$N°$ D'ORDRE : 02/2016$-D/MT$

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté de Mathématiques

THESE

Présentée pour l'obtention du grade de

DOCTEUR EN SCIENCES

En : MATHÉMATIQUES

Spécialité : Recherche Opérationnelle (Méthodes stochastiques)

Par : DAOUD MALIKA

Sujet

Exploitation de l'Information pour résoudre les problèmes d'Optimisation Combinatoire multicritères

Soutenue publiquement, le 15/12/2016, devant le jury composé de :

Mr KHELLADI Abdelkader	Professeur	à l'USTHB	Président.
Mr CHAABANE Djamal	Professeur	à l'USTHB	Directeur de thèse.
Mr ABBAS Moncef	Professeur	à l'USTHB	Examinateur.
Mr AIDENE Mohammed	Professeur	à l'UMMTO	Examinateur.
Mr BOUROUBI Sadek	Professeur	à l'USTHB	Examinateur.
Mr MOULAI Mustapha	Professeur	à l'USTHB	Examinateur.
Mr PIRLOT Marc	Professeur	à l'UMONS	Examinateur.
Mr RADJEF Mohammed Said	Professeur	à l'UAMB	Examinateur.

Remerciements

Je tiens à exprimer ma gratitude à mon directeur de thèse de Doctorat, le Professeur CHAABANE Djamal de l'Université des Sciences et de la Technologie Houari Boumedienne qui a bien voulu me proposer un sujet et d'avoir suivi le travail jusqu'à son terme. Ses remarques et ses conseils fructueux, m'ont beaucoup aider à réaliser ce travail.

Je remercie également le Professeur KHELLADI Abdelkader d'avoir accepté de présider ma soutenance, malgré ses nombreuses occupations.

J'exprime ma sincère reconnaissance aux membres du Jury, Monsieur ABBAS Moncef, Professeur à l'USTHB, Monsieur AIDENE Mohammed, Professeur à l'UMMTO, Monsieur BOUROUBI Sadek, Professeur à l'USTHB, Monsieur MOULAI Mustapha, Professeur à l'USTHB, Monsieur PIRLOT Marc, Professeur à l'UMONS et Monsieur RADJEF Mohammed Said, Professeur à l'UAMB qui m'ont fait l'honneur d'accepter d'êtres examinateurs de ce travail.

Qu'il me soit aussi permis de remercier tous ceux qui, par leur présence et leur amitié, ont contribué à ce travail ; la liste est longue et je risque d'oublier des noms.

Je remercie enfin pour leurs soutiens les membres de ma famille et j'adresse un grand remerciement à mon père et ma mère qui m'ont encouragé à poursuivre mes études.

Résumé

La résolution des problèmes multiobjectifs, particulièrement les problèmes d'optimisation combinatoire multiobjectifs se fait ou bien dans l'espace des critères ou dans l'espace de décision en utilisant la résolution à posteriori des programmes linéaires. Cependant, nous avons introduit à travers cette thèse une nouvelle vision pour résoudre un de ces types de problèmes, il s'agit du problème de sac à dos multiobjectif. Deux contributions sont issues de cette vision, la première consiste à développer une nouvelle méthode de réduction montrant la valeur ajoutée dans l'optimisation combinatoire, particulièrement pour le problème de sac à dos biobjectif suivi des expériences numériques comparées aux résultats de Jorge et Gandibleux (2008).

Pour la deuxième contribution, afin de valider l'utilité de la méthode proposée antérieurement, nous l'intégrons dans une méthode de résolution exacte proposée par Jahanshahloo et al.(2005) et nous terminons par des expériences numériques sur quelques instances étudiées.

Les résultats obtenus sont comparés à ceux existants, indiquant l'efficacité de la réduction développée pour toutes les instances étudiées.

Mots clés : problème de sac à dos, solution efficace, réduction, prétraitement, variable régulière, problème d'optimisation multiobjectif.

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Prendre une décision est l'action de tous les jours, le rôle principal de la recherche opérationnelle est d'aider à prendre cette décision en vue d'une gestion efficace, rationnelle et logique.

Le problème de sac à dos est un problème classique qui fait partie des problèmes d'optimisation combinatoire les plus étudiés pendant plusieurs années, en raison de ses nombreuses applications dans le monde réel. En effet, il intervient souvent comme sous problème dans de nombreux domaines et concerne des problèmes de logistique comme le chargement d'avions ou de bateaux, de productive, de finances et d'économie comme la gestion de portefeuilles ou dans l'industrie comme la découpe de matériaux. Malgré la simplicité de la structure de ce problème en variables binaires : une seule contrainte et uniquement des coefficients entiers positifs, il appartient à la classe des problèmes NP-difficile. Cela explique le nombre important d'ouvrages qui lui sont consacrés [60, 70], mais aussi les différents travaux proposant diverses méthodes de résolution. Il existe de nombreuses extensions du problème de sac à dos, selon le domaine des variables (valeurs binaires, entières ou réelles), le nombre de contraintes (unidimensionnel, bidimensionnel ou multidimensionnel), le nombre de profits associés à chaque objet (unicritère ou multicritère), le nombre de sacs,..., etc.

La méthode de séparation et évaluation et la programmation dynamique sont des méthodes de résolution exactes capables de garantir une solution optimale de bonne qualité pour le problème de sac à dos.
L'application directe de ces dernières méthodes peut engendrer un temps d'exécution important, et rapidement les méthodes métaheuristiques deviennent l'unique moyen d'obtenir une solution approchée en un temps raisonnable. La méthode classique telle que la programmation dynamique et la séparation et évaluation peuvent être combinées de manière efficace pour donner naissance à des méthodes coopératives. Afin de diminuer le temps de résolution de manière économique, l'apparition de nouvelles méthodes de réduction semblent particulièrement intéressantes pour la résolution des problèmes d'optimisation combinatoire [4, 84]. Le problème de sac à dos unicritère se résout de manière assez efficace et plusieurs travaux portent sur différentes extensions du problème de sac à dos qui sont beaucoup plus difficiles à résoudre [10, 65].

Les recherches dans le domaine combinatoire sont longtemps restées confinées aux situations où un seul objectif est à optimiser. Cependant, de nombreux cas réels nécessitent l'optimisation simultanée de plusieurs objectifs conflictuels. En fait, la résolution exacte des problèmes d'optimisation combinatoire multiobjectifs est difficile, tant d'un point de vue théorique que pratique, ils sont tous NP-difficiles [67, 93], même lorsque la version monoobjectif du problème appartient à la classe de complexité P [93].

Les travaux d'Ulungu [97] ont été dédiés à l'optimisation combinatoire multiobjectif. L'auteur a observé que les travaux concernant la résolution de certains problèmes monoobjectifs avaient produit des méthodes de résolution très efficaces. Il a cherché à exploiter la structure particulière de ces problèmes dans le contexte multiobjectif, d'où la naissance de la méthode dite en deux phases. Comme son nom l'indique, cette méthode procède au calcul des solutions efficaces en deux phases ; la première calcule les solutions dites supportées tandis que la seconde calcule des solutions dites non-supportées. La méthode a été appliquée à de nombreux autres problèmes d'optimisation combinatoire multiobjectifs. Cependant, les expérimentations numériques montrent que cette méthode peine à passer l'augmentation de la taille des instances résolues. Aussi, elle a été adaptée au problème de sac à dos biobjectif par Visée et al. [98] en 1998 ; dans la deuxième phase de la procédure, les auteurs utilisent une procédure de séparation et évaluation pour le cas monoobjectif de Martello et Toth [70], dans le but de générer des solutions efficaces non-supportées. Ils ont montré que le nombre de solutions non-supportées est très élevé comparant à celui des solutions supportées. De plus, la méthode en deux phases est restée limitée au cas biobjectif [96, 98], jusqu'aux travaux de Przybylski [83]. Ce dernier a proposé une généralisation de la méthode à trois objectifs et plus, ainsi qu'une nouvelle procédure de calcul des solutions non-supportées, en utilisant une procédure dite ranking. En effet, l'application de telle procédure permet d'explorer l'espace de recherche d'une manière performante, en découvrant les solutions dans un ordre favorisant les plus efficaces. Du fait que la méthode en deux phases s'appuie fortement sur la structure du problème à résoudre, il suffit d'une petite variation dans la formulation du problème pour que la procédure en deux phases perde son efficacité, elle ne peut pas être applicable si la structure du problème est changée. Captivo et al. ont proposé en 2003 une méthode exacte basée sur une transformation du problème de sac à dos multiobjectif à un problème du plus court chemin, la performance de leur algorithme est meilleure que celui de Visée et al. [98]. D'autres méthodes exactes et approchées seront exposées au cours de cette thèse.

L'efficacité des méthodes de résolution appliquées au problème de sac à dos monoobjectif est basée sur la notion du core, tel que de nombreux tests expérimentaux sur une large variété d'instances de problèmes de sac à dos ont montré que seul un sous-ensemble contenant un nombre relativement faible d'objets sert à la détermination d'une solution optimale. En effet, les objets pour lesquels l'efficacité est très grande auront de grandes chances de faire partie des solutions optimales, tandis que ceux ayant une efficacité plus proche de zéro seront probablement dans aucune de ces solutions. Le plus souvent, seules les variables dont l'efficacité est moyenne seront déterminantes aux solutions optimales. Ceci est particulièrement vrai si le nombre d'objets est très grand. Cette observation a mené Balas et Zemel [4] à définir la notion du core ; l'ensemble qui contient les objets dont l'efficacité est moyenne. Puchinger et al. [84] ont élargi cette notion du core pour le cas du problème de sac à dos multidimensionnel. Malgré que cette notion n'est pas immédiate pour le cas multiobjectif, Gomes da Silva et al. ont proposé une extension du cas monoobjectif unidimensionnel au cas biobjectif unidimensionnel [44]. Mavrotas et al. ont proposé une extension du cas biobjectif unidimensionnel au cas biobjectif multidimensionnel [73]. Cependant, le fait que le core soit défini à partir de la connaissance d'une solution optimale rend difficile son utilisation dans une méthode de résolution. Ainsi, plusieurs auteurs ont proposé diverses méthodes pour calculer une estimation du core [4, 71, 78]. Jorge et Gandibleux ont proposé de nouvelles propriétés qui servent à réduire a priori la taille du problème de sac à dos multiobjectif [57, 58].

La résolution des problèmes multiobjectifs, particulièrement les problèmes d'optimisation combinatoire multiobjectifs se fait en utilisant la résolution à posteriori des programmes linéaires. Cependant, nous avons introduit à travers cette thèse une nouvelle vision pour résoudre un de ces types de problèmes, il s'agit du problème de sac à dos multiobjectif unidimensionnel en variables binaires.
D'abord, nous étudions l'efficacité de la procédure de réduction proposée par Jorge et Gandibleux [57], nous identifions plusieurs instances auxquelles leur procédure ne fournit aucune réduction de la taille du problème considéré.

Dans la première partie de cette thèse, nous développons une stratégie de réduction en exploitant l'information de données, nous mettons en évidence plusieurs propriétés qui permettent de déterminer a priori la valeur de certaines variables dans toutes les solutions efficaces avant la résolution du problème "prétraitement". La méthode de réduction développée est basée sur l'espace de données, utilise des solutions extrêmes, la borne inférieure et supérieure de la cardinalité d'une solution efficace et la notion d'efficacité d'objet qui est populaire dans le cas monoobjectif et presque absente dans le cas multiobjectif. Cette première contribution a fait l'objet d'une publication dans la revue internationale "INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH", intitulée "New reduction strategy in the biobjective knapsack problem" et présentée oralement dans une conférence internationale, 23rd International Conference on Multiple Criteria Decision Making, MCDM'15, intitulée : Data Pre-treatment for Solving bi-objective Knapsack Problem.

Dans une deuxième partie, nous intégrons la méthode de réduction développée antérieurement dans la méthode de résolution exacte [54] pour valider son utilité. Ainsi, la méthode de résolution comporte deux phases ; la première phase consiste à réduire la taille du problème considéré tandis que la deuxième résout le problème résiduel tiré de la première phase. Cette deuxième contribution a fait l'objet d'une présentation orale intitulée : Effect of Reduction Strategy in the Binary Bi-Objective Knapsack Problem, dans une conférence internationale ; The 11th international Conference on Multiple Objective Programming and Goal Programming (MOPGP'15).

Dans cette thèse, on présente six chapitres :
Le premier chapitre présente le problème de sac à dos uniobjectif et ses extensions, en éclaircissant quelques méthodes de résolution de sa version unidimensionnel, suivi par les procédures de réduction appliquées en prétraitement, initialisées par Balas et Zemel [4] dans sa version unidimensionnel et les travaux de Puchinger et al. [84] en version multidimensionnel.
L'optimisation multiobjectif est présentée dans la première partie du deuxième chapitre, quelques notions de base concernant l'ensemble de solutions efficaces, notion de dominance et enfin les méthodes de résolution classiques. Tandis que la deuxième partie est consacrée à l'optimisation multiobjectif en nombres entiers, en déterminant la notion de solutions supportées et non-supportées, et quelques méthodes de résolution.
Le troisième et quatrième chapitre, présentent les problèmes d'optimisation combinatoire multiobjectifs (MOCO), et le problème de sac à dos multiobjectif (MOKP) ainsi que leurs méthodes de résolution les plus étudiées.
Le cinquième chapitre est autour de la notion du prétraitement, nous présentons les travaux de Gomes et al. (2008) concernant le problème de sac à dos biobjectif et les travaux de Mavrotas et al. (2009) pour le cas multiobjectif multidimensionnel. A la fin du chapitre, nous exposons les travaux de Jorge et Gandibleux (2008) dédiés au problème de sac à dos biobjectif.

Nos contributions font partie du sixième et dernier chapitre qui est divisé en deux parties, la première porte sur un développement d'une nouvelle méthode de réduction, basée sur l'information de données [13, 22, 23], tandis que la deuxième partie porte sur une intégration de la procédure de réduction proposée antérieurement dans la méthode exacte développée par Jahanshahloo et al. [54] dans le but de réduire sa taille et de résoudre le problème de sac à dos multiobjectif en un temps raisonnable. Des expérimentations numériques portant sur plusieurs types d'instances sont présentées à la fin de ce chapitre.

Chapitre 1

Abécédaire sur le problème de sac à dos unicritère (KP)

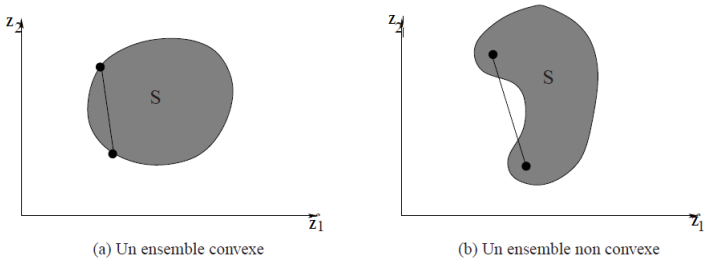
1.1 Introduction

La programmation linéaire a pour objet l'étude et la résolution des problèmes d'optimisation dans lesquels la fonction objectif aussi bien que les contraintes sont des fonctions linéaires en variables de décision. Un programme linéaire est un problème dans lequel les variables de décision sont des réels qui doivent satisfaire un ensemble d'équations et (ou) d'inequations linéaires (dites contraintes) et la valeur d'une fonction linéaire de ces variables appelée "fonction objectif" doit être rendue maximum (ou minimum).
Un problème d'optimisation combinatoire est défini à partir d'un ensemble fini X et d'une application $Z : X \to \mathbb{R}$, il s'agit de déterminer $\tilde{x} \in X$ tel que $Z(\tilde{x}) = \min_{x \in X} (Z(x))$ ou $Z(\tilde{x}) = \max_{x \in X} (Z(x))$.
L'optimisation combinatoire trouve des applications dans des domaines variés : la gestion, l'ingénierie, la conception, la production, les télécommunications, les transports, l'énergie, les sciences sociales et l'informatique,..., etc. De nombreuses applications peuvent être modélisées sous la forme d'un problème d'optimisation combinatoire comme le problème du voyageur de commerce (Traveling Salesman Problem), le problème d'affectation (Assignment Problem), l'ordonnancement de tâches, le problème de couverture (Set Covering Problem) et le problème de sac à dos (Knapsack Problem) appelé aussi problème de chargement.
Dans ce chapitre, nous nous intéressons au problème du sac à dos dans sa version la plus simple dont on présentera ainsi quelques unes de ses nombreuses extensions, citons le problème du sac à dos en variables entières (IKP), le problème du sac à dos multidimensionnel (MKP), le problème du sac à dos multiple ($m-KP$). Des méthodes usuelles de résolution existantes et des procédures de réduction sont présentées à la fin de ce chapitre.

Définition 1.1 La convexité

- Un ensemble S est convexe si pour n'importe quels deux points distincts de cet ensemble, le segment qui relie ces deux points est contenu dans l'ensemble S.
  Autrement dit : un ensemble S est dit "convexe" si :

$$\forall\, x_1, x_2 \in S \text{ et } 0 \leq \lambda \leq 1 \ : \ \lambda x_1 + (1-\lambda) x_2 \in S. \tag{1.1}$$



(a) Un ensemble convexe (b) Un ensemble non convexe

FIGURE 1.1 – Exemples d'ensemble convexe et d'ensemble non convexe

- Une fonction Z d'un convexe S à valeurs dans $\mathbb{R}^n$ est dite convexe si :

$$\forall\, x_1, x_2 \in S \text{ et } 0 \leq \lambda \leq 1 : Z(\lambda x_1 + (1-\lambda)x_2) \leq \lambda Z(x_1) + (1-\lambda)Z(x_2). \quad (1.2)$$

- Le programme mathématique $\max_{x \in X}(Z(x))$ ou $\min_{x \in X}(Z(x))$ est un programme "convexe" si S est un ensemble convexe et si Z est une fonction convexe.

1.2 La version de base de problème de sac à dos

Le problème du sac à dos fait partie des problèmes d'optimisation combinatoire les plus étudiés depuis des décennies. Il peut être traité comme une réduction d'un problème général de la programmation linéaire en variables binaires.
Le contexte de ce problème est le suivant :

- Soit un container (ou cargo ou camion ou satellite ou...) de capacité limité ω en poids (ou en volume).
- n objets, caractérisés par leur valeur c_i et leur poids (ou volume) w_i.

Le problème consiste à définir le chargement optimal du container à l'aide des objets disponibles : quels objets charger dans le container de manière à maximiser la valeur totale du chargement, tout en respectant la limite de capacité du container ?
Le nom "Knapsack" (sac à dos) noté (KP), est folklorique, lui vient de la situation du campeur qui doit sélectionner des objets parmi un grand nombre d'objets possible à emporter dans son sac ; utilisé originellement pour ce problème, le nom lui est resté. Il est aussi parfois dénommé "Cargo loading problem".

Sa formule mathématique peut s'écrire comme suit :

$$(\text{KP}) \begin{cases} \max(Z) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ x_i \in \{0,1\}, i = 1, \ldots, n \end{cases} \quad (1.3)$$

- Nous supposons que tous les profits c_i et les poids w_i, pour tout $i = 1, \ldots, n$ et la capacité du sac à dos ω sont des entiers positifs. Si ce n'est pas le cas, on pourra faire une transformation pour satisfaire cette hypothèse, voir : [60, p. 10] et [93, p. 216].
- Pour éviter toutes solutions triviales, nous allons imposer un certain nombre d'hypothèses sur les données d'entrée du problème du sac à dos qui sera valable et applicable sur la plupart des extensions de (KP) :

1. $\forall i \in \{1, 2, \ldots, n\},\ w_i \leq \omega$,
2. $\sum_{i=1}^{n} w_i > \omega$.

- L'hypothèse (1.) assure que chaque objet peut être chargé dans le sac à dos.
- L'hypothèse (2.) évite le cas triviale où le sac contient tous les objets.

Le problème de sac à dos unidimensionnel se particularise par rapport à un problème linéaire général en variables binaires par l'existence d'une unique contrainte. Il existe plusieurs contextes d'applications pour ce problème, nous citons à titre d'exemples ;

- Le problème d'investissement "Capital Budgeting Problem" : sélection d'investissement $i \in \{1, 2, \ldots, n\}$, nécessitant un capital w_i et correspondant à un rendement c_i et le capital total disponible est représenté par ω.
- Le problème de découpe "Cutting Stock Problem" : découpe d'une plaque rectangulaire (carton, acier, verre, ...) de longueur ω, en différentes pièces possibles $i \in \{1, 2, \ldots, n\}$, de même largeur que la plaque et de longueur pré-définie w_i ; c_i représente le profit correspondant à la pièce i.

Le cas particulier dans lequel le profit et le poids des objects sont les mêmes, $w_i = p_i; i = 1, \ldots, n$ correspond au problème de la somme des sous-ensembles, connu sous le nom de "Subset-sum problem".

Dans cette situation, il convient de trouver le sous-ensemble $J \subset \{1, \ldots, n\}$ tel que la valeur $\sum_{i \subset J} w_i$ est maximale, et soit inférieure à ω. Par exemple dans le problème de découpe, l'objectif est de minimiser la perte ou le déchet $\omega - \sum_{i \subset J} w_i$ de la découpe.

1.2.1 Les extensions du problème de sac à dos (KP)

Parmi les nombreuses extensions du problème de sac à dos citons le cas unidimensionnel, bidimensionnel ou multidimensionnel, aussi selon le domaine des variables ; valeurs binaires, entières ou réelles, selon le nombre de sacs,..., etc. Cette section présente quelques unes d'entre elles [60, 93] :

- (IKP) : le problème du sac à dos en variables entières tel que les variables x_i binaires sont remplacées par des variables x_i entières et bornées $0 \leq x_i \leq b_i$; chaque objet i est disponible pour le chargement en b_i exemplaires.
- (MKP) : le problème du sac à dos à m dimensions ou le problème du sac à dos à multi-dimensionnels , chaque objet i est caractérisé par différentes valeurs w_{ji}, $j = 1, \ldots, m$, (poids, volume,...) et les objets chargés doivent vérifier les m contraintes
$$\sum_{i=1}^{n} w_{ji} x_i \leq \omega_j,\ j = 1, \ldots, m,$$
où $\omega_j,\ j = 1, \ldots, m$ sont les capacités correspondantes du container.

$$\text{(MKP)} \begin{cases} \max(Z) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_{ji} x_i \leq \omega_j \ j = 1, \ldots, m \\ x_i \in \{0, 1\}, i = 1, \ldots, n \end{cases} \tag{1.4}$$

– $(m-KP)$: le problème du sac à dos multiple tel qu'il existe m containers, de capacité respective $\omega_j; j = 1, \ldots, m$, pouvant accueillir les différents objets. Avec les variables binaires, pour $j = 1, \ldots, m$, et $i = 1, \ldots, n$:

$$x_{ji} = \begin{cases} 1 & \text{si l'objet } i \text{ est chargé dans le container } j, \\ 0 & \text{sinon.} \end{cases} \tag{1.5}$$

Le problème peut s'écrire comme suit :

$$(m-KP) \begin{cases} \max \sum_{j=1}^{m} \sum_{i=1}^{n} c_i x_{ji} \\ \sum_{i=1}^{n} w_i x_{ji} \leq \omega_j \; j = 1, ..., m \\ \sum_{j=1}^{m} x_{ji} \leq 1 \; i = 1, ..., n \\ x_{ji} \in \{0,1\}, j = 1, \ldots, m, \; i = 1, \ldots, n \end{cases} \tag{1.6}$$

Remarque 1.1 Le problème $(m-KP)$ est un sous problème du problème d'affectation généralisée (GAP) [93].

– $(m-CKP)$: le problème du sac à dos à choix multiple ; l'ensemble des objets est divisé en m classes ($j = 1, \ldots, m$), la classe j contenant N_j objets. L'objet $j \in \{1, \ldots, N_j\}$ de la classe j est caractérisé par son poids w_{ji} et sa valeur c_{ji}. Le sac à dos doit contenir un objet unique de chaque classe, les m objets sélectionnés doivent maximiser la valeur du chargement afin de ne pas dépasser la capacité ω du sac à dos.
Avec les variables binaires pour $j = 1, \ldots, m, \; i = 1, \ldots, N_j$.

$$x_{ji} = \begin{cases} 1 & \text{si l'objet } i \text{ de la classe } j \text{ est sélectionné,} \\ 0 & \text{sinon.} \end{cases} \tag{1.7}$$

Sa formulation mathématique et d'autres extensions du problème de sac à dos sont détaillées [60, 93], nous terminons ce paragraphe par présenter le problème de sac à dos en variables continues (LKP) dit problème relaxé, ce dernier est obtenu en remplaçant $x_i \in \{0,1\}$ par $x_i \in [0,1]$.

$$(\text{LKP}) \begin{cases} \max(Z) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ 0 \leq x_i \leq 1, \; i = 1, \ldots, n \end{cases} \tag{1.8}$$

Sa résolution s'appuie sur des notions et concepts définis par la suite. Dans la section suivante, nous nous intéresserons au problème du sac à dos unicritère dans sa version de base.

1.3 Relaxation et bornes supérieures et bornes inférieures

Les méthodes de relaxation sont généralement utilisées pour calculer une borne supérieure ou une borne inférieure de la valeur optimale du problème de sac à dos (KP) en se ramenant à des problèmes pour lesquelles des méthodes de résolution efficaces existent. Nous trouvons dans la littérature, la relaxation continue (LKP) qui consiste à relâcher la contrainte d'intégrité des variables, la relaxation lagrangienne qui consiste à étendre le problème au problème relaxé $L(KP, \lambda)$ où ce dernier ne contient pas des contraintes, qui sont inclus dans la fonction objectif comme un terme de pénalité. Geoffrion [40] et Fisher [34] ont montré que cette relaxation pouvait être utilisée de manière efficace pour la résolution du problème de sac à dos multidimensionnel (MKP) [70, p. 19], il existe une relaxation alternative appelée relaxation surrogate ou agrégée ($S(\lambda)$) introduite pour la première fois par Glover [43]. Elle consiste à combiner toutes les contraintes en une seule contrainte. Il existe d'autres relaxations dans [70, 60, 10, 65].

1.3.1 Bornes supérieures et bornes inférieures de Dantzing

Le calcul des bornes inférieures et supérieures est un point important permettant d'affiner les performances de la description de la méthode de résolution, comme l'on verra lors de la description de la méthode Branch & Bound.
Le calcul des bornes nécessaires se fait au début de l'algorithme et elles sont calculées à chaque fois que des variables sont fixées. Le calcul des bornes supérieures est souvent le processus qui consomme le plus de temps d'exécution. Pour cette raison, il est souvent intéressant de résoudre la relaxation continue du problème (KP). La relaxation continue présente un avantage du point de vue de temps de résolution qui résulte de l'utilisation de la méthode du simplexe. En élargissant l'espace des solutions réalisables, nous acceptons de nouvelles solutions hors espace. La plus grande valeur de la fonction objectif associée à ces solutions nous conduit à une borne supérieure. Tandis que pour le calcul des bornes inférieures le premier choix est retenu à partir de l'algorithme 3. Entre autres quelques bornes existantes dans la littérature seront exposées :

a. **Bornes supérieures** :
   - La borne supérieure la plus simple du problème du sac à dos (KP), peut être obtenue en prenant la solution optimale de la relaxation continue du problème du sac à dos (LKP) et en considérant son arrondi à la valeur entière inférieure. On expose le déroulement de la recherche de cette borne développée par Dantzing [21], en présentant quelques définitions importantes et utiles :
     L'important et fameux résultat du problème sac à dos uniobjectif unidimensionnel (KP) est présenté comme suit :
     En 1957, Dantzing [21] a montré qu'il est facile d'obtenir une solution optimale du problème sac à dos relaxé ; les n articles sont triés par ordre décroissant d'efficacité (non increasing profit-to-weight ratios), et faire entrer dans le sac objet par objet jusqu'a ce que le sac soit rempli, il existe un objet noté b qui ne peut pas entrer entièrement dans le sac, cet objet est appelé objet bloquant ou critique ou objet de séparation (the break or critical item or split item).
     La détermination d'une solution optimale du problème de sac à dos relaxé est basée sur les définitions suivantes :

Définition 1.2 On appelle "tightness ratio" le rapport de la capacité du sac à dos sur la somme des poids des objets :

$$r = \frac{\omega}{\sum_{i=1}^{n} w_i}. \tag{1.9}$$

Définition 1.3 Efficacité d'un objet
On appelle efficacité d'un objet i le rapport de son profit sur son poids, noté par $\frac{c_i}{w_i}$.

Supposons que les objets sont triés dans l'ordre décroissant des rapports :

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots \geq \frac{c_n}{w_n}. \tag{1.10}$$

Définition 1.4 Objet de séparation (Element bloquant, Objet critique)
On appelle élément bloquant le premier objet qui ne peut entrer entièrement dans le sac à dos lorsque les objets sont ajoutés par l'ordre cité par l'equation (1.10).
Notons b l'indice correspondant au :

$$b = \min\{j \mid \sum_{i=1}^{j} w_i > \omega\}, \tag{1.11}$$

cet objet vérifie :

$$\sum_{i=1}^{b-1} w_i \leq \omega < \sum_{i=1}^{b} w_i.$$

Théorème 1.1 *Solution optimale de la relaxation linéaire [70]*
Une solution optimale $\widehat{x}$ du problème relaxé (LKP) est donnée par :

$$(\widehat{x}_i^{LKP}) = \begin{cases} 1 \quad si \quad i < b, \\ \dfrac{\omega - \sum_{i=1}^{b-1} w_i}{w_b} \quad si \quad i = b, \\ 0 \quad si \quad i > b. \end{cases} \tag{1.12}$$

La valeur optimale $Z^*(LKP)$ d'une instance est obtenue en prenant les objets dans l'ordre (1.10), jusqu'à l'élément bloquant puis on ajoute la fraction de cet élément permettant de saturer le sac à dos, ainsi on aboutit à la borne supérieure présentée par Dantzing [21] qui est exprimée comme suit :

$$Z^*(LKP) = \sum_{i=1}^{n} c_i \widehat{x}_i^{LKP} = \sum_{i=1}^{b-1} c_i + (\omega - \sum_{i=1}^{b-1} w_i)\frac{c_b}{w_b}.$$

Compte tenu de l'intégralité de c_i et x_i, on a donc :
$U_1 = \lfloor Z^*(LKP) \rfloor = \sum_{i=1}^{b-1} c_i + \lfloor (\omega - \sum_{i=1}^{b-1} w_i)\frac{c_b}{w_b} \rfloor$.

- Martello et Toth [72], [70, p. 20] ont proposé une autre borne supérieure améliore celle donnée par Dantzing, en considérant les deux cas où l'objet b est inclus ou exclu de la solution, on considère les deux solutions suivantes :

$$(\widehat{x}_i^0) = \begin{cases} 1 \quad \text{si} \quad i < b, \\ \dfrac{\omega - \sum_{i=1}^{b-1} w_i}{w_{b+1}} \quad \text{si} \quad i = b+1, \\ 0 \quad \text{si} \quad \text{sinon.} \end{cases} \tag{1.13}$$

$$(\widehat{x}_i^1) = \begin{cases} 1 \quad \text{si} \quad i < b-1 \ \text{ et } \ i = b, \\ \max\{0, 1 - \dfrac{w_b - (\omega - \sum_{i=1}^{b-1} w_i)}{w_{b-1}}\} \quad \text{si} \quad i = b-1, \\ 0 \quad \text{si} \quad \text{sinon.} \end{cases} \tag{1.14}$$

En effet, U^1 et U^2 correspondent respectivement aux deux branches d'une alternative : soit $x_b = 0$, et alors la capacité restante est remplie au mieux avec une fraction de x_{b+1}, soit $x_b = 1$, et alors le dépassement de la capacité est réduit au mieux avec une fraction de x_{b-1}.

$$U^0 = \sum_{i=1}^{n} c_i \widehat{x}_i^0 = \sum_{i=1}^{b-1} c_i + \lfloor (\omega - \sum_{i=1}^{b-1} w_i) \frac{c_{b+1}}{w_{b+1}} \rfloor,$$

et

$$U^1 = \sum_{i=1}^{n} c_i \widehat{x}_i^1 = \sum_{i=1}^{b-1} c_i + c_b + \lfloor (\omega - \sum_{i=1}^{b-1} w_i - w_b) \frac{c_{b-1}}{w_{b-1}} \rfloor.$$

La borne supérieure proposée par Martello et Toth [72] est la suivante : $U_2 = \max\{U^0, U^1\}$. Il est clair que $U_2 \leq U_1$.

D'autres bornes supérieures ont été définies dans [70]. Ces bornes sont peu utilisées dans les méthodes de résolution.

b. **Bornes inférieures** :

Une borne inférieure doit avoir une propriété importante, elle doit correspondre à une solution réalisable pour le problème initial. Il s'agit donc de construire une procédure qui permet de trouver rapidement des solutions approchées du problème. Plusieurs méthodes heuristiques sont proposées dans la littérature [10, p. 48].

1.4 Méthodes de résolution

Nous présentons dans cette section quelques méthodes de résolution exactes et heuristiques.

1.4.1 Procédure de séparation et évaluation (Branch & Bound)

La méthode Branch & Bound dite séparation et évaluation est l'une des méthodes les plus connues pour la résolution des problèmes d'optimisation combinatoire NP-difficiles. Il est à noter que l'énumération de l'ensemble des solutions est souvent peu

réaliste en raison de l'importance de son cardinal. Le concept algorithmique derrière les procédures de séparation et évaluation consiste en une énumération partielle de l'espace des solutions d'une manière précise et intelligente. En général, seule une petite partie de l'espace est explorée explicitement et il est garanti que l'espace non considéré de manière explicite ne contient aucune solution optimale. La recherche par décomposition de l'ensemble des solutions peut être représentée graphiquement par un arbre :

- Chaque sous problème crée au cours de l'exploration est schématisé par un noeud de l'arbre (ou sommet), le noeud racine représentant le problème initial.
- Les branches de l'arbre schématisent le processus de séparation. Elles représentent la relation entre les noeuds.
- Lors de la séparation, un noeud crée un ensemble de noeuds.

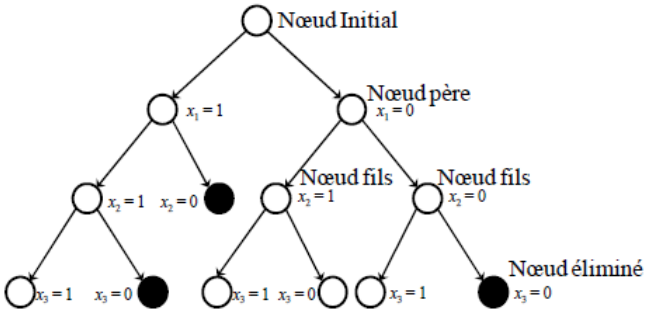


FIGURE 1.2 – Arborescence de Branch & Bound

La méthode de Branch & Bound est basée sur trois principes :

Le principe de Séparation

Le principe de séparation consiste à diviser le problème en un certain nombre de sous problèmes Sol' de l'espace des solutions réalisables Sol. En résolvant tous les sous problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation est appliqué de manière recursive à chacun des sous ensembles tant que celui-ci contient plusieurs solutions.

La procédure de séparation d'un ensemble s'arrête lorsqu'on connaît la meilleure solution de l'ensemble ou on connaît une solution meilleure que toutes celles de l'ensemble ou encore on sait que l'ensemble ne contient aucune solution admissible.

Le principe d'évaluation

Le principe d'évaluation a pour objectif connaître la qualité des noeuds à traiter. En utilisant deux types de bornes, une borne inférieure $\underline{Z}$ et une borne supérieure $\overline{Z}$ du problème initial. La connaissance d'une borne inférieure et supérieure de chaque sous problème permet de stopper l'exploration d'un sous ensemble de solutions qui ne sont pas candidates à l'optimalité : si pour un sous problème la borne supérieure est plus petite que la borne inférieure du problème, l'exploration du sous ensemble correspondant est inutile. D'autre part, lorsque le sous ensemble est suffisamment petit, on procède à une énumération explicite, on résout alors le sous problème correspondant.

