cas bi-objectif.C'est le cas pour le problème d'affectation et le problème de plus court chemin. Dans plusieurs OCMO, la détermination d'un ensemble complet est très difficile. Malgré que les solutions supportées sont obtenues par la résolution d'un programme linéaire pondéré s'il existe un algorithme efficace pour la résolution du problème mono objectif associé ,mais aussi le calcul des solutions non supportées est très intéressant. Or, la détermination de ces solutions nécessite généralement de résoudre des problèmes NP-difficiles, même si le problème mono objectif considéré peut être résolu en temps polynomial.
Par conséquent, une difficulté pratique dans la résolution des MOCO provient de la détermination d'un ensemble X_{NE}. De plus, des expérimentations ont montré que les solutions non supportées, en plus d'être plus difficiles à obtenir, peuvent également être beaucoup plus nombreuses que les solutions supportées [77].
D'un point de vue théorique, la plupart des OCMO sont NP-complets, P-complets et intraitables [21] . En particulier, Hugot a démontré l'intraitabilité du problème de sac à dos multiobjectifs [51]. Cependant, cette propriété apparaît même si le problème mono objectif correspondant est dans la classe P.

1.6 Choix de la méthode d'aide à la décision

Nous rappelons que la résolution d'un problème multiobjectif menant à la détermination d'un ensemble de solutions Pareto.
Pour le choix d'une solution finale à garder, il est nécessaire de faire intervenir l'humain à travers un décideur. Ainsi, avant de se lancer dans la résolution d'un problème multiobjectif, il faut se poser la question du type de méthode d'optimisation à utiliser.
En effet, on peut ranger les méthodes de résolution de problèmes multiobjectif en trois familles, en fonction du moment où intervient le décideur. Ainsi nous pouvons trouver les familles suivantes :

- **Les méthodes à préférence a priori** :
  Dans ces méthodes, l'utilisateur définit le compromis qu'il désire réaliser (il fait par de ses préférences) avant de lancer la méthode d'optimisation.
  Nous retrouvons dans cette famille la plupart des méthodes par agrégation (où les

fonctions objectifs sont fusionnées en une seule).

- **Les méthodes à préférence progressives** :
  Dans ces méthodes, l'utilisateur affine son choix de compromis au fur et à mesure du déroulement de l'optimisation.
  Nous retrouvons dans cette famille les méthodes interactives.

- **Les méthodes à préférence a posteriori** :
  Dans ces méthodes, l'utilisateur choisit une solution de compromis en examinant toutes les solutions extraites par la méthode d'optimisation

Il existe des méthodes d'optimisation multiobjectif qui n'entrent pas exclusivement dans une de ces familles. Par exemple, on peut utiliser une méthode à préférence a priori en lui fournissant des préférences choisies au hasard. Le résultat sera alors un grand nombre de solutions qui seront présentées à l'utilisateur pour qu'il décide de la solution de compromis. Cette combinaison forme alors une méthode à préférence a posteriori.

1.7 Méthodes classiques de résolution

1.7.1 Méthodes de résolution exacte

Il y a très peu de travaux sur les méthodes exactes dans le contexte de la résolution des problèmes d'optimisation multi-objectif,sans doute, à cause de la grande difficulté de ce type de problème.
Les références existantes et qui présentent la plupart des méthodes exactes sont Ulungu [73] et Ehrgott et Gandibleux 2000 [22] .
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.

L'agrégation linéaire

Cette méthode populaire transforme le problème multi-objectif en un problème mono-objectif en combinant linéairement les différents objectifs. Ainsi, le nouveau problème obtenu, car il s'agit alors d'un problème différent, consiste à optimiser $\Sigma_i \lambda_i f_i$.

Le théorème de **Geoffrion**[38] indique qu'en utilisant différentes valeurs pour le vecteur λ, il est possible d'obtenir toutes les solutions supportées du problème multi-objectif initial. Par contre, aucune solution non supportée peut être trouvée par cette méthode.
La méthode d'agrégation linéaire a donc ses limites. Toutefois, elle est intéressante pour des problèmes ayant de nombreux objectifs et/ou un grand nombre de solutions supportées bien réparties. Dans ce contexte, il peut être suffisant de générer les solutions supportées.

La recherche dichotomique

La recherche dichotomique offre un schéma d'application de l'agrégation linéaire permettant d'obtenir les solutions supportées.Elle est initialement proposée par Aneja et Nair [77] pour la résolution exacte du problème de transport bi-objectif.
Elle consiste à trouver les poids $\lambda \in \mathbb{R}^2_>$ qui permettent d'obtenir tous les points supportés extrêmes.
Cette méthode peut être appliquée à n'importe quel problème puisque elle n'utilise aucun caractéristique du problème de transport. La méthode consiste,initialement à déterminer pour chaque objectif la solution optimale associée x^1 etx^2.
Soit S la liste des solutions efficaces supportées trouvées ; cette liste est initialisé par les deux solution optimales x^1 etx^2 respectivement des deux critères Z_1 et Z_2, où leurs valeurs dans l'espace des critères sont$y^1 = (Z_1^1, Z_2^1)$et $y^2 = (Z_1^2, Z_2^2)$.
Les solutions de S sont rangées par ordre croissant du critère Z_1.
Soient x^r et x^s deux solutions consécutives de S.
Une itération de cet algorithme consiste à résoudre un problème $P_\lambda$défini par :
$\lambda_1 = Z_2^r - Z_2^s$ et $\lambda_2 = Z_1^s - Z_1^r$
, où y^r et y^s sont deux points consécutifs avec $Z_1^r < Z_1^s$.
Le vecteur λ correspond à la normale à la droite joignant les y^r et y^s (voir figure 1.5)
Le point y^t obtenu par la résolution de problème paramétrique P_λ est situé entre les deux points y^r et y^s selon deux situations :

1. Si $\lambda.y^t > \lambda.y^r$, alors deux nouveaux problèmes P_λ définis par y^r et y^t, et y^t et y^s doivent être résolus (figure 1.6) ;

2. Si $\lambda.y^t = \lambda.y^r$, la recherche s'arrête.

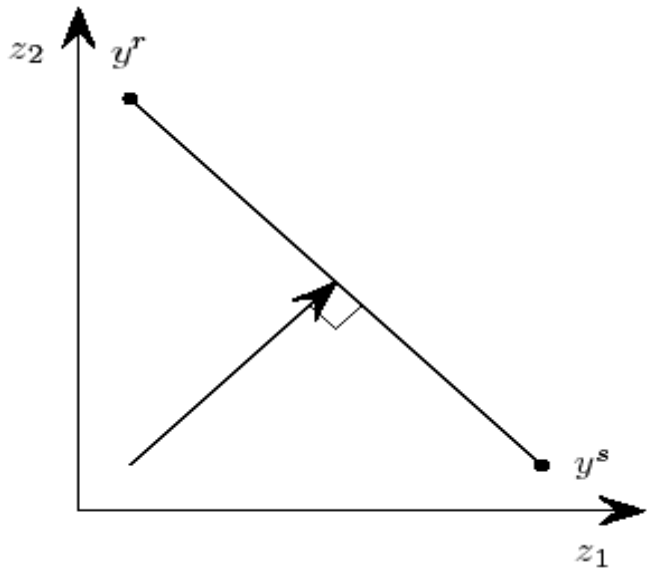

FIG. 1.5 – Problème P défini par y^r et y^s.

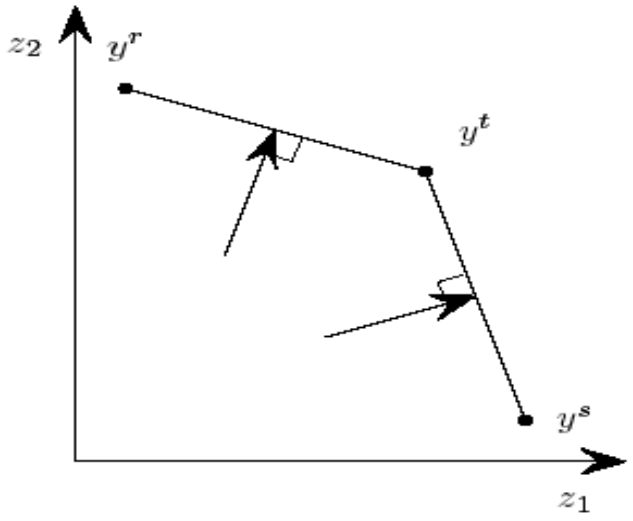

FIG. 1.6 – Un nouveau point supporté est trouvé, la dichotomie continue.

Cette méthode est restée limitée au cas bi-objectif jusqu'à très récemment. Dans ses travaux de thèse, **Przybylski** [53] l'a étendue à trois objectifs et plus en proposant une autre façon

de calculer les poids.
Cette procédure sera détaillée dans le chapitre 4.

Méthode ϵ contrainte

Cette méthode permet de transformer le problème d'optimisation multiobjectif 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[14].
La démarche est la suivante :

- Nous choisissons un critère à optimiser prioritairement
- Nous transformons le problème conservant l'objectif prioritaire et nous nous transformons les autres objectifs en des contraintes d'inégalité

Le problème peut être reformulé de la manière suivante :

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

L'approche par ϵ -contrainte doit aussi être appliquée plusieurs fois en faisant varier le vecteurϵ pour trouver un ensemble de points Pareto optimaux.
Cette approche a l'avantage par rapport à la précédente de ne pas être trompée par les problèmes non convexes.
Ainsi la figure 1.7 illustre, en dimension 2, le cas où un point$(\epsilon; f_{1min})$, de la partie non convexe, est trouvé. La figure 1.7 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 vecteurϵ doit être choisi judicieusement.
Il est clair qu'une bonne connaissance du problème **a priori** est requise.

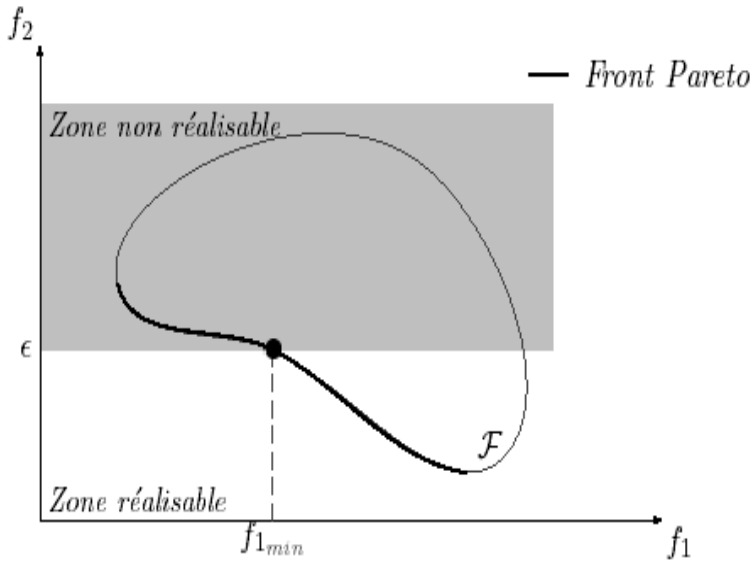

FIG. 1.7 – Interprétation graphique de la méthode

Algorithme en deux phases

La méthode en deux-phases a initialement été proposée par Ulungu et Teghem pour la résolution d'un problème d'affectation bi-objectif [74].
Cette méthode consiste à séparer la détermination des solutions efficaces X_E en deux phases où la première phase cherche à calculer toutes les solutions supportées X_{SE} et la deuxième consiste à déterminer les solutions efficaces restantes X_{NE} (solutions non supportées), et de réutiliser des algorithmes efficaces pour le cas mono-objectif quand il en existe.
Ces algorithmes ayant été conçus pour profiter pleinement de la structure du problème, toute modification de cette structure empêche leur utilisation. C'est pourquoi on s'interdit avec cette méthode d'ajouter des contraintes sur les valeurs des fonctions objectifs.

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 ont obtenu à partir d'une résolution d'une certaine combinaison linéaire des objectifs en utilisant, généralement la variante de Aneja et Nair 1.7.1 [77].

2. **Deuxième phase**
   La deuxième phase consiste alors à la recherche des solutions efficaces non supportées.
   Les solutions trouvées dans la première phase sont utilisées pour réduire l'espace de recherche et déterminer les solutions non supportées.
   Dans le cas bi objectif Ulungu et Teghem proposent alors d'utiliser les solutions supportées trouvées à l'issue de la première phase,pour réduire l'espace de recherche. Pour ce faire, ils se sont basés sur un argument géométrique qu'ils ont établi théoriquement et dont le principe est le suivant : les solutions non-supportées sont forcément dans les triangles rectangulaires basés sur deux solutions supportées consécutives (voir figure 1.8) Ainsi, une recherche de type deuxième phase est exécutée entre chaque couple de solutions supportées adjacentes (voir figure 1.9,1.10).
   À la fin de la deuxième phase, toutes les solutions efficaces sont trouvées.
   La méthode de recherche adoptée durant la deuxième phase dépend du problème étudié. En général, la deuxième phase utilise un schéma de recherche arborescente par séparation et évaluation (Branch and Bound)[24] [Ehrgott et Gandibleux, 2000] en exploitant la structure particulière du problème à résoudre pour rendre la recherche plus efficace.

   La méthode à deux phases semble intéressante dans le sens où elle propose un schéma général pour la résolution exacte des problèmes bi-objectifs de type MOCO. Toutefois, l'application de cette méthode à un problème bi-objectif donné nécessite l'existence d'un algorithme efficace pour la version mono-objectif de ce problème.

Il est à noter que la méthode à deux phases a été appliquée avec succès au problème d'affectation bicritère [Ulungu et Teghem, 1995][75]. Des améliorations de cette méthode ont été proposées et appliquées sur le problème d'affectation bicritère [Przybylski, Gandibleux, et Ehrgott, 2008][65] et celui de flowshop bicritère [Lemesre, Dhaenens, et Talbi, 2007] [55]. Néanmoins, ces améliorations restent moins générales que la méthode originale puisqu'elles tiennent amplement compte des spécificités des problèmes étudiés.

