N° d'ordre :354-c/MT

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

Thèse

Présentée pour l'obtention du **grade** de **DOCTORAT de 3ème Cycle**

En : MATHEMATIQUES

Spécialité : Recherche Opérationnelle et Mathématiques Discrètes

Par : **SATLA Hamou**

Sujet

Contribution algorithmique pour l'optimisation combinatoire multi objectif

Soutenue publiquement, le 27 /09 /2025, devant le jury composé de :

M.AIDER Méziane	Professeur à USTHB	Président
M.CHERGUI Mohamed El-Amine	Professeur à USTHB	Directeur de thèse
Mme.ADICHE Chahrazad	MCA à UMBB	Examinatrice
M.BOUZID Mouaouia Cherif	MCA à ENSTA, Dergana	Examinateur

Table des matières

Table des figures

Liste des tableaux

Remerciement

Louange à Dieu clément et miséricordieux, sans lui rien de tout cela n'aurait pu être.

Je tiens à exprimer ma profonde gratitude au Professeur Mohammed El-Amine CHERGUI, d'abord, d'avoir accepté de m'encadrer dans ce travail, puis pour sa disponibilité et ses conseils tout au long du parcours qui nous a réunis.

J'adresse mes sincères remerciements à Monsieur Méziane AIDER, Professeur à l'USTHB, pour avoir accepté de présider le jury. J'associe à ces remerciements Madame Chahrazad ADICHE, Maitre de Conférences classe A à UMBBoumerdes et Monsieur Mouaouia Cherif BOUZID, Maitre de Conférences classe A à ENSTA, Dergana, , pour avoir accepté d'examiner mon travail.

Je rends hommage à tous les enseignants qui ont, avec bienveillance et passion, éclairé mon parcours de leur savoir.

Je veux remercier aussi tous les membres de mon laboratoire de recherche RECITS de l'USTHB.

Je remercie mes amis : Djamel Hadj, Abdennour, Lyamine, Abdelali, Bilal, Yahia, Abdelghani, Seifeddine, Smail Alouane, Mahmoude, Abderraouf,Brahim, pour leurs amitié.

Dédicaces

A ma chère mère, mes très chers frères ; Abdeljalil, Ahmed, Abdelkader, Mostafa, Hadri, Saidou, toute ma famille SATLA, Demmouche et Hadef, mes amis de Bab-Ezzouar chacun avec son nom

Résumé

Dans le présent travail, nous nous intéressons à la résolution exacte du problème d'affectation multi-objectif (MOAP). Dans une première partie, nous avons développé une méthode exacte fondée sur le principe de séparation et d'évaluation, combinée à l'algorithme hongrois, afin de déterminer l'ensemble des points non dominés. Cette approche a été comparée à deux algorithmes de référence proposés pour les cas bi-objectif et tri-objectif du problème $MOAP$, et notre méthode s'est révélée supérieure dans plusieurs instances.
Par ailleurs, nous avons abordé le problème d'optimisation d'un critère linéaire sur l'ensemble des points non dominés du $MOAP$. La méthode proposée a été confrontée à celle de Lokman, et les résultats obtenus se sont révélés particulièrement prometteurs, notamment en termes de temps de calcul.

Mots clés : Affectation multi-objectif, méthode Hongroise, séparation et évaluation, ensemble des points non dominés, optimisation d'un critère linéaire sur l'ensemble des points non dominés.

Abstract

In the present work, we focus on the exact solution of the multi-objective assignment problem ($MOAP$). In the first part, we developed an exact method based on the separation and evaluation principle, combined with the Hungarian algorithm, to determine the set of non-dominated points. This approach was compared to two benchmark algorithms proposed for the bi-objective and tri-objective cases of $MOAP$, and our method outperformed both in several instances. Furthermore, we addressed the optimization problem of a linear criterion over the set of non-dominated points of the $MOAP$. The proposed method was compared to that of Lokman, and the results obtained were particularly promising, especially in terms of computational time."

Keywords : Multi-Objective Assignment problem, Hungarian Method, branch-and-bound, Set of Non-Dominated Points, Optimization of a Linear Criterion over the Set of Non-Dominated Points.

Classification MSC2000 : 90C27, 90C29, 90C08.