Stratégie de parcours :

La stratégie de parcours est la règle qui permet de choisir le prochain sommet à séparer parmi l'ensemble des sommets de l'arborescence. On peut citer quelques strategies de parcours les plus connues :

- **La profondeur d'abord** : l'exploration privilégie les sous problèmes obtenus par le plus grand nombre de séparations appliquées au problème de départ, (les sommets les plus éloignes de la racine).
- **La largeur d'abord** : cette stratégie favorise les sous problèmes obtenus par le moins de séparations du problème de départ, c'est à dire les sommets les plus proches de la racine (de profondeur la moins élevée).
- **Le meilleur d'abord** : cette stratégie consiste à explorer des sous-problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

Algorithme 1: Branch & Bound pour (KP)

```
1  PSE(↕ x, ↓ i, ↕ x*)
   Input :
   ↓ i : l'indice de l'objet sur lequel opérer
   Output :
   ↕ x : la solution en cours de construction
   ↕ x* : la meilleure solution trouvée.
2  − | La solution est elle réalisable ?
3  if Σ_{j=1}^{i-1} w_i x_i ≤ ω then
4  |  − | Mise à jour de la meilleure solution connue.
5  |  if Z(x) > Z(x*) then
6  |  |  x* := x
7  |  end
8  |  if i ≤ n then
9  |  |  Calculer une borne supérieure Z̄
10 |  |  − | Séparation de la recherche sur les sous espaces disjoints vérifiant
   |  |  x_i = 1 ou x_i = 0.
11 |  |  if Z̄ > Z(x*) then
12 |  |  |  x_i := 1
13 |  |  |  PSE(↕ x, ↓ i + 1, ↕ x*)
14 |  |  |  x_i := 0
15 |  |  |  PSE(↕ x, ↓ i + 1, ↕ x*)
16 |  |  end
17 |  end
18 end
```

De nombreuses procédures de séparation et évaluation ont été proposées, les travaux de Martello et Toth [70] font références à ce sujet.

1.4.2 Programmation dynamique

La programmation dynamique est une méthode classique de résolution exacte qui peut être utilisée pour la résolution d'un grand nombre de problèmes d'optimisation.

Cette technique peut être appliquée quand une solution optimale consiste en une combinaison de solutions optimales de sous problèmes, ce qui est le cas pour le problème de sac à dos. Pour pouvoir s'appliquer avec intérêt, elle exige que les problèmes traités aient une structure particulière de type séquentielle. La plupart des problèmes combinatoires peuvent se mettre sous cette forme et être résolus par programmation dynamique.
Le principe de base de la programmation dynamique est le suivant :

1) On prolonge le problème dans une famille de problème de même nature.
2) On relie par une relation de récurrence les solutions optimales de ces problèmes.

Exemple 1.1 Résolution du problème de sac à dos

$$(KP)\begin{cases} \max(Z) = \sum_{j=1}^{n} c_j x_j \quad w_j \geq 0 \\ \sum_{j=1}^{n} w_j x_j \leq \omega, \quad \omega \text{ et } w_j \text{ des entiers positifs} \\ x_j = 0 \quad \text{ou} \quad 1, \quad j = \overline{1,n} \end{cases} \tag{1.15}$$

Pour tout i de 1 à n et pour tout y de 0 à ω.
Considérons les problèmes $P_i(y)$:

$$P_i(y)\begin{cases} \max(Z) = \sum_{j=1}^{i} c_j x_j \\ \sum_{j=1}^{i} w_j x_j \leq y \\ x_j = 0 \text{ ou } 1, \quad j = \overline{1,i} \end{cases} \tag{1.16}$$

appelons $Z_i(y)$ la valeur de la solution optimale de $P_i(y)$, on a alors :
$Z_{i+1}(y) = \max\{Z_i(y), c_{i+1} + Z_i(y - w_{i+1})\}$ qui exprime le fait que $x_{i+1} = 0$ ou 1.

Les valeurs des variables de la solution optimale $x_i(y)$ tel que :

$$x_i(y) = \begin{cases} 0 & \text{si} \quad Z_{i+1}(y) = Z_i(y) \\ 1 & \text{sinon} \end{cases} \tag{1.17}$$

Algorithme 2: Programmation dynamique pour KP en $\{0-1\}$

Output :
↑ Les performances des solutions optimales

```
for y = 0 : ω do
    Z_0(y) := 0
end
for i ∈ {1, 2, ..., n} do
    for y = 0 : ω do
        if y ≥ w_i then
            Z_i(y) := max{Z_{i-1}(y), c_i + Z_{i-1}(y - w_i)}
        else
            Z_i(y) := Z_{i-1}(y)
        end
    end
end
return Z_n(ω)
```

Dans certaines situations, la programmation dynamique [90, 70, 93, 65, 56] peut être une alternative aux algorithmes Branch & Bound.

Résoudre les problèmes d'optimisation combinatoire est très difficile, en effet pour de tels problèmes, les méthodes exactes exigent un effort calculatoire qui croit exponentiellement avec la taille des instances du problème et souvent cette résolution se heurte à une explosion du nombre de combinaison à explorer et rapidement les méthodes approximatives deviennent l'unique moyen d'obtenir une solution approchée dans un temps raisonnable.

1.4.3 Méthode de résolution approchée

Il s'agit de l'algorithme Glouton "Greedy Algorithm" qui est une procédure de résolution approchée du problème de sac à dos (KP), cette heuristique est basée sur un principe très simple : on commence par la sélection des objets un par un dans un ordre décroissant cité par l'equation (1.10), à condition que les contraintes puissent être satisfaites, jusqu'à ce qu'une solution admissible soit obtenue. L'algorithme 3 est exécuté pour remplir le sac à dos, ceci permet d'obtenir une borne inférieure $\underline{Z}$.

Algorithme 3: Algorithme Glouton

Input :
$\downarrow C$: la matrice de profits
$\downarrow w_i; i = 1, \ldots, n$: les poids de l'objet i
$\downarrow \omega$: capacité du sac à dos
$\downarrow x$: la solution initial
Output :
$\uparrow \underline{Z}$: la borne inférieure
$\uparrow \bar{\omega}$: la capacité résiduelle
$\updownarrow x$: la solution construite par le Glouton

Initialization
Trier les objets par ordre décroissant d'efficacité (1.10)
$0 \longleftarrow \underline{Z}$
$\omega \longleftarrow \bar{\omega}$
for $i = 1 : n$ **do**
  if $w_i \leq \omega$ **then**
    $1 \longleftarrow x_i$
    $\bar{\omega} - w_i \longleftarrow \bar{\omega}$
    $\underline{Z_i} + c_i \longleftarrow \underline{Z_i}$
  else
    $0 \longleftarrow x_i$
  end
end

Cas particulier inadapté par l'algorithme de Glouton

Soit un problème de sac à dos suivant :

$$(\text{KP}) \begin{cases} \max(Z) = c_1x_1 + c_2x_2 \\ w_1x_1 + w_2x_2 \leq \omega \\ x_i \in \{0, 1\}\,,\ i = 1, 2 \end{cases}$$

tel que $n = 2, c_1 = 2, w_1 = 1,$ et $c_2 = \omega, w_2 = \omega$ avec $\omega > 1$.
L'algorithme Glouton choisira en premier l'objet 1, il est plus efficace, empêchant la sélection du second objet. Or, la sélection de ce second objet aurait donné une solution d'une meilleure qualité.
Cette procédure est rapide, peut donner dans le pire des cas des résultats particulièrement mauvais, comme le montre le cas particulier ci-dessus. Elle est ainsi principalement utilisée en préparation pour une résolution plus coûteuse (par exemple pour obtenir une première borne inférieure).

1.4.4 Complexité

La théorie de la complexité en temps des algorithmes a pour objet d'étudier l'évolution du temps de calcul d'un algorithme en fonction de la dimension du problème à traiter. Pour certains problèmes, l'importance des données numériques est essentielle. A tel point que le caractère non polynomial d'un algorithme pour un tel problème peut être provoqué uniquement par la grandeur même des données numériques qui y interviennent. Dans ce cas, si l'on impose une borne supérieure, l'algorithme devient polynomial. Un tel algorithme est appelé pseudo-polynomial, le temps d'exécution d'un tel algorithme (déterministe) est borné supérieurement par un polynôme en dimension du problème et en dimension du plus grand nombre qui intervient dans la description d'une réalisation [26, 93]). Le problème de sac à dos est le plus simple des problèmes non triviaux d'optimisation linéaire en variables binaires, il possède une seule contrainte et uniquement des coefficients entiers positifs. Tel qu'il se trouve en tant que sous problème de nombreux problèmes d'optimisation combinatoire, donc il a une grande importance dans l'optimisation combinatoire malgré la simplicité de sa structure [56]. Le problème de sac à dos est l'un des 21 problèmes NP-difficiles.

1.5 Prétraitement du problème de sac à dos uniobjectif

Le temps d'exécution des algorithmes pour toutes les instances pratique du problème de sac à dos est très important, la diminution de ce temps résulte d'une étude de réduction du nombre de variables modélisant le problème malgré que sa complexité ne change pas. Donc, il serait intéressant de réduire la taille du problème considéré avant sa résolution par des procédures qui fixent a priori le plus grand nombre de variables à leur valeur optimale.

1.5.1 Réduction du problème de sac à dos uniobjectif

Les procédures de réduction consistent à une partition de l'ensemble des variables $\{1, 2, \ldots, n\}$ en trois sous ensembles :

$$J_1 = \{i \in \{1, 2, \ldots, n\} \mid x_i = 1, \text{ dans toutes les solutions optimales de KP}\}.$$

$$J_0 = \{i \in \{1, 2, \ldots, n\} \mid x_i = 0, \text{ dans toutes les solutions optimales de KP}\}.$$

$$F = \{1, 2, \ldots, n\} \backslash (J_1 \cup J_0).$$

Le problème de sac à dos initial peut s'écrire sous une forme réduite aux seules variables de F comme suit :

$$(KP_R)\begin{cases} \max(Z) = \sum_{i\in F} c_i x_i \\ \sum_{i\in F} w_i x_i \leq \hat{\omega} \\ x_i \in \{0,1\}, i \in F \end{cases} \tag{1.18}$$

où $\hat{\omega} = \omega - \sum_{i\in J_1} w_i$.
Les solutions optimales du problème initial sont alors celles du problème réduit, auxquelles les objets de J_1 sont ajoutés.
Ingargiola et Korsh [53] ont proposé une nouvelle méthode pour déterminer les ensembles J_1 et J_0.
L'idée principale est la suivante : si la valeur de x_i prend la valeur b ($b = 0$ ou $b = 1$) produit une solution irréalisable ou moins bonne que la meilleure solution connue alors, x_i doit prendre la valeur $1 - b$ dans toutes les solutions optimales. Soit Z^* la valeur correspondante à une solution réalisable du problème de sac à dos et pour $i \in \{1, 2, \ldots, n\}$, soit Z_i^1 (resp. Z_i^0) une borne supérieure pour le problème considéré calculée en ajoutant la contrainte $x_i = 0$ (resp. $x_i = 1$), alors les ensembles J_1 et J_0 sont définis comme suit :

$$J_1 = \{i \in \{1, 2, \ldots, n\} \mid Z_i^0 < Z^*\},$$
$$J_0 = \{i \in \{1, 2, \ldots, n\} \mid Z_i^1 < Z^*\}.$$

Dans l'algorithme de Ingargiola et Korsh, les bornes supérieures Z_i^1 et Z_i^0 sont calculées en utilisant les bornes de Dantzing [70], mais de nombreuses améliorations ont vu le jour [71, 77].

1.5.2 Notion du core du problème de sac à dos (KP)

De nombreux tests expérimentaux sur une large variété d'instances de problèmes de sac à dos ont montré que seul un sous ensemble d'objets dont l'efficacité ressemble à l'efficacité de l'objet bloquant b intervient dans la construction effective des solutions optimales. Ceci se remarque d'autant plus que le nombre de variables est grand [77]. Cette observation a mené Balas et Zemel [4] à définir la notion du core.
En 1980, Balas et Zemel [4] ont donné une définition précise du core d'un problème de sac à dos, cette définition présentée ci-dessous est basée sur la connaissance d'une solution optimale du problème relaxé (LKP). Plus précisément, ils ont noté qu'une solution optimale d'une instance générée aléatoirement du problème de sac à dos est similaire à une solution optimale du problème de sac à dos relaxé (LKP) donc, ils ont défini le core d'une instance de la manière suivante :

Soit x^* une solution optimale de KP et les objets sont triés suivant un ordre décroissant de l'efficacité d'objet définit par l'équation (1.10), le core est un ensemble défini par :

$$C = \{j_1, \ldots, j_2\}, \tag{1.19}$$

où

$$j_1 = \min\{i \mid x_i^* = 0, i = 1, \ldots, n\}, \tag{1.20}$$

et

$$j_2 = \max\{i \mid x_i^* = 1, i = 1, \ldots, n\}. \tag{1.21}$$

La figure 1.3 représente la notion du core :

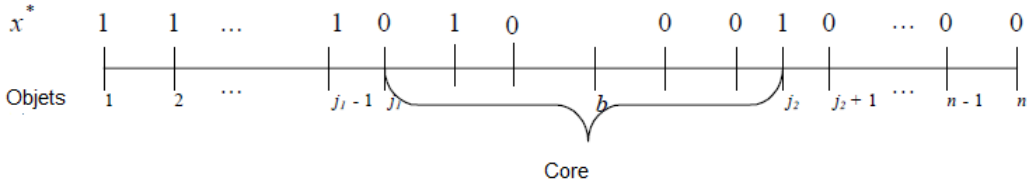


FIGURE 1.3 – Core exacte

Le problème core est alors défini comme suit [70] :

$$(KP_C) \begin{cases} \max(Z) = \sum_{i \in C} c_i x_i \\ \sum_{i \in C} w_i x_i \leq \widetilde{\omega} \\ x_i \in \{0, 1\}, \ i \in C. \end{cases} \tag{1.22}$$

où $C = \{j_1, \ldots, j_2\}$ et $\widetilde{\omega} = \omega - \sum_{i=1}^{j_1 - 1} w_i$.
La taille du core est une petite fraction de n pour de nombreuses instances. Par conséquent, si les valeurs de j_1 et j_2 sont connues a priori, il est possible de ne résoudre que le problème réduit aux objets de C, avec une capacité de $\omega - \sum_{i=1}^{j_1 - 1} w_i$ et de compléter ensuite la solution de manière à ce que : $x_i = 1, \forall i \in \{1, \ldots, j_1 - 1\}$ et $x_i = 0, \forall i \in \{j_2 + 1, \ldots, n\}$.
Le fait que la notion de core est basée sur la connaissance d'une solution optimale rend difficile son utilisation dans les méthodes de résolution. Dans la section suivante, on présente de divers valeurs du cardinal de l'ensemble core estimé par les chercheurs avant la résolution du problème [70].

1.5.3 Notion de core du problème de sac à dos de grande taille

En général, pour les problèmes de large dimension, la taille du core pour de nombreuses classes d'instances est petite par rapport aux nombres d'objets n. Cependant, si on connaît a priori les valeurs j_1 et j_2, on pourra résoudre le problème initial en posant :

$$x_j^* = 1 \quad \text{pour tout} \quad j \in J_1 = \{k \mid \frac{c_k}{w_k} > \frac{c_{j_1}}{w_{j_1}}\},$$

et

$$x_j^* = 0 \quad \text{pour tout} \quad j \in J_0 = \{k \mid \frac{c_k}{w_k} < \frac{c_{j_2}}{w_{j_2}}\},$$

ensuite résoudre le problème réduit par la méthode "Branch & Bound" ou programmation dynamique ou autres méthodes approximatives.

En pratique, les indices j_1 et j_2 ne sont pas connus a priori, mais une meilleure approximation est basée sur la résolution d'un problème core fixé à $C = \{b-u, \ldots, b+u\}$ avec une variation de choix pour la valeur de u et b est l'élément bloquant calculé par

l'équation (1.11), tel que la plupart des algorithmes utilisant la notion du core reposent sur le choix de sa taille $|C| = 2u$, tel que $2u$ représente le nombre d'objets autour de l'objet critique. Plusieurs propositions ont été faites pour trouver le cardinal ou la taille du core. Balas et Zemel [4] ont proposé une valeur constante $|C| = 50$. Martello et Toth [71] ont proposé une valeur pour $u = \sqrt{n}$ puis $u = 2\sqrt{n}$ [69]. Ces différentes valeurs de $|C|$ se rapportent à des interprétations différentes du problème ([77, 65]).

Plusieurs expérimentations ont été menées par Pisinger [77] sur différents types de problèmes de sac à dos pour déterminer la taille minimale du core (moyenne sur 100 instances). Celles-ci sont présentées dans la table 1.1 :

n	non-corrélées	faiblement corrélées	fortement corrélées	la somme des sous-ensembles
100	5	12	13	14
500	8	17	25	13
1000	11	17	36	13
5000	14	21	79	13
10000	17	25	104	13

TABLE 1.1 – Taille minimale du core $|C|$ suivant le nombre d'objets.

- Les instances non-corrélées : les profits et les poids sont générés aléatoirement dans l'intervalle $[1, R]$, où les valeurs de R sont détaillées dans [77].
- Les instances faiblement corrélées : les poids w_i sont générés aléatoirement dans l'intervalle $[1, R]$ et les profits c_i sont générés aléatoirement dans l'intervalle $[w_i - \frac{R}{10}, w_i + \frac{R}{10}],\ i = 1, \ldots, n$.
- Les instances fortement corrélées : les poids w_i sont générés aléatoirement dans l'intervalle $[1, R]$ et les profits $c_i = w_i + 10,\ i = 1, \ldots, n$.

1.5.4 Notion du core du problème de sac à dos multidimensionnel (MKP)

Les définitions précédentes (1.19), (1.20) et (1.21) du core pour le problème de sac à dos unidimensionnel peuvent être étendues au problème de sac à dos multidimensionnel (MKP) sans difficultés. Le problème principal réside dans le fait qu'il n'y a pas une mesure d'efficacité évidente. J.Puchinger et al. [84] ont présenté un nouveau développement du core pour le problème de sac à dos multidimensionnel (MKP) qui est une extension du concept du core classique de cas unidimensionnel.

Mesure d'efficacité du (MKP)

Le core du problème de sac à dos multidimensionnel est défini en dependence du choix de la fonction d'efficacité des objets, puisque aucune mesure d'efficacité évidente est disponible pour ce problème.

Pour obtenir les mesures d'efficacité des objets pour (MKP), on considère la forme générale de l'efficacité d'objets définie en introduisant des valeurs r_i pour chaque contrainte du problème (MKP).

$$e_i(\text{générale}) = \frac{c_i}{\sum_{j=1}^{m} r_j w_{ji}}.$$

Diverses propositions ont été faites pour calculer une estimation de la valeur de r_j dans [79, 60]. Comme il existe plusieurs possibilités pour définir les mesures de l'efficacité d'objets pour MKP, le core du problème de sac à dos multidimensionnel (MKP) doit être définis en fonction d'une mesure d'efficacité spécifique e.

Soit x^* une solution optimale et supposons que les éléments sont triés selon l'efficacité décroissante e, et soit C_e défini par :

$$C_e = \{j_1^e, ..., j_2^e\},$$

où

$$j_1^e = \min\{i \mid x_i^* = 0, i = 1, \ldots, n\},$$

et

$$j_2^e = \max\{i \mid x_i^* = 1, i = 1, \ldots, n\}.$$

Bien que le concept de la notion du core soit une base de l'efficacité des algorithmes appliqués au problème de sac à dos uniobjectif en variables binaires, son utilisation est difficile du fait qu'elle se réfère à une solution optimale déjà trouvée.

1.6 Conclusion

Dans ce chapitre, nous avons exposé quelques extensions du problème de sac à dos unicritère ainsi que les méthodes existantes pour sa résolution exacte et approchée, en faisant le point dans la dernière partie sur des méthodes de prétraitement afin de réduire la taille de ce problème. Le concept du core dédié au problème de sac à dos classique [4] a conduit à des algorithmes très réussis [71, 79, 77]. L'idée principale est de réduire le problème de sac à dos à un ensemble d'éléments dit ensemble core pour lesquels il est difficile de décider si ces éléments entrent dans la construction d'une solution optimale ou non, tandis que toutes les variables en dehors de cet ensemble sont initialement fixées à certaines valeurs, par conséquent résoudre le problème résiduel par une méthode exacte ou approchée.

Les problèmes d'optimisation à objectif unique étaient une première modélisation des problèmes de recherche opérationnelle. À travers le temps on remarque que ce procédé manque de réalisme, il serait impossible de combiner tous les points désirés en une seule fonction objectif, où on néglige un nombre fini d'objectifs et on s'éloigne dans ce cas de la solution réaliste. Une telle situation ne modélise pas correctement un problème.

Afin de s'approcher au maximum de la réalité, de nouvelles méthodes caractérisées par la prise en considération de tous les objectifs apparaissent. Cette multiplicité d'objectifs semble plus logique pour la modélisation des problèmes. Dans le chapitre suivant, on présentera l'aspect de l'optimisation multicritère (MO).

Chapitre 2

Optimisation Multiobjectif (MO)

2.1 Introduction

L'optimisation multiobjectif (MO) ou multicritère est un domaine dont l'importance est justifiée par la nature multiobjectif des problèmes d'optimisation à résoudre dans la réalité. Elle se propose de traiter les problèmes qui nécessitent la satisfaction de plusieurs critères en même temps. Ces critères sont parfois conflictuels et parfois complémentaires. Un des aspects caractérisant l'optimisation multiobjectif est qu'elle fournit un ensemble de solutions réalisant le meilleur compromis entre les critères considérés. Cet ensemble de solutions est appelé l'ensemble optimal de Pareto.
Plusieurs techniques ont été développées pour traiter ces problèmes à objectifs multiples, la plus utilisée est l'algorithme qui maximise plusieurs objectifs par l'intermédiaire de la programmation linéaire paramétrique.

Il existe deux types principaux de méthodes de résolution des problèmes multiobjectifs, les méthodes exactes et les méthodes méta-heuristiques.
Nous commençons ce chapitre par donner la formulation d'un problème multiobjectif, suivie d'un aperçu sur quelques notions de base concernant l'optimisation multiobjectif et la classification des solutions optimales au sens de Pareto dites solutions efficaces. Nous terminons par quelques méthodes de résolution des problèmes multiobjectifs en nombres continus ainsi qu'en nombres entiers.

2.2 Problèmes d'optimisation multiobjectifs

L'optimisation multiobjectif ou la programmation multiobjectif est un processus d'optimisation de deux ou plusieurs objectifs conflictuels (exemple : maximiser la performance d'un véhicule et minimiser la consommation des carbures), simultanément qui vérifie certaines contraintes. On trouve l'optimisation dans de divers champs de Finance et Gestion où la décision optimale sera choisie dans la présence de (Trade-offs), compromis (échange de questions et réponses). Le problème linéaire multiobjectif est défini comme un problème dont on cherche la solution ou le vecteur de décision $x = (x_1, x_2, \ldots, x_n)$ qui satisfait un ensemble de contraintes et optimise un vecteur de fonction objectif $Z = (Z_1, Z_2, \ldots, Z_p)$.

La figure 2.1 représente une modélisation d'un problème biobjectif :

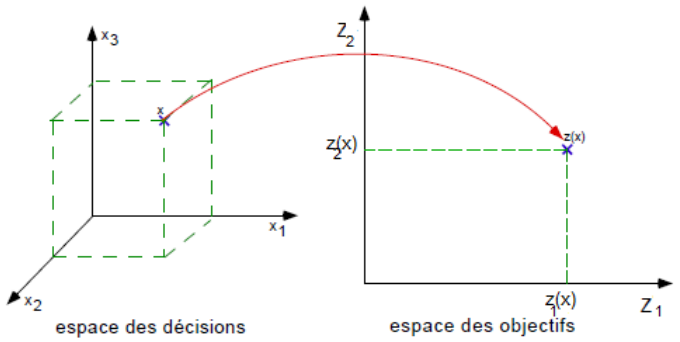


FIGURE 2.1 – Modélisation d'un problème biobjectif

Formulation mathématique

Le problème multiobjectif se présente comme suit :

$$(MOLP)\begin{cases} \text{“}Optimiser\text{”}\, Z_k = \quad C^k x \quad k = \overline{1,p} \\ t.q \qquad x \in X \end{cases} \tag{2.1}$$

où l'ensemble X est déterminé par des contraintes (inéquations, équations) linéaires : $X = \{x \in \mathbb{R}^n \ Ax = b, x \geq 0\}$,
$A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m, \quad C^k \in \mathbb{R}^{1 \times n} \quad k = \overline{1,p}$.
Si de plus, l'ensemble des décisions est restreint aux entiers, le problème devient un problème de programmation linéaire multiobjectif entier défini par :

$$(MOILP)\begin{cases} \text{“}Optimiser\text{”}\, Z_k = \quad C^k x \quad k = \overline{1,p} \\ t.q \qquad x \in D \end{cases} \tag{2.2}$$

où D=$X \cap \mathbb{Z}^n$, $\mathbb{Z}$ est l'ensemble des nombres entiers.

Remarque 2.1 Dans la suite, nous considérons, sans perte de généralité, que toutes les fonctions objectifs sont à maximiser.

Une écriture équivalente du problème MOILP se présente comme suit :

$$(MOILP)\begin{cases} \text{“max”}\, Z_k = \quad C^k x \quad k = \overline{1,p} \\ t.q \qquad x \in D \end{cases} \tag{2.3}$$

2.2.1 Notion de dominance

Soit le problème (MOLP) :

$$(MOLP)\begin{cases} \text{“max”}\, Z_k = \quad C^k x \quad k = \overline{1,p} \\ t.q \qquad x \in X \end{cases} \tag{2.4}$$

Considérons l'application linéaire Z qui associe à chaque vecteur de décision $x \in X$, son image $Z(x) = (c^1 x, ..., c^p x)$ dans l'espace des critères, notée $Y = \{Z(x) \mid x \in X\}$.

Définition 2.1 Espace de décisions et espace des critères

- L'espace $\mathbb{R}^n$ dans lequel se situe l'ensemble des actions X tel que $(X \subseteq \mathbb{R}^n)$ est appelé **espace de décisions**.
- L'espace $\mathbb{R}^p$ dans lequel se situe $Z(X)$ est appelé **espace des critères** ou **espace des objectifs**.

Lorsqu'une solution est associée à une seule valeur, on parle d'un problème monoobjectif, lorsqu'elle est associée à plusieurs valeurs, on parle d'un problème multiobjectif (ou multicritère).
Dans ce dernier cas, une solution ne peut être considérée comme la meilleure de toutes les autres mais, il reste possible de comparer des solutions deux à deux en utilisant l'ordre partiel donné par une relation de dominance au sens de Pareto, définie par la suite.
Une solution Pareto optimale (dite aussi solution efficace), ne peut être améliorée sur un objectif sans dégrader un autre objectif. Selon cette relation, la comparaison de deux solutions montre que soit l'une est meilleure que l'autre, soit qu'elles sont incomparables. Par conséquent on cherche un ensemble de solutions non-dominées, parmi lesquelles on ne peut décider si une solution est meilleure qu'une autre, aucune n'étant systématiquement supérieure aux autres sur tous les objectifs.
On commence par exposer comment caractériser l'intérêt des solutions qui maximise "max" différents objectifs, à travers la notion d'optimalité de Pareto.

Dominance de Pareto

Dans l'ordre de prendre une distinction entre les vecteurs de $\mathbb{R}^p$, on définit trois différents ordres :

Soient $Z^1, Z^2 \in \mathbb{R}^p$,

1. Relation d'ordre faible : $Z^1 \succeq Z^2$ si et seulement si :

$$Z^1_k \geq Z^2_k, \quad \forall k \in \{1, 2, \ldots, p\}.$$

2. Relation d'ordre : $Z^1 \succ Z^2$ si et seulement si :

$$Z^1_k \geq Z^2_k, \quad \forall k \in \{1, 2, \ldots, p\} \text{ et } \exists k \in \{1, 2, \ldots, p\},\ t.q.\ Z^1_k > Z^2_k.$$

3. Relation d'ordre stricte : $Z^1 \succ\succ Z^2$ si et seulement si :

$$Z^1_k > Z^2_k, \quad \forall k \in \{1, 2, \ldots, p\}.$$

Nous définissons par la suite trois différentes notions de dominance au sens de Pareto pour un problème multiobjectif.

Soient $x, \tilde{x} \in X$ et soient $Z(x), Z(\tilde{x})$ leurs images dans l'espace des objectifs $Z(X)$:

Définition 2.2 Notion de Dominance

Si $Z(x) \succeq Z(\tilde{x})$, $Z(x)$ **domine faiblement** $Z(\tilde{x})$.
Si $Z(x) \succ Z(\tilde{x})$, $Z(x)$ **domine** $Z(\tilde{x})$.
Si $Z(x) \succ\succ Z(\tilde{x})$, $Z(x)$ **domine strictement** $Z(\tilde{x})$.

Remarque 2.2 La dominance stricte implique la dominance implique la dominance faible, donc on a :

$$Z(x) \succ\succ Z(\tilde{x}) \Rightarrow Z(x) \succ Z(\tilde{x}) \Rightarrow Z(x) \succeq Z(\tilde{x}).$$

En général, dans les problèmes d'optimisation multiobjectifs, on ne peut pas toujours comparer les solutions contrairement aux problèmes d'optimisation uniobjectifs.

Exemple 2.1

Pour une instance donnée, on présente quatre vecteurs dans l'espace des objectifs pour $p = 2$ (voir, Figure 2.2).

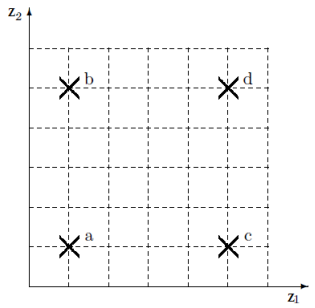


FIGURE 2.2 – Comparaison des points

Toutes les paires des vecteurs peuvent être comparées par la relation de dominance au sens de Pareto : $d \succ\succ a,\ d \succ c,\ d \succ b,\ c \succ a,\ b \succ a$, à part le couple (b, c), car b est incomparable avec c, c'est à dire $b \not\succ c$ et $c \not\succ b$.

On définit par la suite de différents types de solutions :

Définition 2.3 Solution efficace (Efficient solution)
Une solution réalisable $\tilde{x} \in X$ sera dite **efficace** (Pareto optimale) s'il n'existe pas une autre solution réalisable $x \in X$ tel que $Z(x) \succ Z(\tilde{x})$.

Définition 2.4 Ensemble efficace (Efficient set)
L'**ensemble efficace** noté par X_E contient toutes les solutions efficaces.

Définition 2.5 Point non-dominé (Non-dominated point)
L'image $Z(\tilde{x})$ d'une solution efficace $\tilde{x}$ dans l'espace des objectifs est appelé **un point non-dominé**.

Définition 2.6 Frontière de Pareto (Pareto front)
L'image des solutions efficaces X_E sera notée $Y_N = \{Z(x) \mid x \in X_E\}$, est appelé le **front pareto** (Pareto front), autrement dit, le graphe obtenu de l'image des solutions efficaces est appelé la frontière efficace (Pareto front) dans le cas continu.

Définition 2.7 Solution faiblement efficace (Weakly efficient solution)
Une solution réalisable $\tilde{x} \in X$ est appelée solution **faiblement efficace** s'il n'existe pas une autre solution réalisable $x \in X$ tel que $Z(x) \succ\succ Z(\tilde{x})$.

Définition 2.8 Point faiblement non-dominé (Weakly non-dominated point)
L'image d'une solution faiblement efficace est appelé un **point faiblement non-dominé**.

Définition 2.9 Ensemble des solutions faiblement efficaces (Weakly efficient set)

L'**ensemble de solutions faiblement efficaces** noté X_{WE} contient toutes les solutions faiblement efficaces.

L'image de X_{WE}, sera $Y_{NW} = \{Z(x) \mid x \in X_{WE}\}$.

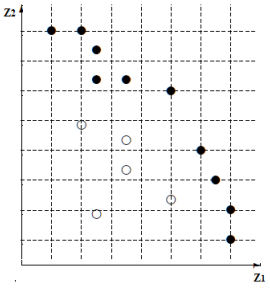


FIGURE 2.3 – Ensemble de solutions faiblement non-dominées Y_{NW}

Les dix points représentent l'image des solutions faiblement efficaces dans l'espace des objectifs Y_{NW}, (voir, Figure 2.3).

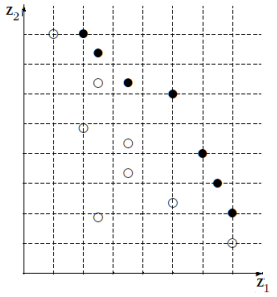


FIGURE 2.4 – La frontière de Pareto Y_N

Les sept points représentent la frontière de Pareto. Il existe trois points qui sont faiblement non-dominés mais ils ne sont pas des points non-dominés, (voir, Figure 2.4).

Remarque 2.3 Une solution faiblement efficace est dominée par une solution efficace.

2.2.2 Optimalité lexicograghique

On introduit dans cette section, une autre définition d'optimalité, il s'agit d'"**Optimalité lexicographique**", cette notion est utilisée lorsqu'il existe un classement d'importance entre les objectifs marquant une préférence absolue, la qualité de la solution suit un ordre total appelé ordre lexicographique. Par exemple pour $p = 2$, on commence par optimiser le premier objectif Z_1 sans tenir compte du deuxième objectif Z_2, s'il existe plusieurs solutions optimales, on optimise le deuxième objectif tout en gardant la valeur optimale obtenue pour le premier objectif.

Définition 2.10 Ordre lexicographique

Soient $Z^1, Z^2 \in \mathbb{R}^p$, et si $Z^1 \neq Z^2$, soit $k^* = \min\{k,\ Z^1_k \neq Z^2_k\}$.

1. Ordre lexicographique : $Z^1 \succeq_{lex} Z^2$ si $Z^1_{k^*} > Z^2_{k^*}$ ou $Z^1 = Z^2$.
2. Ordre lexicographique stricte : $Z^1 \succ\succ_{lex} Z^2$, si $Z^1_{k^*} > Z^2_{k^*}$.

Le problème d'optimisation lexicographique peut être écrit comme suit :

$$\text{lexmax}_{x\in X}(Z_1(x), Z_2(x), \ldots, Z_p(x)). \tag{2.5}$$

Définition 2.11 Solution lexicographiquement optimale

Une solution réalisable $x^* \in X$ est appelée une solution **lexicographiquement optimale**, s'il n'existe pas une autre solution réalisable $x \in X$ tel que $Z(x) \succ\succ_{lex} Z(x^*)$.

En plus, on peut dire que $x^* \in X$ est une solution lexicographiquement optimale si $Z(x^*) \succeq_{lex} Z(x)$, $\forall x \in X$.

Toutes les solutions lexicographiquement optimales sont obtenues par l'algorithme 4 :

Algorithme 4: Les solutions lexicographiquement optimales

Input :
↓ Paramètres : X l'ensemble de solutions réalisable et $Z_1, Z_2, \ldots, Z_p$ les fonctions objectifs du problème multiobjectif (MOLP).
Output :
↑ Paramètres : L'ensemble de solutions lexicographiquement optimales.

Initialisation
$X_1 := X$
$k := 1$
while $k \leq p$ **do**
  Résoudre le problème monoobjectif $\max_{x\in X_k} Z_k(x)$.
  if *le problème précédent admet une solution unique* x_k^* **then**
    Arrêter, x_k^* est l'unique solution optimale du problème d'optimisation lexicographique.
  end
  if $k = p$ **then**
    Arrêter, l'ensemble de solutions optimales du problème d'optimisation lexicographique est :

$$\{x \in X_p : Z_p(x) = \max_{x\in X_p} Z_p(x)\}$$

  end
  $X_{k+1} := \{x \in X_p \mid Z_p(x) = \max_{x\in X_k} Z_k(x)\}$.
end

Théorème 2.1 *[26, p. 129]*

Soit $x^* \in X$ *tel que* $Z(x^*) \succeq_{lex} Z(x)$, $\forall x \in X$ *alors* $x^* \in X_E$.

Il est également possible de définir une solution lexicographiquement optimale pour un autre classement des objectifs différent du classement $(Z_1(x), Z_2(x), \ldots, Z_p(x))$.

Définition 2.12 Solution lexicographiquement optimale globale

On appelle une solution lexicographiquement optimale pour au moins un classement des fonctions objectifs une solution lexicographiquement optimale globale définie comme suit :

Une solution réalisable $x^* \in X$ est appelée une solution lexicographiquement et globalement optimale s'il n'existe pas une autre solution réalisable $x \in X$ tel que $Z^\pi(x) \succeq Z^\pi(x^*)$, pour au moins une permutation $\pi \in \{1, 2, \ldots, p\}$.