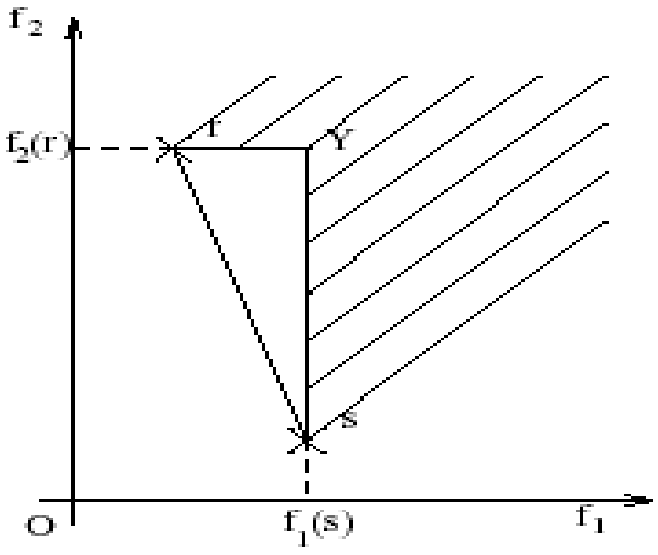

FIG. 1.8 – a

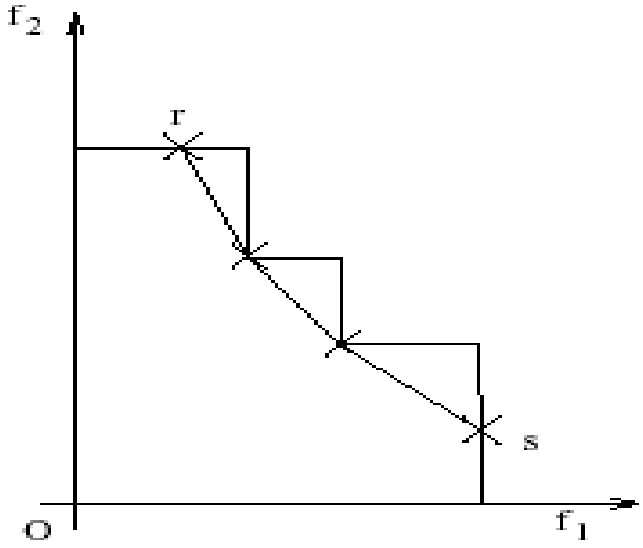

FIG. 1.9 – b

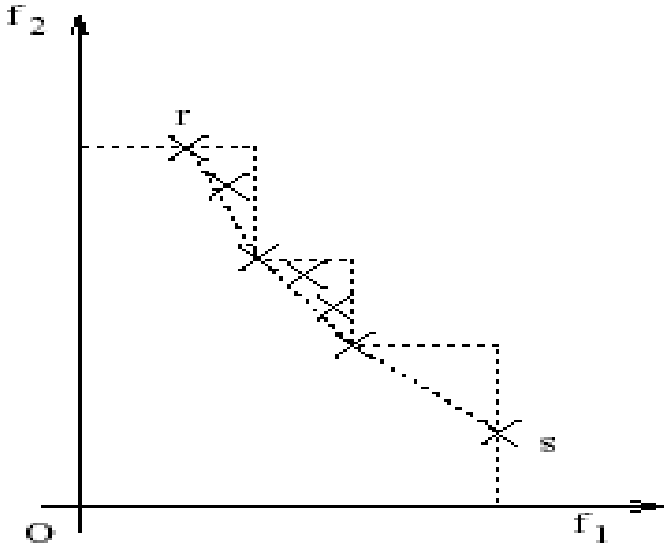

FIG. 1.10 – c

1.7.2 Méta-heuristiques multiobjectif

Du fait que tous les algorithmes exacts sous-entendent une énumération explicite ou implicite de toutes les solutions, de tels algorithmes deviennent alors inefficaces dès que la taille des problèmes (ex., nombre de variables, nombre de fonctions objectifs, nombre de contraintes) augmente de façon considérable. En outre, une grande partie des problèmes de type MOCO sont NP-difficiles [Papadimitriou et Steiglitz, 1982] [Ehrgott, 2005] dans le sens où il n'existe pas (jusqu'à présent !) d'algorithme(s) exact(s) capable(s) de les résoudre en un temps polynomial en fonction de leur taille. Pour illustrer de manière pratique cette problématique, considérons le fameux problème de voyageur de commerce (Traveling Salesman Problem ou TSP).

Ainsi, afin de procurer rapidement des solutions de bonne qualité mais qui ne sont pas nécessairement Pareto-optimales, des méthodes approchées (i.e. ; heuristiques et métaheuristiques) ont été mises au point.

Ces trois dernières décennies ont connu des développements et des améliorations fulgurents des méthodes de résolution approximatives, appelées habituellement "heuristiques et métaheuristiques".

Une heuristique est définie par [66], comme une technique qui détermine de bonnes solutions(i.e proche de l'optimal)en un temps de calcul raisonnable sans pour autant guarantir la réalisabilitéou l'optimalité ou même dans de nombreux cas connaître sa proximité de la solution optimale. La plupart d'entre elles sont conçues pour des problèmes spécifiques et ne peuvent être utilisées pour résoudre d'autres types de problèmes.Au contrire, les métaheuristiques sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile.Contrairement aux méthodes exactes, les méthodes métaheuristiques ne garantissent pas de trouver de manière exacte l'ensemble des solutions Pareto optimales mais une approximation aussi bonne que possible de cet ensemble dans un temps acceptable. Cet approximation est également appelée ensemble de solutions potentiellement efficaces.

Les métaheuristiques ont été appliquées avec succées sur un grand nombre de problèmes académiques et réels : problème d'affectation quadratique, coloriage de graphe, voyageur de commerce,$\cdots$. Le lecteur intéressé peut consulter [13] pour une présentation de quelques applications importantes.

Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. 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.
Ehrgott et Gandibleux [25] classent ces méthodes essentiellement en deux classes :les algorithmes évolutionnaires (avec plusieurs solutions) et les algorithmes d'exploration de voisinage (à solution unique).
Des explications détaillées sur les nombreuses variantes de ces algorithmes sont données dans [26].

Les algorithmes d'exploration de voisinage (à solution unique)

Le principe général le plus utilisé dans les méthodes métaheuristiques est celui de voisinage ; cette dernière est un impact important sur le comportement des métaheuristiques pour les problèmes combinatoires ; tel que à chaque solution d'un problème, nous associons un sous ensemble de solutions. Ce sous ensemble qu'il est possible d'atteindre par une série de transformations de données . Par extension nous désignons par le terme " voisinage" l'ensemble des transformations considérées .
Cette méthode démare avec une solution initiale et une direction de recherche,se traduisant par un poids $\lambda \in R^p_>$. La procédure détermine une approximation des points non dominés corespondants à la valeur données par λ.
Un mécanisme **local** de pondération des objectifs guide la recherche sur une partie de la frontière efficace Y_N. La convergence est donc locale. Ce principe est répété pour plusieurs directions de recherche pour obtenir une approximation complète de Y_N. Ce type d'algorithme a la caractéristique de converger rapidement grâce à l'effort économisé dans la dispersion. Cependant, cette économie rend plus difficile la couverture de Y_N.¨

La Recherche Tabou

La recherche Tabou (Tabu Search) est introduite par Glover [40]. 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 amémoriser 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 diversiffication. 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'intensi fier 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 diversi fication 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. noter que ces dernières années, la recherche Tabou a été appliquée avec succés à un certain nombre de problèmes d'optimisation Combinatoire [41],[3]. Un large éventail d'applications de cette méthode, se retrouve dans [42].

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. Pour être clair, nous essayons de décrire le fonctionnement de l'algorithme du recuit simulé de la façon la plus générale et la plus simple possible. Désignons par F la fonction évaluant la qualité d'une solution donnée (elle correspond souvent à une agrégation linéaire des fonctions objectifs) et par A l'archive des solutions potentiellement non dominées. Alors, le fonctionnement général d'un algorithme de recuit simulé multiobjectifs est le suivant :

– Choisir aléatoirement une solution initiale x.

– Calculer une solution voisine de x ; soit y.

– Calculer l'ecart de qualite entre y et x :

$$\triangle F = F(y) - F(x)$$

– Si $\triangle F \geq 0$, alors remplacer x par y. Sinon, pour accepter (ou ne pas accepter) y, utiliser la probabilite d'acceptation suivante :

$$P = \exp^{\frac{\triangle F}{T}}$$

– Diminuer la temperature T au fur et a mesure du deroulement de l'algorithme.

– Mettre à jour A vis-a-vis de la nouvelle solution retenue.

– Répéter les étapes de 2 à 6 tant que la temperature T n'a pas atteint un seuil minimal fixe à l'avance.

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 [69],[33]. Les deux méthodes les plus populaires sont la méthode MOSA (Multiple Objective Simulated Annealing) proposée par Ulungu et al. [76] 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.

Les algorithmes évolutionnaires(à solution multiple)

On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les algorithmes génétiques [Holland, 1975 ; Goldberg, 1989], les stratégies d'évolution [Schwefel, 1981] et la programmation évolutive [Fogel, 2000]. Ces méthodes se distinguent par la manière de représenter l'information et par la façon de faire évoluer la population d'une génération à l'autre. Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :

- une ***population*** constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné,
- un ***mécanisme d'évaluation des individus*** permettant de mesurer l'adaptation de l'individu à son environnement,
- un ***mécanisme d'évolution de la population*** permettant, grâce à des opérateurs predéfinés, d'éliminer certains individus et d'en créer de nouveaux.

Parmi les composants d'un algorithme évolutionnaire, l'***individu*** et la ***fonction d'évaluation*** correspondent respectivement à la notion de configuration et à la fonction d'évaluation dans les méthodes de voisinage. Le mécanisme d'évolution est composé de plusieurs opérateurs tels que la sélection, la mutation et le croisement.

La ***sélection*** a pour objectif de sélectionner des individus qui vont pouvoir se reproduire pour transmettre leurs caractéristiques à la génération suivante. Le ***croisement*** ou recombinaison est un opérateur permettant de construire de nouveaux individus enfants à partir des caractéristiques d'individus parents sélectionnés. La ***mutation*** effectue des légères modifications de certains individus. Comme exemple des algorithmes évolutionnaires, nous présentons maintenant les algorithmes génétiques [8].

Algorithmes génétiques

Les algorithmes évolutionnaires sont parmi les métaheuristiques à base de population. Ils sont inspirés de la biologie et introduisent le théorème de Darwin dans l'évolution des espèces. Les algorithmes génétiques (AGs) ont été introduits par Holland [Holland, 1975 [49]] comme un modèle de méthode adaptative. Ils s'appuient sur un codage de l'information sous forme de chaînes 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 [8].

Récemment, beaucoup de recherches ont été menées sur l'application des algorithmes évolutionnaires aux problèmes d'optimisation multiobjectif. Celles-ci ont permis de mettre en avant l'intérêt d'utiliser des méthodes d'optimisation basées sur le concept de population. Des exemples d'algorithmes évolutionnaires sont donnés par VEGA (Vector Evaluated Genetic Algorithm [18] , MOGA (Multiple Objective Genetic Algorithm [60]),(NSGA) (non dominated sorting genetic algorithm [70]),SPEA(Strength Pareto evolutionary algorithm [79])ou encore (M-PAES(memetic Pareto archived evolution strategy algorithm [15])

1.7.3 Méthodes hybrides de résolution des problèmes d'OCMO

Une autre façon d'améliorer les performances d'un algorithme consiste à le combiner avec une autre méthode [71].Ce principe général, appelé hybridation, peut s'appliquer pour un grand nombre de méthodes.Une multitude d'algorithmes hybrides ont fait leur apparition ces dernières années.Nous pouvons ainsi citer pour le cas des problèmes multiobjectif : La méthode MOTS([47],[36]) combinant une population et une recherche Tabou, la méthode PSA [16]combinant un algorithme génétique et le récuit simulé, la méthode M-PAES [15]intégrant un schéma généralisant l'implémentation d'un grand nombre d'algorithmes hybrides pour l'optimisation des problèmes multiobjectif. Un cas particulier de l'hybridation entre deux méthodes consiste à combiner une méthode de recherche locale et une méthode exacte : La méthode du recuit simulé hybride [45] appliquée au problème "des enchères combinatoires "(à un seul objectif) combine la métaheuristique du récuit simulé avec une méthode exacte du type "branch and bound".Suivant la manière dont elle effectuée cette combinaison, la méthode hybride considérée donnera lieu soit à une méthode exacte soit à une méthode approchée.

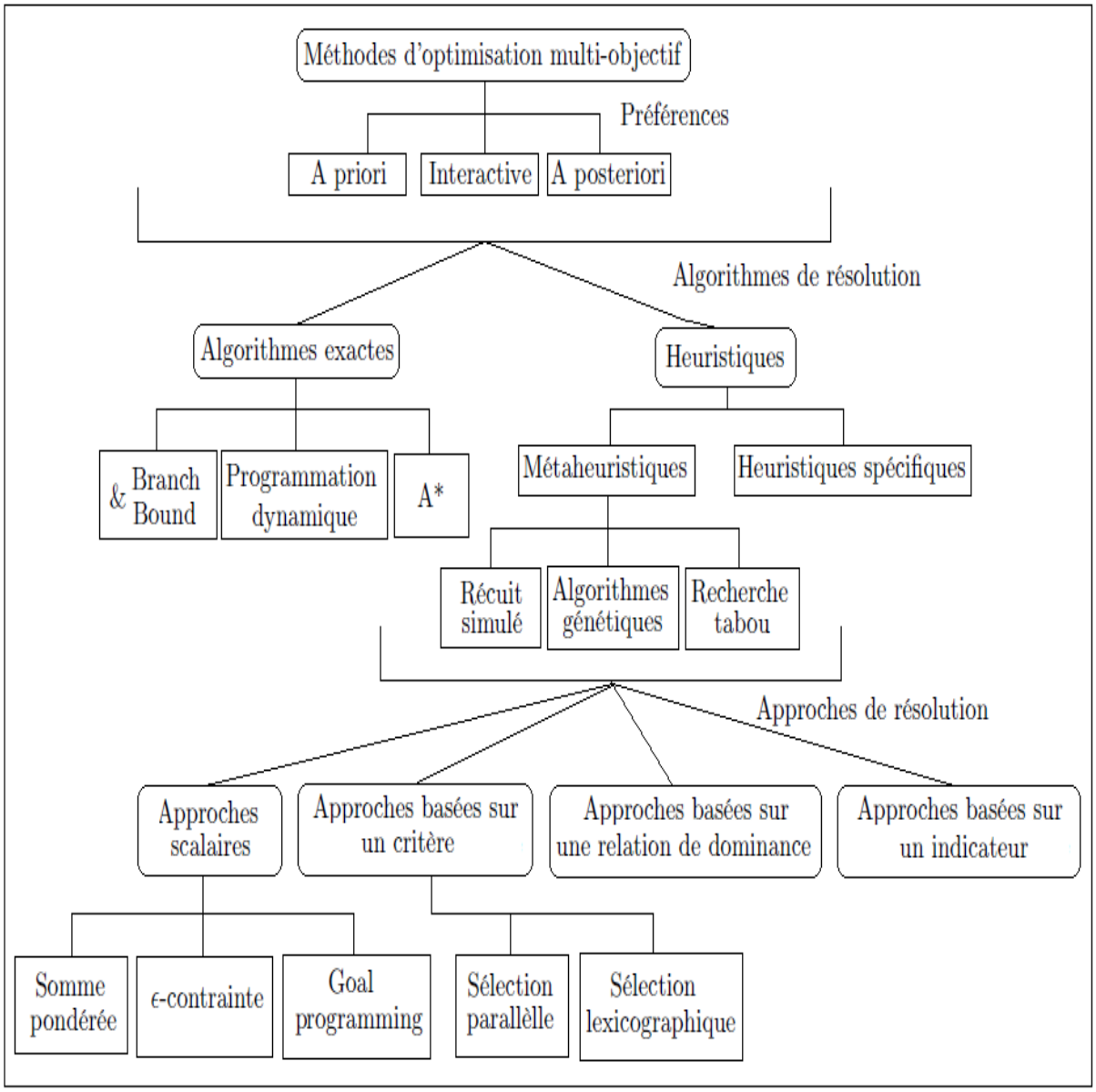

FIG. 1.11 – Classification des approches d'optimisation multi-objectif.

1.8 Conclusion

Dans ce chapitre nous avons donné un aperçu sur les problèmes d'optimisation multiobjectif, la complicité de ces problèmes ainsi les méthodes de résolution proposées.
Dans le chapitre suivant nous allons focaliser notre étude sur un cas particulier de ces problèmes, qui est le problème de sac à dos multiobjectif.