Introduction Générale

La recherche opérationnelle est une discipline scientifique qui vise à modéliser, analyser et résoudre des problèmes complexes d'organisation, de gestion et de prise de décision. En s'appuyant sur des outils mathématiques, des algorithmes et des méthodes informatiques, elle permet d'optimiser les ressources, d'améliorer l'efficacité des systèmes et d'aider à choisir les meilleures alternatives dans des environnements variés tels que la logistique, la finance, les transports et la production.

Les problèmes d'optimisation du monde réel sont le plus souvent modélisés comme des problèmes multi-objectif, puisque plusieurs objectifs concurrents doivent être pris en compte en même temps. Par conséquent, pour de tels problèmes le concept d'optimalité est remplacé par celui de l'efficacité, où une solution est qualifiée comme efficace si son image appartient à un ensemble de compromis de critères, appelé ensemble de points non dominés, sachant qu'un point non dominé est un vecteur pour lequel la valeur d'un critère ne peut être augmenté sans détériorer la valeur d'au moins un autre critère. Pour plus de détails sur l'optimisation multi-objectif, le lecteur pourra consulter les références ((14), (23), (26), (70)).

La résolution d'un problème d'optimisation multi-objectif revient soit à déterminer l'ensemble des points non dominés dans l'espace des critères ou l'ensemble des solutions efficaces dans l'espace des décisions. En général, la cardinalité de ces ensembles augmente avec le nombre d'objectifs, rendant ainsi le problème plus complexe. Dans la littérature, un grand nombre d'approches existent pour résoudre les problèmes d'optimisation multi-objectif linéaire en nombre entiers (MOILP). Récemment, Halffmann et al. (36) divisent les algorithmes pour les problèmes $MOILP$ en sous-groupes spécifiques au problème. Les algorithmes d'un même sous-groupe partagent la même idée du principe de résolution, telle que le principe du "branch-and-bound" ou une méthode de scalarisation distincte. En outre, les méthodes exactes sont inefficaces pour résoudre à grande échelle des problème $MOILP$. L'utilisation des techniques heuristiques et métaheuristiques génèrent de bonnes solutions approximatives de bon compromis dans un temps de calcul raisonnable. Le lecteur peut se référer aux références (76), (22), (45), (50)...

Dans ce mémoire, le problème d'affectation multi-objectif ($MOAP$) constitue le cadre général de notre étude. Le problème $MOAP$ offre une modélisation plus

réaliste des situations rencontrées dans le monde réel et apparaît dans de nombreux contextes ((72), (39)). Il est reconnu comme étant NP-difficile (23) et, à notre connaissance, seules deux approches de résolution spécifiques ont été proposées pour les cas bi-objectif (58) et tri-objectif (60). Par ailleurs, plusieurs méthodes générales dédiées à la programmation linéaire en nombres entiers multi-objectif ((56), (46), (5)) ont été appliquées à des instances particulières du problème $MOAP$.

Dans le présent travail, nous proposons une méthode exacte, fondée sur le principe de séparation et évaluation progressive (branch-and- bound), visant à identifier l'ensemble des points non dominés du problème $MOAP$, sans restriction quant au nombre de critères considérés. Cette méthode intègre une adaptation de la métaheuristique de recherche à voisinage variable multi-objectif ($MOVNS$) afin de générer un front de Pareto initial de qualité. À chaque nœud de l'arborescence de recherche, un problème d'affectation mono-objectif, basé sur l'un des critères du problème $MOAP$, est résolu à l'aide de la méthode hongroise (48). En appliquant simultanément les mêmes opérations de la méthode hongroise aux autres matrices de coûts, une liste de directions de descentes est construite à partir des variables hors base. Cette liste est exploitée dans le processus de branchement : fixer une variable de décision de cette liste à 1 entraîne une diminution d'au moins un critère et permet de réduire les matrices de coûts en supprimant la ligne et la colonne correspondantes. Le problème $MOAP$ réduit est alors résolu selon le même principe. De plus, une phase de prétraitement est mise en œuvre pour éliminer les sous-problèmes irréalisables. Ce travail a fait l'objet d'une publication dans la revue internationale "Pesquisa Operacional"(64).