On présente dans la figure 2.5, les solutions lexicographiquement et globalement optimales dans l'espace des objectifs. Une de ces solutions représente une solution du problème $\text{lexmax}_{x \in X}(Z_1, Z_2)$ et l'autre solution représente une solution du problème $\text{lexmax}_{x \in X}(Z_2, Z_1)$ de l'exemple 2.1.

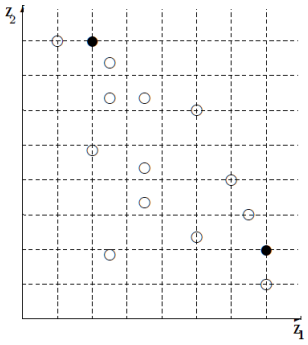


FIGURE 2.5 – Image des solutions lexicographiquement et globalement optimales

Par la suite nous utiliserons la notion des sous ensembles de $\mathbb{R}^p$ comme suit :

- $\mathbb{R}^p_{\succeq} = \{x \in \mathbb{R}^p \mid x \succeq 0\}$.
- $\mathbb{R}^p_{\succ} = \{x \in \mathbb{R}^p \mid x \succ 0\}$.
- $\mathbb{R}^p_{\succ\succ} = \{x \in \mathbb{R}^p \mid x \succ\succ 0\}$.

2.2.3 Ensemble complet minimal et maximal des solutions efficaces

La définition de la frontière de Pareto est relative à l'espace des objectifs. Une solution appartient à la frontière de Pareto si elle n'est dominée par aucune autre solution réalisable. Lorsque deux solutions différentes ont exactement les mêmes valeurs dans l'espace des objectifs, il s'agit de solutions équivalentes. Lors de la résolution d'un problème, on peut trouver deux cas possibles de la cardinalité des solutions efficaces ; si l'ensemble de solutions efficaces est très grand il sera intéressant de ne garder que les solutions dont les images dans l'ensemble des objectifs sont différentes, mais si le problème admet peu de solutions efficaces, il sera intéressant de garder toutes les solutions efficaces y compris dont les images sont identiques (cela dépend du choix de décideur). Nous parlerons alors de recherche d'un ensemble complet minimal, dans le premier cas, et d'un ensemble complet maximal dans le deuxième cas. Nous exposerons par la suite, leur définitions mathématiques.

Définition 2.13 Solutions équivalentes (Equivalent solutions) :
Deux solutions admissibles $x, \tilde{x} \in X$ sont dites **équivalentes**, si :
$Z(x) = Z(\tilde{x})$.

Définition 2.14 Ensemble complet

- Un ensemble complet X_E est un ensemble de solutions efficaces tel que toute solution réalisable $x \in X \setminus X_E$ est soit équivalente, soit dominée par une solution $\tilde{x} \in X_E$.
- L'ensemble complet minimal X_{Em} est un ensemble complet sans solutions équivalentes.
- L'ensemble complet maximal X_{EM} est un ensemble complet contenant toutes les solutions équivalentes.
- Pour chaque point $y \in Y_N$, il existe au moins une solution $x \in X_E$ tel que $Z(x) = y$.

Remarque 2.4

- Tout ensemble complet contient un ensemble complet minimal.
- Trouver un ensemble complet minimal dans le cas multiobjectif est équivalent dans le cas uniobjectif à trouver une seule solution optimale et non pas toutes solutions optimales.

Il existe d'autres sens d'optimalité [67, 26].

2.2.4 Le point Idéal, Nadir, Anti-Idéal et point Utopique

Dans l'optimisation multiobjectif le concept de bornes n'est pas bien développé, les meilleures bornes inférieures et supérieures parmi tous les points non-dominés sont données par le point idéal, le point nadir et le point utopique définis respectivement par :

- **Point Idéal**
  Le point idéal $\overline{Z}$ est le point de $\mathbb{R}^p$ de coordonnées :

$$\overline{Z}_k = Z_k^I = \max_{x \in X}(Z_k(x)), \quad \forall\, k = 1, ..., p.$$

- **Point Nadir**
  Le point Nadir η de $\mathbb{R}^p$ de coordonnées :

$$\eta_k = Z_k^N = \min_{x \in X_E}(Z_k(x)), \quad \forall\, k = 1, ..., p.$$

- **Point Utopique**
  D'un point idéal, on peut se référé au point utopique noté Z^U défini de la manière suivante :

$$Z^U = \overline{Z} + \varepsilon U,$$

  où $\varepsilon > 0$ est un petit nombre positif et $U = (1, 1, \ldots, 1) \in \mathbb{R}^p$ est le vecteur unitaire ; à noter que ce point n'est pas réalisable.
- **Point anti-idéal**
  Le point anti-idéal $\underline{Z}$ est le point de $\mathbb{R}^p$ de coordonnées :

$$\underline{Z}_k = Z_k^{AI} = \min_{x \in X}(Z_k(x)), \quad \forall\, k = 1, ..., p.$$

La figure 2.6 représente de différents points particuliers dans le cas d'un problème d'optimisation bicritère :

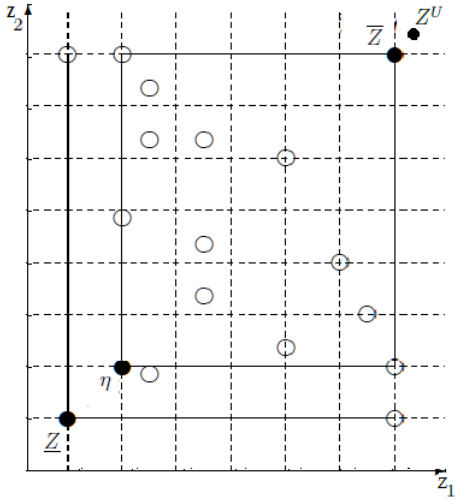


FIGURE 2.6 – Points particuliers

Quelques remarques sur ces derniers points particuliers

- Avec deux objectifs, le point Nadir est obtenu en déterminant deux solutions X^1 et X^2 lexicographiquement optimales par rapport à la permutation $(1, 2)$ et à $(2, 1)$ respectivement, $Z^N = (Z_1(X^2), Z_2(X^1))$. Cette égalité est due à l'ordre naturel des points non-dominés dans le cas biobjectif. Cependant, avec plus de deux objectifs, la détermination du point nadir n'est plus aisée [26].
- Le point idéal ne correspond pas à une solution réalisable car si c'était le cas, cela signifie que les objectifs ne sont pas conflictuels et qu'une solution optimisant un objectif, optimise simultanément tous les autres objectifs, dans ce cas on peut ramener le problème multiobjectif à un problème ayant une seule solution Pareto optimale.
- Les points Z^I et Z^N sont au plus près possible de l'ensemble des points non-dominés. Aucun de ces points n'est réalisable, sauf si tous les objectifs ont une solution optimale commune.
- On note que $Z^I \neq Z^N$, si les objectifs sont considérés conflictuels.

Définition 2.15 Matrice des gains

D'abord, on résolve p problèmes uniobjectifs $\max_{x \in X}(Z_k(x))$. Soit $\hat{x}^k$ une solution optimale du critère $Z_k(x)$, $\forall k = 1, 2, \ldots, p$, alors $Z_k(\hat{x}^k) = \max_{x \in X}(Z_k(x))$.

La matrice carrée de dimension p suivante :

$$\begin{pmatrix} \overline{Z}_1 & \ldots & Z_{1k} & \ldots & Z_{1p} \\ \vdots & & \vdots & & \vdots \\ Z_{k1} & \ldots & \overline{Z}_k & \ldots & Z_{kp} \\ \vdots & & \vdots & & \vdots \\ Z_{p1} & \ldots & Z_{pk} & \ldots & \overline{Z}_p \end{pmatrix}$$

est dite matrice des gains. La diagonale principale de la matrice des gains est formée des coordonnées du point idéal $Z_k^I = Z_k(\hat{x}^k)$, $\forall k = \{1, 2, \ldots, p\}$.

L'approximation du point nadir est obtenue comme suit :
$Z_j^N = \min_{k=1,\ldots,p}(Z_{jk}),\ \forall j = \{1, 2, \ldots, p\}$.

L'algorithme 5, détermine le point nadir et le point idéal :

Algorithme 5: Point Nadir et point idéal (p = 2)

Input :
↓ Paramètres :X l'ensemble de solutions réalisables et $Z = (Z_1, Z_2, \ldots, Z_p)$ est la fonction objectif du problème multiobjectif .
Output :
↑ Paramètres : Le point nadir $Z^N = (Z_1^N, Z_2^N)$ et le point idéal $Z^I = (Z_1^I, Z_2^I)$.

1 Résoudre les deux problèmes uniobjectif $\max Z_1(x)$ et $\max Z_2(x)$. On note les valeurs optimales Z_1^I et Z_2^I.
2 Résoudre le problème uniobjectif $\max Z_2(x)$ avec une contrainte additionnel $Z_1(x) \geq Z_1^I$.
3 Résoudre le problème uni objectif $\max Z_1(x)$ avec une contrainte additionnel $Z_2(x) \geq Z_2^I$.
4 On note par la valeur optimale Z_2^N et Z_1^N respectivement.

Il est facile de voir à partir de la définition du point nadir que l'algorithme 5 calcule Z^N. Les solutions trouvées dans l'étape 1 de cet algorithme ne sont pas nécessairement efficaces (mais sont faiblement efficaces), mais les solutions optimales des problèmes avec contrainte dans l'étape 2 et 3 sont efficaces. Par conséquent, le point nadir peut être facilement obtenu en générant les deux solutions lexicographiquement optimales. Malheureusement, cette approche ne peut pas être généralisée à plus de deux objectifs ($p > 2$) [26].

2.3 Méthodes de résolution

Dans de nombreux problèmes réels, la prise en considération de deux ou plusieurs objectifs (critères) conflictuels est obligatoire, ces objectifs expriment de différentes unités. En réalité les problèmes multicritères sont mal posés, contrairement aux problèmes unicritères, il n'existe pas en général un unique vecteur de valeur qui soit meilleur que tous les autres. Le plus souvent, résoudre un problème multiobjectif (MO) consiste à trouver l'ensemble des solutions optimales au sens de Pareto, c'est-à-dire les solutions telles qu'il n'existe pas de solution au moins aussi bonne sur tous les critères et meilleure sur au moins un critère. D'où vient la notion de la solution optimale au sens de Pareto définie auparavant (Définition 2.3).
Les méthodes de résolution des problèmes multiobjectifs sont des méthodes d'aide à la décision car le choix final sera laissé au décideur. Il existe trois types d'approches pour résoudre un problème multiobjectif (MO) :

2.3.1 Approches de résolution

Quand on parle d'une solution d'un problème multiobjectif, il s'agit d'un ensemble de solutions. Cependant, pour un problème issu de la réalité, une seule solution pourra être retenue, un choix par un décideur doit donc être effectué. Le décideur peut interve-

nir en cours de la résolution du problème multiobjectif, après ou de manière interactive. Donc, il existe trois approches pour que le décideur intervient :

a. **Préférence a priori** : le décideur définit ses préférences entre les différents objectifs avant d'utiliser la méthode de résolution. C'est à dire ramener le problème multiobjectif à un seul objectif au risque d'enlever toutes significations au problème.

b. **Préférence a posteriori** : le décideur choisit la solution de son choix parmi un ensemble de solutions fourni par la méthode de résolution. C'est à dire tenter d'apporter des réponses au problème tenant compte de tous les objectifs (calcul de la matrice gain pour arriver à toutes les réponses)

c. **Préférence progressive** : le décideur affine son choix de compromis au fur et à mesure du déroulement de la méthode d'optimisation.

Certaines méthodes peuvent ne pas être classées dans une seule catégorie. En effet, la méthode qui consiste à agréger les différents objectifs à l'aide de paramètres est une méthode à préférence a priori ; le décideur affecte les paramètres de manière à favoriser un objectif parmi les autres. Cependant, si les paramètres sont affectés aléatoirement et si la méthode est itérée en changeant les paramètres à chaque exécution. Dans ce cas il s'agit d'une méthode à préférence a posteriori.

Donc pour sélectionner une solution efficace ça nécessite une certaine connaissance de la structure de préférence du décideur, cette information est traduite en termes de paramètres.

Définition 2.16 Paramètres de Préférences

1) Un vecteur $\lambda = (\lambda_1, ..., \lambda_p)$ (les λ_k étant appelés coefficients d'importance des p fonctions objectifs ou paramètres de préférence).
2) Des vecteurs de performances ayant des significations particulières, comme le point de référence qui est défini par des niveaux d'aspiration (valeurs souhaitables) sur chaque critère et point de réservation qui est défini par des niveaux de réservation (valeurs non souhaitables) sur chaque critère.

Nous présentons dans la section suivante quelques méthodes fondamentales exactes développées de façon à obtenir un ensemble de solutions efficaces dite solutions Pareto optimales ou une partie du front Pareto. Les méthodes existantes sont classées selon la façon dont le décideur articule ses préférences.

2.3.2 Catégories de méthodes fondamentales

Dans la programmation mathématique multiobjectif, on distingue trois principales méthodes :

- **Les méthodes a priori**

  Le décideur indique l'importance relative des fonctions objectifs, il connaît a priori le paramètre de chaque objectif avant de combiner les différentes fonctions objectifs en une seule fonction d'utilité suivant les préférences du décideur (Théorie l'utilité multi-attribut), donnée par :

a. **Approches scalaires**

Le principe d'une catégorie qui forme les approches dites scalaires est de revenir à un problème monoobjectif via un ensemble de paramètres.
Étant donné un ensemble de paramètre de préférence : $\Lambda = \{(\lambda_1, ..., \lambda_p) \in \mathbb{R}^p\}$, les λ_k sont des coefficients d'importances des fonctions objectifs.
On définit une fonction croissante des objectifs et agrégeons les valeurs des objectifs pour chaque solution :

$$s(Z, \lambda) : \mathbb{R}^p \times \Lambda \to \mathbb{R}$$

où $(\Lambda \subset \mathbb{R}^p)$ est l'ensemble des paramètres. Les fonctions d'agrégation les plus employées sont :

Utilisation de la fonction d'utilité

L'ensemble des méthodes de la théorie de l'utilité multicritère repose sur l'axiome suivant :
Tout décideur essaye d'optimiser une fonction d'utilité

$$S_1(Z, \lambda) = U_k(Z_k) \quad k = \overline{1, p}, \quad (p \text{ fonctions objectifs})$$

Les modèles les plus utilisés sont le modèle additif et le modèle multificatif. U_k représente une fonction de mise à l'échelle du $k^{ème}$ objectif.

Critique :

Dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, soit par manque d'expérience, soit par manque d'information, l'utilisation de ces modèles impose que les objectifs soient compatibles.
Il est donc très difficile d'utiliser ces techniques lorsque l'ensemble des objectifs est composé à la fois d'objectifs qualificatifs et quantitatifs.

Caractérisation à l'aide des paramètres :

La somme pondérée : Cette méthode est largement utilisée en optimisation linéaire multiobjectif, elle consiste à additionner tous les objectifs en affectant à chacun un coefficient de paramètres. Ce coefficient représente l'importance relative que le décideur attribut à l'objectif, donc le problème d'optimisation multiobjectif (MOLP) peut se ramener à un problème de programmation paramétrique monoobjectif de la forme :

$$S_2(Z, \lambda) = \max_{x \in X} \sum_{k=1}^{p} \lambda_k Z_k(x),$$

$$\lambda \in \Lambda, \quad \Lambda = \left\{ \lambda \quad | \quad \sum_{k=1}^{p} \lambda_k = 1, \quad \lambda_k > 0, \quad k = \overline{1, p} \right\}$$

La fonction scalarisante $\sum_{k=1}^{p} \lambda_k Z_k(x)$ permet de caractériser entièrement ou partiellement l'ensemble de solutions efficaces.

Critique :

Cette procédure est efficace et très simple à mettre en oeuvre mais se heurte à quelques problèmes, tels que le choix des paramètres des objectifs et de la compensation entre les différents objectifs.

Choix des paramètres des objectifs
Une solution à ce problème est d'utiliser une combinaison linéaire des objectifs et de faire varier les paramètres de façon à constater l'influence de tel ou tel objectif sur le résultat.
La compensation entre les différents objectifs
Une valeur de base d'un objectif peut être composée par les valeurs des autres objectifs, cela peut conduire le décideur à faire le mauvais choix.

$$S_3(Z,\lambda) = \sum_{k=1}^{p} \lambda_k \left| Z_k - \overline{Z}_k \right|,$$

$\overline{Z}_k$ est la $k^{ème}$ composante du point idéal et $S_3(Z,\lambda)$ mesure la déviation qui sépare l'évaluation des propositions, qui sont généralement sur des points efficaces ou faiblement efficaces, du point d'aspiration ; cette déviation peut être mesurée par d'autres normes :
• Norme L_q pondérée :

$$S_4(Z,\lambda) = \left[\sum_{k=1}^{p} \lambda_k \left| Z_k - \overline{Z}_k \right|^q \right]^{1/q}, \qquad q \in \mathbb{Z}_+^*.$$

• Norme L_q de tchebychev pondérée :

$$S_5(Z,\lambda) = \max_{1\le k\le p} \left\{ \lambda_k \left| Z_k - \overline{Z}_k \right| \right\}.$$

• Norme composée (Tchebychev pondérée augmentée)

$$S_6(Z,\lambda) = \max_{1\le k\le p} \left\{ \lambda_k \left| Z_k - \overline{Z}_k \right| \right\} + \rho \sum_{k=1}^{p} \lambda_k \left| Z_k - \overline{Z}_k \right|, \qquad \rho > 0.$$

Caractérisation à l'aide de points cibles
•Niveau d'aspiration

$$S_7(Z,\lambda) = \left[\sum_{k=1}^{p} \lambda_k \left| Z_k - \hat{Z}_k \right|^q \right]^{1/q}, \qquad q \in \mathbb{Z}_+^*, \qquad \hat{Z}_k \in Z(X),$$

$\hat{Z}_k$ est un point cible qu'on souhaite être très près.
• Niveau de réservation

$$S_8(Z,\lambda) = \left[\sum_{k=1}^{p} \lambda_k \left| Z_k - {Z'}_k \right|^q \right]^{1/q},$$

et des contraintes sur les critères $Z_k \ge Z'_k$.
Z'_k représente un plus petit seuil.

Méthode de la programmation par but "Goal programming"
Dans cette méthode le décideur doit affecter des cibles (niveaux d'aspiration) ou des buts qu'ils souhaitent réaliser pour chacun des objectifs, ces valeurs sont incorporées au problème en tant que contraintes additionnelles, la fonction objectif essayera de réduire au minimum les déviations absolues des cibles aux

objectifs. La forme la plus simple de cette méthode peut être formulé comme suit :

$$\min_{x\in X}\left\{\sum_{k=1}^{p}\lambda_k\left|Z_k - Z_k^*\right|\right\},$$

où Z_k^* représente la cible fixé par le décideur pour le $k^{ème}$ critère. Le critère doit réduire au minimum la somme des valeurs absolues de la différence entre les valeurs à atteindre et les valeurs réellement réalisées. Il est possible d'utiliser cette technique de manière interactive en modifiant à chaque itération les niveaux d'aspiration.

Une formulation plus générale de la programmation but est de remplacer la fonction objectif ci-dessus par la $p^{ème}$ puissance de la déviation $|Z_k - Z_k^*|$. Une telle formulation est appelée la programmation par but généralisée. Différentes applications de cette technique ont été proposée dans Ignizio (1983).

Critique :

Nous pouvons reprendre la critique faite pour la somme pondérée, la méthode est facile à mettre en oeuvre mais la définition des paramètres et des buts à atteindre est une question délicate qui détermine l'efficacité de la méthode.

Cette méthode a l'avantage de fournir un résultat même si un mauvais choix initial a conduit le décideur à donner un ou plusieurs buts non réalisables.

Méthode ϵ-contrainte

Cette méthode est basée sur la maximisation d'un objectif Z_l en considérant que toutes les autres objectifs Z_k avec $k \neq l$ doivent être supérieur à une valeur ϵ_k.

$$P(\epsilon)\begin{cases}\max\limits_{x\in X} Z_l(x)\\ \qquad Z_k(x)\geq\epsilon_k, \qquad \forall k\neq l.\end{cases} \tag{2.6}$$

Le décideur peut ensuite réitérer ce processus sur un objectif différent jusqu'à ce qu'il trouve une solution satisfaisante.

Critique :

La connaissance a priori des intervalles appropriés pour les valeurs de ϵ_k est exigée pour tous les objectifs.

Il existe d'autres méthodes, la méthode de Vincke, Vanderpooten, mini-max, méthode de Zionts et Wallenius et d'autres [16].

b. **Approches non-scalaires**

Une autre catégorie forme les approches dites non-scalaires, ces méthodes ont des mécanismes spécifiques pour chaque objectif et considèrent les objectifs individuellement. Une méthode fait partie de cette catégorie couramment utilisée, il s'agit de la méthode lexicographique. Cette méthode est proposée par Fourman [37].

Méthode lexicographique

Dans cette méthode les critères sont rangés par ordre d'importance par le concepteur, la solution optimale est alors obtenue en optimisant les fonctions objectifs commençant par les plus importantes et le processus continu ainsi selon l'ordre d'importance affecté aux objectifs.

On suppose que les indices des fonctions objectifs indiquent non seulement le nombre de critères mais également la priorité de l'objectif, donc ils représentent les plus et les moins importants critères respectivement.

La méthode lexicographique peut s'exprimer de la manière suivante : supposons que les fonctions objectifs Z_i avec $k = 1, \ldots, p$ classées de telle sorte que si $k < l$ alors Z_k est prioritaire sur Z_l.
Le premier problème à résoudre peut être formulé par :

$$(LP_1) \begin{cases} \max Z_1(x) \\ t.q. \qquad x \in X \end{cases} \tag{2.7}$$

On obtient alors x_1^* la meilleure solution trouvée pour le problème (LP_1) cité par l'equation (2.7), associée à la valeur $Z_1^* = Z_1(x_1^*)$. Cette dernière devient alors une contrainte du problème associé à (LP_2) suivant :

$$(LP_2) \begin{cases} \max Z_2(x) \\ Z_1(x) = Z_1^* \\ t.q. \qquad x \in X \end{cases} \tag{2.8}$$

La meilleure solution trouvée pour le problème (LP_2) est x_2^*, associée à la valeur $Z_2^* = Z_2(x_2^*)$, cette dernière est ajoutée comme contrainte supplémentaire pour la résolution du problème (LP_3).
Les critères sont rangés en ordre de préférence puis optimiser l'un après l'autre dans cette ordre.
A l'itération k on résout le problème :

$$(LP_k) \begin{cases} \max Z_k(x) \quad 1 \leq k \leq p \\ Z_1(x) = Z_1^* \\ Z_2(x) = Z_2^* \\ \vdots \\ Z_{k-1}(x) = Z_{k-1}^* \\ \qquad x \in X \end{cases} \tag{2.9}$$

La solution finale dépend de l'ordre de préférence. L'algorithme 4 dans ce chapitre, fournit les étapes en détail de la méthode lexicographique.
Une méthode alternative pour générer les solutions lexicographiquement optimales, dans le cas d'un problème d'affectation biobjectifs ($p = 2$), se trouve dans l'article [82]. Cette méthode particulière, contrairement à l'algorithme 4 connu de la méthode lexicographique classique de ce chapitre, est basée sur un problème paramétrique, a l'avantage de ne pas modifier la structure du problème.

- **Les méthodes à posteriori**

Le décideur prend sa décision dans un ensemble de solutions, calculé par un solveur multiobjectif. Dans ce cas, la qualité de la décision dépend du choix de la méthode de résolution, car celle-ci va devoir donner un ensemble de résultats le plus représentatif de l'espace des objectifs efficaces. L'inconvénient de ces méthodes est que le nombre de solutions efficaces généré peut être tellement grand de sorte que l'on doit avoir recours à une procédure d'agrégation pour dégager là où les meilleures solutions se trouvent.

En général, ces approches comportent trois étapes :
- Recherche d'une solution initiale réalisable.
- Recherche d'une solution initiale efficace.
- Recherche des solutions efficaces.

- **Les méthodes interactives**

  Le déroulement de la plupart des méthodes interactives se résume par une alternance d'étapes de dialogue et de calcul.

  Critique :

  Dans une méthode interactive, c'est de la dynamique de l'interaction issue l'aide de la décision. Ces modèles exigent donc une connaissance approfondie de la part du décideur des outils utilisés. Ce n'est malheureusement pas souvent le cas. Leur domaine d'application se résume à des situations où le nombre d'actions est très élevé.

2.3.3 Quelques résultats de base

Considérons le problème (MOLP) définit par l'equation (2.4).

Les théorèmes suivants résument les résultats obtenus par Geoffrion (1968) pour le problème paramétrique $P(\lambda)$ donné par l'equation (2.10). Ces résultats mènent à classifier les solutions efficaces en deux catégories, cette classification sera présentée dans la section suivante.

Théorème 2.2 ***Somme Pondérée*** *[41] On peut écrire le problème paramétrique d'un problème multiobjectif comme suit :*

$$P(\lambda)\begin{cases}\max\left(\sum_{k=1}^{p}\lambda_k Z_k(x)\right), & \lambda\in\mathbb{R}^p_{\succ}\\ t.q.\ x\in X\end{cases} \tag{2.10}$$

Si x^* est une solution optimale du problème $P(\lambda)$, alors on a les résultats suivants :

1. *Si $\lambda\in\mathbb{R}^p_{\succ}$ alors, x^* est une solution faiblement efficace supportée X_{WSE}.*
2. *Si $\lambda\in\mathbb{R}^p_{\succ\succ}$ alors, x^* est une solution efficace supportée X_{SE}.*
3. *Si $\lambda\in\mathbb{R}^p_{\succ}$ et si x^* est l'unique solution optimale de $P(\lambda)$ alors, x^* est une solution efficace supportée.*

Théorème 2.3 *[41]*

Soient X et Z tel que $Z(X)=Y$ convexe (dans ce cas $X_E=X_{SE}$),

1. *Si $x^*\in X_{WSE}$, alors il existe $\lambda\in\mathbb{R}^p_{\succ}$ tel que x^* est une solution optimale pour le problème $P(\lambda)$.*
2. *Si $x^*\in X_{SE}$, alors il existe $\lambda\in\mathbb{R}^p_{\succ\succ}$ tel que x^* est une solution optimale pour le problème $P(\lambda)$.*

Théorème 2.4 *Soland (1979)*

Un point $x\in X$ est une solution efficace pour le problème (MOLP) si et seulement si x est une solution optimale du problème $P(\epsilon)$ défini par l'équation (2.6).

2.3.4 Classification des solutions efficaces

Il existe une importante classification des solutions efficaces : les solutions efficaces supportées et les solutions efficaces non-supportées (supported efficient solutions and non-supported efficient solutions). On peut caractériser ces solutions comme suit [26, 67] :

- Les solutions efficaces supportées : notées X_{SE}, elles peuvent être obtenues par résolution du problème d'optimisation combinatoire unicritère paramétrique obtenu par agrégation linéaire de différents objectifs $P(\lambda)$, donnés par l'equation (2.10).
  Les images des solutions efficaces supportées dans l'espace des objectifs sont situées sur la frontière de l'enveloppe convexe de Y, notée $conv(Y)$.
- Les solutions efficaces non-supportées : notées $X_{NE} = X_E \setminus X_{SE}$, cette partie de solutions efficaces ne peut pas être obtenue par résolution du problème uniobjectif paramétrique. Les images des solutions non-supportées ne sont pas situées sur la frontière de l'enveloppe convexe.

La figure 2.7 représente les différents types de solutions trouvées dans l'espace des objectifs pour le cas d'un exemple du problème dont les objectifs sont à maximiser et le nombre d'objectifs est $p = 2$.

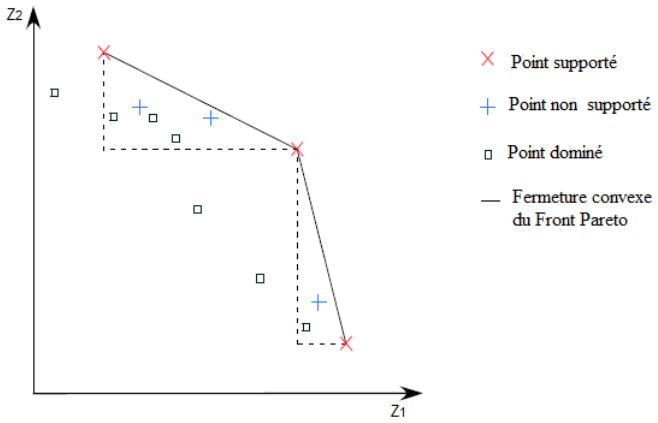


FIGURE 2.7 – Différents types de solutions efficaces

En plus, on peut distinguer deux catégories de solutions supportées, les solutions supportées extrêmes X_{SE1} et les solutions supportées non extrêmes $X_{SE2} = X_{SE} \setminus X_{SE1}$.

a. Les solutions supportées extrêmes (Extreme supported efficient solutions) : leurs images se situent sur un des sommets de $conv(Y)$.
b. Les solutions supportées non extrêmes (Non extreme supported efficient solutions) : leurs images se situent sur les facettes de $conv(Y)$.

On présente un exemple numérique pour extraire la notion des solutions supportées et non-supportées, cet exemple est introduit par Bowman [16]).

Exemple 2.2 Soit :

$$(P^{Bow}) \begin{cases} \max Z_1 = 6x_1 + 3x_2 + x_3 \\ \max Z_2 = x_1 + 3x_2 + 6x_3 \\ x_1 + x_2 + x_3 \leq 1 \\ x_1, x_2, x_3 \in \{0, 1\} \end{cases} \tag{2.11}$$

Il est clair d'après le théorème de Geoffrion que les solutions optimales du problème $(P^{Bow}(\lambda))$ donné par l'équation (2.12) :

$$P^{Bow}(\lambda)\begin{cases}\max Z = \lambda(6x_1+3x_2+x_3)+(1-\lambda)(x_1+3x_2+6x_3)\\ tq \quad x_1+x_2+x_3 \leq 1\\ \quad x_1, \quad x_2, \quad x_3 \in \{0,1\} \quad \text{avec} \quad 0<\lambda<1\end{cases} \tag{2.12}$$

sont des solutions efficaces supportées.

L'ensemble des solutions admissibles du problème est composé de trois solutions $X_1 = (1,0,0)$, $X_2 = (0,1,0)$, et $X_3 = (0,0,1)$. Alors $X_E = \{X_1, X_2, X_3\}$ et $Y_N = \{Z(X_1), Z(X_2), Z(X_3)\} = \{(6,1),(3,3),(1,6)\}$.

La table 2.1 ci-dessous résume toutes les solutions efficaces et les valeurs correspondantes des objectifs.

Z	Z_1	Z_2
X_1	6	1
X_2	3	3
X_3	1	6

TABLE 2.1 – Solutions non-dominées $Z(X_E) = Y_N$

Les solutions X_1 et X_3 sont des solutions efficaces supportées, obtenues du problème $(P^{Bow}(\lambda))$, mais la solution X_2, qui est aussi efficace n'est pas une solution optimale du problème $(P^{Bow}(\lambda))$. Si on représente les points non-dominés, voir (Figure 2.8), on remarque que $(3,3)$ est un point non-supporté tandis que les points $(1,6)$ et $(6,1)$ sont des points supportés. Cependant, le seul point non-dominé établi par un meilleur compromis entre les objectifs est non-supporté.

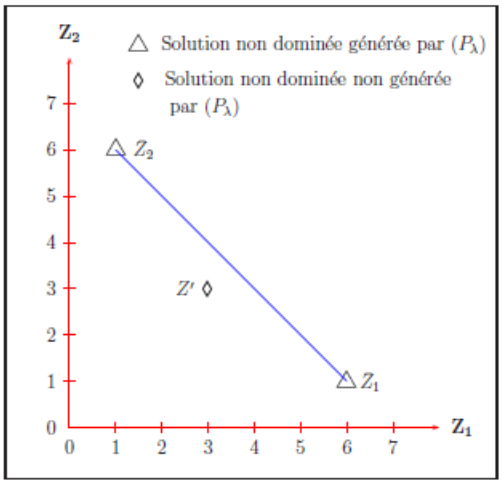


FIGURE 2.8 – Représentation des points non-dominés du problème (P^{Bow}).

Alors, parmi les solutions efficaces, nous pouvons en distinguer deux types de solutions [96] :

- Les solutions efficaces supportées X_{SE} : celles qui sont situées sur l'envelope convexe de l'ensemble de solutions. Ces solutions efficaces X_1 et X_3 ont été obtenues en résolvant le problème paramétrique $P^{Bow}(\lambda)$.

- Les solutions efficaces non supportées X_{NSE} : ces dernières ne peuvent pas être obtenues par résolution du problème paramétrique. La seule solution trouvée est X_2.

Le choix final d'une solution efficace est effectué par un décideur qui décide d'obtenir un compromis reflétant ses demandes, pourrait ne pas être satisfait du seul choix apporté par l'ensemble des solutions supportées. En effet, ces derniers peuvent ne représenter qu'une faible partie de solutions efficaces [98]. En conclusion, la difficulté rencontrée dans la résolution des problèmes (MOILP) est la détermination de l'ensemble des solutions non supportées X_{NSE} qui ne sont pas solutions du problème paramétrique pour aucun paramètre λ.

2.4 Problèmes d'optimisation multiobjectifs en nombres entiers

Dans de nombreuses situations réelles modélisables par la programmation mathématique, les variables qui doivent apparaître dans la modélisation sont supposées à être partiellement ou totalement en nombres entiers. On parle alors de programmation en nombres entiers (ILP : Integer Linear Programming). Les problèmes où toutes les variables doivent être entières sont appelés "problèmes totalement en nombres entiers", ceux dont une partie seulement des variables doivent être entières sont dits "problèmes mixtes". Lorsqu'on impose aux variables de décision du (MOLP) d'être entières, nous aurons le problème (MOILP) définit par l'équation (2.3), dont la structure est complètement différente de celle du (MOLP), cette différence est due au fait que l'ensemble X est convexe par contre l'ensemble D n'est pas convexe, et donc le principe de Geoffrion qui permet de caractériser l'ensemble de solutions efficaces par les solutions optimales du problème paramétrique n'est plus valable, tel qu'il représente qu'une partie des solutions efficaces appelée solutions supportées "supported solutions". Sur cette base l'idée d'efficacité peut être élargie en répartissant l'ensemble des solutions efficaces en deux sous-ensembles. L'ensemble de solutions optimales du $P(\lambda)$ où X est remplacé par D, on le note par X_{SE} l'ensemble des solutions efficaces supportées du problème MOILP et X_{NSE} l'ensemble des solutions efficaces non-supportées.

Dans la section suivante, nous présenterons quelques méthodes de résolution pour les problèmes d'optimisation multiobjectifs en nombre entiers.

2.4.1 Méthodes de résolution

Le développement des méthodes pour la programmation linéaire multiobjectif à variables continues (MOLP) a été entamé sur les problèmes multiobjectifs à variables entières (MOILP). Cependant, la prise en compte des variables entières dans le contexte multiobjectif amène des difficultés particulières, les tentatives de combinaison simple d'une méthode de résolution de MOLP avec une méthode classique pour la résolution des problèmes à variables entières (ILP) ne donne pas en général les résultats voulus. Pour surmonter ces difficultés et obtenir des algorithmes performants, un développement des méthodes spécifiques pour résoudre les problèmes de types (MOILP) est exigé. Il existe un grand nombre de méthodes de résolution dédié spécialement aux problèmes d'optimisation en nombres entiers, nous présenterons quelques unes :

Méthode de Klein et Hannan (1982)

Cette méthode consiste à résoudre progressivement une séquence de programmes linéaires en nombres entiers, avec des contraintes ajoutées à chaque itération.
La méthode génère des solutions efficaces de façon que l'utilisateur ne soit pas obligé de déterminer entièrement l'ensemble des solutions efficaces s'il ne s'intéresse qu'à quelques unes. L'algorithme de la méthode est décrit dans [17].