2

Problèmes de type sac à dos

2.1 Introduction

Le problème de sac à dos (ou knapsack problem (KP)) est un problème classique d'optimisation appartenant à la classe des problèmes NP-difficiles.
Le problème est de remplir un sac à dos supportant un poids maximum c avec des objets ayant un poids w_j,pour $j = 1, \cdots, n$, et un indice de satisfaction de telle sorte que la satisfaction totale (fonction objectif) $Z(x)$ soit maximale.
La question qui se pose est de savoir quels objets doit-on mettre dans le sac ?
Ce problème intervient comme un sous-problème dans plusieurs problèmes, par exemple, les problèmes de découpe (voir Gilomre et Gomory [8]) lors de la génération d'un pivot du simplexe.
Sous le terme de "sac-'a-dos" sont regroupées différentes variantes qui, 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 les diverses formes de problèmes, à savoir sac à dos mono

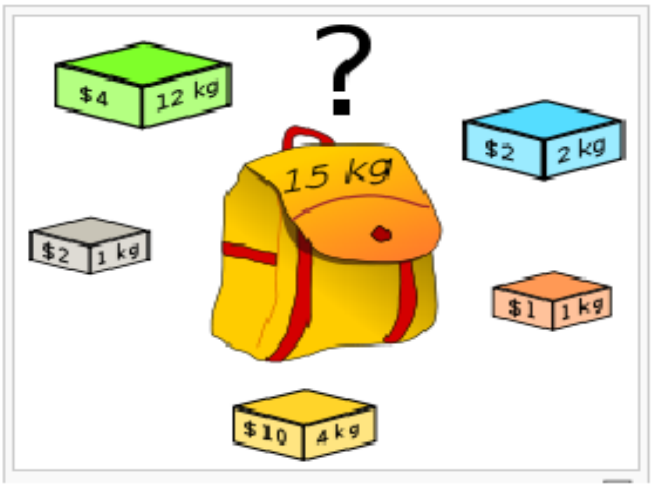

FIG. 2.1 – problème du sac à dos

objectif unidimensionnel, borné et non borné multidimensionnel, sac à dos multiobjectif et sac à dos multiobjectif multidimensionnel, ainsi que l'essentiel des approches de résolution existantes.

2.2 Le problème de sac à dos (knapsack problem (KP))

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

$$(01-KP)\begin{cases} \max Z(x) = \sum_{i=1}^{n} c_i x_i \quad c_i \in \mathbb{N}^* \\ \text{tq :} \\ \sum_{i=1}^{n} w_i x_i \leq w \quad w, w_i \in \mathbb{N}^* \\ x_i \in \{0,1\} \forall i \in \{1, \cdots, n\} \end{cases}$$

où c_i est le profit apporté par la sélection de l'objet i et w_i est son poids.
La contrainte exprime le fait que les objets sélectionnés doivent tenir dans la capacité w du sac à dos.
Si un objet i est sélectionné alors la variable x_i associé prend la valeur 1 sinon elle prend la valeur 0 .

2.2.1 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 multidimensionnel), le nombre de fonction objectif (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. Cette section présente quelques unes d'entre elles.
Le lecteur désireux de connaître plus de détails sur ces variantes ou sur d'autres pourra se référer à [78],[23].

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 2.2.1. (**Efficacité d'un objet**)
On appelle efficacité d'un objet i le rapport de son coût sur son poids, noté $e_i = \frac{c_i}{w_i}$.

Définition 2.2.2. (**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 :
$s = \min\{k : \sum_{i=1}^{k} w_i > w\}$, pour les e_i triés en ordre décroissant.

En suivant l'ordre décroissant des e_i la solution optimale du (LKP) se présente comme suit [56] :

$$x^* := \begin{cases} 1 & Si \quad i = \overline{1, s-1} \\ \frac{w - \sum_{i=1}^{k-1} w_i}{w_s} & si \quad i = s \\ 0 & sinon \end{cases}$$

D'où : $z^*(LKP) = \sum_{i=1}^{s-1} c_i + (w - \sum_{i=1}^{s-1} w_i)\frac{c_s}{w_s}$.

Le problème (LKP) est un problème relaxé de $01KP$
Ainsi, si I est une instance de ce dernier et I^{LP} est l'instance obtenue à partir de I en remplaçant la contrainte $x_i \in \{0,1\}, \forall i \in \{1, \cdots, n\}$ par $x_i \in [0;1], \forall i \in \{1, \cdots, n\}$, alors l'inégalité $z^*(I) \leq z^*(I^{LP})$ se vérifie, où $z^*(I)$ est la valeur optimale des solutions d'une instance I. De plus, LKP appartient à la classe de complexité P.
Pour cela, il est souvent utilisé dans la résolution des variantes à variables entières, dans l'optique d'obtenir une borne supérieure de la solution optimale en un temps raisonnable.

Sac à dos à variables bornées

Le problème défini précédement peut être généralisé en assignant pour chaque objet i une borne b_i à ne pas dépasser ($b_i \leq \frac{w}{w_i}$), nous obtenons donc le problème de sac à dos à variables bornées dont la formulation mathématique est la suivante :

$$\begin{cases} Maximiser Z(x) = \sum_{i=1}^{n} c_i x_i \\ \text{s.c :} \\ \sum_{n}^{i=1} w_i x_i \leq w \\ 0 \leq x_i \leq b_i \qquad i = 1, \cdots, n \\ \quad x_j \quad \text{entier}, \qquad i = 1, \cdots, n \end{cases}$$

Dans le cas où $b_i = +\infty$ le problème s'appelle Sac à dos à variables non bornées (unbounded Knapsack problem). Et pour $b_i = c_i$, ce problème est intitulé "sub-set -sum problem" ou bien la somme des sous ensembles.
Un problème de sac à dos à variables non bornées très particulier est consideré lors que, $c_j = 1, j = 1. \cdots, n$ et en imposant l'égalité pour la contrainte capacité $\sum_{i=1}^{n} w_j x_j = w$, ce problème s'appelle "Change-making problem", cependant ce problème peut être avoir

une extension à variables non bornées (Unbounded change-making problem).

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 peut être considéré comme un programme linéaire en nombres entiers (ILP) classique sous la seule restriction que les coefficients soient positifs et les variables binaires.
Son champ d'application est large, ce qui a grandement contribué à sa popularité.

$$(d-KP)\begin{cases} Maximiser Z(x) = \sum_{i=1}^{n} c_i x_i \qquad c_i \in \mathbb{N}^* \\ \text{s.c :} \\ \sum_{n}^{i=1} w_i^j x_i \leq w_j \quad w_j, w_i \in \mathbb{N}^{d\geq}, \quad j \in \{1, \cdots, d\} \\ \quad x_i \in \{0,1\} \qquad \forall i \in \{1, \cdots, n\} \end{cases}$$

Comme le souligne un récent état de l'art de Fréville [30], cette variante a été intensément utilisée pour modéliser des situations de gestion de portefeuilles ou de sélection de projets. Il est, de plus, largement utilisé dans l'évaluation de méta-heuristiques [1],en particulier pour la recherche tabou et les algorithmes génétiques. Cette variante est intrinsèquement difficile. Ainsi, les méthodes de résolution exacte actuelles se limitent à des instances de quelques centaines de variables pour une dizaine de contraintes.
Ainsi, la meilleure solution pour résoudre une instance de d - KP aujourd'hui consiste en l'utilisation d'heuristiques [32] ou de méta-heuristiques [12],[54][59].

Sac à dos multiple

Le problème de sac à dos multiple (MKP) consiste à répartir un ensemble d'objets dans plusieurs sacs à dos de capacité différentes en respectant les contraintes de capacité de chaque sac et en maximisant le profit total. Une application possible du problème (MKP) est le chargement de cargos (Eilon et Christofides [EC71]).
Ce problème est aussi un cas particulier du problème d'affectation généralisée (PAG).
Le problème (MKP)peut être modélisé par le programme linéaire suivant :

$$(MKP)\begin{cases} Maximiser Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_i x_{ij} \qquad c_i \in \mathbb{N}^* \\ \text{s.c :} \\ \sum_{n}^{i=1} w_i x_{ij} \leq w_j \quad w_j, w_i \in \mathbb{N}^{d\geq}, \quad j \in \{1, \cdots, m\} \\ \sum_{j=1}^{m} x_{ij} \leq 1 \qquad \forall i \in \{1, \cdots, n\} \\ \quad x_i \in \{0,1\} \qquad \forall i \in \{1, \cdots, n\}, j \in \{1, \cdots, m\} \end{cases}$$

Sac à dos multiobjectif

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

$$(01-MOKP)\begin{cases} \max Z_j(x) = \sum_{i=1}^{n} c_i^j x_i \quad j \in \{1, \cdots, p\}, c_i \in \mathbb{N}^p \\ \text{tq :} \\ \sum_{i=1}^{n} w_i x_i \leq w \quad w, w_i \in \mathbb{N}^* \\ x_i \in \{0,1\} \forall i \in \{1, \cdots, n\} \end{cases}$$

2.2.2 Méthodes de résolution

Le problème de sac à dos (KP) a été étudié pour la première fois à la fin des année 50. Dés cette date un grand nombre d'algorithmes et de techniques a été proposé. Peu d'algorithmes exactes pour le problème (KP) existe dans la littérature [5] . L'algorithme le plus connu est de séparation et évaluation qui été proposé par Martello et Toth [27], comme nous trouvons les algorithmes de programmation dynamique.
Le problème de sac à dos mono objectif peut être aussi résolu d'une manière heuristique pour s'approcher de la solution optimale. Une heuristique typique pour ce problème est l'algorithme Glouton qui peut résoudre des problèmes en un temps polynomial.

Méthodes de résolution exacte

- **La procédure de séparation et évaluation**
  Le concept général de ces méthodes se base sur une structure arborescente de recherche de solutions. Chaque nœud de l'arborescence sépare l'espace de recherche en deux sous-espaces et de proche en proche, jusqu' à l'exploration totale de l'espace des solutions.
  Développer tout l'espace de recherche, consiste pour une instance de n éléments, à développer 2^n vecteurs binaires de taille n. Ce qui est excessivement lourd et irréalisable au regard du temps d'exécution nécessaire à l'exploration de toute l'arborescence.
  Les méthodes de séparation et évaluation se basent, en général, sur les principes suivants mais diffèrent entre elles par le choix de stratégies à opérer pour la séparation, le développement de l'arborescence et l'utilisation de la fonction d'évaluation :

  - **La partie séparation** :
    ce principe s'effectue au niveau de l'instance du sac-à-dos. L'élément de séparation peut être pris selon un ordre prédéfini ou selon un élément défini pendant l'exploration.
    Le choix de l'élément de séparation est important pour améliorer les performances de l'algorithme.

  - **Développement** :
    ce principe se fait selon une stratégie de développement qui peut être en profondeur ou par le meilleur d'abord.

  - **La partie évaluation** :
    consiste à calculer la borne inférieure $\underline{Z}$ et la borne supérieure $\overline{Z}$ pour un ensemble X'
    Nous pouvons prendre comme une borne inférieure la valeur de la meilleure solution trouvée jusqu'à cet instant de la recherche, ou la valeur d'une solution obtenue avec une heuristique.
    L'utilisation d'une borne supérieure permet de fermer certaines régions de l'espace à explorer.

Remarque 2.2.1. :
Une bonne borne supérieure doit être rapide à calculer tout en étant serrée, c'est à dire proche de la valeur de la meilleure solution de X'.
Cependant, ces deux qualités s'opposent en pratique.

Si i est l'objet le plus efficace (définition 2.2.1, la borne supérieure la plus simple pour $01KP$ peut être obtenue en relâchant la contrainte d'intégrité $x_i \in \{0,1\}$ et $x_i \in \mathbb{R}$.
Cela donne la borne triviale suivante :

$$U_0 = \lfloor \frac{w}{w_i} c_i \rfloor$$

Une autre manière de calculer une borne supérieure est de relacher la contrainte d'intégrité sur toutes les variables, de manière à obtenir le problème LKP 2.2.1.

$$U_1 = \lfloor z^*(LKP) \rfloor$$

Cette borne est plus serrée que U_0 et peut se calculer en $O(n)$ si les objets ont été initialement triés en ordre décroissant d'efficacité.
Si nous considérons séparément le cas ou l'objet bloquant s est ou n'est pas ajouté à la solution,nous obtenons une troisième borne supérieure définie par Martello et Toth [56] :

– Si s est **exclu** :

$$x_i^0 = \begin{cases} 1 & i = 1, \cdots, s-1 \\ \frac{w - \sum_{i=1}^{s-1} w_i}{w_{s+1}} & i = s+1 \\ 0 & sinon \end{cases}$$

– Si s est **inclus** :

$$x_i^1 = \begin{cases} 1 & i = 1, \cdots, s-2, s \\ \frac{w - \sum_{i=1}^{s-2} w_i - w_s}{w_{s-1}} & i = s-1 \\ 0 & sinon \end{cases}$$

$$U_2 = \sum_{i=1}^{s-1} c_i + \max \lfloor \frac{\overline{w}}{w_{s+1}} c_{s+1} \rfloor, \lfloor c_s (\frac{w_s - \overline{w}}{w_{s-1}}) c_{s-1} \rfloor$$

où s est l'indice de l'élément bloquant et $\overline{w} = w - \sum_{i=1}^{s-1}$.
Cette borne est plus fine que U_1 et il est montré que
$U_2 \leq U_1 \leq U_0$.

Procédure pse($\updownarrow x, \downarrow i, \updownarrow x^*$)
Paramètre $\updownarrow x$: la solution en cours de construction.
Paramètre $\downarrow i$: l'indice de l'objet sur lequel opérer.
Paramètre $\updownarrow x^*$: la meilleure solution trouvée.

- - La solution est-elle réalisable ? - -
1: **si** $\sum_{j=1}^{i-1} w_j x_j \leq \omega$ **alors**
- - Mise à jour de la meilleure solution connue. - -
2: **si** $z(x) > z(x^*)$ **alors**
3: $x^* \leftarrow x$
4: **fin si**
5: **si** $i \leq n$ **alors**
6: calculer une borne supérieure $\bar{z}$
- - Séparation de la recherche sur les sous-espaces disjoints vérifiant $x_i = 1$ ou $x_i = 0$. - -
7: **si** $\bar{z} > z(x^*)$ **alors**
8: $x_i \leftarrow 1$
9: pse$(x, i+1, x^*)$
10: $x_i \leftarrow 0$
11: pse$(x, i+1, x^*)$
12: **fin si**
13: **fin si**
14: **fin si**

FIG. 2.2 – Une procédure de séparation et évaluation pour(01KP)

– **La programmmation dynamique**

La Programmation Dynamique est une méthode exacte de résolution de problèmes d'optimisation séquentielle, due essentiellement à R. Bellman (1957). Bien que très puissante,son cadre d'application est relativement restreint, dans la mesure où les problèmes qu'elle adresse doivent vérifier le principe d'optimalité, ce qui est le cas pour le sac à dos.

Étant donné une instance du problème de sac à dos, on considère le problème restreint à ses j premiers objets et à une capacité d, défini comme suit :

$$KP(j,d) = \begin{cases} \max Z_{j,d} = \sum_{i=1}^{j} c_i x_i \\ \text{tq :} \\ \sum_{i=1}^{j} w_i x_i \leq d \\ x_i \in \{0,1\} \forall i \in \{1, \cdots, j\} \end{cases}$$

Si $z_{j-1,d}$ est connu pour toutes les capacités $d = 0, ..., w$, alors la valeur de $z_{j,d}$ est donnée par la formule suivante [10] :

$$z_{j,d} = \begin{cases} z_{j-1,d} & \text{si} \quad d < w_j \\ \max\{z_{j-1,d}, z_{j-1,d-wj} + cj\} & \text{si} \quad d \geq wj \end{cases}$$

Le premier cas correspond à la situation dans laquelle le sac à dos est trop petit pour contenir l'objet j. Par conséquent, cet objet ne change pas la valeur optimale $z_{j-1,d}$. Si l'objet j peut être contenu dans le sac, ce qui correspond au second cas, il y a deux possibilités : soit l'objet n'appartient pas à la solution optimale, auquel cas la valeur $z_{j-1,d}$ est conservée, soit l'objet appartient à la solution optimale et contribue de c_j à sa valeur mais ne laisse qu'une capacité de $d - w_j$ pour les objets dans $\{1, \cdots, j-1\}$. Évidemment, cette capacité restante pour les objets précédents doit être complétée avec un profit maximum, ce qui est la valeur optimale $z_{j-1,d-w_j}$ du sous problème $KP(j-1, d-w_j)$.

En initialisant $z_{0,d} \longleftarrow 0$ pour $d \in 0, \cdots, w$, le calcul des valeurs $z_{j,d}$ par récursion de $j = 1n$ dans l'algorithme amène à la valeur $z_{n,w}$ des solutions optimales. Cet algorithme ne calcule pas le vecteur correspondant à une solution optimale. Néanmoins, il peut être déduit a posteriori si la table des valeurs $z_{i,d}$ est conservée.

De nombreuses contributions ont été faites pour améliorer la complexité de la résolution de KP par programmation dynamique [50],[72] ou, plus récemment, [29],[63].

Procédure programmation_dynamique()
Sortie La performance des solutions optimales.

1: **pour** $d \in \{0,\ldots,\omega\}$ **faire**
2: $z_{0,d} \leftarrow 0$
3: **fin pour**
4: **pour** j **de** 1 **à** n **faire**
5: **pour** d **de** 0 **à** ω **faire**
6: **si** $d \geq w_j$ **alors**
7: $z_{j,d} \leftarrow \max\{z_{j-1,d}, z_{j-1,d-w_j} + c_j\}$
8: **sinon**
9: $z_{j,d} \leftarrow z_{j-1,d}$
10: **fin si**
11: **fin pour**
12: **fin pour**
13: **retourner** $z_{n,\omega}$

FIG. 2.3 – Programmation dynamique pour (01KP)

Méthodes de résolution approchée

Parmi les méthodes heuristiques pour la résolution du problème $01KP$, on peut citer la méthode dite gloutonne. Un algorithme glouton construit une solution de manière incrémentale, en faisant à chaque pas un choix maximisant une fonction objectif.
Tout d'abord, on ordonne les objets selon l'ordre décroissant du rapport profit par poids (efficacité (définition 2.2.1)).
L'agorithme Glouton consiste donc à sélectionner à chaque étape un élément selon l'ordre précédemment défini. Si l'élément est admissible, c'est-à-dire si son poids ne dépasse pas la capacité restante aprés fixation des autres éléments, alors, il est mis dans le sac sinon, on sélectionne l'élément qui se situe juste après et qui peut être admissible et ainsi de suite de proche en proche jusqu'à épuisement de tous les objets pouvant être mis dans le sac.

Procédure glouton($\updownarrow x$)
Paramètre $\updownarrow x$: la solution construite par le glouton.

1: trier les objets par ordre décroissant d'efficacité
2: $\bar{\omega} \leftarrow \omega$
3: **pour** i **de** 1 **à** n **faire**
4: **si** $w_i \leq \bar{\omega}$ **alors**
5: $x_i \leftarrow 1$
6: $\bar{\omega} \leftarrow \bar{\omega} - w_i$
7: **sinon**
8: $x_i \leftarrow 0$
9: **fin si**
10: **fin pour**

FIG. 2.4 – Algorithme glouton pour(01KP)

2.3 Conclusion

Dans ce chapitre, nous avons présenté une étude sur le problème de Sac-à-dos, sa formulation et ses différentes variantes telles que sac-à-doc monoobjectif,Sac à dos multiple, le problème de sac-à-dos généralisé aux choix multiples et le problème de sac-à-dos multidimensionnel en 0-1 (MOKP).Ainsi ,les méthodes de leur résolution.

3

Pré-traitement et règles de réduction pour le sac à dos multiobjectif

3.1 Introduction

L'application des procédures de réduction de la taille d'une instance d'un problème avant de le résoudre est vraiment intéressante. En particuliers, ces procédures peuvent être utilisés soit dans une résolution exacte ou approchée. L'avantage de ces procédures consiste à améliorer le temps de résolution si le coût de son application est raisonable par rapport à celui de la résolution.

Ces procédures sont disponibles dans le cas de sac à dos mono objectif, ainsi que pour d'autre variantes, comme sa version multi-dimensionnelle [31],[11], mais dans le cas multiobjectif très peu de travail existe.d'aprés [53] seuls Gomes da Silva et al.[44] ont présenté des travaux visant à déterminer, lors d'un pré-traitement pour le sac à dos bi-objectif,la valeur qu'auront certaines variables dans les solutions efficaces.

Nous donnons dans ce chapitre des propositions qui permettent de réduire a priori la taille

des instances du problème de sac à dos multiobjectif. Ces propositions sont un résultat d'une relation de dominance, qui est définie sur les données du problème, ainsi de bornes connues sur la cardinalité des solutions efficaces.

3.2 Pré-traitement pour le sac à dos mono objectif

3.2.1 Réduction de la taille des instances

Les procédures de réduction consistent à séparer l'ensemble $\{1, \cdots, n\}$ des variables en trois sous ensembles :

$$J^1 = \{i \in \{1, \cdots, n\} : x_i = 1, \forall x \in X^*\}$$
$$J^0 = \{i \in \{1, \cdots, n\} : x_i = 0, \forall x \in X^*\}$$
$$F = \{1, \cdots, n\} \backslash (J^0 \cup J^1)$$

où X^* est l'ensemble des solutions optimales.
La résolution du problème initiale peut être faite en résolvant le problème réduit avec seulement les variables de F où la capacité de sac est égale à $w - \sum_{i \in J^1} w_i$.
Les solutions optimales du problème initiale sont celles du problème réduit auquelles les objets de J^1 sont ajoutés.
Pour le calcul des ensembles J^0 et de J^1 nous avons suivi l'idée suivante :
Pour chaque variable x_i si l'affectation $x_i \leftarrow b$ ($b = 0$ ou 1) donne une valeur de z qui est moins bonne qualité que la meilleur solution connue, alors la variable x_i doit prendre la valeur 1-b dans toutes les solutions optimales.
Soit $\underline{z}$ la valeur de z d'une solution réalisable, obtenue par exemple par l'algorithme Glouton décrite dans le paragraphe 2.2.2.
Soit $\overline{z}_i^b$ une borne supérieur calculée en fixant $x_i = b$.
Les ensembles J^0 et J^1 sont définis comme suit :

$$J^{(1-b)} = \{i \in \{1, \cdots, n\} : \overline{z}_i^b < \underline{z}\}$$

Toutes les bornes présentées dans la remarque 2.2.1 peuvent être utilisées pour le calcul de $\overline{z}_i^b$. Ingargiola et Korsh [52] ont présenté le premier algorithme de réduction pour KP, mais de nombreuses améliorations ont vu le jour par la suite [57],[64].

3.3 Pré-traitement et règles de réduction pour le sac à dos multiobjectif

3.3.1 Variables régulières et données du problème

Définition 3.3.1. (Variable régulière)
On dit qu'une variable est régulière si elle n'accepte qu'une seulle valeur dans toutes les solutions efficaces.Les autres variables sont dites irrégulières.

Les ensembles dénotent les indices de ces variables sont définis par :

$$C_0 = \{i \in \{1, \cdots, n\} : x_i = 0, \forall x \in X_{E_M}\}$$
$$C_1 = \{i \in \{1, \cdots, n\} : x_i = 1, \forall x \in X_{E_M}\}$$

Remarque 3.3.1. Dans une instance de 50 variables,aux coefficients générés aléatoirement et sans corrélation dont l'ensemble des solutions efficaces est constitué de 34 solutions,où 9 variables sont toujours égales à 0 et 18 autres sont toujours égales à 1.
La figure 3.1 représente les profits et poids des objets, associés au caractère régulier des variables correspondantes.
L'image en haut à gauche illustre la régularité des valeurs des variables dans les solutions efficaces, pour une instance de sac à dos bi-objectif de taille 50. Les données des variables toujours à zéro sont indiquées par des triangles, celles des variables toujours à un sont illustrées par des cercles. Les données représentées par des carrés correspondent à des variables intervenant avec les deux valeurs dans X_{EM}. Les autres images sont des projections de la première sur chaque paire d'axes. Nous pouvons observer que les variables toujours à zéro sont dans une partie de l'espace où les profits sont faibles et aux poids importants, alors que celles toujours égales à un sont associées aux grands profits et aux poids faibles.
Il n'y a cependant aucune séparation nette de l'espace entre chaque groupe de variables. En effet, nous pouvons constater que des variables intervenant avec les deux valeurs sont présentes dans tout l'espace.

Nous pouvons représenté les données du problème de sac à dos par l'ensemble des vecteurs suivants :

$$V = \{v^i \in \mathbb{Z}^{p+1} : v^i = (c_i^1, \cdots, c_i^p, -w_i), i \in \{1, \cdots, n\}\}$$

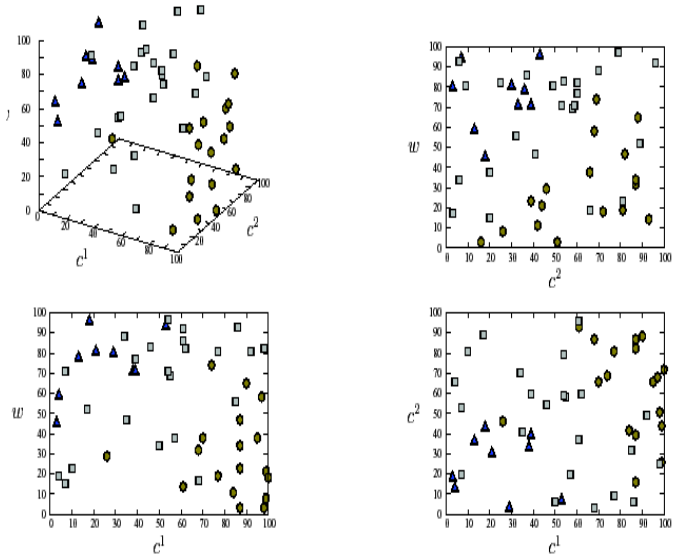

FIG. 3.1 – Régularité des valeurs des variables dans les solutions efficaces.

Définition 3.3.2. (Dominance des données)
Soit v^i ,$v^j \in V$.
La donnée v^i domine la donnée v^j ($v^i \geq_V v^j$) si $c^i \geq c^j$ et $w_i \leq w_j$

Cette définition signifie que,si l'objet j est choisi dans une solution ,alors la substitution de cet objet par l'objet i améliore la qualité de catte solution.

Remarque 3.3.2. :
Dans le cadre de recherche d'un ensemble complet maximal de solutions, la relation de dominance de la définition 3.3.2 ne peut s'appliquer directement aux éléments de V.
En effet, considérons l'exemple suivant :

Exemple 3. *:*
soient $v^1 = (15, 15, -9)$ *et* $v^2 = (15, 15, -13)$ *d'une instance quelconque de sac à dos bi objectif. D'après la définition 1.2,* $v^1 \geq v^2$*.Or,la substitution de l'objet 2 par l'objet 1 n'améliore pas la qualité d'une solution.Par conséquent, nous pouvons pas considérer que la donnée 1 domine la donnée 2.*

Néanmoins, nous pouvons remarquer que bien que la séparation de la comparaison du profit et du poids soit impérative ici, elle ne l'est pas dans le cadre d'un ensemble complet

non maximal.

Une définition des sous-ensembles de $\{1, \cdots, n\}$ correspondant à ceux de V dans lesquels la relation de dominance $\geq_V$ intervient est donnée ci-dessous.

Définition 3.3.3. Soit $v^i \in V$.

1. L'ensemble des indices j des données v^j dominant la donnée v^i est nommé ensemble **préféré** et est noté :

$$Pref(v^i) = \{j \in \{1, \cdots, n\} : v^j \geq_V v^i\}.$$

2. L'ensemble des indices j des vecteurs v^j dominés par la donnée v_i est nommé ensemble **dominé** et est noté :

$$Dom(v^i) = \{j \in \{1, \cdots, n\} : v^i \geq_V v^j\}.$$

D'après ces définitions, nous obtenons l'équivalence suivante :

$$i \in Pref(v^j) \Longleftrightarrow j \in Dom(v^i)$$

3.3.2 Bornes sur la cardinalité des solutions efficaces

Glover [39] a introduit la notion de la borne inférieure et de la borne supérieure de la cardinalité d'une solution optimale x pour le problème de sac à dos unidimensionnel mono-objectif.
Ces bornes sont définies comme suit :

$$LB(w) = \max\{s : \sum_{i=1}^{s} w_i \leq w\} \text{ où } w_i \geq w_{i+1}, \forall i \in \{1, \cdots, n\}$$
$$UB(w) = \max\{s : \sum_{i=1}^{s} w_i \leq w\} \text{ où } w_i \leq w_{i+1}, \forall i \in \{1, \cdots, n\}$$

LB(w) et UB(w) sont clairement indépendantes des fonctions objectifs.
Gandibleux et Fréville ont généralisé leur utilisation au cas multi-objectif à travers la proposition suivante [35] :

Proposition 3.1. *:*
Si $x \in X_{E_M}$ *alors* $LB(w) \leq \sum_{i=1}^{n} x_i \leq UB(w)$

Dans la section suivante, ces bornes sont utilisées conjointement aux ensembles Pref(.) et Dom(.) pour dériver des propriétés des variables régulières depuis l'ensemble des solutions efficaces [53].

3.3.3 Propriétés des variables régulières

Proposition 3.2. *:*
Soit $x \in X_{E_M}.\forall i \in \{1, \cdots, n\}$,*si* $x_i = 0$ *alors* $x_j = 0, \forall j \in Dom(v^i)$

Preuve :
Immédiat à partir de la définition de dominance 3.3.2 sur V et la définition 3.3.3 .

Proposition 3.3. *:*
Soit $x \in X_{E_M}.\forall i \in \{1, \cdots, n\}$,*si* $x_i = 0$ *alors* $x_j = 1, \forall j \in Pref(v^i)$

Preuve :
Immédiat à partir de la définition de dominance 3.3.2 sur V et la définition 3.3.3 .

En appliquant l'équivalence de la définition 3.3.3 sur ces deux propositions, nous obtenons les propriétés suivantes :

- Soit $x \in X_{E_M}.\forall i, j \in \{1, \cdots, n\}$,si $x_i = 0$ et $i \in Pref(v^j)$ alors $x_j = 0$

- Soit $x \in X_{E_M}.\forall i, j \in \{1, \cdots, n\}$,si $x_i = 0$ et $i \in Dom(v^j)$ alors $x_j = 1$

Proposition 3.4. *:*
Soit $i \in \{1, \cdots, n\}.Si \mid Pref(v^i) \mid \geq UB(w)$ *alors* $x_i = 0, \forall x \in X_{E_M}$.

Preuve :
Par contradiction, supposons qu'il existe $x \in X_{E_M}$ telle que $x_i \neq 0$, c'est à dire telle que $x = (x_1, \cdots, x_i = 1, \cdots, x_n$. puisque $\mid Pref(v^i) \mid \geq UB(w)$ alors il existe un $j \in Pref(v^i$ tel que $x_j = 0$. De plus, nous savons que $v^j \geq v^i$, ou de manière équivalente,$w_j \leq w_i$ et $c_j \geq c_i$. Par conséquent nous pouvons construire une solution réalisable $x' = (x_1, \cdots, x_i = 0, \cdots, x_j = 1, \cdots, x_n)$, avec $z(x') \geq z(x)$. Cela implique que $x \notin X_{E_M}$, ce qui est une contradiction.

Proposition 3.5. *:*
Soit $i \in \{1, \cdots, n\}$.*Si* $\sum_{j \in Pref(v^i)} w_j + w_i > w$ *alors* $x_i = 0, \forall x \in X_{E_M}$.

Preuve :
Par contradiction, supposons qu'il existe $x \in X_{E_M}$ telle que $x_i \neq 0$, c'est à dire telle que $x = (x_1, \cdots, x_i = 1, \cdots, x_n$.puisque $\sum_{j \in Pref(v^i)} w_j + w_i > w$ alors il existe un $j \in Pref(v^i)$ tel que $x_j = 0$. De plus, nous savons que $v^j \geq v^i$, ou de manière équivalente,$w_j \leq w_i$ et $c_j \geq c_i$. Par conséquent nous pouvons construire une solution réalisable $x' = (x_1, \cdots, x_i = 0, \cdots, x_j = 1, \cdots, x_n)$, avec $z(x') \geq z(x)$. Cela implique que $x \notin X_{E_M}$, ce qui est une contradiction.