Méthode de Villareal-Karwan

Cette méthode est une extension des techniques de résolution du problème de sac à dos à l'aide de la programmation dynamique.
Dans le problème (MOILP), les coefficients des matrices C et A sont supposés entiers positifs et $N_j \quad (j = 1, 2, \ldots, n)$ est un entier positif considéré comme borne supérieure de la variable x_j.
En posant $Y^i = \sum_{j=1}^{i} a^j x_j$ où $a^j(m, 1)$ est la $j^{ème}$ colonne de la matrice A, introduisons le problème suivant :
A la $i^{ème}$ itération :

$$(P_i(Y^i)) \begin{cases} \max \sum_{j=1}^{i} C^j x_j \\ t.q \\ \quad \sum_{j=1}^{i} a_j x_j \leq Y^i \\ \quad 0 \leq x_j \leq N_j, \quad j = 1, \ldots, i \end{cases} \tag{2.13}$$

où $0 \leq Y^i \leq b$ et $X_E(P_i(Y^i))$ est l'ensemble de solutions efficaces correspondant.
Ce problème peut être résolu en considérant la programmation dynamique multiobjectif à l'itération i.

$$\begin{cases} H_i(Y^i) = \text{“max”}(c^i x_i \oplus H_{i-1}(Y^{i-1})) \\ a_i \leq Y^i - Y^{i-1} \\ 0 \leq x_j \leq N_j \end{cases} \tag{2.14}$$

c^i est le $i^{ème}$ vecteur de la matrice C et $H_{i-1}(Y^{i-1})$ correspond à l'ensemble de solutions de l'étape $i-1$, avec un second membre de contraintes Y^{i-1} et $\oplus$ indique que la somme s'effectue sur tous les éléments de $H_{i-1}(Y^{i-1})$.

Ces équations récursives sont appliquées pour chaque vecteur second membre Y^i, $0 \leq Y^i \leq b$ et pour chaque $i^{ème}$ itération du problème.
Dans l'étape finale n, $X_E(P) = X_E(P_n(b))$.

Méthode de Chalmet et al.

Cette méthode a été proposée par Chalmet et al. pour résoudre un problème biobjectif en variables discrètes.
Le problème monocritère $\max_{x \in D} \{\lambda Z_1(x) + (1 - \lambda) Z_2(x)\}$ est résolu afin de générer l'ensemble des solutions supportées X_{SE} du problème (MOILP), puis l'ensemble des solutions non-supportées qui sont déterminées comme suit :
Soient $Z^1 = (Z_1^1, Z_2^1)$ et $Z^2 = (Z_1^2, Z_2^2)$ deux points non-dominés, adjacents parmi les points déjà obtenus.
On cherche s'il y a une solution non-dominée entre ces points en résolvant le problème

suivant :

$$(P')\begin{cases}\max \lambda Z_1(x)+(1-\lambda)Z_2(x)\\ t.q \quad Z_1(x)\geq Z_1^1\\ \quad\ Z_2(x)\geq Z_2^2\\ \quad\ x\in D\\ \quad\ \lambda\in[0,1]\,.\end{cases} \tag{2.15}$$

Soit Z^l la solution efficace s'il y a des solutions à ce problème.
La même procédure est appliquée ensuite avec les paires (Z^1, Z^l) et (Z^l, Z^2). S'il n'existe pas de solution pour le problème (P') alors Z^1 et Z^2 sont deux solutions non-dominées adjacentes. La méthode prend fin lorsque toutes les paires ont été examinées.

Méthode de Gonzales, Reeves et Franz : Cette méthode est interactive, elle comporte deux phases :

- Un sous-ensemble E de solutions efficaces (supported solutions) est d'abord déterminée (ensembles de solutions préférées par le décideur).
- Les solutions non supportées sont ensuite présentées l'une après l'autre au décideur, si x^*(est l'une d'elles) est préférable à une solution x de E, x est rejetée de E et elle est remplacée par x^* (l'algorithme de la méthode est décrit dans [17]).

Méthode de Alves et Climaco

Cette méthode est interactive, elle combine l'utilisation de la distance de Tchebychev avec la technique des coupes de Gomory (1963), à chaque itération, l'algorithme proposé fournit la solution non-dominée la plus proche du point de référence fixé par le décideur. L'analyse des informations données par le décideur, à chaque étape de dialogue, permet d'ajuster le point de référence de l'itération suivante [17].

Méthode MOMIX de Teghem et Kunch (1986)

C'est une version interactive de la méthode Branch & Bound. Elle comporte deux phases :

La phase 1

C'est une phase d'initialisation, il s'agit de déterminer le premier point efficace x^0 en minimisant la norme de Tchebychev. Le problème suivant est résolu pour $m = 0$.

$$(P_m)\begin{cases}\min \delta\\ t.q. \quad \beta_{m,k}(M_{m,k}-c^k x)\leq \delta \quad k=1,2,...,p\\ \quad\ x\in X_m\end{cases} \tag{2.16}$$

où $X_0 = X$; $M_{m,k}$ la valeur optimale de la $k^{ème}$ fonction objectif sur X_m et $\beta_{m,k} = \frac{\alpha_{m,k}}{\sum_{i=1}^{p}\alpha_{m,k}}$ le paramètre de la $k^{ème}$ fonction objectif par exemple $\alpha_{m,k} = \frac{M_{m,k}-n_{m,k}}{max(|n_{m,k}|,|M_{m,k}|)}\frac{1}{\|C^k\|}$;
où $n_{m,k}$ est la $k^{ème}$ composante du point nadir sur X_m.

La phase 2

La méthode Branch & Bound interactive est utilisée en deux étapes :

Première étape (Procédure descendante "depth first") :

Soit la $m^{ème}$ itération.

- Soit x^{m-1} le $m^{ème}$ compromis et $Z_k = C^k x^{m-1}$ la valeur correspondante pour la $i^{ème}$ fonction objectif.
- Le DM indique le critère $l_m(1) \in \{1, 2, ..., p\}$ à améliorer en priorité.

- Un nouveau compromis est obtenu en résolvant le problème (P_m) avec $X_m = X_{m-1} \cap \left\{x | Z_{lm(1)}(x) > Z_{lm(1),(m-1)}\right\}$ de façon que le critère $l_m(1)$ est amélioré.

Test d'arrêt

Des tests d'arrêts ont été définis, on cite quelques uns :

a) $X_m = \phi$.
b) $M_{k,m} - m_{k,m} \leq \epsilon_k$ pour toutes les valeurs k ($\epsilon_k > 0$, fixé).
c) $\hat{Z}$, le vecteur des meilleurs valeurs des fonctions objectifs fournies par le meilleur compromis trouvé est préféré au vecteur idéal M_m.
d) Un nombre Q d'itérations a été effectué.

Si aucune amélioration n'a été apporté à $\hat{Z}$ après un nombre Q d'itérations fixé par le décideur, la première étape de la procédure s'arrête.

Deuxième étape (Procédure remontante " backtracking") :

La région d'admissibilité négligée à chaque itération de la procédure descendante est scannée pour un éventuel meilleur compromis que $\hat{Z}$ dans cette région. Les différentes opérations sont :

- Le noeud correspondant au compromis x^{m-1} est séparé en p branches.
- Soit $l_m(k); \quad k = 1, 2, ..., p$ l'ordre de priorité dans lequel le décideur veut voir améliorer les p critères par rapport à ce compromis ;
- Les p sous noeuds sont obtenus par adjonction respective des contraintes suivantes :

1) $Z_{lm}(1) > Z_{lm}^{m-1}(1)$.
2) $Z_{lm}(2) > Z_{lm}^{m-1}(2), \quad Z_{lm}(1) \geq Z_{lm}^{m-1}(1)$.
$\vdots$
$p)$ $Z_{lm}(p) > Z_{lm}^{m-1}(p), \quad Z_{lm}(k) \geq Z_{lm}^{m-1}(k). \quad k = 1, 2, \ldots p.$

L'ensemble de solutions admissibles est divisé et à chaque sous noeud on résout le problème (P_m).

Les mêmes tests d'arrêts sont utilisables.

Il reste à citer la méthode de Ralph Stuer et Choo, la méthode de Macott et Soland, l'algorithme de Moulaî et celui de Chaabane et d'autres [16, 17].

2.5 Conclusion

Nous avons rapporté dans la première partie de ce chapitre les principales définitions nécessaires à la présentation des problèmes d'optimisation multiobjectifs en nombre continus (MOLP), et quelques méthodes de résolution développées à ces problèmes. Entre autre, nous avons mis en évidence les difficultés inhérentes aux problèmes (MOILP) à travers quelques résultats de base donnés par le théorème de Géoffrion (1968). D'autres méthodes de résolution ont été présentées pour la catégorie des problèmes d'optimisation multiobjectifs en nombres entiers. Cependant, il existe un nombre important de problèmes en nombres entiers qui ne peuvent se formuler que comme problèmes à variables binaires. Le chapitre suivant est dédié aux problèmes d'optimisation combinatoire multiobjectifs (MOCO), nos travaux se situent dans ce dernier domaine.

Chapitre 3

Problèmes d'Optimisation Combinatoire Multiobjectifs (MOCO)

3.1 Introduction

Un nombre considérable de problèmes en nombres entiers est formulé comme problèmes à variables qui ne peuvent prendre que deux valeurs 0 ou 1, il s'agit des problèmes d'optimisation combinatoire multiobjectifs (MOCO : Multi-Objective Combinatorial Optimisation). L'optimisation combinatoire occupe une place très importante en recherche opérationnelle et en mathématiques discrètes. Son importance se justifie d'une part par la grande difficulté des problèmes associés et d'autre part par le nombre important d'applications réelles pouvant être modélisées sous sa forme : tel que le problème d'affectation "Assignement Problem", le problème de voyageur de commerce "Traveling Salesman Problem", le problème de couverture d'ensemble "Set Covering", le problème de transport, le problème de location, le problème du plus court chemin, le problème de sac à dos "Knapsack Problem",..., etc. Bien que les problèmes d'optimisation combinatoire sont souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles. Dans ce chapitre, nous exposerons la formulation des problèmes (MOCO) ainsi que les différents types de problèmes d'optimisation multicritères (MOCO) et nous terminerons par quelques méthodes de résolution concernant ce type de problèmes.

3.2 Quelques applications classiques des problèmes (MOCO)

Les problèmes d'optimisation combinatoire multiobjectifs (MOCO) sont au centre d'un grand intérêt dans la littérature depuis plusieurs années. Le problème MOCO peut s'écrire comme suit :

$$(MOCO)\begin{cases} \text{“max”}\, Z_k = \quad C^k x \quad k = \overline{1,p} \\ t.q.\ \ x \in D \\ D \subset \{0,1\}^n \end{cases} \tag{3.1}$$

La structure combinatoire est associé à différents problèmes dans la littérature. Certains exemples seront envisagés et définis dans la section suivante, parmi les nombreux types

de problèmes MOCO existants [28, 29].

3.2.1 Problème d'affectation multicritère

Le problème d'affectation est un problème qui tient une place centrale dans le domaine de l'optimisation combinatoire, par le fait qu'il possède de nombreux liens avec d'autres problèmes. Il est un cas particulier du problème de transport, aussi il se présente comme une relaxation d'un problème plus complexe, il s'agit du problème de voyageur de commerce [93]. Le problème d'affectation "Assignment problem"(AP) est un problème spécifique des problèmes linéaires en variables binaires. Il consiste à affecter n personnes (ressources, équipements, localisations,...) à n tâches (travaux, activités, services,...). A titre d'exemple, affecter des ouvriers à des travaux, des professeurs à des classes, des équipages à des vols aériens, ou encore, les différents services d'un hôpital aux différents emplacements disponibles,...,etc.

Désignons par $i = 1, \ldots, n$: les personnes et par $j = 1, \ldots, n$: les tâches. Notons c_{ij}^k le coefficient du $k$ème critère. Ces valeurs peuvent être un coût associé à l'affectation d'une personne i à une tâche j, une durée d'affectation, un indice de préférence exprime par une personne i, ou un facteur de fiabilité.

Introduisons les variables binaires :

$$x_{ij} = \begin{cases} 1 & \text{si la personne } i \text{ est affecté à la tâche } j, \\ 0 & \text{sinon.} \end{cases} \tag{3.2}$$

Les contraintes du problème d'affectation s'écrivent de la façon suivante :

- La contrainte donnée par l'equation (3.3) veut dire que la personne i doit être affectée à une et une seule tâche.

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n. \tag{3.3}$$

- La contrainte donnée par l'equation (3.4) veut dire que la tâche j ne peut être affectée qu'à une et une seule personne.

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n. \tag{3.4}$$

Le problème d'affectation multiobjectif peut s'écrire comme suit :

$$(MOAP) \begin{cases} \text{“min”} Z_k = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij}, \quad k = 1, \ldots, p \\ \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\ x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n. \end{cases} \tag{3.5}$$

Dans le cas uniobjectif, le problème d'affectation est un problème de complexité polynomial (appartient à la classe P), et il est particulièrement un des plus simple de l'optimisation combinatoire. Il possède une matrice totalement unimodulaire (matrice

dont tous les sous déterminants valent $-1,\ 0$ ou $+1$). Par contre dans le cas multiobjectif, il est un des problèmes NP-difficiles. Un tableau représente les résultats concernant la complexité de quelques problèmes classiques d'optimisation combinatoire dans leurs versions multicritère et unicritère se trouve dans [67, Chapitre 2, p 30].

3.2.2 Problème de transport multicritère

Le problème de transport multicritère est une extension du problème d'affectation multicritère [93, 97]. Supposons qu'on transportent un produit à partir de r usines (dépôts) disposant de quantités $q_1, q_2, \ldots, q_r$ vers s destinations (ateliers, magasins,...) où des demandes $d_1, d_2, \ldots, d_s$ doivent être satisfaites. Le problème n'aura de solution que dans la mesure où la quantité totale disponible permet de satisfaire la demande totale, c'est à dire $\sum_{i=1}^{r} q_i \geq \sum_{j=1}^{s} d_j$. On associe des valeurs c_{ij}^k le coût de transport, délai de livraison, sécurité de livraison, quantité fournie au transport d'une unité de produit de l'origine i à la destination j. Il s'agit de déterminer comment approvisionner les dépôts à partir des ateliers de manière à minimiser le coût total et le délai de livraison et d'autres objectifs de transport.

Soit x_{ij} la quantité de biens transporté de i vers j ; le problème de transport multicritères peut se formuler comme suit :

$$(MOTP)\begin{cases} \text{“min”} \sum_{i=1}^{r}\sum_{j=1}^{s} c_{ij}^k x_{ij}, \quad k = 1, \ldots, p \\ \sum_{j=1}^{s} x_{ij} = q_i, \quad i = 1, \ldots, r, \\ \sum_{i=1}^{r} x_{ij} = d_j, \quad j = 1, \ldots, s, \\ x_{ij} \geq 0, entier \quad \forall i, j. \end{cases} \tag{3.6}$$

3.2.3 Problème de voyageur de commerce multicritère

Le problème du voyageur de commerce multicritère, noté (MOTSP) "Multi-Objective Travelling Salesman Problem", fait partie de l'optimisation combinatoire. Ce problème était l'objet de plusieurs études spécifiques et complexes, donnant lieu à deux ouvrages intéressants cités dans [93] faisant une synthèse des progrès réalisés dans le cas uniobjectif. Il est simple à définir : Soit un voyageur de commerce qui doit réaliser une tournée en visitant une et une seule fois n villes tout en minimisant les divers frais et risques occasionnés par ces déplacements.

Considérons un graphe $G(N, A)$ avec $N = \{1, 2, \ldots, n\}$ l'ensemble de sommets et $A = \{(i, j); i, j \in N, i \neq j\}$ l'ensemble d'arêtes ; les n sommets représentent les villes et les arrêtes (i, j) représentent les routes entre deux villes. Si les $c_{ij}^k, \quad k = 1, \ldots, p$ sont les diverses valeurs (distance ou coût) associés au déplacement de la ville i vers la ville j. Le problème revient à rechercher un cycle hamiltonien qui soit de coût minimum. Soit x_{ij} la variable qui vaut 1 si l'on effectue le trajet allant de i vers j et 0 sinon. Le problème $MOTSP$ peut se formuler :

$$(MOTSP)\begin{cases} \text{“min”} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{k} x_{ij}, \quad k=1,\ldots,p \\ \sum_{i=1}^{n} x_{ij} = 1, \quad \forall j. \\ \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i. \\ \sum_{i\in T}\sum_{j\in \overline{T}} x_{ij} \geq 1, \quad \forall T \subset N, \quad \overline{T} = N \backslash T \\ x_{ij} \in \{0,1\}, \quad \forall i,j \end{cases} \tag{3.7}$$

Ce problème très simple est l'un des plus connus en recherche opérationnelle, correspond à un problème NP-difficile. La difficulté du problème réside dans le fait qu'il ne peut y avoir de sous-tours dans la permutation, vu son caractère cyclique. Contrairement au problème d'affectation uniobjectif qui est un problème de complexité polynomial, le problème de voyageur de commerce (TSP) est NP-difficile.

3.2.4 Problème de couverture multicritère

Le problème de couverture multicritères noté MOSCP (Multiobjective Set Covering Problem) peut se présenter comme suit : considérons que nous devons atteindre tout un ensemble de localités et que nous pouvons prendre un certain nombre de routes ayant chacune k coûts $c_j^k,\ k=1,\ldots,p$ (p coûts associés à chaque route) d'utilisation pour y arriver. Il s'agirait donc de trouver les "meilleures" routes possibles pour atteindre toutes les localités, c'est à dire déterminer l'ensemble de routes des coûts minimaux couvrant toutes les localités.
Définissons la notion de couverture mathématiquement : Soit $I = \{1,2,\ldots,m\}$ un ensemble de m objets, $i \in I$. Considérons n sous ensembles H_j de I, $j \in J = \{1,2,\ldots,n\}$. Une couverture de I est un ensemble de H_j tel que $j \in J^*$ vérifiant la condition $\bigcup_{j\in J^*} H_j = I$. Soit $c_j^k,\ k=1,\ldots,p$ le coût associé à H_j pour l'objectif k, le problème multicritère de couverture peut s'écrire de la façon suivante :

$$(MOSCP)\begin{cases} \text{“min”} Z^k = \sum_{j=1}^{n} c_j^k x_j, \quad k=1,\ldots,p \\ \sum_{j=1}^{n} t_{ij} x_j \geq 1, \quad \forall i \in I, \\ \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i, \\ x_j \in \{0,1\}. \end{cases} \tag{3.8}$$

où

$$t_{ij} = \begin{cases} 1 & \text{si } i \in H_j, \\ 0 & \text{sinon.} \end{cases} \tag{3.9}$$

et

$$x_j = \begin{cases} 1 & \text{si } H_j \text{ appartient à la couverture,} \\ 0 & \text{sinon.} \end{cases} \tag{3.10}$$

et $c_j^k > 0, \quad \forall (k,j)$.

3.2.5 Problème du plus court chemin multicritère

Le problème du plus court chemin noté (MOSPP)" Multiple Objective Shortest Path Problem" est un problème qui peut être une interprétation du problème de sac à dos. Son contexte est le suivant :

On considère un graphe orienté acyclique $G(N, A)$ où $N = 1, \ldots, n$ est un ensemble de sommets et A est un ensemble d'arcs reliant des couples de sommets. A chaque arc (i, j) on associe p valeurs positives $c^k_{(i,j)}, \quad k = 1, \ldots, p$; pouvant représenter un coût, un temps, une distance,..., de parcours de i vers j. Soit $1 \in N$ et $n \in N$ deux sommets fixés, l'ensemble de tous les chemins reliant 1 à n est noté (CH). A chaque chemin $K \in CH$ correspond aux valeurs $Z_k(K) = \sum_{(i,j)\in K} c^k_{(i,j)}, \quad k = 1, \ldots, p.$

Le problème du plus court chemin multicritère consiste à minimiser ces p valeurs. Sa formulation mathématique peut s'écrire de la façon suivante :

$$(MOSPP) \begin{cases} \text{“min”} Z^k = \sum_{i=1}^{n} \sum_{(i,j)\in K} c^k_{(i,j)} x_{ij}, \quad k = 1, \ldots, p \\ \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} 0 & \text{si } i \neq 1, n \\ 1 & \text{si } i = 1 \\ \text{-1} & \text{si } i = n \end{cases} \end{cases} \tag{3.11}$$

où

$$x_{ij} = \begin{cases} 1 & \text{si l'arc } (i, j) \in K, \\ 0 & \text{sinon.} \end{cases} \tag{3.12}$$

3.2.6 Problème de sac à dos multicritère

Désormais, nous nous intéressons à un problème dont la structure mathématique peut décrire aussi un grand nombre de problèmes d'optimisation, il s'agit du problème du sac à dos, ce problème a été abondamment traité dans la littérature pour les raisons suivantes : sa formulation simple, son interprétation économique propre (problème de chargement et d'investissement), il apparaît souvent comme sous-problème d'autres problèmes d'optimisation combinatoire (problème d'affectation généralisé,...). Après avoir vu de quoi consiste ce problème dans sa version unicritère (voir chapitre 1) et les études larges dans la littérature concernant ce problème [60, 70]. Nous allons consacré le chapitre 4 à une étude détaillée de sa structure dans le cas multiobjectif et ses méthodes de résolution développées dans la littérature.

Il existe d'autres types de problèmes d'optimisation multicritères (MOCO) [28, 29].

3.3 Différentes méthodes de résolution pour les problèmes MOCO

Dans les problèmes MOCO, le nombre de solutions efficaces non supportées est très important comparant avec celui des solutions efficaces supportées [98]. Il n'existe pas de réponses théoriques pour ce phénomène, mais pour plusieurs problèmes MOCO, on observe que le nombre de solutions efficaces supportées croit de façon polynomial avec la taille de l'instance tandis que le nombre de solutions non supportées croit exponentiellement avec la taille du problème [67, chapitre 7,8].

Une littérature abondante et riche concernant les méthodes de résolution des problèmes d'optimisation combinatoire multiobjectifs (MOCO), exactes et heuristiques. On présentera dans ce chapitre les méthodes souvent utilisées [66, 67, 26, 27, 28, 29]. Le but de ces méthodes est la détermination du front pareto ou une approximation du front pareto de différents problèmes MOCO. Pour se faire il existe deux approches principales de résolution d'un problème d'optimisation multicritère qui varient entre les méthodes exactes (Branch & Bound : procédure de séparation et évaluation), et les méthodes heuristiques et les méta-heuristiques.

3.3.1 Les méthodes exactes

Les méthodes exactes ont la particularité de garantir l'optimalité de la solution obtenue. De nombreuses méthodes utilisent des techniques de scalarisation pour résoudre des problèmes multiobjectifs reformulés en problèmes monoobjectifs. On expose dans cette sous section, quelques méthodes de résolution développées pour les problèmes MOCO, on commence par les méthodes scalarisantes, la méthode des sommes pondérées telles que l'algorithme de Benson [9], la méthode ϵ contrainte [50] ensuite la méthode mixte dite "the hybrid method" et on termine par la méthode en deux phases qui a été appliquée sur un grand nombre de problèmes d'optimisation biobjectifs et moins de travaux concernant plus de deux objectifs [96, 98, 83].

Les méthodes scalarisantes

L'approche traditionnelle pour résoudre les problèmes d'optimisation multiobjectifs est la scalarisation. Cette méthode populaire transforme un problème multiobjectif à un problème monoobjectif en combinant linéairement les différents objectifs. La scalarisation est une technique simple et naturelle développée pour résoudre les problèmes d'optimisation multiobjectifs en utilisant des sommes pondérées [9]. Le problème d'optimisation multiobjectif est transformé à un problème uniobjectif suivant :

$$\max_{x \in X} \sum_{k=1}^{p} \lambda_k Z_k(x), \qquad \text{avec } \lambda_k \in \mathbb{R}^p_{\succ}.$$

Le théorème de Geoffrion [41] indique qu'en utilisant différentes valeurs pour le vecteur λ, il est possible d'obtenir toutes les solutions supportées du problème multiobjectif. Par contre, aucune solution non-supportée ne peut être trouvée par cette méthode. Cependant, pour les problèmes MOCO, l'ensemble de solutions non-dominées est généralement non convexe. Cette méthode peut donner seulement une approximation de l'ensemble de solutions efficaces X_E, composé des solutions supportées. Parmi les premiers papiers étudiants la résolution exacte des problèmes MOCO, citons la méthode développée par Aneja et Nair [3], cette dernière est basée sur le calcul des solutions supportées du problème de transport biobjectif. Elle consiste à trouver les paramètres qui permettent d'obtenir tous les points supportés extrêmes. La méthode n'utilise aucune spécificité du problème de transport, donc elle peut être appliquée à n'importe quel autre problème. Elle est souvent utilisée pour déterminer un ensemble de solutions supportées d'un problème d'optimisation combinatoire au cas biobjectif.

Les fonctions de sommes pondérée de Tchebychev [92]

Une autre approche connue dans la littérature concernant l'optimisation est la scalarisation des fonctions de Tchebychev où la première apparition était en 1976 [11], la méthode génère des solutions non supportées.
Soit $\|Z^0-Z(x)\|_\lambda$ est la distance de la pondération de Tchebychev $Z(x) = (Z_1(x), \ldots, Z_p(x))$ à Z^0, le point de référence sur l'espace des objectifs, tel que $Z^0 \geq Z(x), \forall x \in X$. Donc :

$$\|Z^0 - Z(x)\|_\lambda = \max_{k=1,\ldots,p} \{\lambda_k(Z_k^0 - Z_k(x))\}.$$

La solution optimale de $\min_{k=1,\ldots,p} \|Z^0 - Z(x)\|_\lambda$ est au moins une solution faiblement efficace, et le paramètre $\lambda \in \Lambda = \{(\lambda_1, \lambda_2, \ldots, \lambda_p) : \ \lambda_k \succ 0, \ \forall k, \ \sum_{k=1}^{p} \lambda_k = 1\}$ de $\min_{k=1,\ldots,p} \|Z^0 - Z(x)\|_\lambda$ génère toutes les solutions efficaces.
On reformule la dernière phrase de la manière suivante :
Le problème paramétrique uniobjectif de Tchebychev est donné comme suit :

$$\min_{k=1,\ldots,p} \|Z^0 - Z(x)\|_\lambda \tag{3.13}$$

On obtient le théorème suivant :

Théorème 3.1 *[92, chapitre 14]*
Soit $\hat{x}$ une solution optimale de (3.13) :

- *Si $\lambda \succ 0$ alors $\hat{x} \in X_{WE}$.*
- *Si $\hat{x} \in X_E$, alors il existe $\lambda \succ 0$ tel que $\hat{x}$ est une solution optimale pour le problème cité par l'equation (3.13).*

On remarque que l'hypothèse de convexité n'est pas nécessaire pour la fonction de somme pondérée de Tchebychev et avec cette technique, il est ainsi possible de générer toutes les solutions efficaces en variation de paramètre λ. Bien que ces propriétés sont intéressantes, la méthode a été peu utilisée pour résoudre les problèmes MOCO, en raison de sa difficulté dans la résolution et au fait que la solution optimale du problème donné par l'équation (3.13) est faiblement efficace.

Les fonctions de sommes pondérée de Tchebychev augmentée

Les fonctions de sommes pondérée de Tchebychev augmentée est une extension des fonctions de sommes pondérée de Tchebychev classique.

Si on note par $F(\lambda, x)$ la distance de tchebychev pondérée au point idéal Z^I, tel que $F(\lambda, x) = \|Z(x) - Z^I\|_\lambda$, La fonction scalarisante de Tchebychev augmentée est définie comme suit :

$$\max_{x \in X} F(\lambda, x) + \rho \sum_{k=1}^{p} Z_k(x) \tag{3.14}$$

avec ρ une petite valeur positive. L'avantage de la fonction de Tchebychev augmentée est que contrairement à la fonction de Tchebychev classique, la solution optimale du problème (3.14) est une solution efficace [26].

Dichotomie

La dichotomie propose une procédure simple, facile pour le calcul des solutions supportées, initialement proposée par Aneja et Nair (1979) [3] dédié à la résolution exacte du problème de transport biobjectif (problème à minimiser). Cette procédure a ignoré l'ensemble des solutions non-supportées.
Soit X_{SE} l'ensemble de solutions efficaces supportées. La méthode est initialisée par la détermination de deux solutions X^1 et X^2 lexicographiquement optimales par rapport aux objectifs $lexmax_{x\in X}\big(Z^1(x), Z^2(x)\big)$ et $lexmax_{x\in X}\big(Z^2(x), Z^1(x)\big)$ respectivement. Durant la recherche dichotomie, des solutions de l'ensemble X_{SE} sont triées par ordre croissant des valeurs Z_1, une itération de cet algorithme consiste à résoudre un problème P_λ défini par $\lambda_1 = Z_2^r - Z_2^s$ et $\lambda_2 = Z_1^s - Z_1^r$ où Z^r et Z^s sont deux points consécutifs avec $Z_1^r < Z_1^s$ et donc $Z_2^r > Z_2^s$. Le vecteur λ correspond à la normale de droite joignant les points Z^r et Z^s. Cet algorithme est initialisé par $X^r = X^1$ et $X^s = X^2$. Toutes les solutions obtenues du problème paramétrique P_λ sont énumérées, ces dernières peuvent être extrêmes ou non-extrêmes. Dans une situation particulière où le problème paramétrique admet une unique solution, on a $X^{t1} = X^{t2}$.

Soit $R = \{X^t, \ \ t \in T\}$ l'ensemble des solutions optimales du problème paramétrique P_λ, où T est l'ensemble d'indices tel que $|T|$ est le nombre de solutions optimales. Deux cas sont possibles :

a. Si $\{X^r, X^s\} \cap R = \emptyset$, donc toutes les solutions X^t sont des nouvelles solutions supportées et elles sont rajoutées à X_{SE}. On calcule ensuite, la valeur minimale et maximale de $Z_1(x)$ dans $\{X^t, \ \ t \in T\}$. Soit X^{t1} et X^{t2} les solutions où le minimum et le maximum sont atteints. Deux nouveaux problèmes paramétriques sont considérées, l'un défini par les solutions X^r et X^{t1} et l'autre est défini par les deux solutions X^{t2} et X^s. Il n'est pas nécessaire de considérer un autre problème paramétrique par deux solutions de l'ensemble R car le paramètre λ obtenu est le même que celui défini par les solutions X^r et X^s et par conséquent aucune nouvelle solution ne peut être obtenue.

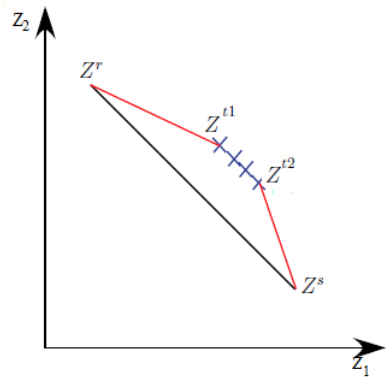


FIGURE 3.1 – Cas a. Ensemble de solutions efficaces extrêmes Y_{SE1}

b. Si $\{X^r, X^s\} \cap R \neq \emptyset$, alors toutes les solutions X^t différentes de X^r et X^s sont des solutions supportées non-extrêmes.

La méthode dichotomie n'utilise aucune spécificité du problème de transport, elle peut donc être appliquée à n'importe quel problème d'optimisation. Elle est souvent utilisée pour la détermination d'un ensemble de solutions efficace supportées d'un problème d'optimisation combinatoire biobjectif, comme l'on verra par la suite. Cette méthode

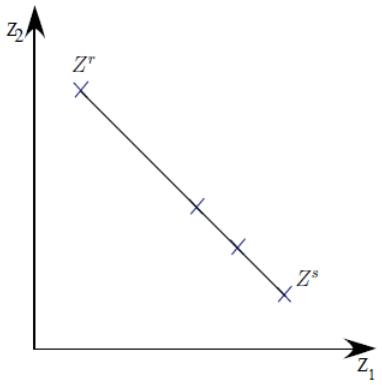


FIGURE 3.2 – Cas b. Ensemble de solutions efficaces non extrêmes Y_{SE2}

est restée limitée au cas biobjectif jusqu'aux travaux de thèse de Przybylski [83], ce dernier a proposé une généralisation de la méthode à trois objectifs et plus, en proposant une autre façon de calculer les paramètres.

La méthode ϵ-contrainte

Il existe une autre méthode différente de la précédente, il s'agit de la méthode ϵ-contrainte. C'est une méthode connue pour une meilleure technique, résout les problèmes d'optimisation multiobjectifs, il n'y a aucune aggregation des critères, un seul objectif Z_i est optimisé (maximisé ou minimisé) tandis que les autres objectifs sont transformés à des contraintes. Le problème d'optimisation s'écrit alors :

$$P(\epsilon)\begin{cases}\max\limits_{x\in X} Z_i(x) \\ Z_k(x) \geq \epsilon_k, \quad k = \overline{1,p}, \quad k \neq i \quad \text{et} \quad \epsilon \in \mathbb{R}^p.\end{cases} \tag{3.15}$$

La méthode ϵ-contrainte a été introduite pour la première fois par Haimes et al [50], elle permet de générer toutes les solutions efficaces (à noter que la solution du problème donné par (3.15) peut être faiblement efficace, alors dans ce cas l'optimisation lexicographique peut être utilisée). Contrairement à la méthode des sommes pondérées, aucune hypothèse de convexité n'est nécessaire.

Remarque 3.1 On peut remarqué que cette méthode peut être interprétée comme une itération de la méthode lexicographique décrite par l'algorithme 4.

L'objectif Z_i représente l'objectif important ou l'objectif préféré. L'ensemble Pareto optimal est généré en faisant varier les valeurs de ϵ_k. Il est ainsi possible de générer l'ensemble Pareto optimal en résolvant un seul ensemble de problèmes d'optimisation uniobjectifs. Il est à noter que cette méthode capable de générer les solutions non-supportées contrairement à la méthode d'agrégation. Cependant, l'ajout des contraintes modifie le polyèdre, ce qui peut avoir un effet négatif sur le calcul d'une borne inférieure (par exemple).

Proposition 3.1 Soit $\hat{x}$ est une solution optimale du problème donnée par l'équation (3.15), pour quelques i , alors $\hat{x}$ est faiblement efficace.

Proposition 3.2 Soit $\hat{x}$ est la solution optimale unique du problème donnée par l'équation (3.15), pour quelques i , alors $\hat{x}$ est une solution efficace supportée et donc efficace.

Théorème 3.2 *Une solution réalisable $\hat{x} \in X$ est dite efficace si et seulement si $\exists \hat{\epsilon} \in \mathbb{R}^p$ telle que $\hat{x}$ est optimale du problème donné par l'équation (3.15) pour tout $i = 1, \ldots, p$.*

Preuve : La démonstration des propositions précédentes (3.1),(3.2) ainsi que le théorème (3.2) se trouve dans [26].

Remarque 3.2 Avec des choix appropriés pour le vecteur ϵ, toutes les solutions efficaces peuvent être obtenues [50]. C'est à dire l'ensemble de toutes les solutions efficaces trouvées depend du choix appropriés de la valeur de ϵ.

Parmi les avantages d'une méthode exacte du type ϵ-contrainte est qu'elle est facile à mettre en oeuvre et possède peu de problèmes de stockage que d'autres méthodes, puisque chaque solution efficace est obtenue en résolvant un nouveau programme linéaire en variables binaires. Afin d'éliminer les solutions faiblement efficaces, l'algorithme 6 cité dans la référence [5], représente une alternative de l'algorithme classique de la méthode ϵ-contrainte.

Algorithme 6: $\epsilon-$contrainte

↑ L'ensemble de solutions efficaces X_E.
Générer une solution optimale X^1_{opt} de $\max_{x \in X} Z_1(x)$.
Générer une solution optimale X^2_{opt} de $\max_{x \in X} Z_2(x)$.
Générer une solution optimale X^1 de $\max_{x \in X}\{Z_2(x),\ \ Z_1(x) \geq Z_1(X^1_{opt})\}$.
$X_E := \{X^1\}$, $j := 1$
while $Z_2(X^j) < Z_2(X^2_{opt})$ **do**
    Générer une solution optimale X^{j+1} de
    $\max\{\big(Z_2(X^2_{opt}) - Z_2(X^j)\big)Z_1(x) + Z_2(x) \ \mid \ x \in X\ Z_2(x) \geq Z_2(X^j) + 1\}$
    $X_E := X_E \cup \{X^{j+1}\}$, j :=j+1 ;
end
return X_E.

La méthode mixte

La méthode mixte "Hybrid method", est une combinaison entre la méthode des sommes pondérées et la méthode ϵ-contrainte. Le problème à résoudre est de la forme :

$$P(mixte) \begin{cases} \max\limits_{x \in X} \sum\limits_{k=1}^{p} \lambda_k Z_k(x) \\ Z_k(x) \geq Z_k(x^0), \quad k = 1, \ldots, p \\ x \in X \end{cases} \tag{3.16}$$

– $\lambda \in \mathbb{R}^p_{\succ}$.
– x^0 est une solution réalisable du problème donné par l'équation (3.1).