Proposition 3.6. *:*
Soit $i \in \{1, \cdots, n\}$.*Si* $n- \mid Dom(v^i) \mid \leq LB(w)$ *alors* $x_i = 1, \forall x \in X_{E_M}$.

Preuve :
Par contradiction, supposons qu'il existe $x \in X_{E_M}$ telle que $x = (x_1, \cdots, x_i = 0, \cdots, x_n$ et que $n- \mid Dom(v^i) \mid \leq LB(w)$. Puisque $x_i = 0$, d'après la proposition 3.2, $x_j = 0, \forall j \in Dom(v^i)$.
Par conséquent, il reste $n- \mid Dom(v^i) \mid -1$ variables pour lesquelles l'affectation de la valeur 0 ou 1 reste à déterminer. Cependant, $n- \mid Dom(v^i) \mid -1 < LB(w)$; par conséquent, selon la proposition 3.1, $x \notin X_{E_M}$, ce qui est une contradiction.

Proposition 3.7. *:*
Soit $i \in \{1, \cdots, n\}$.*Si* $\sum_{j \notin Dom(v^i)} w_j \leq w$ *alors* $x_i = 1, \forall x \in X_{E_M}$.

Preuve :
Par contradiction, supposons qu'il existe $x \in X_{E_M}$ telle que $x = (x_1, \cdots, x_i = 0, \cdots, x_n$ et que $\sum_{j \notin Dom(v^i)} w_j \leq w$. Puisque $x_i = 0$,d'après la proposition 3.2 ,$x_j = 0$, $\forall j \in Dom(v^i)$. Soit x' une solution avec $x'_j = 0, \forall j \in Dom(v^i)$,et $x'_j = 1, \forall j \notin Dom(v^i)$. Puisque $\sum_{j \notin Dom(v^i)} w_j \leq w$, alors x' est réalisable. De plus, $z(x') \geq z(x)$. Cela implique que $x \notin X_{E_M}$, ce qui est une contradiction.

3.4 Expérimentations numériques

Les propositions précédentes ont été appliquées sur un ensemble d'instances d'après [53].
Nous noterons $C'_0 \subseteq C_0$ (respectivement $C'_1 \subseteq C_1$) l'union des variables fixées par les

propriétés 3.4 et 3.5 (respectivement 3.6 et 3.7).
Les ensembles C_0 et C_1 sont calculés à partir des solutions déterminées par la résolution exacte de ces instances.
Nous présentons dans la première partie de cette section les instances utilisées. Puis, dans la seconde partie, les résultats obtenus sur toutes les instances.

3.4.1 Instances du problème de sac à dos

Instances du problème de sac à dos bi-objectif

Les instances du problème de sac à dos bi-objectif sont séparées en trois groupes selon plusieurs caractéristiques précisées ci-dessous.Les intervalles des coefficients sont détaillés dans la table 3.1. Dans tous les cas, la capacité du sac à dos est $w = \lfloor 0,5 * \sum_{i=1}^{n} w_i \rfloor$.
Les instances du groupe A sont séparées en quatre sous-groupes,comme décrit dans [53] les trois derniers étant une variation du premier. Pour une taille d'instance donnée, le vecteur de poids est identique dans chaque sous-groupe. Le premier, A-1, est constitué de dix instances aux coefficients non corrélés et générés aléatoirement. Les instances du sous-groupe A-2 ont été crées à partir de A-1 en remplaçant c^2 par c^1, en ordre inversé (c'est-à-dire que $c_i^2 = c_{n-i+1}^1, \forall i$). Les vecteurs des coûts de celles de A-3 ont des valeurs générées par plateaux de longueurs tirées aléatoirement dans l'intervalle $[1; \lfloor \frac{n}{10} \rfloor]$. Enfin, les instances du groupe A-4 ont été crées à partir de A-3 en remplaçant c^2 par c^1, en ordre inversé (comme A-2 par rapport à A-1).
Les instances du groupe B sont séparées en trois sous-groupes,comme décrit dans [53] celles du sous-groupe B-1 ont leurs coefficients non corrélés. Les instances du sous-groupe B-2 sont des instances non conflictuelles où c^1 est corrélé positivement avec c^2. Enfin, celles du sous-groupe B-3 sont non conflictuelles où c^1 est positivement corrélé à w.
Les instances du dernier groupe C sont séparées en quatre sous-groupes,comme décrit dans [53]le sous-groupe C-1 contient des instances aux coefficients non corrélés. Celles de C-2 sont des instances non conflictuelles où c^1 est corrélé positivement avec c^2. C-3 est constitué d'instances conflictuelles où c^1 et c^2 sont corrélés négativement,de même que C-4 où, en plus, w est corrélé positivement avec c^1 et c^2.

Instances du problème de sac à dos tri-objectif

Les instances du problème de sac à dos tri-objectif seront représentées par le groupe D par la suite, contenant deux sous-groupes, nommés D-1 et D-2. Le premier est constitué

Groupe	c^1	c^2	w
A-1 à A-4	[1 ; 100]	[1 ; 100]	[1 ; 100]
B-1	$10 \times[1;300] et 10 \times [1;1000]$	$10 \times[1;300] et 10 \times [1;1000]$	$10 \times[1;300] et 10 \times [1;1000]$
B-2	$c_i^1 \in [c_i^2 - 100; c_i^2 + 100]$	[111 ; 1000]	[1 ; 1000]
B-3	$c_i^1 = w_i + 100$	[1 ; 1000]	[1 ; 1000]
C-1	[1 ; 1000]	[1 ; 1000]	[1 ; 1000]
C-2	[111 ; 1000]	$c_i^1 \in [c_i^2 - 100; c_i^2 + 100]$	[1 ; 1000]
C-3	[1 ; 1000]	$c_i^2 \in [\max\{900 - c_i^1, 1\}, \min\{1100 - c_i^1, 1000\}]$	[1 ; 1000]
C-4	[1 ; 1000]	$c_i^2 \in [\max\{900 - c_i^1, 1\}, \min\{1100 - c_i^1, 1000\}]$	$w_i \in [-200; 200] + c_i^1 + c_i^2$

TAB. 3.1 – Coefficients utilisés pour générer les instances.

Groupe	c^1	c^2	c^3	w
D-1	[1 ; 1000]	[1 ; 1000]	[1 ; 1000]	[1 ; 1000]
D-2	[1 ; 1000]	$c_i^2 \in [1; 1001 - c_i^1]$	$c_i^3 \in [\max\{900 - c_i^1 - c_i^2, 1\}, \min\{1100 - c_i^1 - c_i^2, 1001 - c_i^1\}]$	[1 ; 1000]

TAB. 3.2 – Coefficients utilisés pour générer les instances.

d'instances non corrélées tandis que celles du second sont conflictuelles, où c^1 et c^2 sont corrélés négativement, de même que c^3 avec c^1 et c^2. La table 3.2 représente les intervalles des coefficients générés aléatoirement. Dans tous les cas, la capacité du sac à dos est $w = \lfloor 0,5 * \sum_{i=1}^{n} w_i \rfloor$.

3.4.2 Résultats expérimentaux

Instances du problème de sac à dos bi-objectif

Les figures 3.2, $\cdots$, 3.5 présentent le nombre de variables régulières trouvées pour chaque instance, pour chaque propriété, ainsi que le nombre $\mid C_0' \cup C_1' \mid$ de variables régulières différentes et le nombre $\mid C_0 \cup C_1 \mid$ calculé après une résolution exacte des instances. D'après ces figures, nous remarquons que les résultats obtenus montrent que $\mid C_0' \cup C_1' \mid$ est loin de $\mid C_0 \cup C_1 \mid$ mais reste néanmoins une quantité non négligeable. De plus, la propriété 3.5 est clairement plus performante que la 3.4 pour fixer des variables à 0. De même que la 3.7 est meilleure que 3.6 pour les fixer à 1.
Les propriétés n'apportent rien sur les instances des groupes B-3, C-3 et C-4. Ce résultat est attendu dans la mesure où, par construction, les données de ces instances ne présentent aucune dominance. Cependant, elles ne sont pas pour autant exemptes de variables régulières, comme le montrent les valeurs reportées dans la figure 3.6.

Instances du problème de sac à dos tri-objectif

La figures 3.7 et ref39 présentent le nombre de variables régulières trouvées pour chaque instance, pour chaque propriété, ainsi que le nombre $| C'_0 \cup C'_1 |$ de variables régulières différentes et le nombre $| C_0 \cup C_1 |$ calculé après une résolution exacte des instances.
La mème remarque quedans le cas bi-objectif,$| C'_0 \cup C'_1 |$ est loin de $| C_0 \cup C_1 |$.
Les propriétés 2 à 3.2 sont ici quasiment sans apport, seule la propriété 3.3 permet de déterminer dans tous les cas la valeur de quelques variables.
Nombre de variables Pour les instances de D-2, aucune des propriétés énoncées précédemment ne fixe de variables.Cela revient du fait de leur construction, ces instances ne véhiculent aucune dominance et, par conséquent Le nombre de variables régulières pour ces instances est reporté dans la figure 3.9 et nous pouvons constater, à nouveau, que leur nombre est important.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
A-1	50	0	1	0	10	11	27
	100	0	0	1	5	5	53
	150	0	7	1	11	18	83
	200	2	3	0	14	17	116
	250	1	6	0	12	18	139
	300	0	3	2	23	26	173
	350	2	5	1	26	31	213
	400	1	7	5	26	33	252
	450	0	5	3	22	27	270
	500	0	13	3	34	47	301
A-2	50	0	2	1	7	9	24
	100	0	0	0	4	4	51
	150	0	1	1	8	9	78
	200	0	5	2	10	15	108
	250	1	5	0	13	18	145
	300	1	6	2	26	32	180
	350	4	8	3	23	31	203
	400	3	5	3	34	39	249
	450	2	6	3	25	31	249
	500	3	11	1	38	49	305

FIG. 3.2 – Nombre moyen de variables fixées par les propriétés pour les instances du groupe A.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
A-3	50	1	2	0	1	3	23
	100	3	7	0	4	11	61
	150	0	4	0	6	10	81
	200	0	3	1	16	19	105
	250	1	3	2	12	15	136
	300	5	10	0	27	37	186
	350	0	2	0	27	29	193
	400	4	15	4	32	47	239
	450	1	2	3	25	27	291
	500	14	18	9	66	84	364
A-4	50	0	1	0	1	2	19
	100	0	3	0	7	10	59
	150	2	4	0	6	10	75
	200	0	0	2	18	18	102
	250	1	4	3	17	21	146
	300	1	5	1	37	39	207
	350	0	14	1	31	45	239
	400	2	3	0	13	16	185
	450	6	11	7	27	38	328
	500	3	11	0	63	74	346

FIG. 3.3 – Nombre moyen de variables fixées par les propriétés pour les instances du groupe A.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
B-1	50	0	0	0	1	1	19
		0	3	0	2	5	25
		0	0	2	6	6	29
		0	3	0	4	7	28
		1	1	0	1	2	23
		1	1	0	3	4	26
		0	0	0	3	3	25
		1	2	1	3	5	27
		0	0	0	8	8	31
		1	2	0	4	6	27
		0	2	0	2	4	24
		1	2	0	7	9	30
		0	1	0	8	9	30
		1	3	0	2	5	25
		0	1	2	6	7	22
		0	0	0	3	3	25
		1	1	1	4	5	21
		1	4	1	8	12	33
		0	0	0	2	2	28
		1	1	1	3	4	22

FIG. 3.4 – Nombre moyen de variables fixées par les propriétés pour les instances du s.groupe B-1.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
B-2	50	3	9	2	17	26	46
	100	5	10	3	30	40	100
	150	7	14	3	37	51	139
	200	13	16	7	48	64	197
	250	15	19	8	58	77	237
	300	16	26	10	68	94	281
	350	20	28	13	88	116	328
	400	19	32	16	101	133	373
	450	20	35	18	111	146	420
	500	22	38	20	131	169	468
	600	25	51	24	148	199	560
	700	28	55	28	169	224	658
	800	32	63	30	182	245	751
	900	35	69	32	210	279	841
	1000	41	76	34	234	310	934

FIG. 3.5 – Nombre moyen de variables fixées par les propriétés pour les instances du s.groupe B-2.

3.5 Conclusion

Dans ce chapitre, nous avons présenté une analyse des données pour fixer les valeurs des variables a priori.Nous avons utilisé une nouvelle relation de dominance sur les données du problème et des bornes sur la cardinalité des solutions efficaces pour définir des nouvelles propriétés afin de déterminer les valeurs des variables a priori. Les experimentations ont montrées que l'application de ces propriétés est utile dans le cas où les coefficients du problèmes sont indépendants, en particulier dans le cas bi-objectif.Cependant, de nombreuses variables ne peuvent pas être obtenus par ces dérivées, même si elles sont visiblement régulières après calcul de l'ensemble complet des solutions efficaces.

Sous-groupe	Taille	$\lvert C_0 \cup C_1 \rvert$	Sous-groupe	Taille	$\lvert C_0 \cup C_1 \rvert$
B-3	50	29	C-3	100	28
	100	69		200	69,1
	150	100		300	111,4
	200	140		400	156,3
	250	180		500	192,7
	300	212	C-4	100	4,1
	350	—		150	7,9
	400	—		200	12,7
	450	—		250	15,9

FIG. 3.6 – Nombre de variables régulières par les instances bi-objectif où au moins un vecteur de coût est corrélé au vecteur des poids. Les solutions efficaces des instances du sous-groupe B-3 sont inconnues à partir de 350 variables.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
C-1	100	0,8	0,5	0,8	7,1	7,6	56,9
	200	1,0	0,4	1,2	12,6	13,0	116,4
	300	1,6	0,7	1,7	20,8	21,5	176,4
	400	2,7	1,5	2,5	26,2	27,7	238,3
	500	2,1	0,9	3,2	34,7	35,6	300,4
	600	2,5	0,9	3,2	41,5	42,4	366,4
	700	2,8	1,5	4,1	46,9	46,9	420,2
C-2	600	21,0	4,4	23,0	148,1	152,5	558,4
	700	24,9	4,9	27,3	161,7	166,6	656,7
	800	28,6	4,9	30,9	191,3	191,3	747,5
	900	33,9	6,4	34,0	216,9	223,3	848,4
	1000	37,2	7,6	38,5	241,1	248,7	941,2
	2000	75,1	15,8	75,0	477,7	493,5	1888,6
	3000	116,1	21,2	113,5	722,2	743,4	2838,8
	4000	153,6	29,0	150,4	963,0	992,0	3784,4

FIG. 3.7 – Nombre moyen de variables fixées par les propriétés pour les instances du groupe C.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
D-1	5	0,0	0,1	0,0	0,4	0,5	2,9
	10	0,1	0,0	0,0	0,6	0,6	3,0
	20	0,0	0,1	0,0	0,5	0,6	5,9
	30	0,1	0,0	0,1	0,6	0,6	9,7
	40	0,2	0,0	0,0	0,4	0,4	11,2
	50	0,1	0,0	0,0	0,6	0,6	25,4
	60	0,2	0,0	0,0	0,7	0,7	33,9
	70	0,0	0,0	0,0	1,2	2,2	37,2
	80	0,1	0,0	0,0	0,7	0,7	41,7
	90	0,1	0,0	0,0	1,0	1,0	48,2
	100	0,2	0,0	0,3	1,2	1,2	53,3
	110	0,0	0,1	0,1	1,8	1,9	62,2
	120	0,2	0,0	0,2	1,0	1,0	66,4
	130	0,2	0,0	0,0	1,8	1,8	70,1
	140	0,0	0,0	0,1	1,7	1,7	73,6
	150	0,2	0,0	0,0	1,5	1,5	81,0
	160	0,2	0,1	0,2	1,9	2,0	88,6
	170	0,0	0,2	0,0	2,0	2,2	94,4
	180	0,0	0,0	0,1	2,3	2,3	102,5
	190	0,0	0,0	0,1	2,8	2,8	111,5
	200	0,3	0,1	0,3	2,3	2,4	112,5
	210	0,0	0,1	0,1	2,9	3,0	116,5
	220	0,0	0,0	0,1	2,3	2,3	125,9
	230	0,0	0,0	0,0	2,0	2,0	128,7
	240	0,2	0,0	0,2	3,6	3,6	138,1
	250	0.1	0.2	0.2	3.8	4.0	143.2

FIG. 3.8 – Nombre moyen de variables fixées par les propriétés pour les instances du groupe D.

Sous-groupe	Taille	prop. 3.4	prop. 3.5	prop. 3.6	prop. 3.7	$\lvert C'_0 \cup C'_1 \rvert$	$\lvert C_0 \cup C_1 \rvert$
D-1	260	0,2	0,0	0,0	4,1	4,1	154,7
	270	0,1	0,0	0,2	3,2	3,2	151,4
	280	0,0	0,0	0,2	4,2	4,2	163,8
	290	0,0	0,0	0,4	5,3	5,3	166,4
	300	0,1	0,0	0,3	3,7	3,7	172,7
	310	0,3	0,1	0,3	3,4	3,5	185,0
	320	0,1	0,0	0,2	4,4	4,4	187,3
	330	0,1	0,0	0,1	4,1	4,1	195,5
	340	0,2	0,2	0,2	4,8	5,0	194,4
	350	0,0	0,1	0,3	4,8	4,9	203,4
	360	0,2	0,3	0,3	5,2	5,5	208,5
	370	0,2	0,1	0,4	4,8	4,9	220,7
	380	0,2	0,1	0,2	4,2	4,3	225,1
	390	0,1	0,1	0,1	4,4	4,5	230,4
	400	0,1	0,0	0,6	4,7	4,7	236,5
	410	0,2	0,1	0,4	4,3	4,4	239,3
	420	0,0	0,2	0,3	6,7	6,9	250,2
	430	0,2	0,0	0,3	5,7	5,7	252,4
	440	0,1	0,1	0,3	5,5	5,6	262,1
	450	0,0	0,1	0,2	5,5	5,6	268,1
	460	0,4	0,1	0,3	5,2	5,3	269,8
	470	0,4	0,1	0,3	6,5	6,6	276,1
	480	0,1	0,1	0,5	8,2	8,3	285,1
	490	0,2	0,3	0,2	4,9	5,2	288,8
	500	0,2	0,0	0,5	8,0	8,0	299,2
	550	0,4	0,3	0,3	8,3	8,6	333,1
	600	0,3	0,2	0,9	7,9	8,1	363,4
	650	0,0	0,1	0,8	10,2	10,3	393,1
	700	0,1	0,1	0,6	10,6	10,7	423,2

FIG. 3.9 – Nombre moyen de variables fixées par les propriétés pour les instances du groupe D.

4

La méthode en deux phases pour le problème de sac à dos bi-objectif unidimensionnel en variables binaires

4.1 Introduction

La méthode en deux-phases présente un schéma de résolution exacte très intéressant car très général et qui ne dépend pas du problème. Dans ce chapitre, nous présentons la méthode en deux phases appliquée au problème de sac à dos biobjectif, ainsi l'adaptation des procédures de réduction avant la résolution.

4.2 La méthode en deux phases pour le problème de sac à dos bi-objectif unidimensionnel en variables binaires

Le problème de sac à dos bi-objectif peut être défini par :

$$(01-biKP)\begin{cases}\max Z_j(x)=\sum_{i=1}^{n} c_i^j x_i \quad j\in\{1,2, c_i\in\mathbb{N}^p\\ \text{tq :}\\ \sum_{i=1}^{n} w_i x_i \leq w \quad w, w_i\in\mathbb{N}^*\\ x_i\in\{0,1\}\forall i\in\{1,\cdots,n\}\end{cases}$$

Visée et al.[77] ont proposé une méthode en deux phases dans laquelle l'exploration est faite en utilisant une procédure de séparation et évaluation [77]. Comme son nom l'indique, cette méthode est composé en deux phases :

- La première phase consiste à calculer les solutions efficaces supportées X_{SE} et exactement X_{SE1_m}.
  Pour le calcul de l'ensemble complet minimal des solutions efficaces supportées extrèmes,nous suivons le schéma dichotomique proposé par Anéja et Nair.
  Initialement, l'ensemble S des solutions efficaces supportées est initialisé par les deux solutions lexicographique optimales correspondantes aux $(z_1(x), z_2(x))$ et $(z_2(x), z_1(x))$, respectivement.
  Pour calculer une solution lexicographique optimale, nous résolvons les deux problèmes mono objectif p_1 et p_1 telque :

$$\begin{cases}\max Z_i(x)\\ \text{tq :}\\ \sum_{i=1}^{n} w_i x_i \leq w \quad w, w_i\in\mathbb{N}^*\\ x_i\in\{0,1\}\forall i\in\{1,\cdots,n\}\end{cases}$$

  pour i= 1 à 2.
  les solutions trouvées x^1 et x^2 sont les solutions optimales des problèmes p_1 et p_1 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 ($x^r = x^2 et x^s = x^{1'}$).

Ensuite, nous résolvons un problème (p_λ) où le vecteur λ représente graphiqement la normale de la droite joignant les deux points x^r et x^s où $\lambda = (z_2(x^r) - z_2(x^s), z_1(x^s) - z_1(x^r))$.

La solution obtenue par la résolution de ce problème est appellée x^t

Noton $\overline{y^r y^s}$ le segment reliant les deux points $y^r = (z_1(x^r), z_2(x^r))$ et $y^s = (z_1(x^s), z_2(x^s))$ dans $\mathbb{R}^2$.

- Si $z(x^t) \cap \overline{y^r y^s} = \emptyset$, cela signifie que $z(x^t)$ n'est pas sur le segmant reliant les deux points $y^r et y^s$ comme l'indique la figure 4.2, dans ce cas nous avons deux problèmes (p_λ) à résoudre $y^r et y^t$ et $y^t et y^s$

- Autrement dit, la recherche dichotomique est terminée comme l'indique la figure 4.1 quand la solution supprortée x^t n'est pas extrème et elle est sur le même segment reliant x^ret x^s

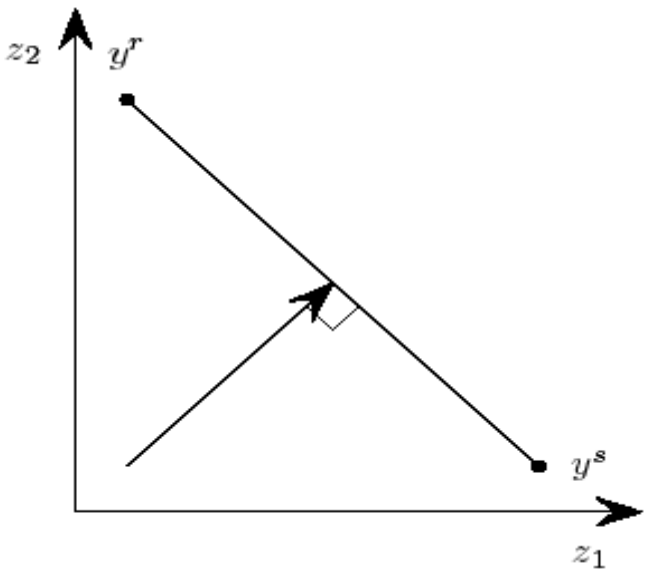

FIG. 4.1 – Problème P défini par y^r et y^s.

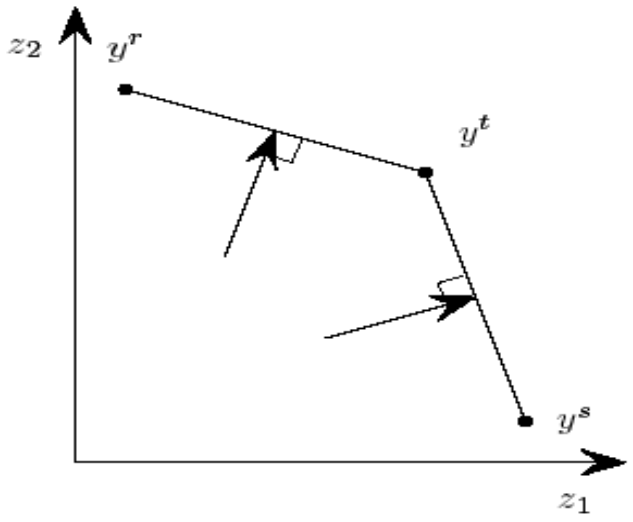

FIG. 4.2 – Un nouveau point supporté est trouvé, la dichotomie continue.

– La seconde phase consiste à trouver les solutions non supportées.
Dans cette phase,Paquete et al. (2004) et Angel et al. (2004) ont commencé leur méthode par des solutons générées aléatoirement,tandisque, nous utilisons les solutions supprotées trouvées dans la première phase qui permettent de réduire l'espace de recherche.
L'espace de recherche restant à explorer dans le cas bi objectif est constitué d'un ensemble de traingles. Chaque triangle est construit de la manière suivante : deux points y^r et y^s adjacents dans Y_{SN}, avec $y_1^r < y_1^s$, définissent un triangle $\triangle(y^r, y^s)$ dont l'hypoténuse est donnée par le segment les reliant et dont l'angle droit est situé en (y_1^r, y_2^s), soit au point d'intersection des cônes de dominance $y^s - R^2$ et $y^r - R^2$.
Par extension de la définition de la section 1.4.4 , ce point est nommé **point nadir local**.
Pour chaque triangle nous associons un problème mono objectif p_λ qui est résolu par une procédure de séparation et évaluation, avec $\lambda = (y_2^r - y_2^s, y_1^s - y_1^r)$.

– **La partie évaluation** :
Si y^r et y^s sont les deux points décrivant le triangle, alors $\underline{z}_1 = y_1^r$ est une bonne borne inférieure au regard du premier objectif pour les solutions non supportées dont les images sont incluses dans le triangle.

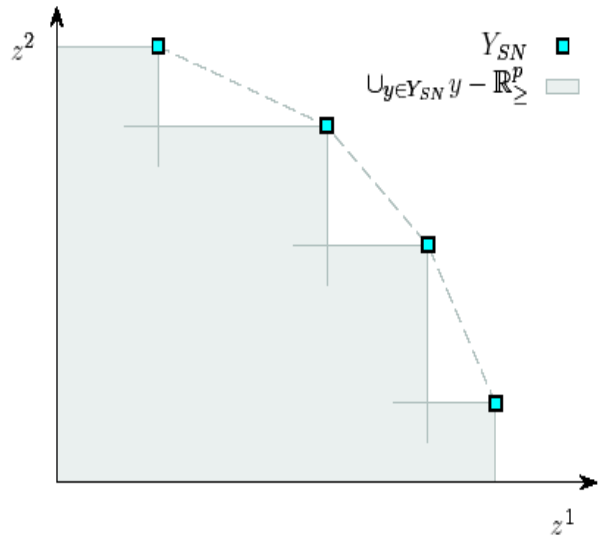

FIG. 4.3 – L'espace de recherche pendant la seconde phase se décrit comme un ensemble de triangles. La ligne en pointillés représente l'enveloppe convexe de Y_N ainsi que les hypoténuses. Les angles droits des triangles se situent à l'intersection des cônes de dominance de points supportés adjacents.

De même pour $\underline{z}_2 = y_2^s$ sur le second objectif.
Enfin, une borne inférieure au problème P_λ est donnée par $\underline{z}^\lambda = \lambda y^N$, où $y_N$est le point nadir local défini par $y^N = (y_1^r, y_2^s)$.
Ces bornes vont être changer à chaque fois que des nouvelles solutions efficaces sont trouvées dans le triangle.
Soit $\{y^1, \cdots, y^m\}$ les points réalisables potentiellement non dominés connus dans le triangle, tels que $y_1^i < y_1^{i+1}$ pour tout $i \in \{1, \cdots, m-1\}$.
En posant $y^0 = y^r, y^{m+1} = y^s$, la meilleur valeur de la borne inférieure présentée dans [77] est

$$\underline{z}^\lambda = \min_{i \in \{0, \cdots, m\}} \{\lambda_1 y_1^i + \lambda_2 y_2^{i+1}\}$$

L'évaluation des noeuds dans la procédure de séparation et évaluation consiste à calculer séparément la borne supérieure des objectif z_1,z_2 et z_λ.
Si une de ces bornes est inférieure à la borne inférieure correspondante, alors le noeud est fermé.

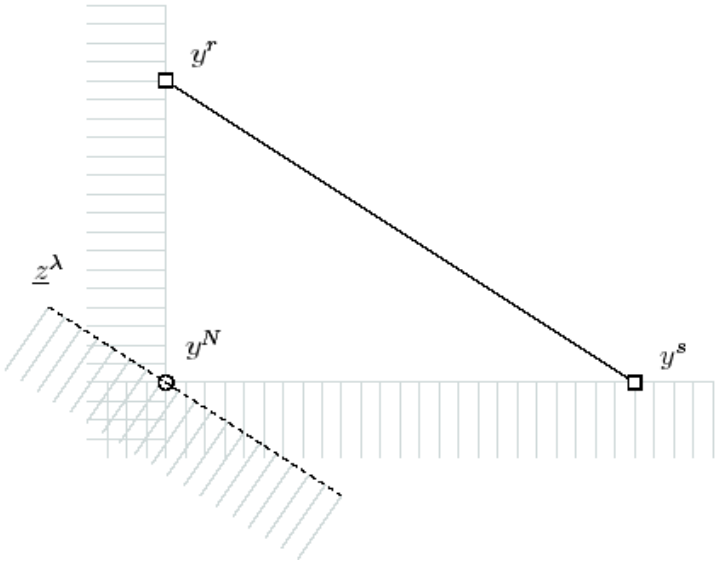

FIG. 4.4 – Illustration des bornes lors de l'exploration d'un triangle dans la seconde phase. Les rayures représentent les régions ôtée de l'espace de recherche par ces bornes. Les carrés (□) représentent les performances des solutions décrivant le triangle. Le rond (○) illustre le point nadir local utilisé pour le calcul de la borne $\underline{z}^\lambda$, représentée ici par une ligne en pointillés.

– **La partie séparation** :
Nous associons au noeud S_0 de l'arbre, une des solutions x^r ou x^s. Supposons x^s, et notons $\{j/x_j^s = 1\} = \{p_1, \cdots, p_q\}$, l'ensemble des indices des variables égale à 1 dans cette solution.
Nous créons q sous noeud $S_1, \cdots, S_q$ qui sont caractérisés par 1 j'usqu'à q variables fixes de la manière illustrée dans la figure 4.6 Suivons cette méthode , les ensembles des solutions admissibles dans chaque sous noeud sont disjoints.Chaque noeud est examiné séparément.
Soit L la liste des solutions non supportées.
Initialement,L est vide et à la fin du procédure L,va contenir les solutions efficaces non supportées relatives aux triangles $\triangle(y_r, y_s)$.

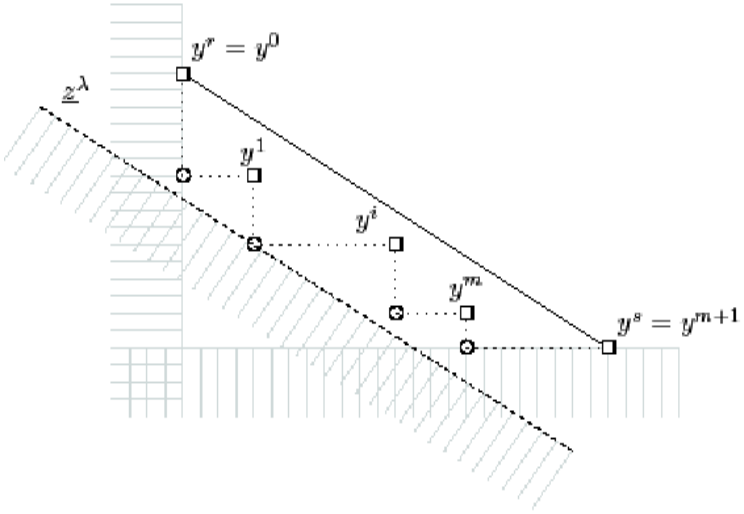

FIG. 4.5 – La borne inférieure $\underline{z}^\lambda$.