Il existe une relation entre la méthode des sommes pondérées et la méthode ϵ-contrainte, le théorème (3.3) montre cette relation.

Théorème 3.3 *Chankong and Haimes (1983)*

- *Supposons que $\hat{x}$ est une solution optimale du problème paramétrique $\max_{x\in X} \sum_{k=1}^{p} \lambda_k Z_k(x)$.*
  - *Si $\lambda_k > 0$ alors, il existe $\hat{\epsilon}$ tel que $\hat{x}$ est une solution optimale du problème donné par l'équation (3.15).*
- *Supposons que X est un ensemble convexe et Z_k sont des fonctions convexes (voir définition (1.1)).*
  - *Si $\hat{x}$ est une solution optimale du problème cité dans l'équation (3.15) pour quelques k, il existe $\hat{\lambda} \in \mathbb{R}^p_>$ tel que $\hat{x}$ est optimale pour $\max_{x\in X} \sum_{k=1}^{p} \hat{\lambda}_k Z_k(x)$.*

D'autres méthodes scalarisantes sont largement expliquées dans [26, 28, 29].

La méthode en deux phases

La méthode en deux phases a été développée par Ulungu et Teghem en 1995 [96] pour résoudre les problèmes MOCO. Dans sa version original, la méthode résout le problème d'affectation biobjectif (BAP). L'idée générale est de construire l'ensemble des solutions efficaces X_E en séparant la recherche des solutions supportées X_{SE} et la recherche des solutions non supportées X_{NSE}, et d'utiliser 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 afin de rechercher les solutions efficaces (comme par exemple l'ajout de contraintes pour les problèmes MOILP). Dans cette section, on présentera cette méthode en détail. Comme son nom l'indique cette méthode se compose en deux phases :

- **Phase 1 : Détermination des solutions efficace supportées X_{SE}**
  On présente la première phase dans la méthode en deux phases qui reste inchangeable dans tous les algorithmes en deux phases. Elle consiste à la construction de l'ensemble des solutions efficaces supportées à l'aide de la méthode dichotomie proposée par Aneja & Nair [3], basée sur le théorème de Géoffrion [41], qui génère toutes les solutions efficace supportées X_{SE} (extrêmes et non-extrêmes) à l'aide d'un problème uni-critère dont la fonction objectif est une agrégation linéaire de deux objectifs. Cette phase est identique à celle de Aneja & Nair [3] pour le problème de transport, rappelons cependant que ces auteurs se contentaient de cette seule phase, en ignorant l'ensemble X_{NSE}.
- **Phase 2 : Détermination des solutions efficace non-supportées X_{NSE}**
  Cette phase est la partie la plus originale du travail élaboré par Ulungu et Teghem (1995). L'objectif est de rechercher, pour chaque paire (X^r, X^s) de l'ensemble de solutions de X_{SE}, s'il existe des solutions efficaces non supportées X^u) vérifiant :

$$\begin{pmatrix} Z_1^{(r)} & < & Z_1^{(u)} & < & Z_1^{(s)} \\ & et & & & \\ Z_2^{(r)} & > & Z_2^{(u)} & > & Z_2^{(s)} \end{pmatrix}$$

  des solutions pareilles seraient situées à l'intérieur du triangle $\Delta[YZ^{(s)}Z^{(r)}]$.
  La deuxième phase génère toutes les solutions efficaces non-supportées X_{NSE}. Cette deuxième phase est généralement enumerative. La première technique est d'utiliser les bornes inférieures et supérieures, et de fixer quelques variables, dans

le but de réduire cette énumération dans chaque triangle. Cependant cette seconde phase prendra une forme différente de la première puisque une méthode Branch & Bound sera appliquée pour tout couple de solutions efficaces supportées adjacentes. cette deuxième phases est bien détaillé dans les références [97, 96, 98]. La seconde technique, développée par Przybylski et al.[82] est d'utiliser un algorithme ranking pour résoudre les problèmes paramétriques. Ce dernier travail est dédié au problème d'affectation, Przybylski et al.[82] ont montré que cette approche est plus efficace et simple à appliquer que la première technique.

La méthode en deux phases est à l'origine adaptée pour résoudre les problèmes biobjectifs. Elle a été appliquée sur un grand nombre de problèmes d'optimisation combinatoires biobjectifs (BOCO) tel que le problème d'affectation [96], et le problème de sac à dos [98] avec une approche de type séparation et évaluation en deuxième phase. En 2008, la méthode a été appliquée au problème d'affectation avec une approche de type ranking [82], au problème de flot de coût minimum en variables entières, et d'arbre couvrant de poids minimum, ainsi qu'aux problèmes d'ordonnancement biobjectifs. Pour plus de détail le lecteur pourra se référé à [28, 29, 82, 67]. L'inconvénient de cette méthode est que de nombreux problèmes doivent être résolus, même si seulement un ensemble complet minimal est recherché. En outre, des algorithmes de ranking, qui sont efficaces pour la deuxième phase ne sont pas populaires dans les problèmes d'optimisation combinatoire. D'autre part, la méthode respecte la structure du problème considéré. En effet, l'adaptation de la première phase aux problèmes d'optimisation multiobjectifs avec $p = 3$ n'est pas facile, comme l'a souligné Przybyslki et al.[81]. Malgré ces difficultés, Przybyslki et al.[81] et Ozpeynirci [75] ont généralisé la méthode. L'étude de la performance de ces dernières méthodes n'a pas été faite.

La méthode en deux phases de Carlo Filippi et Elisa Stevanato [33]

Dans ce travail, une variante de la méthode en deux phases pour les problèmes d'optimisation combinatoire biobjectifs a été exploitée. Tout d'abord, les auteurs analysent la procédure énumérative, souvent utilisée dans la littérature pour résoudre des problèmes d'optimisation combinatoires biobjectifs, et convient pour résoudre les problèmes généraux. Ils ont montré que la procédure génère l'ensemble exacte de solutions efficaces X_E en résolvant exactement $2|X_E| - 1$ problèmes uniobjectifs. Deuxièmement, ils intègrent leur procédure dans la méthode en deux phases, où les solutions efficaces supportées sont calculées dans la première phase et les solutions non-supportées sont calculées dans la deuxième phase. Ensuite, ils testent leur méthode sur le problème de voyageur de commerce biobjectif. La procédure développée surpasse la méthode dite ϵ-contrainte, utilisée pour résoudre les problèmes d'optimisation combinatoire biobjectifs.

La méthode de Gokhan Kirlik et Serpil Sayin [62]

Dans cette étude, un nouvel algorithme est proposé pour générer toutes les solutions efficaces pour les problèmes d'optimisation combinatoire multiobjectifs en nombre discrets. Dans cet algorithme, la recherche est faite sur plus de $p-1$ rectangles où p représente le nombre d'objectifs dans le problème et pour chaque rectangle, deux étapes sont utilisées pour résoudre le problème. L'algorithme est basé sur la fonction scalarisante ϵ-contrainte, ainsi il est comparé à des études sur le problème de sac à dos multiobjectif

et le problème d'affectation multiobjectif. La méthode est très compétitive en termes de temps de résolution et le nombre de problèmes d'optimisation résolus.

Un très grand nombre de méthodes de résolution existent pour le cas biobjectif [96, 98, 33, 56], beaucoup moins de littérature disponible pour le cas où le nombre d'objectifs augmente [83, 62]. Bien que les méthodes exactes permettent de trouver des solutions optimales pour des problèmes de taille raisonnable, le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante, au temps de calcul qui se positionne en premier lieu, ainsi que le nombre croissant d'objectifs qui contribue largement à le faire exploser. Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si l'optimalité n'est pas essentielle. En effet, ces méthodes sont utilisées depuis longtemps par de nombreux chercheurs.

3.3.2 Les méthodes approximatives

Résoudre des problèmes d'optimisation combinatoire multiobjectifs NP-difficiles est très ardu, en effet pour de tels problèmes, les méthodes exactes exigent un effort calculatoire qui croit exponentiellement avec la taille des instances du problème, et rapidement, les méthodes méta-heuristiques deviennent l'unique moyen d'obtenir une bonne solution en un temps raisonnable. Ces méthodes méta-heuristiques sont principalement basées sur des méthodes de recherche locale (méthodes de descente, tabou, recuit simulé) ou des méthodes évolutives (algorithmes génétiques, colonies de fourmis, Scatter Search). Une façon intuitive de mesurer la distance entre une solution approchée et une solution optimale est la garantie relative de performance qui borne le rapport maximum entre l'approximation et la valeur de la solution optimale. Cette expression de la distance par un pourcentage de la valeur de la solution optimale a l'avantage d'être indépendante de l'ordre de grandeur des coefficients du problème. Nous proposons d'exposer quelques méthodes méta-heuristiques pour la résolution des problèmes d'optimisation combinatoires multiobjectifs NP-difficiles.

Définition 3.1 Garantie relative de performance (monoobjectif)
Soit $Z^*(I)$ la valeur optimale des solutions de l'instance I et $Z^A(I)$ la valeur obtenue par l'algorithme d'approximation A.

L'algorithme A est un algorithme d'approximation avec une garantie relative de performance égale à k, $0 < k < 1$ si l'inégalité $\frac{Z^A(I)}{Z^*(I)} \geq k$ se vérifie pour toute instance I.

Un algorithme à garantie relative de performance égale à k sera appelé une k-approximation. Pour mettre en évidence la différence entre l'approximation et la valeur de la solution optimale, on définit $\epsilon = 1 - k$ et on parlera de $1 - \epsilon$ approximation vérifiant ; $\frac{Z^A(I)}{Z^*(I)} \geq 1 - \epsilon \iff \frac{Z^*(I) - Z^A(I)}{Z^*(I)} \leq \epsilon$; pour toute instance I. Cette définition est valable dans le cas multiobjectif :

Définition 3.2 Garantie relative de performance (multiobjectif)
Soit une instance I d'un problème multiobjectif ; soit X^A les solutions réalisables non-dominées obtenues par l'algorithme A.

- $x^A \in X^A$ est une $1-\epsilon$ approximation de $x^* \in X_E$ si $Z_k(x^A) \geq (1-\epsilon)Z_k(x^*) \quad \forall k \in \{1, 2, \ldots, p\}$.
- X^A est une $(1-\epsilon)$ approximation de la frontière efficace de I si, $\forall x^* \in X_E$, $\exists x^A \in X^A$ tel que x^A soit une $(1-\epsilon)$ approximation de x^*.

Il est logique que le temps d'exécution d'un algorithme d'$1-\epsilon$ approximation augmente lorsque ϵ diminue. Plus la solution demandée doit être de bonne qualité, plus il faut du temps pour la trouver. Ainsi, il est intéressant de pouvoir limiter cette durée d'exécution dans une certaine mesure.

Définition 3.3 Algorithme d'approximation en temps polynomial
Un algorithme d'$(1-\epsilon)$ approximation est un algorithme d'(ϵ) approximation en temps polynomial (PTAS : Polynomial Time Approximation Scheme) si sa complexité en temps d'exécution est polynomial en n.

Définition 3.4 algorithme d'approximation en temps totalement polynomial
Un algorithme d'$(1-\epsilon)$ approximation est un algorithme d'ϵ-approximation en temps totalement polynomial (FPTAS, Fully PTAS) si sa complexité en temps d'exécution est polynomial en n et polynomial en $\frac{1}{\epsilon}$.

La frontière efficace d'une instance d'un problème MOCO peut contenir un nombre arbitrairement grand de solutions. Par conséquent, un PTAS pour un MOCO a l'avantage de donner une approximation ayant une qualité garantie, mais aussi de présenter un ensemble de solutions relativement petit au décideur. Des méta-heuristiques multi-objectifs sont souvent utilisées pour approximer un ensemble complet. Leur efficacité pour la résolution en pratique de problèmes très difficiles est à l'origine du regain d'intérêt pour les MOCO dans les années 1990 ([56, 67]).

La figure 3.3 représente un exemple d'une frontière de Pareto et plusieurs approximation de cette frontière.

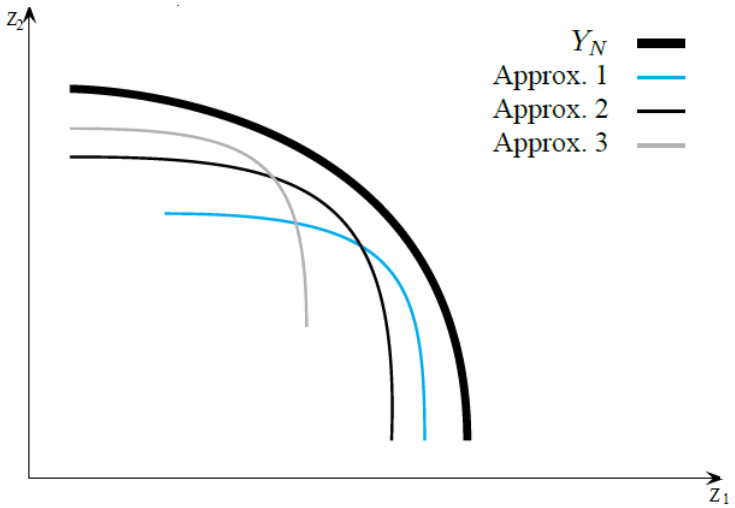


FIGURE 3.3 – Approximation d'une frontière efficace et ses approximations.

La figure 3.3 illustre le problème en présentant plusieurs approximations d'une même frontière. Aucune de ces approximations n'est meilleure qu'une autre, cet exemple montre que la notion de bonne qualité n'est pas évidente en optimisation multiobjectifs.

Une literature existante concernant des algorithmes approximatifs pour les problèmes multiobjectifs, en 1979, [51] développe FPTAS pour le problème du plus court chemin multicritères, ce problème a été aussi étudié par Warburton en 1987 [99], En 1990, Ruhe et Fruhwirth [88] présentent un algorithme ϵ-approximation pour le problème flot bicritères. Dans [89], Safer et Orlin étudient les conditions nécessaires et suffisantes pour qu'il existe un FPTAS pour un problème MOCO. Ils présentent dans [89] un FPTAS pour divers problèmes de flot, de sac à dos et d'ordonnancement. En particulier, ils prouvent l'existence d'un FPTAS pour le problème de sac à dos multiobjectif. En 2009, Bazgan et al. ont proposé un FPTAS pour ce problème [7, 6]. Glasser et al. ont aussi proposé un PTAS pour le problème de couverture de disque multiobjectif [18]. De même, Tsaggouris et al. ont présenté un FPTAS pour le problème de plus court chemin multiobjectif [94]. Il existe quelques travaux concernant les problèmes MOMKP et les problèmes MOTSP [67, Chapitre 4,5].

3.3.3 Les méthodes méta-heuristiques

les résultats disponibles des méthodes de résolution exacte pour le cas multiobjectif se cantonnent à quelques centaines de variables et prennent assez de temps. Il est alors naturel de s'orienter vers des méthodes approchées pour résoudre les plus grandes ou les plus difficiles des instances de problèmes MOCO. D'où la naissance des méta-heuristiques, ces dernières sont des concepts généraux pour produire des heuristiques, dont leur but est de résoudre une grande classe de problèmes d'optimization. Elles sont souvent utilisées pour approximer un ensemble complet de solutions efficaces. Ehrgott et Gandibleux [27] proposent un état de l'art des méta-heuristiques pour les problèmes MOCO, la plupart des méta-heuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à l'explosion des problèmes combinatoire. Une méta-heuristique peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné. Plusieurs d'entre elles sont souvent inspirées par des systèmes naturels dans de nombreux domaines tels que : la biologie (algorithmes évolutionnaires et génétiques) la physique (recuit simulé), et aussi l'éthologie (algorithmes de colonies de fourmis). On distingue deux catégories : les algorithmes de recherche locale et les algorithmes évolutionnaires (à population) [29, 27], la troisième catégorie est un hybride entre les deux dernières catégories [27].

- **Méthodes de recherche locale** : Dans les algorithmes d'exploration de voisinage, à partir d'une solution initial la procédure à travers un mécanisme local d'agrégation des objectifs, fournit une partie de la frontière Pareto correspondante à une direction donné λ. Ce principe est répété pour plusieurs directions de recherche pour approximer l'ensemble de solutions efficaces (l'ensemble de points non-dominés dans l'espace des objectifs). L'efficacité de ces algorithmes dépend fortement de la direction λ.
- **Algorithme évolutionnaire** : Contrairement à la première catégorie, où seulement une seule solution est considérée, pour cette catégorie toute la population contribue à la détermination de plusieurs solutions potentiellement efficaces parallèlement.
- **Des approches hybrides** : Des méthodes hybridant les algorithmes de la première et la deuxième catégorie ont donné naissance aux méthodes MOGLS (Multiple Objective Genetic Local Search), MOGTS (Multiple Objective Genetic Tabu

Search) et d'autres [27].
Ces méthodes sortent du cadre de nos travaux, nous nous limiterons juste à mentionner par la suite quelques contributions existantes. Le lecteur souhaitant approfondir sur ces méthodes pourra consulter à [67, 27, 29].
De différentes méthodes méta-heuristiques ont été développées, parmi ces méthodes la plus populaire est recuit simulé (simulated annealing (SA))[61, 97, 95, 38], la recherche tabou (tabu search (TS))[42, 39], VEGA (Vector Evaluated Genetic Algorithm [24]. MOGA (Multiple Objective Genetic Algorithm, des méthodes évolutionnaires comme algorithmes génétiques, colonies de fourmis, la recherche de voisinage, les algorithmes GRASP, et Scatter Search [67, 27, 29]. Contrairement aux méthodes exactes, les méthodes méta-heuristiques ne garantissent 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. Le principal inconvénient de ces procédures est la difficulté d'obtenir une bonne approximation de la frontière efficace, comme le mentionne la figure 3.3. Pour mesurer la qualité des approximations, il existe un ordre sur l'ensemble de solutions potentiellement efficace, pour être en mesure de dire qu'une approximation est meilleure qu'une autre. Une comparaison entre deux approximations en utilisant la notion de dominance au sens de Pareto est récapitulé dans un tableau, voir [67, p.44, chapitre 3].

3.4 Conclusion

Dans ce chapitre, nous avons présenté quelques types de problèmes d'optimisation combinatoire multiobjectifs (MOCO) existants suivis de différentes méthodes de résolution dédiées à ces problèmes, à savoir les méthodes scalarisantes, la méthode ϵ-contrainte et la méthode populaire en deux phases. Les méthodes approximatives et méta-heuristiques ont été brièvement présentées. Nous consacrons le chapitre suivant à un problème dont sa structure est la plus simple mais en résolution est le plus complexe des problèmes d'optimisation combinatoire (MOCO), il s'agit du problème de sac à dos multiobjectif.

Chapitre 4

Problème de sac à dos multiobjectif (MOKP)

4.1 Introduction

Bien que le problème de sac à dos uniobjectif largement étudié (Martello & Toth [70], Kellerer [60]) possède plusieurs situations pratiques, le cas unicritère prouvé NP-complet, ne puisse pas tenir compte des nombreux aspects pertinents (plusieurs critères) de la réalité, ainsi le cas multicritère semble être plus réaliste.
Le problème de sac à dos multiobjectif est une extension naturelle du problème de sac à dos monoobjectif. La taille de l'ensemble efficace peut croître de façon exponentielle avec le nombre d'objets dans le sac à dos. Ceci est prévu car les algorithmes proposés sont basés sur des méthodes d'énumération implicites telles que la programmation dynamique, les procédures de séparation et évaluation ou des heuristiques, en particulier les méta-heuristiques rapprochant un ensemble efficace [29]. Plusieurs méthodes de résolution disponibles dans la littérature, on présentera dans la section suivante celles souvent utilisées pour la résolution du problème de sac à dos multicritère.

4.2 Les méthodes de résolution

On distingue deux grandes classes pour résoudre le problème de sac à dos multicritère : les méthodes exactes et les méthodes approximatives :

4.2.1 Les méthodes exactes

Les méthodes exactes sont connues pour résoudre de façon efficace le problème de sac à dos multiobjectif (MOKP). On présente une partie de ces méthodes avec quelques précisions à propos du nombre d'objets et la taille du problème considéré.

- La méthode en deux phases a été adaptée au problème de sac à dos multiobjectif par Visé et al. [98], en première phase, les auteurs utilisent la méthode dichotomie pour chercher toutes les solutions supportées, tandis que la deuxième phase, la méthode Branch & Bound est utilisée dans le but de générer toutes les solutions non supportées. Ils testent leurs méthodes sur des instances du problème biobjectif générées aléatoirement. La capacité du sac à dos ω est égal à la moitié de la totalité des poids. La taille limite des instances est à 500 objets.

- En 2000, Klamroth et Wiecek [63] présentent plusieurs formulations de programmation dynamique, malheureusement la méthode ne dépasse pas la partie théorique (pas d' expériences numériques).
- Il existe une analogie entre le problème de sac à dos et le problème du plus court chemin, tel que le problème KP peut être résolu de manière similaire à la recherche du plus court chemin. Cette similarité reste valide dans le cas multiobjectif [56]. Captivo et al. [12] ont proposé en 2003, une méthode exacte basée sur une transformation du problème BOKP au problème du plus court chemin biobjectif. Les auteurs ont évalué leur procédure sur trois types d'instances du problème de sac à dos biobjectif : aléatoire, faiblement corrélées et fortement corrélées. Ils ont résolu la première catégorie à 320 objets, la deuxième catégorie à 800 objets et la dernière à 900 objets. Ils ont montré que le temps d'exécution de la méthode est meilleur que celui de la méthode de Visée et al. L'algorithme implémenté retourne néanmoins à un ensemble complet minimum de solutions efficaces :

Algorithme 7: Plus court chemin pour $BOKP$

Input :
$\downarrow G$: le graphe construit
Output :
$\uparrow$ Les performances de solutions efficaces

$-- |S_i(y)$ est l'ensemble des performances obtenu au sommet $Z_i(y)$
$S_i(y) := (0, 0)$
for $i \in \{1, 2, \ldots, n\}$ **do**
  for $y = 0 : \omega$ **do**
    if $y \geq w_i$ **then**
      $S_i(y) := S_{i-1}(y) \cup \{d + c_i \ : \ d \in S_{i-1}(y - w_i)\}$
    else
      $S_i(y) := S_{i-1}(y)$
    end
  end
end
return $\bigcup_{y=0}^{\omega} S_n(y)$

- **Méthode exacte (2005) [54]**
On présente la méthode exacte développée par Jahanshahloo et al.[54], cette méthode est proposée pour détecter toutes les solutions efficaces d'un problème de programmation linéaire multiobjectif en $\{0, 1\}$, la méthode peut être aussi appliquée pour le problème de sac à dos multiobjectif en variables binaires.
On considère le problème de sac à dos biobjectif suivant :

$$(\text{BOKP}) \begin{cases} \text{“max”}(Z^k) = \sum_{i=1}^{n} c_i^k x_i, & k = 1, 2. \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ x_i \in \{0, 1\}, \quad i = 1, \ldots, n. \end{cases} \tag{4.1}$$

Pour résoudre le problème (BOKP) donné par l'equation (4.1), on considère les deux problèmes de sac à dos uniobjectifs $KP_k,\ (k = 1, 2)$:

$$(KP_k)\begin{cases} \max Z^k(x) \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ x_i \in \{0,1\}, \quad i = 1, \ldots, n. \end{cases} \tag{4.2}$$

Soit $G_0 = \{x^*_{i_1}, x^*_{i_2}, \ldots, x^*_{i_\alpha}\}$ l'ensemble de solutions optimales du problème KP_k et $L_0 = \{i_1, i_2, \ldots, i_\alpha\}$.

– Si l'ensemble G_0 est vide, alors on résout le problème défini par :

$$(\text{SKP})\begin{cases} \max \sum_{k=1}^{2} Z^k(x) \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ x_i \in \{0,1\}, \quad i = 1, \ldots, n \end{cases} \tag{4.3}$$

Supposons que $G_0 = \{x^*_{i_1}, x^*_{i_2}, \ldots, x^*_{i_\beta}\}$ est l'ensemble de solutions optimales du problème (SKP) donné par l'equation (4.3) et $L_0 = \{i_1, i_2, \ldots, i_\beta\}$. Notons que $x^*_q \in G_0$ tel que $q \in L_0$ est une solution efficace du problème (BOKP) donné par l'equation (4.1).

– Si l'ensemble G_0 est non vide, on détermine toutes les solutions optimales du problème défini par :

$$(SKP_1)\begin{cases} \max \sum_{k=1}^{2} Z^k(x) \\ Z^k(x) > Z^k(x^*_q) - Mt_{kq}, \quad k = 1, 2,\ q \in L_0 \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ t_{1q} + t_{2q} \leq 1, \qquad t_{1q}, t_{2q} \in \{0,1\}, \\ x_i \in \{0,1\}, \quad i = 1, \ldots, n. \end{cases} \tag{4.4}$$

Dans le cas $t_{kq} = 1$ $(k = 1, 2)$, la contrainte $Z^k(x) > Z^k(x^*_q) - Mt_{kq}$ est redondante, sinon, la contrainte est non redondante. La contrainte $t_{1q} + t_{2q} \leq 1$ $(q \in L_0)$: implique qu'au moins une des contraintes $Z^k(x) > Z^k(x^*_q) - Mt_{kq}, \quad k = 1, 2$ est non redondante (on peut choisir $\max\limits_{1\leq k\leq 2}\left\{\sum\limits_{i=1}^{n} c_i^k\right\}$ comme une borne inférieure de la valeur de M).

Maintenant, on suppose que $A = \{x^*_{i_{j+1}}, x^*_{i_{j+2}}, \ldots, x^*_{i_{j+l}}\}$ est l'ensemble des solutions optimales du problème (SKP1) donné par l'équation (4.4), où $j = \alpha$ ou β.

- Si A est vide, G_0 est l'ensemble des solutions efficaces du problème (BOKP).
- Si A est non vide, l'ensemble des solutions efficaces est $G_1 = G_0 \cup A$.

Dans le but de déterminer toutes les autres solutions efficaces du problème (BOKP) donné par l'equation (4.1), on procède de la façon suivante :
pour chaque $x^*_q \in A$, on ajoute les trois contraintes suivantes au problème (SKP_1) :

$$Z^k(x) > Z^k(x^*_q) - Mt_{kq}, \quad k = 1, 2.$$

$$t_{1q} + t_{2q} \leq 1.$$

et deux $(0,1)$ variables au problème (SKP_1), t_{1q} et t_{2q}. Par conséquent, le problème (SKP_1) peut s'écrire comme suit :

$$(SKP_2) \begin{cases} \max \sum_{k=1}^{2} Z^k(x) \\ Z^k(x) > Z^k(x_q^*) - Mt_{kq}, \quad k = 1,2 \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ t_{1q} + t_{2q} \leq 1 \;\; q = i_1, \ldots, i_{j+1}, i_{j+2}, \ldots, i_{j+l} \;\; (j = \alpha \text{ ou } \beta). \\ x_i \in \{0,1\}, \quad i = 1, \ldots, n. \end{cases} \tag{4.5}$$

Ce processus continue jusqu'à ce que le problème (SKP_2) donné par l'équation (4.5) devient irréalisable. Si les problèmes (SKP), (SKP_1), (SKP_2) admettent des solutions optimales alternatives, on doit déterminer toutes ces solutions. Cependant cet action décroît le nombre d'itérations de l'algorithme et garantie l'obtention de toutes les solutions efficaces du problème (BOKP). On termine cette méthode par donner quelques théorèmes utilisés :

Théorème 4.1 *Si $G_l = \{x_{1_l}^*, x_{2_l}^*, \ldots, x_{f_l}^*\}$ l'ensemble de toutes les solutions optimales du l^{th} problème (KP_l) donné par l'équation (4.2), alors au moins une de ces solutions est efficace pour le problème $(BOKP)$.*

Théorème 4.2 *Chaque solution optimale du problème (SKP) est une solution efficace pour le problème $(BOKP)$.*

Remarque 4.1 Pour trouver l'ensemble de solutions efficaces, un effort calculatoire est exigé, pour chaque itération de l'algorithme proposé par Jahanshahloo et al.[54] , l'ajout de trois contraintes :

$$Z^k(x) > Z^k(x_q^*) - Mt_{kq}, \quad k = 1,2,$$

$$t_{1q} + t_{2q} \leq 1$$

et deux variables binaires t_{1q} et t_{2q} est exigé pour réduire le domaine, à la fin, au moins une solution efficace est trouvée. Comme le nombre des solutions possibles est fini, l'algorithme converge.

- En 2006, Figuera et al. [32] ont étendu les résultats de Captivo et al. Les auteurs ont présenté des algorithmes pour la construction de quatre modèles de réseau tous représentants les problèmes MOKP et la transformation du MOKP au problème du plus court chemin multiobjectif. Ils ont testé leurs travaux sur différents types, mais les résultats ne sont pas présentés.
- **Algorithme exacte (2007) [55]**
  Les auteurs ont proposé une nouvelle procédure basée sur la formulation de la programmation linéaire pour trouver toutes les solutions efficaces du problème de sac à dos biobjectif en variables $\{0,1\}$ en résolvant un nombre limité de problèmes uniobjectifs. Ils montrent que la méthode proposée est capable de générer l'ensemble efficace avec moins de contraintes et de variables que la méthode développée par Jahanshahloo et al. [54]. Un exemple est présenté illustre toutes les étapes de la procédure. Dans cette section, on présente cette procédure en détail.

La méthode bénéficie considérablement au développement de la méthode présentée auparavant proposée par Jahanshahloo et al. [54] dédiée au problème linéaire multiobjectif en variables $\{0,1\}$. La procédure développée peut être appliquée au problème de sac à dos $\{0,1\}$ multiobjectif de façon particulière.
L'avantage de la procédure développée par Jolai et al.[55] comparant avec la procédure générale de Jahanshahloo et al. [54] pour MOLP, qui admet moins de contraintes et de variables. A chaque itération le problème contient $m+3*(k+1)$ contraintes et $n+2*(k+1)$ variables, donc le nombre de contraintes et variables augmentent par trois contraintes et deux variables, respectivement. Par contre pour cette procédure le nombre de contraintes et variables sont constants et égales à $m+5$ contraintes et $n+2$ variables (pour $k>1$). La méthode peut être valider à d'autres problèmes ayant des propriétés similaire au problème de sac à dos. La méthode n'a pas dépassée la description théorique et un exemple didactique (pas de résultats numériques).

- Bazgan et al. [5] ont présenté une méthode de résolution exacte basée sur la programmation dynamique pour le problème de sac à dos multiobjectif. Celle-ci est similaire à l'algorithme du plus court chemin. Les auteurs utilisent plusieurs relations de dominance complémentaires pour éliminer des solutions partielles durant la résolution.
  Chaque relation de dominance se concentre sur des considérations spécifiques. Il est alors souhaitable de faire usage à des relations de dominance complémentaires. En outre, au moment de décider d'utiliser une relation de dominance, un arrêt doit être fait entre la capacité potentielle de jeter de nombreux états et le temps qu'il faut pour être vérifiée.
  Nous présentons maintenant les trois relations de dominance utilisées dans la méthode. Les deux premières relations sont très faciles à mettre en place, par contre la dernière est difficile à établir, elle est considérée en raison de sa complémentarité avec les deux autres.
  a. La première de ces relations fondée sur l'observation du fait que lorsque la capacité résiduelle $\omega - y$ associée à un état $S_i(y)$ est supérieure ou égale à la somme des poids des objets restants, alors la seule façon d'obtenir une solution efficace à partir de $S_i(y)$ est d'ajouter tous ces objets.
  b. La deuxième relation est basée sur la dominance de Pareto. Soit $s \in S_i(y)$ et $s' \in S_i(y')$; si $y \leq y'$, alors tous les objets pouvant être sélectionnés à partir de $S_i(y')$ peuvent aussi l'être à partir de $S_i(y)$. Ainsi, si $s \geq s'$ alors l'état s permettra d'obtenir des solutions au moins aussi bonnes qu'à partir de s', ce dernier peut donc être rater pour la suite.
  c. La troisième relation de dominance est la plus coûteuse à évaluer. En effet, elle est basée sur la comparaison entre des extensions spécifiques d'un état et une limite supérieure des extensions d'un autre état. Pour plus de détails [5, p. 10].

  Ils ont testé leur algorithme sur différents types d'instances : aléatoire (type A), fortement corrélés (type B), faiblement non corrélés (type C) et fortement non-corrélés (type D) pour le cas biobjectif et pour les instances de trois objectifs, type A et C. En moins de deux heures de temps de calcul sur un bi-Xeon 3,4 GHz avec 3072 Mo de RAM, ils ont résolu des instances biobjectif de type A avec 700 objets, de type B avec 4000 objets, de type C avec 500 objets et sur le type D avec 250 objets. Ils ont comparé leurs résultats avec la méthode de Captiva et al.

et avec une méthode ϵ-contrainte couplé avec le solveur ILOG Cplex 9.0, ils ont obtenu de meilleurs résultats et aussi ils ont résolu les cas de grande tailles. Pour les cas de trois objectifs, en raison de l'explosion de la cardinalité de l'ensemble de solutions efficaces, ils ont pu résoudre les cas de type A qu'avec un maximum de 110 objets et les instances de type C avec un maximum de 60 objets, il est remarquable que cette méthode est la première méthode exacte qui a été adaptée aux cas de trois objectifs pour les problèmes MOKP.

- J.Jorge et X.Gandileux [58], ont présenté des améliorations sur les bornes utilisées pour l'exploration d'un triangle dans l'algorithme en deux phases pour le problème de sac à dos biobjectif $\{0, 1\}$, tel que décrit par Visée et al.[98]. Ils ont aussi proposé une nouvelle procédure en deux phases pour ce problème, utilisant un ranking pour l'exploration des triangles. Les expérimentations numériques montrent une amélioration clair de la procédure en deux phases utilisant des procédures de séparation et évaluation lorsque les bornes améliorées sont utilisées. Quand à l'approche ranking, les expérimentations indiquent qu'elle est la plus performante comparée à l'algorithme utilisant des procédures de séparation et évaluation et comparée à la programmation dynamique de Bazgan et al., sauf quelques classes d'instances particulières. Pour plus de détails [56].
- On peut reprendre la méthode de Gokhan Kirlik et Serpil Sayin [62].
- A. Rong et al. [86] ont présenté un algorithme de programmation dynamique pour générer la frontière de Pareto pour le problème de sac à dos biobjectif en nombres entiers. L'algorithme développé répond à un problème réduit construit après l'application des techniques de fixation de variables. Les résultats numériques obtenus à partir de différents types du problèmes de sac à dos biobjectifs montrent l'efficacité de l'algorithme proposé. En 2014, les mêmes auteurs ont proposé une autre méthode exacte pour le problème de sac à dos biobjectif en nombres entiers [87].

4.2.2 Les méthodes approximatives

Le passage au cadre multiobjectif rend implicitement les problèmes MOKP plus ardus, en particulier du fait de la grandeur du nombre de solutions efficaces. Des algorithmes pour le cas monoobjectif peuvent résoudre des instances ayant plusieurs centaines de milliers de variables en un temps raisonnable, les résultats disponibles pour les algorithmes de la version multiobjectif sont limités à quelques centaines de variables, les méthodes approchées sont les seules remèdes pour résoudre les plus grandes ou les plus difficiles des instances. Nous présentons dans cette section des méthodes approximatives dans le but de produire un PTAS et FPTAS. Bien que les méthodes approximatives sortent du cadre de nos travaux, nous citons quelques-unes dédiées aux problèmes MOKP : Erlebach et al.[30] ont présenté en 2002 un FPTAS efficace et applicable pour les problèmes MOKP, aussi ils ont présenté un algorithme PTAS pour les problèmes MOMKP basé sur la programmation linéaire, Dans 2006, Kumar et Banerjee [64] ont présenté un algorithme stochastique pour obtenir $(1 + \epsilon)$-approximations. Récemment, en 2009, Bazgan et al.[6] ont proposé un algorithme FPTAS, l'idée principale de leurs approche est basée sur la programmation dynamique et l'utilisation de plusieurs relations de dominance complémentaires pour écarter une partie de solutions dominées. Ils ont comparé leur algorithme FPTAS avec celui de Erlebach et al. et ils ont montré que leurs résultats sont meilleurs.

4.2.3 Les méthodes heuristiques

Nous présentons une synthèse des méthodes heuristiques qui a pour but trouver une bonne approximation de l'ensemble de solutions efficaces.