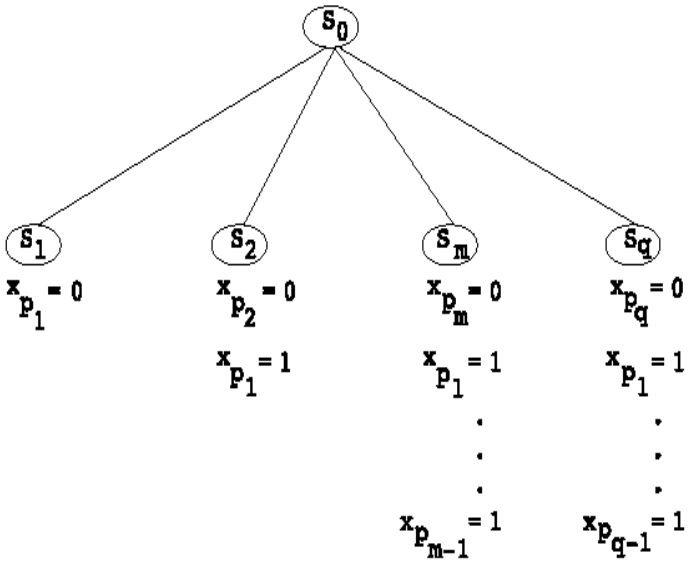

FIG. 4.6 – principe de séparation

4.3 Application de la procédure de réduction dans la méthode en deux phases

D'après [53] les auteurs appliquent une procédure de réduction dans la méthode en deux phases.En effet, dans la première phase ,il s'agit à chaque fois d'un problème de sac à dos mono objectif d'où l'application de l'algorithme de réduction est évidente.
Si,dans cette phase, y^r et y^s sont deux points supportés utilisés pour orienter la maximisation à une étape de la dichotomie, alors la borne inférieure utilisée dans l'algorithme de réduction ainsi que lors de l'initialisation de la PSE est
$\underline{z}^\lambda = \lambda y^r = \lambda y^s$.

Durant la deuxième phase, l'algorithme de réduction consiste à calculer l'union des variables fixées par l'application de l'algorithme de réduction de problème sac à dos mono-objectif en considérant séparément z_1, z_2 et z_λ avec les bornes correspondantes. Pour tout $b \in \{0,1\}$, $j \in \{1,2\}$, $J^b(j)$ est l'ensemble J^b obtenu pour le problème mono-objectif P_j et la borne inférieure $\underline{z}_j$ avec l'algorithme décrit dans la section 2.2.2. Soit $J^b(\lambda)$,$b \in \{0,1\}$ les ensembles obtenus de la même manière pour le problème P_λ et la borne $\underline{z}^\lambda$.

Les variables fixées à $b \in \{0,1\}$ avant l'exploration du triangle sont alors celles de $J^b = J^b(1) \cup J^b(2) \cup J^b(\lambda)$. L'exploration est alors réduite au seuls objets de l'ensemble $F = \{1, \cdots, n\} \setminus (J^1 \cup J^0)$, avec une capacité de sac à dos égale à $w - \sum_{i \in j^1} w_i$.
Dans notre travail, nous avons utilisé les propositions montionées dans le chapitre précédent : nous avons représenté les donées sous la forme des vecteurs ,pour chaque donnée, nous avons déterminé les ensembles Pref(.) et Dom(.) ainsi nous avons calculé les bornes LB(w) et UB(w).Enfin, et avant l'application de la méthode en deux phases, nous avons appliqué les propositions 3.5 pour fixer des variables à 0 et la 3.7 pour les fixer à 1.

Algorithme 1 phase1

1: **Initialisation. pour** i=1 à 2 Résoudre les problèmes mono objectif p_i

$$\begin{cases} \max Z_i(x) \\ \text{tq :} \\ \sum_{i=1}^{n} w_i x_i \leq w \quad w, w_i \in \mathbb{N}^* \\ x_i \in \{0,1\} \forall i \in \{1, \cdots, n\} \end{cases}$$

les solutions trouvées sont x^r et x^s tel que $y_1^r < y_1^s$.$S := \{y^r, y^s\}$

2: **fin pour**

3: Choisir deux points supportés adjacents y^r et y^s.

4: Construire et résoudre

$$(P_\lambda) = \begin{cases} \max \lambda \cdot Z(x) \\ \text{tq :} \\ \sum_{i=1}^{n} w_i x_i \leq w \quad w, w_i \in \mathbb{N}^* \\ x_i \in \{0,1\} \forall i \in \{1, \cdots, n\} \end{cases}$$

, avec $\lambda = (y_2^r - y_2^s, y_1^s - y_1^r)$, obtenir $y^t \in \mathbb{N}^2$.

5: Si $\lambda \cdot y^t \neq \lambda \cdot y^r$, conserver y^t.$S := S \bigcup \{y^t\}$

6: Si il reste des adjacences non traitées, retourner en 3.

7: **S**i $\lambda \cdot y^t = \lambda \cdot y^r$ ou les adjacences dans S sont tous traitées.**ALLER** en algorithme 2

Algorithme 2 phase2

1: Choisir un point x tel que $x = x^r$ ou $x = x^s$ du triangle $\triangle(y^r, y^s)$ par exemple x^r ,calculer $\underline{z}_1 = y_1^r$,$\underline{z}_2 = y_2^s$,$\underline{z}^\lambda = \lambda \cdot (y_1^r, y_2^s)$.

2: **tant que**$z_1(x) \geq \underline{z}_1$ et $z_2(x) \geq \underline{z}_2$ et $z^\lambda(x) \geq \underline{z}^\lambda$ **faire**

3: Déterminer l'ensemble $J^1 = \{j/x_j^r = 1\} = \{p_1, \cdots, p_q\}$.

4: **si** $J^1 = \emptyset$ **TERMINER**.

5: **sinon**

6: **pour** $i := p_1$ à p_q **tel que** $i \in J^1$ **faire**

7: $x_i = 0$

8: $x_j = 1$, $j \in J^1$ et $j < i$ et $j > p_1$ Résoudre le problème (P_λ)

9: **si** le problème admet une solution x **alors**

10: Calculer z_1, $z_2 z^\lambda$. $S := S \bigcup \{(z_1(x), z_2(x))\} = \{y^1, \cdots, y^m\}$. $y^0 = y^r, y^{m+1} = y^s$

$$\underline{z}^\lambda = \min_{i \in \{0, \cdots, m\}} \{\lambda_1 y_1^i + \lambda_2 y_2^{i+1}\}$$

ALLER en 2.

11: **sinon TERMINER**.

12: **fin pour** .

13: **fin si**

14: **fin tant que**

4.4 Implémentation et résultats

La méthode en deux pahases est implementée sur machine en se servant des avantages du langage de programmation mathématique le Matlab 7 (matrix laboratory 7).

Cet algorithme a été implementer sur une machine dont les caractéristiques :(processor P4, FSB 800 Mb, DDR 3 GB) en utilisant le même langage de programmation (matlab 7) et tester sur un ensemble d'instances de sac-à-dos biobjectifs des groupes A et B présentés dans la section 3.4.1. Ces algorithmes sont exécutés une seule fois.

Instances/Nombre de variables	A_1			
	t_1	t_2	T	SE
50	4.703	**55.172**	59.875	7
100	10.657	**1.1156e+003**	1.1263e+003	15
150	57.893	**3.697e+003**	3.7549e+003	25
200	60.783	**4.279e+003**	4.34e+003	32
250	64.12	**4.921e+003**	5.e+003	50
300	67.265	**5.743e+003**	5.81e+003	62
350	69.31	**6.81e+003**	6.88e+003	66
400	72.43	**8.367e+003**	8.439e+003	74
450	76.31	**9.676e+003**	9.852e+003	84
500	79.68	**1.063e+004**	1.071e+004	98

TAB. 4.1 – *Table des résultats pour les instances de s. groupe* A_1.

Instances/Nombre de variables	A_2			
	t_1	t_2	T	SE
50	7.031	**171.25**	178.281	6
100	7.016	**2667**	2.6740e+003	13
150	56.367	**3.457e+003**	3.5133e+003	27
200	61.892	**4.198e+003**	4.259e+003	35
250	63.623	**4.891e+003**	4.9546e+003	52
300	66.951	**5.67e+003**	5.747e+003	60
350	70.218	**6.681e+003**	6.7512e+003	69
400	74.66	**7.961e+003**	8.0356e+003	50
450	73.68	**9.346e+003**	9.4196e+003	74
500	78.84	1.0481e+004	1.0559e+004	90

TAB. 4.2 – *Table des résultats pour les instances de s. groupe* A_2

t_1 **:** Temps d'exécution de la première phase par seconde ;

t_2 **:** Temps d'exécution de la deuxième phase par seconde ;

T : Temps total d'exécution par seconde ;

SE : Nombre de solutions efficaces ;

. : représente une virgule.

Instances/Nombre de variables	A_3			
	t_1	t_2	T	SE
50	16.14	**158.187**	174.327	9
100	48.531	**1.1612e+003**	1.2097e+003	18
150	58.593	**3.8099e+003**	3.8685e+003	27
200	62.69	**4.368e+003**	4.4307e+003	30
250	65.894	**4.8351e+003**	4.9010e+003	25
300	67	**5.8621e+003**	5.9291e+003	45
350	69.12	**7.65e+003**	7.7191e+003	61
400	72.968	**8.867e+003**	8.94e+003	77
450	75.68	**9.371e+003**	9.4467e+003	79
500	78	**9.965e+003**	1.0043e+004	85

TAB. 4.3 – *Table des résultats pour les instances de s. groupe A_3.*

Instances/Nombre de variables	A_4			
	t_1	t_2	T	SE
50	15.86	**179.396**	195.2560	7
100	38.21	**1.218e+003**	1.2562e+003	16
150	56.965	**4.168e+003**	4.2250e+003	25
200	63.86	**4.765e+003**	4.8289e+003	33
250	66.58	**5.35e+003**	5.4166e+003	30
300	67.861	**6.764e+003**	6.8319e+003	45
350	70.296	**7.674e+003**	7.7443e+003	59
400	73.869	**8.43e+003**	8.5038e+003	64
450	78.82	**9.168e+003**	9.2468e+003	74
500	81.582	**9.6972e+003**	9.7787e+003	89

TAB. 4.4 – *Table des résultats pour les instances de s. groupe A_4*

Instances/Nombre de variables	B_1			
	t_1	t_2	T	SE
50	4.312	**53.063**	57.375	7

TAB. 4.5 – *Table des résultats pour les instances de s. groupe B_1*

4.5 Discussion

Nous pouvons remarquer que le temps de résolution croit rapidement avec la taille de l'instance et, même, la façon dont celle-ci a été générée influe directement sur ce temps.

– Pour les instances du groupe A, nous pouvons remarquer qu'elle peuvent être

Instances/Nombre de variables	B_2			
	t_1	t_2	T	SE
50	2.68	**45.148**	47.828	6
100	3.6	**49.65**	53.2500	11
150	4.89	**52.74**	57.63	21
200	9.65	**68.89**	78.54	25
250	12.32	**72**	84.32	31
300	14.1	**75.29**	89.39	35
350	20.31	**89.56**	109.8700	42
400	38.96	**1.25e+003**	1.2890e+003	56
450	46.98	**2.56e+003**	2.6070e+003	45
500	58.593	**3.2099e+003**	3.2685e+003	63
550	60.351	**4.64e+003**	4.7004e+003	69
600	68.12	**5.96e+003**	6.0281e+003	74

TAB. 4.6 – *Table des résultats pour les instances de s. groupe B_2*

Instances/Nombre de variables	B_3			
	t_1	t_2	T	SE
50	62.3	**5.687e+003**	5.7493e+003	5
100	66	**7.654e+003**	7720	7
150	77	**9.4867e+003**	9.5637e+003	4

TAB. 4.7 – *Table des résultats pour les instances de s. groupe B_3.*

résolues jusqu'à 500 variables.

– Pour les instances du s.groupe B-1, nous pouvons remarquer que le temps de résolution est acceptable car elles sont de petite taille (50 variables).

– Pour les instances du s.groupe B-2, nous pouvons remarquer qu'elle peuvent être résolues sans difficulté en allant j'usqu'à 600 variables car ces instances sont générées avec une corrélation positive entre les objectifs.

– Pour les instances du s.groupe B-3, nous ne pouvons pas résoudre ces instances au delà de 150 variables car ces instances sont générées avec une corrélation positive entre un objectif et le vecteur des poids.

Nous pouvons remarquer que le temps de résolution est consommé dans la deuxième phase où nous avons utilisé la méthode de séparation et évaluation : cette méthode est une méthode exacte. Elle exploite systématiquement la totalité de l'espace de recherche. Dans ce type de méthodes, l'ensemble des solutions trouvées "front Pareto" est efficace. Cependant elle est, de grande complexité, ce qui rend le temps d'exécution très important.

4.6 Conclusion

Dans ce chapitre, nous avons présenté la méthode en deux phases appliquée au problème de sac-à-dos biobjectif unidimensionnel en variables binaires.
Son intérêt réside dans une décomposition de l'espace de recherche et l'utilisation de méthodes mono-objectifs pour les différentes résolutions successives (recherche des extrêmes, résolution des agrégations...). Appliquer la méthode en deux-phases pour la résolution d'un problème bi-objectif nécessite donc d'avoir une méthode mono-objectif efficace (si possible polynomiale). C'est le cas pour notre problème traité , et c'est ce qui rend la méthode performante pour ce problème.

Conclusion générale

Dans ce mémoire, nous avons présenté l'application de la méthode en deux phases sur le problèmes de sac à dos multiobjectif unidimentionnel en variables binaires (ou MultiObjectiv knapsack problem (01-MOKP)), cas : biobjectif, ainsi q'une procédure de pré-traitement appliquée avant la résolution de ce problème.

D'abord, nous avons abordé et étudié la programmation multiobjectif. Ensuite nous avons présenté le problème de sac-à-dos et ses différentes variantes, en nous focalisant sur le cas uni-dimensionnel multiobjectif (biobjectif). Ainsi, nous avons étudié une procédure de pré-traitement qui permette de réduire la taille d'une instance d'un problème, appliquée au problème de sac à dos mono objectif, puis dans le cas multiobjectif.
Enfin, nous avons appliqué la méthode en deux phases sur le problème de sac à dos biobjectif unidimentionnel en variables binaires ainsi que la procédure de prétraitement.

Afin de tester l'efficacité de notre méthode, nous avons étudié et programmé cette méthode, qui a été implémenté sur machine sous le logiciel matlab 7, puis une étude comparative à été menée sur un ensemble de problèmes tests, dont les résultats numériques montrent l'efficacité de notre approche, en termes de nombre de variables et de qualité de l'ensemble des solutions malgrès que le temps d'exécution est grand.

Perspectives de recherche

Nous nous sommes concentrés dans ce mémoire sur l'optimisation combinatoire multiobjectif, dont l'objectif est de fournir l'ensemble le plus complet possible des solutions

Pareto à un problème donné. En guise de perspectives, on peut citer :

- Adapter la procédure de séparation et évaluation au cas multiobjectif.
- Développer d'autres mécanismes de reduction de problème.
- Améliorer la vitesse de l'algorithme, en développant de nouvelles bornes pour la procédure de séparation et évaluation ou essayer d'hybrider la procédure de séparation et évaluation avec des metaheuristiques, telles que les algorithmes génétiques, colonies de fourmis . . . etc.

Bibliographie

[1] E. AARTS , J. RÉDS et K. LENSTRA.*Local Search in Combinatorial Optimizatio*n. Princeton University Press, 2003.

[2] ADICHE.C.*Approche déterministe et stochastique de résolution de problèmes d'optimisation.*Thèse de Doctorat,Université des sciences et de la technologie HOUARI BOUMEDIENE, 2009.

[3] ARMENTANO, V. A., CLAUDIO, J. E. (2004). *An application of a multiobjective tabu search algorithm to a bicriteria owshop problem.* Journal of Heuristics , 10 (5), 463-481.

[4] ASLI.A.*Approche hybride pour les problémes d'optimisation combinatoire multiobjectif : cas des problèmes de type sac-à-dos.*Mémoire de magistère,Université des sciences et de la technologie HOUARI BOUMEDIENE, 2010.

[5] E. BALAS et E. ZEMEL.*An algorithm for large zero-one knapsack problems.* Operations Research, 28 :1130–1154, 1980.

[6] C. BAZGAN, H. HUGOT et D. VANDERPOOTEN.*Solving efficiently the 0-1 multi-objective knapsack problem.* Computers Operations Research, 36(1) :260–279, 2009.

[7] C. BAZGAN, H. HUGOT et D. VANDERPOOTEN.*Implementing an efficient FPTAS for the 0–1 multi-objective knapsack problem.* European Journal of Operational Research, 198(1) :47–56, 2009.

[8] V.BARICHARD.*Approches hybrides pour les problèmes multiobjectifs.* Thèse de doctorat,Université d'angers ,Novembre 2003.

[9] V.Barichard et J.HAO.*Un algorithme hybride pour le problème de sac à dos multi-objectifs.*Université d'Angers,2002.

[10] R. E. BELLMAN. *Dynamic Programming.* Princeton University Press, 1957.

[11] S. BALEV, N. YANEV, A. FRÉVILLE et R. ANDONOV. *A dynamic programming based reduction procedure for the multidimensional 0-1 knapsack problem.* European Journal of Operational Research, 186(1) :63–76, April 2008.

[12] P. CAPPANERA et M. TRUBIAN. *A Local-Search-Based Heuristic for the Demand-Constrained Multidimensional Knapsack Problem.* INFORMS Journal on Computing, 17(1) :82–98, 2005.

[13] C.A.COELLO and G.B. LAMONT. *Applications of Multi-Objective Evolutionary Algorithms.* world Scietific Publishing co., 2004.

[14] Y.COLLETTE ;P.SIARY.*Multiobjective optimisation.*Cranfield Bedford MK43 0AL UK,August 2003.

[15] D.W.CORNE and J.D.KNOWLES.*M-paes :a memetic algorithm for multiobjective optimization.*In Proceedings of the 2000 Congres on Evolutionary Computation, pages 325-332,2000.

[16] P.Czyżak and A.JASKIEWICZ.*Pareto Simulated Annealing - a metaheuristic technique for multiple objective combinatorial optimization.*Journal of Multi-Criteria Decision Analysis, 7 : 34-47, 1998.

[17] M. DAOUD.*Sur quelques problèmes d'optimisation multiobjectif.*Mémoire de magistère,Université des sciences et de la technologie HOUARI BOUMEDIENE, 2008.

[18] J. David SCHAFFER.*Multiple Objective Optimization with Vector Evaluated Genetic Algorithms.*Dans Proceedings of the 1st International Conference on Genetic Algorithms, pages 93–100,Mahwah,NJ, USA, 1985. Lawrence Erlbaum Associates, Inc.

[19] DHAENENS-FLIPO. c. *optimisation combinatoire multi-objectif :apport des méthodes coopératives et contribution à l'extraction de connaissances.* PhD thesis, Université des Sciences et Technologies de Lille U.F.R. d'I.E.E.A., 2005.

[20] F.Y. EDGEWORTH. *Mathematical Physics.* P. Keagan, London, 1881.

[21] M. EHRGOTT. *Multicriteria Optimization*, volume 491 de Lecture Notes in Economics and Mathematical Systems. Springer Verlag, Berlin, 2000.

[22] M.EHRGOTT, AND X.GANDIBLEUX, *An annotated bibliography of multiobjective combinatorial optimisation.*

[23] M. EHRGOTT. *A discussion of scalarization techniques for multiple objective integer programming.*Annals of Operations Research, 147(1) :343–360, 2006.

[24] M. EHRGOTT et X. GANDIBLEUX.*A survey and annotated bibliography of multiobjective combinatorial optimization*. OR Spektrum, 22(4) :425–460, 2000.

[25] M. EHRGOTT et X. GANDIBLEUX.*Multiple Criteria Optimization : State of the Art Annotated Bibliographic Surveys*, volume 52 de Kluwer's International Series in Operations Research and Management Science, pages 369–444. Kluwer Academic Publishers, 2002.

[26] M. EHRGOTT et X. GANDIBLEUX. *Approximative solution methods for multiobjective combinatorial optimization*. TOP, 12(1) :63, 2004.

[27] M. EHRGOTT, J. PUERTO et A. RODRIGUEZ-CHIA. *Primal-Dual Simplex Method for Multiobjective Linear Programming*. Journal of Optimization Theory and Application, 134(3) :483–497, September 2007.

[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] D. EL BAZ et M. ELKIHEL. *Load balancing methods and parallel dynamic programming algorithm using dominance technique applied to the 0-1 knapsack problem*. Journal of Parallel and Distributed Computing, 65(1) :74–84, 2005.

[30] A. FRÉVILLE. *The multidimensional 0-1 knapsack problem : An overview*. European Journal of Operational Research, 155 :1–21, 2004.

[31] A. FRÉVILLE et G. PLATEAU. *An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem*. discrete applied mathematics, 49(1-3) :189–212, 1994.

[32] K. FLESZAR et K. S. HINDI. Fast, *effective heuristics for the 0-1 multi-dimensional knapsack problem*. Computers Operations Research, 36(5) :1602–1607, 2009.

[33] T. FRIESZ, G.ANANDALINGAM ,N.MEHTA, K.NAM, ,S. SHAH, and R.TOBIN, (1993). *The multiobjective equilibrium network design problem revisited : A simulated annealing approach*. European Journal of Operational Research, 65 :44-57.

[34] L.GALAND. *Méthodes exactes pour l'optimisation multicritère dans les graphes : recherche de solutions de compromis*.Thèse de Doctorat, Université de Paris VI, UFR d'Ingénierie,2008.

[35] X. GANDIBLEUX et A. FRÉVILLE. *Tabu search based procedure for solving the 0-1 multiobjective knapsack problem : the two objectives case.* Journal of Heuristics, 6 :361–383, 2000.

[36] X.GANDIBLEUX ,N.MEZDAOUI and A.FRÉVILLE. *A multiobjective tabu search procedure to solve combinatorial optimization problems.*In lecture Notes in Economics and Mathematical Systems,volume 455,pages 291-300.Springer.

[37] M. GAREY and D. JOHNSON, *Computers and Intractability : a Guide to the Theory of NP-Completness*, w. H. Freeman and Company, San Francisco, USA, 1979.

[38] A.M. GEOFFRION. *Proper efficiency and the theory of vector minimization.* Journal of Mathematical Analysis and Applications, 22 :618-630, 1968.

[39] F. GLOVER. *A multiphase-dual algorithm for the zero-one integer programming problem.* Operations 25Research, 13(6) :879–919, 1965.

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

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

[42] F.GLOVER,M. LAGUNA, (1997). *Tabu search.* Kluwer Academic Publishers.

[43] K.GHAZLI. *Optimisation continue d'une fonction linéaire sur l'ensemble des solutions efficaces d'un problème linéaire stochastique multiobjectif.*Mémoire de magistère,Université des sciences et de la technologie HOUARI BOUMEDIENE, 2008.

[44] C. Gomes da SILVA, J. CLÍMACO et J. R. FIGUEIRA. *Core problems in 0, 1 Bicriteria Knapsack Problems.* Dans Proceedings of the 7th international conference on MultiObjective Programming and Goal Programming (MOPGP' 06), June 2006. 12-14 Juin 2006.

[45] Y.GUO, B.LIM, B.RODRIGUES and Y.ZHU. *Heuristics for a bidding problem.*In Journal of computers and Operations Research-Volume 33 no 8 :2179-2188,2006.

[46] P.HANSEN. *Bicriterion path problems. Dans G. FANDEL et T. GAL, réds., Multiple Criteria Decision Making Theory and Application, volume 177 de Lecture Notes in Economics and Mathematical Systems, pages 109–127.* Springer Verlag, Berlin, 1979.

[47] M.P.HANSEN. *Tabu search for multiobjective optimization :MOTS.*In proceedings of 13th International Conference on MCDM,1997.

[48] H.HARFOUCHE. *Contribution des métaheuristiques dans l'optimisation multiobjectif.*Mémoire de magistère,Université des sciences et de la technologie HOUARI BOUMEDIENE ,JUIN 2010.

[49] j.h.HOLLAND. *Adaptation in natural and artifcial systems.* Phd thesis, University of Michigan Press, 1975.

[50] E. HOROWITZ et S. SAHNI. *Computing partitions with applications to the knapsack problem.* Journal of ACM, 23 :277–292, 1974.

[51] H. HUGOT. *Approximation et énumération des solutions efficaces dans les problèmes d'optimisation combinatoire multiobjectifs.* Thèse de Doctorat, Université Paris-Dauphine, octobre 2007.

[52] G. P.INGAGORLIA et J. F. KORSH. *Reduction algorithm for zero-one single knapsack problem.* Management Science, 20 :460–463, 1973.

[53] J.JORGE. *Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires.* Thèse de Doctorat, Université de Nantes, Mai 2010.

[54] M. KONG, P. TIAN et Y. KAO. *A new ant colony optimization algorithm for the multidimensional Knapsack problem.* Computers Operations Research, 35(8) :2672–2683, 2008.

[55] J. LEMESRE, C. DHAENENS et E. G. TALBI. *An exact parallel method for a bi-objective permutation flowshop problem.* European Journal of Operational Research, 177(3) :1641–1655, 2007.

[56] S. MARTELLO et P. TOTH. *Knapsack Problems : Algorithms and Computer Implementations,* chapitre 1-4. John wiley sons, 1990.

[57] S. MARTELLO et P. TOTH. *A new algorithm for the 0-1 knapsack problem.* Management Science, 34(5) :633–644, 1988.

[58] H.MEUNIER. *Algorithmes évolutionnaires parallèles pour l'optimisation multiobjectif de réseaux de télécommunications mobiles.*PhD thesis,Thèse pour obtenir le grade de docteur. Université de Lille, 2002.

[59] R. J. MORAGA, G. w. DEPUY et G. E. WHITEHOUSE. *Meta-RaPs approach for the 0-1 multidimensional Knapsack problem.* Computers Industrial Engineering, 48(1) :83–96, 2005.

[60] T.MURATA et H. ISHIBUCHI. *MOGA : Multi-objective genetic algorithms*. Dans Proceedings of the 2nd IEEE International Conference on Evolutionary Computing, Perth, Australia, pages 289–294. IEEE Service Center, Piscataway, NJ, 1995.

[61] C.OLIVA. *Techniques Hybrides de Propagation de Contraintes et de Programmation Mathématique*. Thèse Université d'avignon et des pays de vaucluse.

[62] V. PARETO. *Cours d'économie politique*. Rouge, Lausanne, 1896.

[63] U. PFERSCHY. *Dynamic programming revisited : improving knapsack algorithms. Computing*, 63(4) :419–430, 1999.

[64] D. PISINGER. *A minimal algorithm for the 0-1 knapsack problem*. Operations Research, 45 :758– 767, 1997.

[65] A. PRZYBYLSKI, X. GANDIBLEUX et M. EHRGOTT. *Two phase algorithms for the bi-objective assignment problem*. European Journal of Operational Research, 185(2) :509–533, oct 2008.

[66] C.REEVES. *Modern HeuristicTecheniques for combinatorial Problems.*Advanced Topics in Computer Science.McGrawHill,London,1995

[67] A.SBIHI. *Les méthodes hybrides en optimisation combinatoire :algorithmes exacts et heuristiques.*Thèse de Doctorat, Université de PARIS-I,2003.

[68] P.SERAFINI. *Some consideration about computational complexity for multiobjective combinatorial problems.*In J.Jahn and w.Krabs, editors, Recent advances and historical development of vector optimisation volume 294 of Lecure Notes in Economics, 1987.

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

[70] N.SRINIVAS and K.DEB. *Multiobjective optimization using non dominated sorting genetic algorithms,*Evolutionary Computation, 2(3) : 221-248, 1994.

[71] E-G.TALBI. *Une Taxinomie des métaheuristiques hybrides.*DansROADEF'2000,2000.

[72] P. TOTH. *Dynamic programming algorithms for the Zero-One Knapsack Problem.* Computing,25(1) :29–45, 1980.

[73] E. L.ULUNGU. *Optimisation Combinatoire Multicritère : Détermination de l'ensemble des Solutions Efficaces et Méthodes Itératives*. PhD thesis, 1997.

[74] E. L. ULUNGU. *Optimisation combinatoire multicritère : détermination de l'ensemble des solutions efficaces et méthodes interactives*. Thèse de Doctorat, Université de Mons-Hainaut, Faculté des sciences, oct 1993.

[75] E. L. ULUNGU et J. TEGHEM. *The two phases method : An efficient procedure to solve biobjective combinatorial optimization problems.* Foundations of Computing and Decision Sciences, 20(2) :149–165, 1995.

[76] 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.

[77] J. VISÉE, J. TEGHEM, M. PIRLOT et E. L. ULUNGU. *Two-phases Method and Branch and Bound Procedures to Solve the Bi-objective Knapsack Problem.* Journal of Global Optimization, 12 :139- 155, 1998.

[78] C. WILBAUT. *Heuristiques hybrides pour la résolution de problèmes en variables 0-1 mixtes.*Thèse de Doctorat, Université de Valenciennes et du Hainaut-Cambresis, Septembre 2006.

[79] E.ZITZLER and L.THIELE. *An evolutionary algorithm for multiobjective optimization :The stength Pareto approach.* Technical report, Swiss Federal Institute of Technology (Zurich),1998.

Résumé

Résumé : *Approche multicritère pour le problème de sac à dos.*

Dans ce mémoire, nous nous intéressons à l'étude d'une méthode exacte pour la résolution d'un problème d'optimisation combinatoire multiobjectif, cas du problème de sac à dos bi-objectif unidimensionnel en variables binaires. Cette méthode est appelée la méthode en deux phases, son principe est basé sur les travaux de Visée et al, 1998 [77]. Nous étudions d'abord une procédure de pré-traitement qui permet de réduire la taille d'une instance d'un problème, appliquée au problème de sac à dos mono objectif, puis nous donnons plusieurs propriétés permettant de déterminer a priori une partie de la structure de toutes les solutions efficaces.Enfin, nous nous attachons à décrire l'application de la méthode de deux phases dans notre problème, ainsi la procédure de pré-traitment appliquée.

Mots clés : *Optimisation combinatoire multiobjectif, sac à dos, méthode exacte, méthode en deux phases, pré-traitement.*

Abstract : *multicriter approach for Knapsack problem.*

In this paper, we focus on the study of an exact method for solving a multiobjective combinatorial optimization problem, case of the unidimensional bi-criteria knapsack problem with binary variables. This method is called the method in two phases, its principle is based on the work of Visée et al, 1998 [77]. First, we study a preprocessing procedure that reduces the size of an instance of a problem, applied to the problem of single-objective knapsack, then we give several properties to determine a priori part of the structure of all efficients solutions. Finally, we strive to describe the application of the method of two phases in our problem, and the procedure of preprocessing Applied.

Keywords : *Multi-objective combinatorial optimization - Knapsack-exact method-Two phase method-preprocessing.*