- En 1993, Ulungu [97] a présenté dans sa thèse la première adaptation de la méthode SA aux problèmes MO, il s'agit de la méthode MOSA. Il a adapté la méthode pour le problème MOKP [95]. Il a résolu les instances biobjectifs aléatoires (les mêmes instances que dans la méthode exacte par Visé et al. [98]) avec jusqu'à 500 objets.
- Ben Abdelaziz et al [8] ont proposé en 1997 une méthode basée sur la recherche tabou. Ils résolvent des instances aléatoires biobjectifs jusqu'à 100 objets. Ils ont fait une extension de la méthode réalisée en 1999, en integrant une recherche tabou dans des algorithmes génétiques.
- Czyzak et Jaskiewicz ont adapté une méthode de recuit simulé [20] pour résoudre (MOKP). Ils ont adapté le cas de 2, 3 et 4 objectifs générés aléatoirement avec 800 objets au maximum.
- Gandibleux et Fréville [39] ont proposé en 2000 une méthode de recherche tabou, appelée MOTS, appliquée au problème de sac à dos biobjectif. Ils ont utilisé de différentes techniques pour réduire l'espace de décision. Ils ont testé leur méthode sur des instances aléatoires dans le cas biobjectif 50 et 100 objets. Aucune comparaison avec d'autres méthodes heuristiques a été réalisée.
- Ils existent aussi les travaux de Gandibleux et al. établis en 2001, les travaux de C. Gomes et al. en 2006 et en 2007 et d'autres travaux [67, 27]. Le principal inconvénient de ces procédures est la difficulté d'obtenir une bonne approximation de la frontière efficace.

Bien que l'efficacité des procédures de résolution dans le cas unicritère se base sur la notion du core, dans le cas multicritère peu de travaux existants dans cet axe, il nous semble intéressant de se pencher vers cet axe. Le chapitre suivant rapporte les travaux consacrés à la fixation de variables, avant la résolution du problème de sac à dos multiobjectif dans sa version unidimensionnel et multidimensionnel.

Chapitre 5

Prétraitement pour le problème de sac à dos multiobjectif

5.1 Introduction

Les procédures de réduction essayent à travers différentes techniques de diminuer de façon efficace le temps d'exécution des méthodes de résolution exactes ou approchées pour toutes les instances pratiques du problème de sac à dos, bien que cela ne change pas la complexité des algorithmes. Ces procédures sont disponibles pour sa version monoobjectif unidimensionnel et multidimensionnel [4, 84] (voir : Chapitre 1).

Nous exposerons dans la première partie de ce chapitre les travaux de Gomes da Silva et al. [44], les auteurs proposent une extension de la notion du core du cas monoobjectif au cas biobjectif. Les travaux de G.Mavrotas et al. [73] concernant le cas multiobjectif multidimensionnel prennent leur part dans la deuxième partie. Dans la troisième partie, nous présenterons les travaux de J.Gorge et X. Gandibleux [58, 57], les auteurs proposent de nouvelles propriétés visant à réduire a priori la taille du problème de sac à dos biobjectif, suivis d'un exemple didactique.

5.2 Notion de core du problème de sac à dos multiobjectif

Pour le problème du sac à dos uniobjectif unidimensionnel, le core (noyau) est défini comme un sous-ensemble d'objets (voir : Définition 5.3) dont leur efficacité est similaire à l'efficacité de l'élément bloquant définit au chapitre 1 (voir : Définition 1.11).

Les algorithmes les plus efficaces pour résoudre le problème de sac à dos uniobjectif en variables $\{0, 1\}$ sont basés sur la notion du core et le problème core, malheureusement, ce concept est resté limité au cas uniobjectif unidimensionnel [4], jusqu'aux travaux de J. Puchinger [84] en 2006, là où les auteurs ont proposé une extension au cas uniobjectif multidimensionnel et par conséquent ils ont défini le concept du core pour le problème du sac à dos uniobjectif multidimensionnel,(voir : Chapitre1). Gomes da Silva et al. [44] ont validé l'existence de tel ensemble core pour le problème de sac à dos biobjectif unidimensionnel en variables $\{0, 1\}$. En 2009, Mavrota et al. [73] ont proposé une adaptation du concept du core pour le problème de sac à dos biobjectif multidimensionnel en variables $\{0, 1\}$. Cette extension n'était pas triviale, les auteurs

ont mis en oeuvre la notion du core de ce dernier cas en synthétisant les deux œuvres, celui de J. Puchinger [84] et de Gomes et al. [44].

La première section de ce chapitre sera consacré aux travaux de Gomes da Silva et al. [44] tandis que les travaux de Mavrota et al. [73] seront présentés dans la deuxième section.

5.2.1 Notion de core multiobjectif unidimensionnel

La notion du core du problème de sac à dos monoobjectif est basée sur le tri de l'efficacité d'objet en ordre décroissant [4], cet ordre ne se trouve pas dans le cas multiobjectif. C.Gomes da Silva et al. ont proposé une généralisation du core pour le problème de sac à dos biobjectif [44]. On présente dans cette section le déroulement de cette généralisation :

Le problème de sac à dos biobjectif peut s'écrire comme suit :

$$(BOKP)\begin{cases} \text{“max”}(Z^k) = \sum_{i=1}^{n} c_i^k x_i,\ k=1,2, \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ x_i \in \{0,1\},\ i=1,\ldots,n. \end{cases} \tag{5.1}$$

où x_i est une variable de décision égal à 1 si l'objet i est sélectionné dans le sac à dos et 0 sinon. On suppose que :

a. Toutes les données $\omega,\ c_i$ et w_i, pour $i=1,\ldots,n$, sont des entiers positifs.

b. $\sum_{i=1}^{n} w_i > \omega$, dans l'ordre d'éviter toutes les solutions triviales.

C.Gomes da Silva et al. [44], ont défini le problème core comme suit :

$$(BOKP_C)\begin{cases} \max(Z_1) = \sum_{i\in C} c_i^1 x_i \\ \max(Z_2) = \sum_{i\in C} c_i^2 x_i \\ \sum_{i\in C} w_i x_i \leq \tilde{\omega} \\ x_i \in \{0,1\},\ i \in C. \end{cases} \tag{5.2}$$

où $\widetilde{\omega} = \omega - \sum_{i\in\{1,2,\ldots,n\}\setminus C} w_i$.

Les fonctions objectifs peuvent être agrégées en un seul objectif. Dans le cas biobjectif une agrégation des fonctions peut être exprimée par :
$Z(x,\lambda) = \lambda Z_1(x) + (1-\lambda)Z_2(x)$ avec $0 \leq \lambda \leq 1$. Soit $\Im$ une famille des fonctions paramétriques $Z(x,\lambda)$, le core est défini dans le cas biobjectif comme suit :

Définition 5.1 Soit $\Im$ une famille des fonctions paramétriques $Z(x,\lambda)$, le core d'une solution efficace x^t pour le problème de sac à dos biobjectif (BOKP) est le plus petit core en considérant chaque fonction de $\Im$ séparément, obtenu en remplaçant les objets par ordre décroissant de leur efficacité pondérée $\dfrac{\lambda c_i}{w_i}$.

Remarque 5.1 Cette définition s'étend au cas multiobjectif $p \geq 2$.

Considérons qu'il existe p solutions efficaces et q fonctions objectifs, le core associé à une solution efficace x^t en tenant compte des fonctions $Z(x, \lambda^k)$ est défini :

$$C^{k,t} = \{j_1^{k,t}, \ldots, j_2^{k,t}\}, \tag{5.3}$$

où

$$j_1^{k,t} = \min\{i \mid x_i^t = 0, \quad i = 1, \ldots, n\}, \tag{5.4}$$

et

$$j_2^{k,t} = \max\{i \mid \quad x_i^t = 1, \quad i = 1, \ldots, n\}. \tag{5.5}$$

(Si $j_1^t > j_2^t$, on pose $C^t = \emptyset$) et où les objets sont ordonnés suivant un ordre décroissant des rapports

$$e_i(\lambda) = \frac{\lambda^k c_i^1 + (1 - \lambda^k) c_i^2}{w_i}, \qquad i = 1, \ldots, n. \tag{5.6}$$

Le core du problème de sac à dos biobjectif d'une solution x^t est défini par :

$$C^{k*,t} = argmin_{k=1,\ldots,q}\{|C^{k,t}|\}. \tag{5.7}$$

Déterminer le core d'un problème BOKP d'une solution efficace exige que toutes les fonctions $Z(x, \lambda)$ de $\Im$ soient analysées. Pour obtenir le plus petit ensemble core, une estimation se fait de la fonction déterminante l'ensemble core pour obtenir l'ensemble efficace. Pour déterminer l'ensemble core d'une solution efficace, les objets du problème de sac à dos en variables $\{0, 1\}$ doivent être triés par un ordre décroissant de l'efficacité d'objet :

$$e_i(\lambda) = \frac{\lambda c_i^1 + (1 - \lambda) c_i^2}{w_i} = \frac{c_i^2}{w_i} + \frac{c_i^1 - c_i^2}{w_i}, \qquad 0 \leq \lambda \leq 1. \tag{5.8}$$

Le rapport donné par l'equation (5.8) est une fonction de λ, pour cette raison, ce rapport n'est pas bien défini, mais il peut vérifié l'inégalité suivante :

$$\min\{\frac{c_i^1}{w_i}, \frac{c_i^2}{w_i}\} \leq e_i(\lambda) \leq \max\{\frac{c_i^1}{w_i}, \frac{c_i^2}{w_i}\}. \tag{5.9}$$

Pour déterminer le core du problème paramétrique, on doit trier le rapport défini par l'équation(5.8) par ordre décroissant donné par :

$$e_{l_1}(\lambda) \geq e_{l_2}(\lambda) \geq \ldots \geq e_{l_n}(\lambda), \quad \text{où } \{l_1, l_2, \ldots, l_n\} = \{1, 2, \ldots, n\}. \tag{5.10}$$

A chaque fois que λ varie dans $[0, 1]$, l'ordre défini dans (5.10) n'est pas stable. La détermination des valeurs de λ pour que l'ordre soit stable est bien détaillé dans [44].

Pour détecter l'ensemble core C, le problème biobjectif est transformé à un problème uniobjectif, prenant l'exemple suivant : soit la fonction paramétrique $Z(x, \frac{1}{2}) = \frac{1}{2}Z_1(x) + \frac{1}{2}Z_2(x)$

En utilisant cette fonction particulière $Z(x, \lambda)$, l'ensemble core C est composé de 21 objets à la droite de la figure 5.1 (11 objets dans le cas de la même figure à gauche) autour de l'objet bloquant.

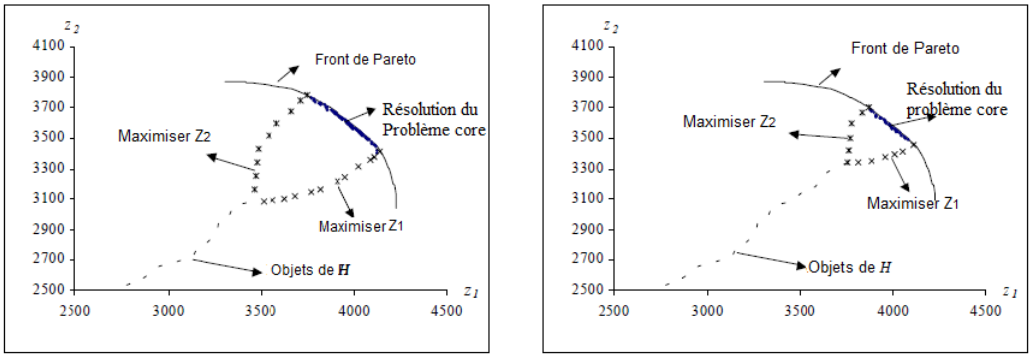


FIGURE 5.1 – Core exacte

Dans le but d'éclaircir la notion du core dans le cas d'un problème de sac à dos biobjectif, on présente un exemple didactique :

Exemple 5.1

$$(BOKP)\begin{cases} \max(Z_1) = 91x_1 + 38x_2 + 8x_3 + 77x_4 + 93x_5 + 13x_6 + 41x_7 + 81x_8 + 11x_9 + 93x_{10} \\ \max(Z_2) = 35x_1 + 49x_2 + 86x_3 + 19x_4 + 50x_5 + 90x_6 + 21x_7 + 59x_8 + 63x_9 + 55x_{10} \\ 33x_1 + 45x_2 + 23x_3 + 19x_4 + 70x_5 + 100x_6 + 63x_7 + 70x_8 + 55x_9 + 10x_{10} \leq 244 \\ x_i \in \{0,1\} \end{cases} \tag{5.11}$$

Ce problème admet cinq solutions efficaces listées dans la table 5.1 :

i	1	2	3	4	5	6	7	8	9	10
X^1	1	0	1	1	0	1	0	0	1	1
X^2	1	0	1	1	1	0	0	1	0	1
X^3	1	1	1	0	0	0	0	1	1	1
X^4	1	1	1	0	1	0	0	0	1	1
X^5	0	1	1	1	1	0	0	1	0	1

TABLE 5.1 – Solutions efficaces

La table 5.2 résume tous les points non-dominés.

Z	Z_1	Z_2
X^1	293	348
X^2	443	304
X^3	320	347
X^4	332	338
X^5	388	318

TABLE 5.2 – Points non-dominés

En utilisant les formules précédentes données par les équations (5.3), (5.4) et (5.5), pour trois différentes fonctions $Z(\lambda, x)$ tel que les valeurs de $\lambda \in \{(0,1),(1,0),(0.5,0.5)\}$, la table 5.3 représente l'ensemble core associé à toutes les cinq solutions efficaces.

$Z(\lambda, x) = 0Z_1 + 1Z_2$	$Z(\lambda, x) = 1Z_1 + 0Z_2$	$Z(\lambda, x) = \frac{1}{2}Z_1 + \frac{1}{2}Z_2$
$C^{1,1} = \{2, 1, 4, 6\}$	$C^{2,1} = \{5, 8, 2, 7, 3, 9, 6\}$	$C^{3,1} = \{5, 8, 2, 9, 6\}$
$C^{1,2} = \{9, 2, 1, 4, 6, 8, 5\}$	$C^{2,2} = \{2, 7, 3\}$	$C^{3,2} = \{8, 2\}$
$C^{1,3} = \{4, 6, 8\}$	$C^{2,3} = \{4, 1, 5, 8, 2, 7, 3, 9, 6\}$	$C^{3,3} = \{4, 3, 1, 5, 8, 2, 9\}$
$C^{1,4} = \{4, 6, 8, 5\}$	$C^{2,4} = \{4, 1, 5, 8, 2, 7, 3, 9\}$	$C^{3,4} = \{4, 3, 1, 5, 8, 2, 9\}$
$C^{1,5} = \{9, 2, 1, 4, 6, 8, 5\}$	$C^{2,5} = \{1, 5, 8, 2, 7, 3\}$	$C^{3,5} = \{1, 5, 8, 2\}$
$C^{1,*} = \{9, 2, 1, 4, 6, 8, 5\}$	$C^{2,*} = \{4, 1, 5, 8, 2, 7, 3, 9, 6\}$	$C^{3,*} = \{4, 3, 1, 5, 8, 2, 9, 6\}$

TABLE 5.3 – Ensemble core pour différentes fonctions $Z(\lambda, x)$

D'après cet exemple, on peut conclure que :
- Pour le même problème, plusieurs fonctions paramétriques peuvent être construites pour déterminer l'ensemble core.
- L'ensemble core dépend de la solution efficace considérée.
- La même solution est associé à différent ensemble core.

Aussi l'exemple montre que pour plus d'un objectif, le rapport profit sur poids n'est pas bien définit, il dépend de la fonction paramétrique considérée.

De la même manière que le cas uniobjectif, l'ensemble core est le plus petit core donné d'une solution efficace (voir : Définition 5.1, Equation (5.7)).

Donc l'ensemble core exacte des solutions efficaces X^1, X^2, X^3, X^4, X^5 est donné respectivement par $C^{1,1}, C^{3,2}, C^{1,3}, C^{1,4}, C^{3,5}$.

Remarque 5.2 Un autre exemple de taille $n = 7$ est détaillé concernant la détermination des valeurs de λ [44].

En conclusion, les auteurs ont proposé une extension du concept core au cas d'un problème de sac à dos biobjectif en variables $\{0, 1\}$, cependant, cette extension n'est pas évidente puisque plusieurs core peuvent être définis pour chaque solution efficace. Les auteurs appuient leurs propositions par de nombreuses expérimentations sur cinq familles d'instances. Il ont noté que les caractéristiques du cas uniobjectif sont valables au cas biobjectif [44]. Aussi, ils ont observé que pour une instance donnée, la taille du core est relativement insensible à l'intérieur d'une même famille. Cette taille peut énormément varier d'une famille à une autre.

Enfin, les auteurs proposent deux d'algorithmes de résolution utilisant le core, respectivement pour une résolution approchée et exacte, mais pas de résultats numériques.

5.2.2 Notion de core pour le cas multiobjectif multidimensionnel

Mavrotas et al. [73] ont combiné les résultats proposés par Gomes et al. [44] sur le core pour le problème de sac à dos biobjectif (BOKP) avec ceux de Puchinger et al. [84] sur le core pour le problème de sac à dos monoobjectif multidimensionnel, et ils ont exploité le concept du core pour la résolution du problème de sac à dos biobjectif multidimensionnel [73].

Les auteurs traitent dans cet article [73] le problème du sac à dos biobjectif multidimensionnel, ils proposent l'adaptation du concept du core qui a été utilisé efficacement au cas du problème de sac à dos uniobjectif multidimensionnel. L'idée principale du concept du core est basée sur le principe "diviser pour conquérir". Au lieu de résoudre un problème avec n variables, ils résolvent plusieurs sous-problèmes avec une fraction de n variables. L'idée principale est la suivante :

- Selon Gomes da Silva et al. [44], en première étape, ils résolvent le problème relaxé qui produit l'ensemble des solutions efficaces extrêmes (X_{EE}), ensuite ils affectent à chaque solution efficace extrême un intervalle de paramètre λ correspond à un programme linéaire multiobjectif (MOLP).
- Pour chaque solution efficace extrême et en utilisant les valeurs de paramètres λ pour agréger les deux fonctions objectifs, on transforme le problème à un problème uniobjectif. Ensuite, nous profitons de la méthode de Puchinger et al.[84] concernant le concept du core pour le problème du sac à dos multidimensionnel.
- Pour chaque solution efficace extrême, nous affectons le core approprié (selon Puchinger et al. [84] l'efficacité de dual). Nous ajustons le problème biobjectif (MKP) aux variables du core spécifiques et on résout avec la méthode Branch & Bound, afin de générer le front Pareto.
- Nous répétons ce processus pour toutes les solutions efficaces extrêmes et à la fin nous fusionnons l'ensemble front pareto pour obtenir une représentation de l'ensemble complet du front Pareto du problème initial.

L'idée peut être exprimée par un graphe illustratif. La figure 5.2 explique la méthode développée pour la notion du core pour le problème de sac à dos biobjectif multidimensionnel, le front de Pareto du problème détendu est représenté par la ligne A, B, C, D. Pour chacune des quatre solutions efficaces extrêmes, nous déterminons l'ensemble core. Les variables qui sont en dehors du core sont fixées soient à 0 ou 1 (variables régulières), et la partie fixe correspond aux points A', B', C', D'. Les quatre solutions obtenues en résolvant le problème BOMKP par la méthode Branch & Bound (MCBB) implique que les variables du core produisent les solutions optimales localement pertinentes du front Pareto. Par exemple à partir des variables fondamentales du core de A, les solutions optimales localement Pareto A_1, A_2, A_3 sont produites. Après que toutes les solutions efficaces extrêmes soient visitées, les solutions optimales localement Pareto sont fusionnées pour obtenir la représentation de l'ensemble Pareto du problème original du sac à dos biobjectif multidimensionnel (BOMKP).

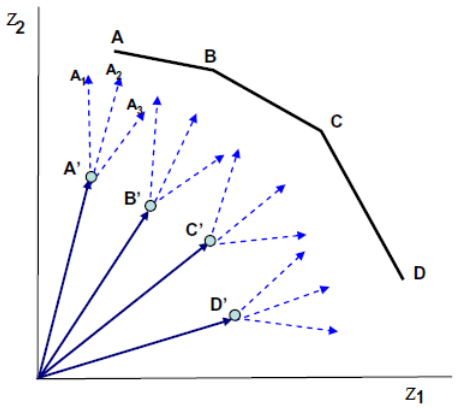


FIGURE 5.2 – Représentation graphique du core d'un problème BOMKP

Toutes les étapes de la méthode développée et plus de détails sont dans la référence [73].

5.3 Règles de réduction pour le problème de sac à dos multiobjectif

Dans cette section, on présentera les travaux élaborés par J.Jorge et X.Gandibleux [58, 57], les auteurs ont remarqué que le fait que le core d'un problème soit défini à partir de la connaissance d'une solution optimale rend difficile son utilisation dans une méthode de résolution. Ils ont proposé un prétraitement basé sur des propriétés et quelques notions existantes dans la littérature, pour extraire un ensemble de variables dites régulières. Ces variables prennent une seule valeur qui vaut à 0 ou 1 dans toutes les solutions efficaces.

5.3.1 Caractère régulier des variables

À travers une instance particulière aux coefficients générés aléatoirement et sans corrélation, les auteurs dans [58, 57, 56] ont remarqué que cette instance de taille 50 accepte 34 solutions efficaces, où 9 variables prennent la valeur 0 dans toutes les solutions efficaces, tandis que 18 prennent la valeur 1 dans toutes les solutions efficaces et les autres restantes interviennent dans les deux cas (prennent les valeurs 0 ou 1.)

On présente cette instance du problème de sac à dos biobjectif par la figure (5.3) qui montre les profits et poids des objets, associés au caractère régulier des variables correspondantes.

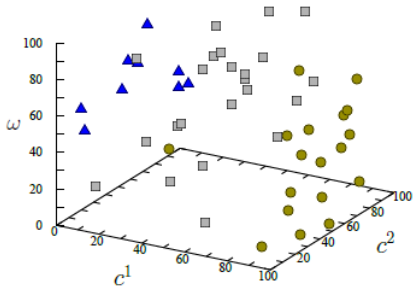


FIGURE 5.3 – Caractère régulier des variables

Les variables égales à 0 sont indiquées par des triangles, celles des variables égales à 1 sont illustrées par des cercles. Les données représentées par des carrés correspondent à des variables intervenant avec les deux valeurs dans toutes les solutions efficaces.

Les auteurs ont observé que certaines données semblent groupées selon le type des variables auxquelles elles sont attachées. En particulier, 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, ils ont constaté que des variables intervenant avec les deux valeurs sont présentées dans tout l'espace. On présente un exemple didactique expliquant ce caractère de variables :

Exemple 5.2 On considère le problème de sac à dos biobjectif $(BOKP)$:

$$(BOKP)\begin{cases} \max(Z_1) = 2x_1 + 2x_2 + 5x_3 + 9x_4 + 8x_5 + 6x_6 \\ \max(Z_2) = 8x_1 + 2x_2 + 6x_3 + 2x_4 + 5x_5 + 8x_6 \\ \quad 8x_1 + 8x_2 + 7x_3 + 5x_4 + 4x_5 + 2x_6 \leq 17 \\ \quad x_i \in \{0,1\} \end{cases} \tag{5.12}$$

L'ensemble efficace est représenté par la table 5.4 :

i	1	2	3	4	5	6
X^1	1	0	1	0	0	1
X^2	1	0	0	0	1	1
X^3	0	0	1	0	1	1
X^4	0	0	1	1	0	1
X^5	0	0	0	1	1	1

TABLE 5.4 – Solutions efficaces

On remarque que la variable $x_2 = 0$ et la variable $x_6 = 1$ dans les cinq solutions efficaces.

L'ensemble efficace situé dans l'espace des objectifs est représenté dans la table 5.5 :

X	X^1	X^2	X^3	X^4	X^5
$Z(X)$	$(13, 22)$	$(16, 21)$	$(19, 19)$	$(20, 16)$	$(23, 15)$

TABLE 5.5 – Les points non-dominés

Les points non-dominés sont représentés par la figure 5.4 :

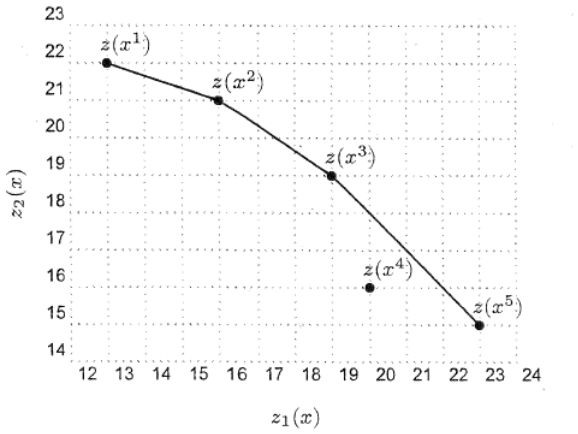


FIGURE 5.4 – l'ensemble des points non-dominés $Y_N = Z(X_E)$

5.3.2 Variables régulières et relation de dominance

Cette section présente les définitions des éléments nécessaires pour les propriétés présentées par la suite.

Définition 5.2 Variables régulières

Soit X_E l'ensemble complet de solutions efficaces du problème de sac à dos biobjectif donné par l'équation (5.1).

$C_0 = \{i \in \{1, \ldots, n\} \,|\, x_i = 0,\ \forall x \in X_E\}$

$C_1 = \{i \in \{1, \ldots, n\} \,|\, x_i = 1,\ \forall x \in X_E\}$

Les ensembles C_0 et C_1 sont les indices des variables régulières, celles qui n'acceptent qu'une seule valeur dans toutes les solutions efficaces.

L'espace déterminé par les données du problème de sac à dos est l'ensemble des vecteurs donné par :

$$V = \left\{v^i \in \mathbb{N}^n \times \mathbb{Z} \mid v^i = (c_i^1, \ldots, c_i^p, -w_i),\ i \in \{1, \ldots, n\}\right\} \tag{5.13}$$

Définition 5.3 On considère $v^i,\ v^j \in V$. Un vecteur v^i domine v^j ($v^i \succ v^j$), si $(c_i^1, \ldots, c_i^p, -w_i) \geq (c_j^1, \ldots, c_j^p, -w_j)$ pour au moins une inégalité stricte.

Définition 5.4 Soit $v^i \in V$. L'ensemble de tous les indices j de vecteurs v^j dominant le vecteur de donnée v^i est appelé ensemble préféré et noté par :

$$P(v^i) = \{j \in \{1, \ldots, n\} \ \mid \ v^j \succ v^i\}. \tag{5.14}$$

Définition 5.5 Soit $v^i \in V$. L'ensemble des indices j des vecteurs v^j dominés par le vecteur v^i est appelé ensemble dominé et noté

$$D(v^i) = \{j \in \{1, \ldots, n\} \ \mid \ v^i \succ v^j\}. \tag{5.15}$$

Bornes sur la cardinalité des solutions efficaces

Une borne inférieure LB et une borne supérieure UB de la cardinalité d'une solution optimale x pour le problème de sac à dos uniobjectif unidimensionnel ont été introduites par Glover [43]. Ces bornes sont définies comme suit :

Définition 5.6

$$LB = \max\{s \ \mid \ \sum_{i=1}^{s} w_i \leq \omega\} \quad \text{tel que } \ w_i \geq w_{i+1},\ \ \forall i \in \{1, \ldots, n-1\} \tag{5.16}$$

$$UB = \max\{s \ \mid \ \sum_{i=1}^{s} w_i \leq \omega\} \quad \text{tel que } \ w_i \leq w_{i+1},\ \ \forall i \in \{1, \ldots, n-1\} \tag{5.17}$$

Les bornes LB et UB sont clairement indépendantes des fonctions objectifs, cette observation a mené Gandibleux et Fréville [39] à généraliser leurs utilisation au cas multiobjectif à travers la proposition suivante :

Proposition 5.1 Soit X_E l'ensemble complet de solutions efficaces (5.1). Si $x \in X_E$, alors :

$$LB \leq \sum_{i=1}^{n} x_i \leq UB.$$

Ces bornes seront utilisées conjoignement avec les deux ensembles dominé $D(v^i)$ et préféré $P(v^i)$ définis respectivement par les équations (5.14), (5.15) pour construire de nouvelles propriétés détectant un ensemble de variables régulières.

5.3.3 Propriétés des variables régulières

Dans cette section des conditions proposées dans [56] sous lesquelles une variable soit régulière dans l'ensemble des solutions efficaces.

Proposition 5.2 Soit $x \in X_E$.
$\forall i \in \{1, 2, \ldots, n\}$, si $x_i = 0$ alors $x_j = 0$, $\forall j \in D(v^i)$.

Proposition 5.3 Soit $x \in X_E$.
$\forall i \in \{1, 2, \ldots, n\}$, si $x_i = 1$ alors $x_j = 1$, $\forall j \in P(v^i)$.

Proposition 5.4 Soit $i \in \{1, 2, \ldots, n\}$.
Si $|P(v^i)| \geq UB$ alors $x_i = 0$, $\forall x \in X_E$.

Proposition 5.5 Soit $i \in \{1, 2, \ldots, n\}$.
Si $\sum_{j \in P(v^i)} w_j + w_i > \omega$ alors $x_i = 0$, $\forall x \in X_E$.

Proposition 5.6 Soit $i \in \{1, 2, \ldots, n\}$.
Si $n - |D(v^i)| \leq LB$ alors $x_i = 1$, $\forall x \in X_E$.

Proposition 5.7 Soit $i \in \{1, 2, \ldots, n\}$.
Si $\sum_{j \notin D(v^i)} w_j < \omega$ alors $x_i = 1$, $\forall x \in X_E$.

Preuve : La preuve des propositions (5.2), (5.3), (5.4), (5.5), (5.6), (5.7) sont dans la référence [56].

Détermination de l'ensemble de variables régulières

L'algorithme 8 détermine l'ensemble de variables régulières C'_0 et C'_1 détecté par J.Jorge et X.Gandibleux [56, 57, 58].

Algorithme 8: Détermination de l'ensemble de variables régulières.

Input :
$\downarrow c_i^k$: le vecteur profit de chaque objectif ; $k = 1, 2, \ldots, p$
$\downarrow w_i : i = 1, \ldots, n$: le poids de chaque objet
$\downarrow \omega$: la capacité du sac à dos
Output :
$\uparrow C_0', C_1'$

```
Calculer LB et UB
for i ∈ {1, 2, ..., n} do
    Calculer |P(v^i)| et |D(v^i)|
    if |P(v^i)| ≥ UB ou Σ_{j∈P(v^i)} w_j + w_i > w then
        C_0' = C_0' ∪ {i}
    else
        n − |D(v^i)| ≤ LB ou Σ_{j∉D(v^i)} w_j < w
    end
    C_1' = C_1' ∪ {i}
end
return C_0', C_1';
```

On présente le déroulement de l'algorithme 8 sur l'exemple 5.2. D'une part, cette instance accepte cinq solutions efficaces, listées dans la Table 5.4, chacune de ces cinq solutions vérifie $x_2 = 0$ et $x_6 = 1$. D'une autre part, la borne inférieure $LB = 2$ et la borne supérieure $UB = 3$. Dans le but de détecter un ensemble de variables régulières, nous appliquons les propriétés (5.2), (5.3), (5.4), (5.5), (5.6), (5.7) :
Les vecteurs de données sont :

$$v^1 = \begin{pmatrix} 2 \\ 8 \\ -8 \end{pmatrix}, \; v^2 = \begin{pmatrix} 2 \\ 2 \\ -8 \end{pmatrix}, \; v^3 = \begin{pmatrix} 5 \\ 6 \\ -7 \end{pmatrix},$$

$$v^4 = \begin{pmatrix} 9 \\ 2 \\ -5 \end{pmatrix}, \; v^5 = \begin{pmatrix} 8 \\ 5 \\ -4 \end{pmatrix}, \; v^6 = \begin{pmatrix} 6 \\ 8 \\ -2 \end{pmatrix}.$$

Dans cet exemple, la table 5.6 représente les ensembles préférés et dominés de chaque objet i tel que $i = 1, 2, \ldots, 6$:

i	$P(v^i)$	$D(v^i)$
v^1	$\{6\}$	$\{2\}$
v^2	$\{1, 3, 4, 5, 6\}$	$\emptyset$
v^3	$\{6\}$	$\{2\}$
v^4	$\emptyset$	$\{2\}$
v^5	$\emptyset$	$\{2\}$
v^6	$\emptyset$	$\{1, 2, 3\}$

TABLE 5.6 – Ensembles préférés et dominés

D'après la table 5.6, on constate que :

- $P(v^2) = \{1, 3, 4, 5, 6\}$, $\mid P(v^2) \mid = 5 \geq UB$ tel que $UB = 3$, on conclu de la propriété 5.4 que $x_2 = 0$ dans toutes les solutions efficaces.
- $D(v^6) = \{1, 2, 3\}$, d'après la propriété 5.6, $n - |D(v^i)| \nleq LB(\omega)$ tel que $n = 6$, $|D(v^6)| = 3$ et $LB = 2$, mais d'après la propriété 5.7, $\sum_{j \notin D(v^i)} w_j < \omega$, on a : $w_4 + w_5 + w_6 = 5 + 4 + 2 = 11 < 17$, alors $x_6 = 1$ dans toutes les solutions efficaces.

En conclusion J.Jorge et X.Gandibleux ont développé des propriétés basées sur quelques notions existantes dans la littérature, détectant un ensemble de variables régulières avant la résolution du problème de sac à dos biojectif.

5.4 Conclusion

On a présenté dans ce chapitre, trois cas concernant une réduction de la taille du problème de sac à dos multiobjectif unidimensionnel et multidimensionnel avant sa résolution. Le premier cas s'agit de la notion du core dédiée au problème de sac à dos biobjectif unidimensionnel inspiré du cas uniobjectif unidimensionnel ensuite l'extension au cas biobjectif multidimensionnel et enfin nous avons exposé un prétraitement établi par J.Jorge et X.Gandibleux [58, 57, 56]. La détection de variables régulières est basée sur des concepts introduits en détail dans ce chapitre. L'espace de recherche a été réduit ce qui peut conduire à une diminution du temps de calcul nécessaire pour la résolution du problème $MOKP$. Les résultats théoriques sont évalués à la lumière présentant deux objectifs, et appuyés par des expérimentations numériques [57]. Ces expérimentations montrent que la procédure de réduction est utile pour les instances où les coefficients sont indépendants, en particulier pour le cas biobjectif, la procédure de réduction n'a pas été adaptée aux instances où les coefficients d'un objectif et de la contrainte sont corrélés. Cependant, ces expérimentations numériques montrent que de nombreuses variables n'obéissent pas aux propriétés développées, même si elles sont visiblement régulières après calcul de l'ensemble complet des solutions efficaces. Cette observation nous a motivé à faire une étude plus approfondie sur des approches fixant a priori ces variables. Dans le chapitre suivant nous proposons d'autres propriétés pour remédier aux cas non étudiés.

Chapitre 6

Une nouvelle procédure de réduction pour le problème de sac à dos multiobjectif

6.1 Introduction

Il existe très peu de méthodes exactes de réduction dédiées au problème de sac à dos multiobjectif. Aussi, il nous a semblé intéressant de voir ce qui pouvait être fait dans ce sens. Ceci est l'objectif de notre première partie de ce travail de thèse. Après une étude des méthodes de réduction existantes, une méthode utilisée en biobjectif a attiré notre attention. Il s'agit de la méthode expliquée dans la section 3 du chapitre 5.

L'observation faite dans la conclusion du chapitre précédent, nous a conduit à proposer une nouvelle procédure de réduction pour déterminer a priori un ensemble de variables régulières pour le problème de sac à dos multiobjectif en se basant sur plusieurs propriétés. Ensuite, nous intégrons cette procédure dans une méthode exacte pour étudier son utilité dans la résolution du problème considéré. Dans la section suivante, la formulation du problème de sac à dos multiobjectif en variables binaires est présentée, nous définissons l'espace de données du problème, inspiré de l'efficacité d'objet du cas unicritère, nous décrivons théoriquement et techniquement notre algorithme de réduction suivi d'un exemple didactique expliquant toutes les étapes et montrant la valeur ajoutée dans l'optimisation combinatoire, particulièrement pour le problème de sac à dos biobjectif en variables binaires.

Les expérimentations numériques en fin de ce chapitre, nous permettent de tirer des interprétations et conclusions sur la performance de la procédure de réduction développée et son impact sur le temps d'exécution globale.

6.2 Espace de données du problème MOKP

Le problème de sac à dos multiobjectif peut s'écrire comme suit :

$$(MOKP)\begin{cases} \text{“max”}(Z^k) = \sum_{i=1}^{n} c_i^k x_i \quad k = \overline{1,p} \\ \sum_{i=1}^{n} w_i x_i \leq \omega \\ x_i \in \{0,1\} \quad i = \overline{1,n} \end{cases} \tag{6.1}$$

où :

1. c_i^k, w_i et w sont des entiers positifs et $x = (x_1, x_2, \ldots, x_n)$ dans l'ordre d'éviter toutes solutions triviales, on suppose que :
2. $w_i \leq \omega, \quad i = \overline{1,n}$.
3. $\sum_{i=1}^{n} w_i > \omega$.

Rappelons la définition de l'efficacité d'un objet dans le cas d'un problème de sac à dos unicritère unidimensionnel.

Définition 6.1 Efficacité d'un objet

On appelle efficacité d'un objet le rapport de son profit sur son poids, noté par $e_i = \dfrac{c_i}{w_i}$.

Remarque 6.1 – La notion d'efficacité d'un objet dans le cas unicritère est différente de la notion d'efficacité d'une solution dans le cas multicritère.

Soient C_0 et C_1 les ensembles d'indices qui correspondent aux variables régulières (voir : définition 5.2). On définit l'espace de données du problème de sac à dos multicritère de la manière suivante :

Définition 6.2 Espace de données du problème de sac à dos multicritère

Soit l'ensemble E de vecteurs E^i de composantes e_i^k, $i = 1, \ldots, n, \ k = 1, \ldots, p$. Alors :

$$E = \left\{ E^i \in \mathbb{R}^n \ \mid \ E^i = (e_i^1, e_i^2, \ldots, e_i^p), \ i \in \{1, \cdots, n\} \right\} \text{ où } e_i^k = \frac{c_i^k}{w_i}, \qquad k = 1, \ldots, p. \tag{6.2}$$

Définition 6.3 Soit $E^i, \ E^j \in E$. Un vecteur E^i domine E^j $(E^i \succ E^j)$, si :

$$e_i^k \geq e_j^k \quad \forall k \in \{1, 2, \ldots, p\}, \text{ et } \exists k \in \{1, 2, \ldots, p\} \ t.q. \ e_i^k > e_j^k.$$

Définition 6.4 Soit $E^i \in E$.

L'ensemble de tous les indices j de vecteurs E^j dominant le vecteur de données E^i est appelé ensemble préféré donné par :

$$P(E^i) = \{j \in \{1, \ldots, n\} \ \mid \ E^j \succ E^i\}.$$

Définition 6.5 Soit $E^i \in E$.

L'ensemble des indices j des vecteurs E^j dominés par le vecteur de données E^i est appelé ensemble dominé donné par :

$$D(E^i) = \{j \in \{1, \ldots, n\} \mid E^i \succ E^j\}.$$

6.3 Description de l'algorithme développé

On présente dans cette section, le déroulement de la méthode proposée ainsi que sa description théorique et technique suivie d'un exemple didactique en fin de cette section.

6.3.1 Description théorique

L'algorithme proposé s'articule autour de deux étapes, la première, détermine p solutions efficaces supportées extrêmes, tandis que la deuxième, détecte un ensemble de variables régulières (voir : la définition 5.2).
On note $X^1, X^2, \ldots, X^p$, p solutions efficaces supportées extrêmes du problème $MOKP$ définit par l'équation (6.1) tel que $X^k = (x_1^k, x_2^k, \ldots, x_n^k),\ k = 1, 2, \ldots, p$.
On définit aussi les ensembles J_0 et J_1 par :

$$J_0 = \{j \in \{1, \ldots, n\} \mid x_j^1 = x_j^2 = \cdots = x_j^p = 0\} \tag{6.3}$$

$$J_1 = \{j \in \{1, \ldots, n\} \mid x_j^1 = x_j^2 = \cdots = x_j^p = 1\} \tag{6.4}$$

Sans perdre de généralité, on considère le problème de sac à dos biobjectif ($p = 2$).
Pour chercher deux solutions efficaces supportées extrêmes X^1 et X^2 du problème BOKP, on propose la méthode lexicographique :

1. La solution lexicographiquement optimale X^1 correspondante au problème $\text{lexmax}_{x \in X}\big(Z^1(x), Z^2(x)\big)$ est obtenue de la façon suivante :
   En première étape, on résout le premier problème uniobjectif Z^1, avec le paramètre λ égal à $(1, 0)$, soit X_{opt}^1 une solution optimale. Si cette solution est unique alors elle est efficace et la solution lexicographiquement optimale X^1 est égale à X_{opt}^1. Sinon, s'il existe plusieurs solutions optimales, alors X_{opt}^1 est une solution faiblement efficace, dans ce cas, pour améliorer le deuxième objectif sans dégrader le premier, on optimise le deuxième problème uniobjectif $Z^2(x)$ sous la contrainte additionnelle $Z^1(x) = Z^1(X_{opt}^1)$. C'est à dire : $\max\limits_{x \in X}\{Z_2 \mid Z_1(x) = Z_1(X_{opt}^1)\}$, soit X^1 une solution efficace parmi l'ensemble de solutions lexicographiquement optimales du problème.
2. La deuxième solution lexicographiquement optimale X^2 correspondante au problème
   $\text{lexmax}_{x \in X}\big(Z^2(x), Z^1(x)\big)$ est obtenue de la même manière.

Les ensembles d'indices correspondants aux variables régulières notés par D_0 et D_1, sont calculés de la manière suivante :

a. Pour $i \in J_0$, les objets dominés par au moins UB objets "en terme d'efficacité" sont groupés à la fin, dont les valeurs de variables correspondantes sont égales à $x_i = 0$ dans toutes les solutions efficaces.
b. Pour $i \in J_1$, les objets dominant au moins $n - LB$ objets "en terme d'efficacité" sont groupés en premier, dont les valeurs de variables correspondantes sont égales à $x_i = 1$ dans toutes les solutions efficaces.

Mathématiquement, on écrit :

$$D_0 = \{i \in J_0 \mid |P(E^i)| \geq UB\} \tag{6.5}$$

tel que $P(E^i)$ est l'ensemble préféré défini ci-dessus (voir : définition 6.4) et UB est la borne supérieure de cardinalité des solutions réalisables pour le problème biobjectif (voir : définition 5.6, l'equation 5.17).

$$D_1 \quad = \quad \{i \in J_1 \ | \ |D(E^i)| \geq n - LB\} \tag{6.6}$$

tel que $D(E^i)$ est l'ensemble dominant défini ci-dessus (voir : définition 6.5) et LB est la borne inférieure de cardinalité des solutions réalisables pour le problème biobjectif. (voir : définition 5.6, l'equation 5.16).

6.3.2 Description technique

Dans cette section, on présente une description technique de l'algorithme proposé, cet algorithme détecte un ensemble d'indices noté D_0 et D_1 dont les variables correspondantes vaux 0 et 1 respectivement.

Algorithme 9: Détermination des ensembles D_0 et D_1

Input :
$\downarrow c_i^k$: le vecteur profit de chaque objectif $k = 1, 2, \ldots, p$
$\downarrow w_i, i = 1, \ldots, n$: les poids des objets
$\downarrow w$: la capacité du sac à dos
Output :
$\uparrow D = D_0 \cup D_1$: l'ensemble d'indices de variables régulières

Initialisation
$D_0 \leftarrow \varnothing, D_1 \leftarrow \varnothing$;
Trouver le vecteur d'efficacité d'objet de chaque objectif.
for $i \leftarrow 1$ ***to*** n **do**
  for $j \leftarrow 1$ ***to*** p **do**
    $e^j(i) \leftarrow \frac{c^j(i)}{w(i)}$;
  end
end
Définir l'efficacité d'objet dans le cas d'un problème multiobjectif ;
for $i \leftarrow 1$ ***to*** n **do**
  $E^i \leftarrow [e^1(i), e^2(i), \ldots, e^p(i)]$
end
Calculer LB et UB;
Calculer p solutions efficaces supportées $X^1, \ldots, X^p$;
($p \leftarrow 2$ pour KP biobjectif);
Définir;
$J_0 \leftarrow \{j \in \{1, \ldots, n\} \,|\, x_j^1 = x_j^2 = \cdots = x_j^p = 0\}$;
$J_1 \leftarrow \{j \in \{1, \ldots, n\} \,|\, x_j^1 = x_j^2 = \cdots = x_j^p = 1\}$;
$m_0 \leftarrow |J_0|, m_1 \leftarrow |J_1|$;
$D_0 \leftarrow \emptyset$;
for $i \leftarrow 1$ ***to*** m_0 **do**
  Calculer $P(E^{(J_0(i))})$;
  if $|P(E^{(J_0(i))})| \geq UB$ **then**
    $D_0 \longleftarrow D_0 \bigcup \{J_0(i)\}$
  end
end
return D_0;
$D_1 \leftarrow \emptyset$;
for $i \leftarrow 1$ ***to*** m_1 **do**
  Calculer $D(E^{(J_1(i))})$;
  if $|D(E^{(J_1(i))})| \geq n - LB$ **then**
    $D_1 \longleftarrow D_1 \bigcup \{J_1(i)\}$
  end
end
return D_1;

6.3.3 Exemple didactique

Cet exemple est présenté expliquant le déroulement de l'algorithme 8 de Jorge et Gandibleux [58, 56]. Bien que l'exemple contienne trois variables régulières la procédure ne détecte aucune de ces variables. Partant de cette remarque, nous proposons de dérouler notre procédure de réduction sur cet exemple montrant l'avantage de l'algorithme 9 et la valeur ajoutée :

Exemple 6.1

$$(BOKP)\begin{cases} \max(Z_1) = 11x_1 + 4x_2 + 5x_3 + 1x_4 + 7x_5 + 13x_6 + 4x_7 + 3x_8 \\ \max(Z_2) = 9x_1 + 8x_2 + 2x_3 + 2x_4 + 16x_5 + 5x_6 + 9x_7 + 4x_8 \\ \qquad\quad 4x_1 + 3x_2 + 2x_3 + 1x_4 + 8x_5 + 7x_6 + 6x_7 + 5x_8 \leq 20 \\ \qquad\quad x_i \in \{0,1\} \end{cases} \tag{6.7}$$

La table 6.1 représente les trois solutions efficaces obtenues en résolvant le problème (BOKP)(6.7) :

i	1	2	3	4	5	6	7	8
X^1	1	0	1	1	0	1	1	0
X^2	1	1	1	1	1	0	0	0
X^3	1	0	0	1	1	1	0	0

TABLE 6.1 – Solutions efficaces

La première, la quatrième et la huitième colonnes sont des indices des variables régulières correspondants à $C_1 = \{1, 4\}$ et $C_0 = \{8\}$.
On appliquons la réduction établi par J.Jorge et X.Gandibleux [57], on obtient :

- Les vecteurs de données :

$$v^1 = \begin{pmatrix} 11 \\ 9 \\ -4 \end{pmatrix},\ v^2 = \begin{pmatrix} 4 \\ 8 \\ -3 \end{pmatrix},\ v^3 = \begin{pmatrix} 5 \\ 2 \\ -2 \end{pmatrix}, v^4 = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix},$$

$$v^5 = \begin{pmatrix} 7 \\ 16 \\ -8 \end{pmatrix},\ v^6 = \begin{pmatrix} 13 \\ 5 \\ -7 \end{pmatrix},\ v^7 = \begin{pmatrix} 4 \\ 9 \\ -6 \end{pmatrix},\ v^8 = \begin{pmatrix} 3 \\ 4 \\ -5 \end{pmatrix}.$$

- La borne inférieure $LB = 2$ et la borne supérieure $UB = 5$.

- $P(v^8) = \{1, 2\}$ et $D(v^1) = \{7, 8\}$ et $D(v^4) = \{\emptyset\}$ et en utilisant les propriétés dans [57], on obtient : $|P(v^8)| \not\geq UB$ et $w_1 + w_2 + w_8 = 13 \not\geq w$, donc on ne peut rien dire sur la valeur de x_8, même si $x_8 = 0$ dans toutes les solutions efficaces, aussi $8 - |D(v^1)| = 6 \not\leq LB$ et $w_1 + w_2 + w_3 + w_4 + w_5 + w_6 \not\leq w$ qui ne peut pas déterminer la variable x_1, même si $x_1 = 1$ dans toutes les solutions efficaces. La même manière pour la valeur x_4, on ne peut pas la déterminer par les propriétés développées dans [57].

En conclusion, l'ensemble d'indices de variables régulières détecté dans [57] est donné par $C_1' = \emptyset$ et $C_0' = \emptyset$.
Maintenant, on applique notre procédure de réduction sur l'exemple 6.1, en parcourant les trois étapes suivantes :

1. On définit l'espace de données E de vecteurs :

$$E^1 = \begin{pmatrix} 2,75 \\ 2,25 \end{pmatrix},\ E^2 = \begin{pmatrix} 1,33 \\ 2,66 \end{pmatrix},\ E^3 = \begin{pmatrix} 2,5 \\ 1 \end{pmatrix}, E^4 = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

$$E^5 = \begin{pmatrix} 0,87 \\ 2 \end{pmatrix},\ E^6 = \begin{pmatrix} 1,85 \\ 0.71 \end{pmatrix},\ E^7 = \begin{pmatrix} 0,66 \\ 1,5 \end{pmatrix},\ E^8 = \begin{pmatrix} 0,6 \\ 0,8 \end{pmatrix}.$$

2. Deux solutions efficaces supportées extrêmes sont calculées par la méthode lexicographique données par l'algorithme 4 :
$X^1 = (1,0,1,1,0,1,1,0)$ et $X^2 = (1,1,1,1,1,0,0,0)$.

3. Les ensembles J_0 et J_1 sont définis (voir : l'equation (6.3) et (6.4)) par :
$J_0 = \{8\}$ et $J_1 = \{1,3,4\}$.
Dans cet exemple, la table 6.2 représente l'ensemble préféré et l'ensemble dominé de chaque objet i en utilisant les définitions 6.4 et 6.5 :

i	$P(E^i)$	$D(E^i)$
E^1	$\emptyset$	$\{3,4,5,6,7,8\}$
E^2	$\emptyset$	$\{4,5,7,8\}$
E^3	$\{1\}$	$\{6,8\}$
E^4	$\{1,2\}$	$\{5,7,8\}$
E^5	$\{1,2,4\}$	$\{7,8\}$
E^6	$\{1,3\}$	$\emptyset$
E^7	$\{1,2,4,5\}$	$\{8\}$
E^8	$\{1,2,3,4,5,7\}$	$\emptyset$

TABLE 6.2 – Ensembles préférés et dominés

L'ensemble dominé par le vecteur E^1 est : $D(E^1) = \{3,4,5,6,7,8\}$, donc $|D(E^1)| \geq n - LB$, d'après l'equation (6.6), on conclut que : $1 \in D_1$ et donc $x_1 = 1$.
L'ensemble dominé par le vecteur E^3 est $D(E^3) = \{6,8\}$, on a $|D(E^3)| \ngeq n - LB$ donc, on ne peut rien dire sur x_3.
L'ensemble dominé par le vecteur E^4 est $D(E^4) = \{5,7,8\}$, comme $|D(E^4)| \ngeq n - LB$ on ne peut rien dire sur x_4.
L'ensemble dominant le vecteur E^8 est $P(E^8) = \{1,2,3,4,5,7\}$, on a $|P(E^8)| \geq UB$, d'après l'équation (6.5) on conclut que : $8 \in D_0$ alors $x_8 = 0$.

Par conséquent, la procédure proposée détecte quelques variables régulières $D_1 = \{1\}$ et $D_0 = \{8\}$, ces dernières n'ont pas été détectées par la procédure développée dans [57].

6.4 Expérimentations Numériques

Les instances du problème de sac à dos prélevées du site :
http ://xgandibleux.free.fr/MOCOlib/instances/MOKP/, sont formées de trois types $1A$, $1B$, $1C$, selon plusieurs caractéristiques précisées par la suite.

D'une part, ces instances sont utilisées pour valider l'algorithme proposé et d'autre part les instances ont été résolues à l'aide d'une méthode exacte. L'ensemble de solutions efficaces permet de calculer C_0 et C_1 définis ci-dessus. Leurs cardinaux servent à borner les résultats pouvant être obtenus à l'aide d'une méthode de réduction et évaluer la qualité des propriétés développées. La première partie de cette section présente les instances utilisées ainsi que les résultats obtenus de l'algorithme proposé sur toutes les instances comparés avec ceux de Jorge et Gandibleux (2008). Puis, la seconde partie présente les résultats obtenus de l'impact de cet algorithme sur le temps d'exécution globale sur quelques instances comparés avec ceux de la méthode de résolution proposée par Jahanshahloo et al. (2005).

6.4.1 La détection de l'ensemble de variables régulières

L'algorithme 9, est implémenté dans un environnement MATLAB 11 sur un PC Intel(R) Pentium(R)M Processor 2.13 GHz. On utilise CPLEX 12.2 pour résoudre le problème de sac à dos biobjectif.
On adopte les notations suivantes :

- $2KPn - r$: le problème de sac à dos biobjectif avec n objets et le rapport de la capacité sur la somme des poids égal à $r \in 10^{-2} \times [11, 91]$.
- $\mid C_E \mid = \mid C_0 \cup C_1 \mid$: le nombre d'indices de variables régulières obtenu par une méthode exacte donnée par le site mentionné auparavant.
- $\mid C_J \mid$: le nombre d'indices de variables régulières donné dans [57].
- $\mid D \mid = \mid D_0 \cup D_1 \mid$: le nombre d'indices de variables régulières donné par notre algorithme.
- T : le temps d'exécution de notre algorithme.
- $R_J = \dfrac{\mid C_J \mid}{\mid C_E \mid} * 100\%$.
- $R_D = \dfrac{\mid D \mid}{\mid C_E \mid} * 100\%$.

- **Résultats des instances du type 1A**
  Ce type contient 5 instances du problème de sac à dos biobjectifs unidimensionnels en variables $\{0, 1\}$. Les vecteurs profits et poids sont uniformément générés. Les instances contiennent entre 50 et 500 objets. Le rapport r appartient à l'intervalle $[0.11, 0.92]$, (voir : table 6.3).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50 - 11$	31	10	28	0	28	0.59
$2KP50 - 50$	20	2	6	2	8	0.71
$2KP50 - 92$	48	28	1	34	35	0.5
$2KP100 - 50$	56	10	8	21	29	2.49
$2KP500 - 41$	inconnu	inconnu	60	10	70	6.97

TABLE 6.3 – Nombre de variables régulières fixées à 0 ou 1 pour $1A$.

D'après la table 6.3, la résolution de l'instance $2KP500 - 41$ fournit 70 variables régulières par contre $\mid C_E \mid$ et $\mid C_J \mid$ sont inconnus.
La différence entre les résultats R_J et R_D en pourcentage est dans l'intervalle 14.58% et 58.06% (voir : fig.6.1).

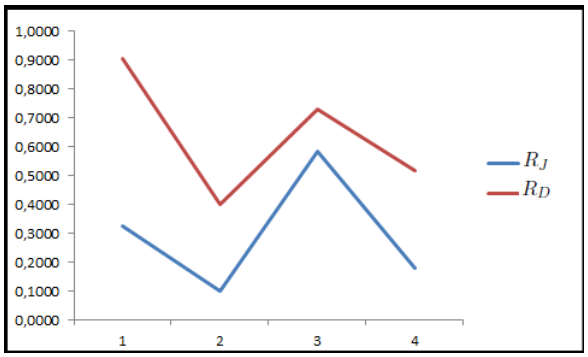


FIGURE 6.1 – Pourcentage des variables régulières pour le type 1A

- **Résultats des instances de type** $1B$
  Ce type contient 27 données correspondent à 09 instances du problème de sac à dos biobjectifs unidimensionnels en variables $\{0, 1\}$. Le rapport r est fixé à 0.5. Trois variantes A, B et C, de 09 instances sont explorées, où les profits et les poids dans les deux premières variantes sont générées uniformément dans l'intervalle $[1, 100]$.

  a. Le type $1B/A$, les vecteurs profits et poids sont générés uniformément dans $[1, 100]$, (voir : table.6.4).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	27	11	6	10	16	0.52
$2KP100$	53	5	5	13	18	0.45
$2KP150$	83	18	14	24	38	0.95
$2KP200$	116	17	15	34	49	1.45
$2KP250$	139	18	15	39	54	2.23
$2KP300$	173	26	17	54	71	2.54
$2KP350$	213	31	21	58	79	3.75
$2KP400$	252	33	28	72	100	4.58
$2KP450$	270	27	35	88	123	5.47

TABLE 6.4 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1B/A$

La différence en pourcentage est aussi significative, elle est entre 18.52% et 35.56% (voir : fig.6.2).

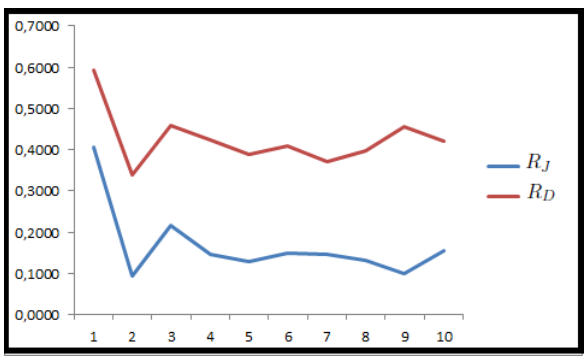


FIGURE 6.2 – Pourcentage des variables régulières pour le type $1B/A$

b. Les instances de type $1B/B$ ont été crées à partir de $1B/A$ en remplaçant le deuxième vecteur profit par le premier dans l'ordre inverse ($c_i^2 = c_{n-i+1}^1,\ \forall i$), (voir : table.6.5).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	24	9	3	9	12	0.26
$2KP100$	51	4	7	14	21	0.52
$2KP150$	78	9	7	23	30	0.87
$2KP200$	108	15	11	25	36	1.26
$2KP250$	145	18	14	38	52	2.03
$2KP300$	180	32	18	51	69	2.23
$2KP350$	203	31	25	51	76	3.82
$2KP400$	249	39	29	64	93	4.87
$2KP450$	249	31	25	77	102	6.10

TABLE 6.5 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1B/B$

Dans cette catégorie, toutes les instances produisent une différence importante entre les rapports R_J et R_D (voir : fig.6.3)

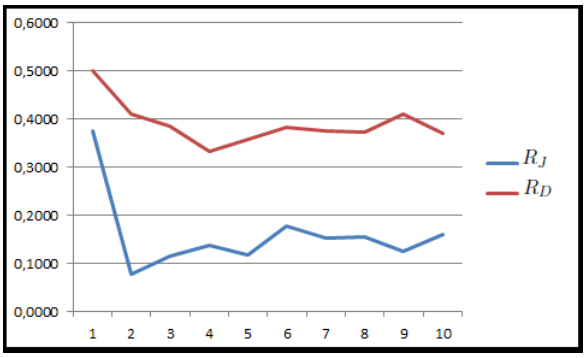


FIGURE 6.3 – Pourcentage des variables régulières pour le type $1B/B$

c. Le type d'instances $1B/C$, les composantes du vecteur profit générées par plateaux de longueurs tirées aléatoirement dans l'intervalle ($[1, \lfloor \frac{n}{10} \rfloor]$), (voir : table 6.6).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	23	3	3	8	11	0.21
$2KP100$	61	11	11	17	28	0.63
$2KP150$	81	10	13	27	40	0.86
$2KP200$	105	19	10	28	38	1.45
$2KP250$	136	15	11	43	54	2.03
$2KP300$	186	37	28	46	74	2.86
$2KP350$	193	29	10	54	64	3.82
$2KP400$	239	47	41	71	112	4.77
$2KP450$	291	27	28	87	115	5.96

TABLE 6.6 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1B/C$

Ce type d'instances présente les mêmes résultats, toutes les instances produisent une différence clair (voir : fig.6.4).

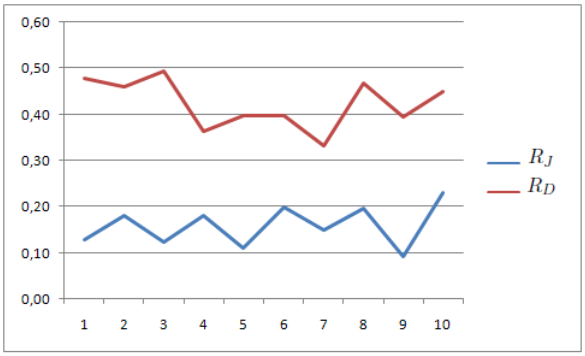


FIGURE 6.4 – Pourcentage des variables régulières pour le type $1B/C$

Les résultats précédents concernant le type d'instances $1B$ conduisent aux conclusions suivantes :

(a) Une grande similarité est observée entre les classes $1B/A$ et $1B/B$.

(b) $D \subset C_E$, sauf la première instance pour la classe $1B/C$. D'une part, notre procédure produit l'indice $31 \in D_1$ mais $31 \notin C_1$ et d'une autre part, toutes les solutions montrent qu'il existe seulement une solution efficace parmi 79 solutions, sa composante x_{31} égale à zéro tandis que toutes les autres solutions efficaces leur composante x_{31} égale à 1, ce qui conduit lors de la résolution du problème de sac à dos biobjectif que notre procédure pourra rater une solution parmi 79 solutions efficaces.

(c) $|C_J| < |D| < |C_E|$, la différence entre $|D|$ et $|C_J|$ est significative.

(d) La moyenne de temps d'exécution est acceptable, entre 0.21 et 6.10 secondes.

(e) Pour toutes les instances, la taille du problème est limitée à 450.

- **Résultats des instances de type 1C**
  Ce type contient 38 données de 3 variantes du problème de sac à dos biobjectifs unidimensionnels en variables $\{0, 1\}$: non corrélées, noté "UNCOR", faiblement corrélées noté "WEAK" et fortement corrélées noté "STRONG". Le rapport est fixé à $r = 0.5$.

  a. UNCOR : cette variante est composée de 20 instances de 50 variables. Les profits et les poids sont générés uniformément ; 10 entre eux sont dans l'intervalle $[1, 300]$ et les autres sont dans l'intervalle $[1, 1000]$, (voir : la table.6.7).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	19	1	3	6	9	0.61
$2KP50$	25	5	7	6	13	0.51
$2KP50$	29	6	5	9	14	0.43
$2KP50$	28	4	5	6	11	0.57
$2KP50$	23	2	2	9	11	0.60
$2KP50$	26	4	5	7	12	0.43
$2KP50$	25	3	1	8	9	0.61
$2KP50$	27	5	6	8	14	0.76
$2KP50$	31	8	4	10	14	0.76
$2KP50$	27	6	5	8	13	0.55
$2KP50$	24	4	3	$\underline{8}$	11	0.55
$2KP50$	30	9	5	9	14	0.49
$2KP50$	30	9	$\underline{3}$	8	11	0.57
$2KP50$	25	5	6	5	11	0.55
$2KP50$	22	7	4	10	14	0.52
$2KP50$	25	3	6	6	12	0.42
$2KP50$	21	5	2	8	10	0.59
$2KP50$	33	12	8	10	18	0.60
$2KP50$	28	2	3	5	8	0.53
$2KP50$	22	4	2	10	12	0.77

TABLE 6.7 – Nombre de variables régulières fixées à 0 ou 1 pour le type UNCOR

Concernant ce type d'instances, les mêmes résultats sont obtenus (voir : fig.6.5).

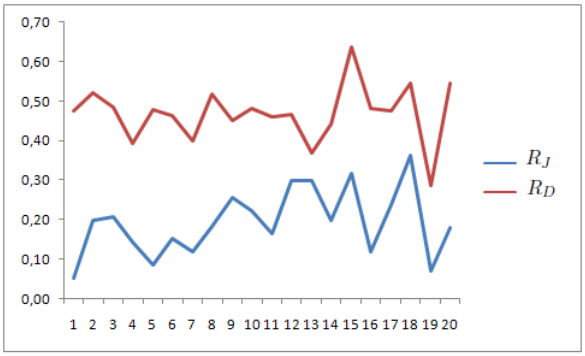


FIGURE 6.5 – Pourcentage des variables régulières pour le type UNCOR

b. WEAK : 09 instances corrélées de taille entre 50 et 450. Les vecteurs poids sont générés uniformément dans [1,1000]. Le deuxième vecteur profit prend ces valeurs dans [111,1000]. Le premier vecteur profit est choisi aléatoirement dans l'intervalle $[c_i^2 - 100, c_i^2 + 100]$, (voir : la table.6.8).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	46	26	14	12	26	0.58
$2KP100$	100	40	28	27	55	1.17
$2KP150$	139	51	40	39	79	1.42
$2KP200$	197	64	55	53	108	1.97
$2KP250$	237	77	65	66	131	2.77
$2KP300$	281	94	79	79	158	3.88
$2KP350$	328	116	92	92	184	4.42
$2KP400$	373	133	104	107	211	6.69
$2KP450$	420	146	118	117	235	6.50

TABLE 6.8 – Nombre de variables régulières fixées à 0 ou 1 pour le type WEAK

Concernant le type d'instances faiblement corrélées, l'algorithme proposé produit plus que la moitié de l'ensemble exacte de variables régulières, ce qui conduit à réduire la taille du problème à la moitié (voir : fig.6.6).

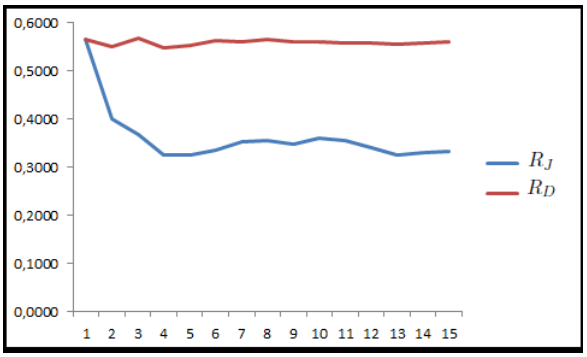


FIGURE 6.6 – Pourcentage des variables régulières pour le type WEAK

c. STRONG : 09 instances fortement corrélées d'une taille entre 50 et 450 sont utilisées. Les poids sont générés uniformément dans $[1, 1000]$. Le deuxième vecteur profit prend ces valeurs dans l'intervalle $[1, 1000]$. Le premier vecteur profit égal à $w_j + 100$, (voir : table.6.9).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	29	–	4	10	14	0.37
$2KP100$	69	–	8	21	29	0.75
$2KP150$	100	–	12	33	45	1.20
$2KP200$	140	–	16	44	60	1.87
$2KP250$	180	–	19	53	72	2.09
$2KP300$	212	–	23	63	86	3.27
$2KP350$	–	–	31	76	107	3.53
$2KP400$	–	–	34	85	119	4.87
$2KP450$	–	–	40	92	132	5.92

TABLE 6.9 – Nombre de variables régulières fixées à 0 ou 1 pour le type STRONG

Pour les instances fortement corrélées, on note que notre approche produit un ensemble de variables régulières mais pour les autres méthodes, à partir de la sixième instance ($2KP350 - - > 2KP450$), les résultats sont inconnus (voir : fig.6.7).

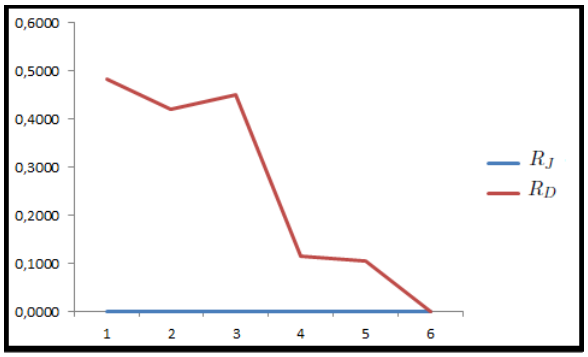


FIGURE 6.7 – Pourcentage des variables régulières pour le type STRONG

Finalement, concernant le type d'instances $1C$, nos résultats se résument comme suit :

a. $D \subset C_E$, sauf le type d'instances UNCOR pour la 11ème et 13ème instances, il existe seulement un indice $39 \in D_1$, (resp.$19 \in D_0$, mais $39 \notin C_1$, (resp. $19 \notin C_0$).
   Par exemple, l'indice $39 \in D_1$ mais $39 \notin C_1$, la composante $x_{39} = 1$ apparaît dans la majorité des solutions efficaces, et $x_{39} = 0$ apparaît dans une solution parmi tout l'ensemble efficace.

b. $|C_J| < |D| < |C_E|$.

c. Pour les instances de type UNCOR et les instances de type WEAK, la moyenne de temps d'exécution est acceptable entre 0.42 et 0.77 secondes.

d. Pour les instances de type STRONG, notre approche détecte un ensemble de variables régulières D pour toutes les instances, par contre, la réduction de Jorge et Gandibleux (2008) ne détecte aucune variable régulière. La moyenne du temps d'exécution est acceptable entre 0.37 et 5.92 secondes.

Remarque 6.2 Nous avons utilisé dans un premier temps Matlab 11 pour résoudre le problème de sac à dos biobjectif, on a remarqué que : (voir : Annexe A)

- Le temps d'exécution pour résoudre le problème de sac à dos biobjectif est très élevé, comparant avec celui du Cplex 12.2.
- Les instances ont été résolues pour une taille plus grande, on est arrivé jusqu'à $n = 1000$ pour l'instance WEAK.
- Pour le type $1A$, l'instance de taille 500 a consommé plus d'une heure en temps d'exécution.

6.4.2 L'impact des variables régulières sur le temps de résolution

On adopte les notations suivantes :

- $\mid D \mid = \mid D_0 \cup D_1 \mid$: le nombre d'indices correspond aux variables régulières produites par notre méthode de réduction.
- N_R : la taille du problème résiduel est calculée : $N_R = n - \mid D \mid$.
- T_E et M_E sont respectivement le temps d'exécution et le nombre de solutions efficaces de la méthode exacte développée dans [54].
- T_R et M_R sont respectivement le temps d'exécution et le nombre de solutions efficaces de notre procédure de réduction adaptée à la méthode exacte [54].

– $T = \frac{T_R}{T_E} * 100\%$: le pourcentage de temps d'exécution de la procédure de réduction adaptée à la méthode exacte T_R et le temps d'exécution de la même méthode exacte sans cette réduction T_E.

a. **Résultats des instances de type 1A**

Instances	Méthode exacte [54]		Méthode de réduction [22]				
	M_E	$T_E(s)$	$\mid D \mid$	N_R	M_R	$T_R(s)$	$T\%$
$2KP50-11$	43	85.47	28	22	42	51.30	60.02
$2KP50-50$	51	408.70	8	42	51	399.35	97.71
$2KP50-92$	2	0.48	35	15	2	0.48	100
$2KP100-50$	$*$	$*$	29	71	$*$	$*$	$*$

TABLE 6.10 – Différence de temps d'exécution entre la méthode exacte et la méthode de réduction pour le type $1A$.

On note que pour les instances de type $1A$ (voir : table 6.10), le temps d'exécution de notre méthode de réduction [22] est meilleure que le temps consommé par la méthode exacte [54]. Dans la cas où, le problème admet deux solutions efficaces, le temps d'exécution est le même dans les deux méthodes (exemple $2KP50-92$) mais, si le problème admet un nombre important de solutions efficaces, un effort calculatoire est exigé pour cette méthode itérative même si la taille du problème est réduite (exemple : $2KP100-50$, $n=100 -> N_R=71$). Pour cette dernière raison, la ligne de cette instance est vide $(*)$.

b. **Résultats des instances de type 1C**

On note que pour toutes les instances de type UNCOR, (voir : table 6.11), le temps d'exécution fournit par notre méthode [22] est meilleur que celui de la méthode exacte [54]. Le pourcentage de temps d'exécution T est significatif entre 65.30% et 97.66%.

Instances	Méthode exacte [54]		Méthode de réduction [22]				
	M_E	$T_E(s)$	$\mid D \mid$	N_R	M_R	$T_R(s)$	$T\%$
$2KP50$	59	425.40	9	41	59	399.03	93.80
$2KP50$	37	106.85	13	37	37	93.25	80.64
$2KP50$	39	115.63	14	36	39	112.93	97.66
$2KP50$	34	62.74	11	39	34	54.39	86.69
$2KP50$	40	186.25	11	39	40	127.86	68.64
$2KP50$	40	107.75	12	38	40	88.84	82.45
$2KP50$	38	119.76	9	41	38	103.14	86.12
$2KP50$	30	92.50	14	36	30	78.15	84.48
$2KP50$	33	50.10	14	36	33	37.09	74.03
$2KP50$	44	140.99	13	37	44	133.16	94.44
$2KP50$	56	353.51	11	39	56	336.58	95.21
$2KP50$	49	193.05	14	36	49	126.07	65.30
$2KP50$	27	21.93	11	39	26	18.44	84.08
$2KP50$	47	182.67	11	39	47	136.94	74.96
$2KP50$	65	766.56	14	36	65	678.10	88.46
$2KP50$	45	153.90	12	38	45	122.49	79.59
$2KP50$	62	679.80	10	40	62	460.92	67.80
$2KP50$	19	13.94	18	32	19	8.59	61.62
$2KP50$	34	63.52	8	42	34	44.66	70.30
$2KP50$	80	1736.94	12	38	80	1337.3	76.99

TABLE 6.11 – Différence de temps d'exécution entre la méthode exacte et la méthode de réduction pour le type UNCOR.

c. **Résultats des instances de type WEAK**

Instances	Méthode exacte [54]		Méthode de réduction [22]				
	M_E	$T_E(s)$	$\mid D \mid$	N_R	M_R	$T_R(s)$	$T\%$
$2KP50$	3	8.79	26	24	2	0.58	6.59
$2KP100$	1	1.17	55	45	1	1.17	100
$2KP150$	8	5.43	79	71	8	4.00	73.66
$2KP200$	2	1.97	108	92	2	1.97	100
$2KP250$	7	10.98	122	118	7	9.97	90.80
$2KP300$	16	81.53	158	142	16	59.07	72.45
$2KP350$	25	193.87	184	166	25	106.65	55.01
$2KP400$	25	344.19	211	189	24	276.46	80.32
$2KP450$	–	–	235	215	35	873.71	–

TABLE 6.12 – Différence de temps d'exécution entre la méthode exacte et la méthode de réduction pour le type WEAK.

On note que pour les instances de type WEAK, (voir : table 6.12), le temps d'exécution de notre méthode de réduction est meilleur que celui de la méthode exacte pour la majorité des instances à part les instances $2KP100$ et $2KP200$ dont le problème admet respectivement 1 et 2 solutions efficaces. Pour la dernière instance $2KP450$, notre méthode de réduction produit un ensemble de solutions efficaces, par contre la méthode exacte n'a détecté aucun ensemble à cause de la version académique du (cplex12.6) qui est limitée par la grandeur des valeurs (Profits et Poids).

En conclusion, les tables (6.10), (6.11) et (6.12), montrent l'effet de la stratégie de réduction sur le temps d'exécution globale.

Pour toutes les instances, le signe " $-$ " signifie que la méthode ne peut pas produire des résultats et le signe " $*$ " signifie que la méthode produit une grande énumération, dans ce cas, pas de résultats.

Pour analyser la performance de la stratégie de réduction développée, on a utilisé les instances citées antérieurement correspondantes à différentes classes :

- L'algorithme de réduction est particulièrement efficace pour toutes les instances de type $1B/A$, $1B/B$, WEAK et STRONG.
- Pour quelques instances, où les données dépendent de la taille du problème, par exemple, quelques instances de type $1B/C$ et UNCOR, l'ensemble d'indices de variables régulières D_0 (resp. D_1) est inclus dans C_0 (resp.C_1), mais il existe exactement un indice inclus dans D_0 (resp. D_1), mais il n'est pas inclus dans C_0 (resp. C_1), ce qui peut conduire à rater un nombre négligeable de solutions efficaces comparant au nombre important de solutions efficaces.

Pour ce type d'instances, on procède de la manière suivante :
Basé sur la relation de dominance au sens de Pareto utilisée pour l'efficacité d'objet, on trie les objets dans un ordre croissant selon $|P(E^i)|$ (ou dans un ordre décroissant selon $|D(E^i)|$) comme suit :

a. Pour $i \in J_0$, les objets dominés par **un grand nombre** d'objets (plus que UB objets) (en terme d'efficacité d'objet) sont groupés à la fin, où les rapports $E^i,\ i = 1, \cdots, n$ sont trop petits "les objets d'une efficacité faible" dont les valeurs correspondantes $x_i = 0$ dans toutes les solutions efficaces.

b. Pour $i \in J_1$, les objets dominants **un grand nombre** d'objets (plus que $n - LB$ objets) (en terme d'efficacité d'objet) sont groupés au début, où les rapports $E^i,\ i = 1, \cdots, n$ sont importants "les objects d'une efficacité forte" et les valeurs correspondantes $x_i = 1$ dans toutes les solutions efficaces.

Pour assurer **le grand nombre**, une certaine valeur u est ajoutée dans les deux derniers cas de la manière suivante :

- Généralement, les valeurs exactes des seuils pour lesquels on pourra détecter un ensemble de variables régulières D_0 et D_1 ne peuvent pas être identifiées a priori, mais une bonne approximation de ces valeurs est UB (resp.$n - LB$), c'est la raison pour laquelle on pourra rater quelques fois un nombre négligeable (1 ou 2) solutions parmi un nombre important de solutions efficaces. On essaye de remédier ce cas en ajoutant aux seuils une certaine valeur additif u.

Les ensembles D_0 et D_1 sont calculés de la manière suivante :

$$D_0 \ = \ \{i \in J_0 \ | \ |P(E^i)| \geq UB + u\} \tag{6.8}$$

$$D_1 \ = \ \{i \in J_1 \ | \ |D(E^i)| \geq n - LB + u\} \tag{6.9}$$

où, u est un nombre entier positif.

Proposition 6.1 $u \in [0, \min(LB-1, n-UB-1)]$.

Preuve :
Le nombre maximum d'indices j (resp. le nombre minimum) tel que E^j domine E^i au sens de Pareto est $n-1$ (resp.0). Dans ce cas on peut écrire :
$0 \leq |P(E^i)| \leq n-1$, et de l'equation (6.8), on a $|P(E^i)| \geq UB + u$,
donc, $UB + u \leq |P(E^i)| \leq n-1 \Longrightarrow UB + u \leq n-1$ alors, on déduit :

$$u \leq n - UB - 1 \tag{6.10}$$

Le nombre maximum d'indices j (resp. le nombre minimum) tel que E^j est dominé par E^i au sens de Pareto est $n-1$ (resp.0). Dans ce cas on peut écrire :
$0 \leq |D(E^i)| \leq n-1$ et de (6.9) on a : $|D(E^i)| \geq n - LB + u$,
donc, $n - LB + u \leq |D(E^i)| \leq n-1 \Longrightarrow n - LB + u \leq n-1$ alors, on déduit :

$$u \leq LB - 1 \tag{6.11}$$

De l'equation (6.10) et (6.11), on conclut

$$u \leq \min(LB-1, n-UB-1)$$

ce qui implique :

$$u \in [0, \min(LB-1, n-UB-1)].$$

Exemple 6.2 On présente un exemple didactique dont la procédure de réduction pourra rater une solution efficace :

$$(BOKP) \begin{cases} \max(Z_1) = 11x_1 + 5x_2 + 7x_3 + 13x_4 + 3x_5 \\ \max(Z_2) = 9x_1 + 2x_2 + 16x_3 + 5x_4 + 4x_5 \\ \qquad\quad 4x_1 + 2x_2 + 8x_3 + 7x_4 + 5x_5 \leq 16 \\ \qquad\quad x_i \in \{0,1\} \end{cases} \tag{6.12}$$

Les solutions efficaces sont listées dans la table 6.13 :

i	1	2	3	4	5
X^1	1	1	0	1	0
X^2	1	1	1	0	0
X^3	1	0	0	1	1

TABLE 6.13 – Solutions efficaces

Notre procédure de réduction détecte deux indices de variables régulières $D_1 = \{1\}$ et $D_0 = \{5\}$. Ainsi, la résolution du problème résiduel fournit deux solutions efficaces dont les extensions sont X^1 et X^2, ce qui conduit à rater la solution efficace X^3, car son cinquième indice égal à 1.

Remarque 6.3 Si on applique la proposition 6.1, on récupère la solution ratée.

6.5 Conclusion

Dans ce chapitre, un nouvel algorithme réduisant la taille du problème de sac à dos biobjectif en variables binaires a été présenté. Ce prétraitement est basé sur l'information de données pour fixer quelques variables à 0 ou 1 avant la résolution du problème considéré. Pour étudier l'efficacité de la procédure développée, nous avons comparé nos résultats avec ceux de Jorge et Gandibleux [57]. Des expérimentations numériques montrent que la procédure de réduction est utile pour toutes les instances étudiées $1A$, $1B$ et $1C$ (voir : les tables 6.3-6.9). La conséquence est que la taille du problème considéré est réduite (la moitié ou plus par exemple pour l'instance "STRONG"), ce qui a fait une diminution du temps de calcul nécessaire pour résoudre les instances $1A$ et $1C$: "STRONG et WEAK" (voir : les tables 6.10-6.12). On a remarqué aussi que pour toutes les instances étudiées, les conditions utilisant les bornes LB et UB dans [57] ne fixent que rarement des variables régulières, par contre pour la méthode développée leur utilisation détermine plusieurs variables régulières. Pour les instances de type "$1B/C$" et "UNCOR", il existe des cas inadaptés pour notre algorithme exemple 6.2, la borne inférieure LB et supérieure UB sont utiles pour la détermination de la valeur exacte du nombre additionnel u pour quelques instances de ce type d'instances. Cette observation encourage la réalisation d'une étude plus approfondie sur des approches fixant des variables régulières a priori.

Conclusion générale et Perspectives

Les procédures de réduction sont populaires dans les problèmes d'optimisation combinatoire monoobjectifs mais presque absentes en cas multiobjectif. Leur but est d'explorer de manière efficace l'espace de recherche et permettent de résoudre un problème dans un temps acceptable. Nos études ont été dédiées au problème de sac à dos multiobjectif en variables binaires $\{0, 1\}$, ce dernier est classique dans l'optimisation combinatoire, présent comme sous problème dans de nombreux problèmes d'optimisation combinatoire.

Après l'introduction de la thèse, dans **le premier chapitre**, nous avons exposé la formulation du problème de sac à dos uniobjectif et ses extensions, et quelques méthodes de résolution, en mettant le point sur la notion du core proposée par Balas et Zemel [4] dans le cas unidimensionnel ensuite les travaux établis par Puchinger et al. [84] dans sa version multidimensionnel.
L'optimisation multiobjectif est présentée dans **le deuxième chapitre**, quelques notions de base concernant la dominance au sens de Pareto, l'ensemble de solutions efficaces, et enfin les méthodes de résolution classiques. La notion de solutions supportées et non supportées a été aussi mise en évidence pour le cas multiobjectif en nombre entier.
Tandis que **le troisième et le quatrième chapitre**, ont été réalisés pour un état d'art des différents types de problèmes d'optimisation combinatoire (MOCO) et les méthodes de résolution dédiées à ce genre de problème, nous avons vu que les méthodes exactes ne sont pas faciles à appliquer et qu'ils ne peuvent être utilisés que pour résoudre les problèmes MOCO de petite taille. Ensuite nous avons présenté le problème de sac à dos multiobjectif (MOKP) et quelques méthodes de résolution les plus étudiées en mettant le point sur la méthode de Jahanshahloo et al. (2005) utilisée dans nos travaux.

Le cinquième chapitre a été autour des procédures de réduction, commençant par les travaux de Gomes et al. (2008) concernant le problème de sac à dos biobjectif ensuite les travaux de Mavrotas et al. (2009) pour le cas multiobjectif multidimensionnel. En fin de ce chapitre, nous avons exposé les travaux de Jorge et Gandibleux (2008) pour le problème de sac à dos biobjectif suivi d'un exemple didactique. Bien que les résultats établis par Jorge et Gandibleux (2008) montrent que la procédure de réduction est utile pour les instances où les coefficients sont indépendants, en particulier pour le cas biobjectif, de nombreuses variables n'obéissent pas aux propriétés développées même si elles sont régulières après calcul de l'ensemble complet des solutions efficaces. De plus, la procédure de réduction n'est pas adaptée aux instances où les coefficients d'un objectif et de la contrainte sont corrélés. Ce qui nous a motivé à viser deux objectifs dans cette thèse :

Dans **le sixième et le dernier chapitre**, nous avons développé une méthode de réduction appliquée en prétraitement en exploitant la recherche de deux solutions efficaces supportées extrêmes combinées avec la notion de dominance au sens de Pareto sur l'efficacité d'objet. Nos études s'adressent au problème de sac à dos multiobjectif unidimensionnel en variables binaires.

Dans la deuxième partie de nos travaux, nous avons intégré la réduction développée antérieurement dans la méthode exacte de Jahanshahloo et al. (2005), on a mis en oeuvre la résolution du problème réduit qu'il nous a permis de résoudre le problème dans un temps acceptable.

Les résultats obtenus sont comparés aux résultats existants dans la littérature issus de site http ://xgandibleux.free.fr/MOCOlib/instances/MOKP/ confirment l'efficacité de l'algorithme proposé.

Enfin, les résultats portants sur les procédures de réduction appliquées en prétraitement sont encourageants. De telles procédures permettent de réduire la taille du problème de sac à dos multiobjectif et donc de le résoudre dans un temps raisonnable si le temps de leur application est acceptable. Une procédure de réduction est performante lorsqu'elle est applicable en un temps raisonnable, ce qui n'est pas immédiat lorsqu'un problème quelconque se présente.

Les perspectives de ces travaux sont nombreuses, on évoquera prochainement les suivantes :

- Étendre cette étude au problème de sac à dos multidimensionnel.
- Prendre en considération plus de deux objectifs.
- Intégrer cette réduction dans des méthodes approximatives ou méta-heuristiques en prenant en compte des solutions optimales des deux objectifs et pas forcément des solutions efficaces extrêmes.
- Le cas déterministe sera investi pour l'environnement stochastique.

Bibliographie

[1] M. Abbas, D. Chaabane, An algorithm for solving multiple objective integer linear programming problem, RAIRO 36 : 351-364, 2002.

[2] M. Abbas, D. Chaabane, Optimizing a Linear Function over an Integer Efficient Set, European Journal of Operational Research 174 : 1140-1161, 2006.

[3] Y. P. Aneja et K. P. K. Nair. Bicriteria transportation problem. Management Science, 25 : 73-78, 1979.

[4] E.Balas, E.Zemel. An algorithm for large zero-one knapsack problems.Operational Research, 28 : 1130-1154, 1980.

[5] C. Bazgan, H. Hugot, and D. Vanderpooten. Solving efficiently the $\{0-1\}$ multi-objective knapsack problem. Computers & Operations Research, 36(1) : 260-279, 2009.

[6] C. Bazgan, H. Hugot, and D. Vanderpooten. Implementing an efficient FPTAS for the $\{0-1\}$ multi-objective knapsack problem. European Journal of Operational Research, 198(1) : 47-56, 2009.

[7] C. Bazgan, H. Hugot et D. Vanderpooten. A Practical Efficient FPTAS for the 0-1 Multiobjective Knapsack Problem, volume : 4698 de Lecture Notes in Computer Science, pages 717-728. Springer Berlin/Heidelberg, septembre 2007.

[8] F. Ben Abdelaziz and S. Krichen. A tabu search heuristic for multiobjective knapsack problems. Technical Report RRR 28-97, Rutgers Center for Operations Research, Piscataway, USA, 1997.

[9] H. P. Benson. Existence of Efficient Solutions for Vector Maximization Problems. Journal of Optimization Theory and Applications, 26(4) : 569-580, 1978.

[10] V.Boyer. Contribution à la programmation en nombres entiers. Thèse de doctorat, Université de Toulouse, 2007.

[11] V.J. Bowman. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thiriez and S. Zionts, editors, Multiple criteria decision making, pages 76-86. Springer. Lecture Notes in Economics and Mathematical Systems, Berlin, 1976.

[12] M. E. Captivo, J.C.N. Climaco, J.R. Figueira, E.Q.V. Martins, and J. L. Santos. Solving bicriteria $0,1$ knapsack problems using a labeling algorithm. Computers & Operations Research, 30(12) : 1865-1886, 2003.

[13] D. Chaabane and M. Daoud. Data Pre-treatment for Solving bi-objective Knapsack Problem, The 23rd International Conference on Multiple Criteria Decision Making (MCDM'15), Hamburg, Germany, 2nd-7th August, 2015.

[14] D. Chaabane, B. Brahmi and Z. Ramdani. The augmented weighted Tchebychev norm for optimizing a linear function over an integer efficient set of a multicriteria linear program. Intl. Trans. in Op. Res. 19 : 531-545, 2012.

[15] D. Chaabane and M.Pirlot. A method for optimizing over the integer efficient set. Journal of Industrial and Management Optimization. volume 6, Number 4, 811-823, 2010.

[16] D. Chaabane. Contribution à l'Optimisation Multicritère en Variables Discrètes : Dissertation soumise à la Faculté Polytechnique de Mons, en vue l'obtention du titre de Docteur en Sciences Appliquées, 2007.

[17] D. Chaabane. Optimisation Multicritère en nombres entiers. Thèse de doctorat, U.S.T.H.B, Faculté de Mathématiques, 2005.

[18] G. Christian, R. Christian et S. Heinz. Multiobjective Disk Cover Admits a PTAS, volume : 5369 de Lecture Notes in Computer Science, pages 40-51. Springer Berlin/Heidelberg, décembre 2008.

[19] Y. Collette, P. Siarry. Optimisation Multiobjectif. Groupe Eyrolles, 2002.

[20] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing-a metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis, 7 : 34-47, 1998.

[21] G. Dantzig. Discrete variable extremum problems. Operations Research, 5 : 226-227, 1957.

[22] M. Daoud and D. Chaabane. New reduction strategy in the biobjective knapsack problem. Intl. Trans. in Op. Res. 00, 1-24, 2016.

[23] M. Daoud and D. Chaabane. Effect of Reduction Strategy in the Binary Bi-Objective Knapsack Problem, 11th International Conference on Multiple Objective Programming and Goal Programming (MOPGP'15), Tlemcen, Algeria, 13-15 December, 2015.

[24] 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.

[25] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. Computers & Operations Research, 34 : 2674-2694, 2007.

[26] M. Ehrgott. Multicriteria Optimization. Second edition. Springer, Berlin, 2005.

[27] M. Ehrgott et X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. TOP, Vol.12, No.1, pp 1-89, 2004.

[28] M. Ehrgott and X. Gandibleux. Multiple Criteria Optimization : State of the Art Annotated Bibliographic Surveys, volume 52, pages 369-444. Kluwer Academic Publishers, Boston, MA, 2002.

[29] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum, 22 : 425-460, 2000.

[30] T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multiobjective knapsack problems. Management Science, 48(12) : 1603-1612, 2002.

[31] Fabien Degoutin et Xavier Gandibleux : Un retour d'experiences sur la résolution de problèmes combinatoires bi-objectifs. Journée Programmation Mathématiques Multiobjectifs, 2002.

[32] J.R. Figuera, M. Wiecek, and G. Tavares. Multiple criteria knapsack problems : Network models and computational results. In Proceedings of the multi-Objective Programming and Goal Programming Conference (MOPGP'06), Tours, 2006.

[33] C.Filippi, E.Stevanato. A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits. Algorithmic Operations Research Vol.7 : 125-139, 2013.

[34] M. L. Fisher, "The Lagrangean Relaxation Method for Solving Integer Programming Problems", Management Science, vol. 27, 1981, pages 1-18.

[35] A. Fréville, The multidimensional 0-1 knapsack problem : an overview, European Journal of Operational Research 155, 1-21, 2004.

[36] A. Fréville and G. Plateau. An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. Discrete Applied Mathematics, 49 : 189-212,1994.

[37] M. Fourman. Compaction of symbolic layout using genetic algorithms. In First International Conference on Genetic Algorithm, pages 141–153, 1985.

[38] X. Gandibleux, J. Jorge, S. Martinez et N. Sauer. Premier retour d'expérience sur le flowshop biobjectif et hybride à deux étages avec une contrainte de blocage particulière. Avril 2006. 6ème Conférence Francophone de Modélisation et Simulation MOSIM'06.

[39] X. Gandibleux and A. Freville. Tabu Search Based Procedure for Solving the 0-1 Multi-Objective Knapsack Problem : The Two Objectives Case. Journal of Heuristics, 6(3) : 361–383, August 2000.

[40] A. Geoffrion, "The Lagrangean Relaxation for Integer Programming", Mathematical Programming Study, vol. 2, pages 82-114, 1974.

[41] A. M. Geoffrion. Proper Efficiency and the theory of Vector Maximization. Journal of Mathematical Analysis and Applications, 22 : 618-630, 1968.

[42] F. Glover and M. Laguna. Tabu Search. Kluwer Academic, Dordrecht, 1997.

[43] F.Glover. A multiphase-dual algorithm for the zero-one integer programming problem. Operations Research, 13(6) : 879-919, 1965.

[44] C. Gomes da Silva, J. Climaco, and J. R. Figueira. Core problems in bi-criteria $\{0, 1\}$-knapsack problems. Computers & Operations Research, 35(1) : 2292-2306, 2008.

[45] C. Gomes da Silva, J. R. Figueira, and J. Climaco. Integrating partial optimization with scatter search for solving bi-criteria {0,1}-knapsack problems. European Journal of Operational Research, 177 : 1656-1677, 2007.

[46] C. Gomes da Silva, J. Climaco, and J. R. Figueira. A scatter search method for bi-criteria $\{0, 1\}$-knapsack problems. European Journal of Operational Research, 169 : 373-391, 2006.

[47] C. Gomes da Silva, J. Climaco, and J. R. Figueira. Scatter search method for the bicriteria multi-dimensional $\{0, 1\}$-knapsack problem using surrogate relaxation. Journal of Mathematical Modelling and Algorithms, 3(3) : 183-208, 2004.

[48] R. E. Gomory, "Recent Advances in Mathematical Programming", Chapitre An algorithm for integer solution to linear programs, pages 269-302, McGraw-Hill, 1963.

[49] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs", Bulletin of the American Mathematical Society, vol. 64, pages 275-278, 1958.

[50] Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man, and Cybernetics, 1 : 296-297, 1971.

[51] P. Hansen. Bicriterion path problems. Lecture Notes in Economics and Mathematical Systems, 177 : 109-127, 1979.

[52] J.H. Holland. Adaptation in natural and artificial systems. PhD thesis, University of Michigan Press, 1975.

[53] G. P. Ingagorlia et J. F. Korsh. Reduction algorithm for zero-one single knapsack problem. Management Science, 20 : 460-463, 1973.

[54] G.R. Jahanshahloo, F. Hosseinzadeh, N. Shoja, G. Tohidi. A method for generating all efficient solutions of $0-1$ multi-objective linear programming problem, Applied Mathematical Computations 169 874-886, 2005.

[55] F. Jolai, M.J. Rezaee, M. Rabbani, J. Razmi, and P. Fattahi. Exact algorithm for biobjective $0-1$ knapsack problem. Applied Mathematics and Computation, 194(2) : 544-551, 2007.

[56] J. Jorge, Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires. Thèse de Doctorat de l'Université de Nantes, 2010.

[57] J.Jorge, X.Gandibleux, and M.Wiecek. A Priori Reduction of the Size of the Binary Multiobjective Knapsack Problem, MOPGP'08 Multi-Objective Programming and Goal Programming, 2008.

[58] J. Jorge and X. Gandibleux. Nouvelles propositions pour la résolution exacte du problème de sac à dos bi-objectif unidimensionnel en variables binaires, février 2007. FRANCORO V-MOSIM'07.

[59] N, Jozefowiez. Habilitation à diriger les recherches, Optimisation combinatoire multi-objectif :des méthodes aux problèmes, de la Terre à (presque) la Lune, 2013.

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

[61] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Science, 220 : 671-680, 1983.

[62] G. Kirlik, S.Sayin. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. European Journal of Operational Research 232 : 479-488, 2014.

[63] K. Klamroth et M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. Naval Research Logistics, 47 : 57–76, 2000.

[64] R. Kumar and N. Banerjee. Analysis of a multiobjective evolutionary algorithm on the $\{0-1\}$ knapsack problem. Theoretical Computer Science, 358(1) : 104-120, 2006.

[65] M. E. Lalami. Contribution à la résolution de problèmes d'optimisation combinatoire : méthodes séquentielles et parallèles. Thèse de doctorat, Université de Toulouse III, 2012.

[66] T. Lust and J.Teghem. The multiobjective multidimensional knapsack problem :a survey and a new approach.Intl. Trans. in Op. Res. 00, 1-26, 2012.

[67] T. Lust. New metaheuristics for solving MOCO problems : application to the knapsack problem, the traveling salesman problem and IMRT optimization. Thèse de doctorat, Université de Mons, 2010.

[68] R. Malhotra, H.L. Bhatia and M.C.Puri. Bi-criteria assignement problem. Opsearch, volume 19, Number 2, 1982.

[69] S. Martello, P. Toth. "Upper bounds and algorithms for hard $\{0-1\}$ knapsack problems". Operations Research, Vol. 45, pp. 768-778, 1997.

[70] S. Martello and P. Toth. Knapsack Problems : Algorithms and Computer Implementations.New York :Wiley ; 1990.

[71] S. Martello et P. Toth. A new algorithm for the $\{0-1\}$ knapsack problem. Management Science, 34(5) : 633-644, 1988.

[72] S. Martello and P. Toth. "An upper bound for the zero-one knapsack problem and a branch and bound algorithm". European Journal of Operational Research, 1 : 169-175, 1977.

[73] G. Mavrotas, J. R. Figuera et K. Florios. Solving the bi-objective multidimensional knapsack problem exploiting the concept of core. Applied Mathematics and Computation, 215(7) : 2502-2514, Décembre 2009.

[74] 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.

[75] O. Ozpeynirci. Approaches for multiobjective combinatorial optimization problems. PhD thesis, University of Economics, Department of Logistics Management, Izmir, Turkey, 2008.

[76] D. Pisinger. Where are the hard knapsack problems ? Computers & Operations Research, 32(9) : 2271-2284, 2005.

[77] D. Pisinger. A minimal algorithm for the $\{0-1\}$ knapsack problem. Operations Research, 45 : 758-767, 1997.

[78] D. Pisinger. An expanding-core algorithm for the exact 0-1 knapsack problem. European Journal of Operational Research, 87 : 175-187, 1995.

[79] D.Pisinger. Algorithms for Knapsack Problems. Ph.D, Thesis February, 1995.

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

[81] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two recursive algorithms for finding all nondominated extreme points in the outcome set of multi-objective integer programme. INFORMS Journal on Computing, in press, 21(4), 2009.

[82] 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, 2008.

[83] A.Przybylski. Thèse de doctorat, Méthode en deux phases pour la résolution exacte de problèmes d'optimization combinatoire comportant plusieurs objectives :nouveaux développements et application au problème d'affectation linéaire. Université de Nantes, 2006.

[84] J.Puchinger, G.R.Raidl, U.Pferschy. The core concept for the multidimensional knapsack problem. In Gottlieb, J., Raidl, G.R. (eds) EvoCOP, Vol. 3906 of Lecture Notes in Computer Science. Springer, Berlin, pp. 195-208, 2006.

[85] M. Rosenblatt and Z. Sinuany-Stern. Generating the discrete efficient frontier to the capital budgeting problem, Operations Research, Vol. 37(3), pp. 384-394, 1989.

[86] A. Rong and J. R. Figueira. A reduction dynamic programming algorithm for the bi-objective integer knapsack problem.European Journal of Operational Research 231 : 299-313, 2013.

[87] A. Rong and J. R. Figueira. Dynamic programming algorithms for the bi-objective integer knapsack problem. European Journal of Operational Research, Vol. 236, pp. 85-99, 2014.

[88] G. Ruhe and B. Fruhwirth. ϵ-optimality for bicriteria programs and its application to minimum cost flows. Computing, 44 : 21-34, 1990.

[89] H.M. Safer et J.B. Orlin. Fast approximation schemes for multi-criteria combinatorial optimization. Rapport technique 9756-95, MIT Sloan School of Management, 1995.

[90] M.Sakarovitch : Optimisation Combinatoire (Méthodes mathématiques et algorithmiques), 1984.

[91] R.Steuer, and E, Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. Mathematical Programming, 26, 326-344, 1983.

[92] R. Steuer. Multiple Criteria Optimization : Theory, Computation and Applications. John Wiley & Sons, New-York, 1986.

[93] J. Teghem. Cours de Recherche Opérationnelle, 2010.

[94] G. Tsaggouris et C. Zaroliagis. Multiobjective Optimization : Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications. Theory of Computing Systems, 45(1) : 162-186, 2009.

[95] E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA Method : A Tool for Solving Multiobjective Combinatorial Optimization Problems. Journal of Multi-Criteria Decision Analysis, 8(4) : 221-236, 1999.

[96] E.L. Ulungu and J. Teghem. The two-phases method : An efficient procedure to solve biobjective combinatorial optimization problems. Foundation of Computing and Decision Science, 20 : 149-156, 1995.

[97] 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, Mons, Belgique, 1993.

[98] M. Visée, J. Teghem, M. Pirlot and and E.L. Ulungu. Two-phases Method and Branch and Bound Procedures to Solve the Bi-objective Knapsack Problem. Journal of Global Optimization, Vol. 12, pp. 139-155, 1998.

[99] A. Warburton. Approximation of Pareto optima in multiple-objective shortest-path problems. Operations Research, 35(1) : 70-79, 1987.

Annexe A

Détection de l'ensemble de variables régulières

L'algorithme 9 décrit dans le chapitre 6, a été implémenté dans un environnement MATLAB 11 sur un PC Intel(R) Pentium(R)M Processor 2.13 GHz.
Les résultats suivants ont été détectés en utilisant Matlab 11 pour résoudre le problème de sac à dos biobjectifs.

- Résultats des instances de type 1A
  Ce type contient 5 instances du problème de sac à dos biobjectifs unidimensionnels en variables $\{0, 1\}$. Les vecteurs profits et poids sont uniformément générés. Les instances contiennent entre 50 et 500 objets. Le rapport r appartient à l'intervalle $[0.11, 0.92]$, (voir : table.6.3).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50-11$	31	10	28	0	28	14.80
$2KP50-50$	20	2	6	2	8	6.28
$2KP50-92$	48	28	1	34	35	1.74
$2KP100-50$	56	10	8	21	29	5.36
$2KP500-41$	⁄	⁄	⁄	⁄	⁄	⁄

TABLE A.1 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1A$

- Résultats des instances de type $1B$
  Ce type contient 30 données correspondent à 10 instances du problème de sac à dos biobjectifs unidimensionnels en variables $\{0, 1\}$. Le rapport r est fixé à 0.5. Trois variantes A, B et C, de 09 instances sont explorées, où les profits et les poids dans les deux premières variantes sont générées uniformément dans l'intervalle $[1, 100]$.
  a. La classe $1B/A$, les vecteurs profits et poids sont générés uniformément dans $[1, 100]$, (voir : table.6.4).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	27	11	6	10	16	4.89
$2KP100$	53	5	5	13	18	14.47
$2KP150$	83	18	14	24	38	39.16
$2KP200$	116	17	15	34	49	53.95
$2KP250$	139	18	15	39	54	14.00
$2KP300$	173	26	17	54	71	17.55
$2KP350$	213	31	21	58	79	45.70
$2KP400$	252	33	28	72	100	38.90
$2KP450$	270	27	35	88	123	79.51
$2KP500$	301	47	41	86	127	79.60

TABLE A.2 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1B/A$

b. Les instances de type $1B/B$ ont été crées à partir de $1B/A$ en remplaçant le deuxième vecteur profit par le premier dans l'ordre inverse ($c_i^2 = c_{n-i+1}^1$, $\forall i$),(table.6.5).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	24	9	3	9	12	4.21
$2KP100$	51	4	7	14	21	9.81
$2KP150$	78	9	7	23	30	35.47
$2KP200$	108	15	11	25	36	46.84
$2KP250$	145	18	14	38	52	84.17
$2KP300$	180	32	18	51	69	8.39
$2KP350$	203	31	25	51	76	15.67
$2KP400$	249	39	29	64	93	97.36
$2KP450$	249	31	25	77	102	82.22
$2KP500$	305	49	32	81	113	58.06

TABLE A.3 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1B/B$

c. Le type d'instances $1B/C$, les composantes du vecteur profit générées par plateaux de longueurs tirées aléatoirement dans l'intervalle ($[1, \lfloor \frac{n}{10} \rfloor]$), (voir : table.6.6).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	23	3	3	$\underline{8}$	11	2.79
$2KP100$	61	11	11	17	28	18.78
$2KP150$	81	10	13	27	40	14.23
$2KP200$	105	19	10	28	38	39.41
$2KP250$	136	15	11	43	54	119.27
$2KP300$	186	37	28	46	74	50.75
$2KP350$	193	29	10	54	64	87.27
$2KP400$	239	47	41	71	112	161.45
$2KP450$	291	27	28	87	115	104.83
$2KP500$	364	84	69	95	164	139.56

TABLE A.4 – Nombre de variables régulières fixées à 0 ou 1 pour le type $1B/C$

- Résultats des instances de type 1C

a. UNCOR : Cette variante est composée de 20 instances de 50 variables. Les profits et les poids sont générés uniformément ; 10 entre eux sont dans l'intervalle $[1, 300]$ et les autres sont dans l'intervalle $[1, 1000]$, (voir : table.6.7).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	19	1	3	6	9	3.88
$2KP50$	25	5	7	6	13	4.83
$2KP50$	29	6	5	9	14	3.60
$2KP50$	28	4	5	6	11	2.72
$2KP50$	23	2	2	9	11	4.42
$2KP50$	26	4	5	7	12	3.56
$2KP50$	25	3	$\underline{2}$	8	10	6.50
$2KP50$	27	5	6	8	14	7.63
$2KP50$	31	8	4	10	14	8.82
$2KP50$	27	6	5	8	13	3.47
$2KP50$	24	4	3	$\underline{8}$	11	4.25
$2KP50$	30	9	5	9	14	5.02
$2KP50$	30	9	$\underline{3}$	8	11	7.76
$2KP50$	25	5	6	5	11	5.24
$2KP50$	22	7	4	10	14	6.42
$2KP50$	25	3	6	6	12	2.68
$2KP50$	21	5	2	8	10	5.61
$2KP50$	33	12	8	10	18	5.84
$2KP50$	28	2	3	5	8	3.16
$2KP50$	22	4	2	10	12	4.37

TABLE A.5 – Nombre de variables régulières fixées à 0 ou 1 pour le type 1C

b. WEAK : 15 instances corrélées de taille entre 50 et 1000. Les vecteurs poids sont générés uniformément dans [1,1000]. Le deuxième vecteur profit prend ces valeurs dans [111,1000]. Le premier vecteur profit est choisi aléatoirement dans l'intervalle $[c_i^2 - 100, c_i^2 + 100]$, (voir : table.6.8).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	46	26	14	12	26	5.62
$2KP100$	100	40	28	27	55	11.61
$2KP150$	139	51	40	39	79	39.89
$2KP200$	197	64	55	53	108	104.22
$2KP250$	237	77	65	66	131	152.32
$2KP300$	281	94	79	79	158	490.74
$2KP350$	328	116	92	92	184	287.32
$2KP400$	373	133	104	107	211	807.58
$2KP450$	420	146	118	117	235	321.64
$2KP500$	468	169	130	132	262	408.63
$2KP600$	560	199	155	158	313	299.96
$2KP700$	658	224	179	188	367	648.46
$2KP800$	751	245	205	212	417	1069.88
$2KP900$	841	279	229	241	470	468.12
$2KP1000$	934	310	252	272	524	810.83

TABLE A.6 – Nombre de variables régulières fixées à 0 ou 1 pour le type WEAK

c. STRONG : 09 instances fortement corrélées d'une taille entre 50 et 450 sont utilisées. Les poids sont générés uniformément dans $[1, 1000]$. Le deuxième vecteur profit prend ces valeurs dans l'intervalle $[1, 1000]$. Le premier vecteur profit égal à $w_j + 100$, (voir : table.6.9).

Instances	$\mid C_E \mid$	$\mid C_J \mid$	$\mid D_0 \mid$	$\mid D_1 \mid$	$\mid D \mid$	$T(s)$
$2KP50$	29	–	4	10	14	14.15
$2KP100$	69	–	8	21	29	7333.42
$2KP150$	100	–	12	33	45	37.715
$2KP200$	140	–	16	44	60	7454.11
$2KP250$	180	–	19	53	72	7589.42
$2KP300$	212	–	–	–	–	–
$2KP350$	–	–	–	–	–	–
$2KP400$	–	–	–	–	–	–
$2KP450$	–	–	–	–	–	–

TABLE A.7 – Nombre de variables régulières fixées à 0 ou 1 pour le type STRONG