Dans cette thèse, nous présentons également une méthode exacte, reposant sur le même principe que précédemment, mais avec des adaptations ciblées, visant à résoudre le problème d'optimisation d'un critère linéaire sur l'ensemble des points non dominés de $MOAP$. Cette approche permet d'éviter l'exploration exhaustive de tous les points non dominés. Dans ce cas, à chaque nœud de l'arborescence de recherche, un problème d'affectation mono-objectif, relatif au nouveau critère d'optimisation, est résolu à l'aide de la méthode hongroise. Plusieurs règles de sondage des nœuds sont introduites afin d'éviter l'exploration des domaines non prometteurs ; ils ne contiennent pas une solution optimale et/ou une solution efficace de $MOAP$.

Cette thèse s'articule principalement autour de quatre chapitres. Le premier est dédié à la présentation des notions fondamentales, des concepts clés et des méthodes existantes relatives à la programmation linéaire multi-objectif en nombres entiers $MOILP$. Le deuxième chapitre se concentre sur le problème $MOAP$, en exposant les approches exactes développées dans la littérature pour les cas bi-objectif et tri-objectif. Notre contribution principale, consistant en une méthode exacte pour résoudre le problème $MOAP$ sans restriction sur le nombre de critères, est détaillée dans le troisième chapitre. Le quatrième chapitre présente une nouvelle méthode visant à optimiser un critère linéaire sur l'ensemble des points non dominés du $MOAP$.

Ce chapitre inclut également des définitions et des résultats liés à l'optimisation d'un critère linéaire sur l'ensemble des points non dominés du $MOILP$. Enfin, ce travail est conclu par une synthèse des résultats obtenus et la proposition de perspectives de recherche futures.

Chapitre 1

Problèmes d'optimisation combinatoire multi-objectif

1.1 Introduction

De nombreux secteurs industriels, tels que les transports, l'agronomie, les télécommunications, l'environnement et l'économie, sont confrontés à des problèmes d'optimisation combinatoire impliquant des objectifs souvent conflictuels. L'optimisation combinatoire multi-objectif (MOCO) vise à traiter ces situations en considérant simultanément plusieurs critères. Les premières contributions théoriques dans ce domaine remontent au XIX^{e} siècle, avec les travaux de Francis Ysidro Edgeworth (21), qui introduisit en 1881 la notion d'optimalité dans son ouvrage Mathematical Psychics. Cette approche fût ensuite généralisée par Vilfredo Pareto (28) dans son cours d'économie politique publié en 1896, établissant ainsi les fondements de l'optimalité au sens de Pareto.

Les méthodes de résolution des problèmes d'optimisation combinatoire multi-objectif se classent généralement en deux catégories : les méthodes exactes et les métaheuristiques. Les méthodes exactes visent à déterminer l'ensemble complet des solutions optimales de Pareto, mais leur complexité computationnelle limite leur application à des problèmes de petite taille ou à un nombre restreint de critères. En revanche, les métaheuristiques, telles que les algorithmes génétiques, le recuit simulé ou la recherche tabou, offrent une flexibilité et une efficacité accrues pour explorer des espaces de solution vastes et complexes. Elles sont particulièrement adaptées aux problèmes de grande dimension, où les méthodes exactes deviennent impraticables. Cette adaptabilité explique leur prédominance dans la littérature scientifique et leur large adoption dans les applications industrielles.

Dans ce chapitre, nous rappelons en premier lieu la formulation d'un problème d'optimisation en nombres entiers. Ensuite, nous donnons un aperçu sur quelques notions de base liées à l'optimisation multi-objectif et la classification des solutions

au sens de Pareto. Nous terminons par donner quelques méthodes de résolution de ces problèmes.

1.2 Sur l'optimisation combinatoire multi-objectif discrète

Un problème d'optimisation linéaire multi-objectif ($MOLP$) (ou multicritère) peut être défini comme un problème dont on cherche à satisfaire un ensemble de contraintes et optimise un vecteur de fonctions objectifs. Le problème $MOLP$ est formulé comme suis :

$$(P)\begin{cases} Maximizer\ Z(x) = (z_1(x), z_2(x), ..., z_p(x)) \\ x \in X. \end{cases}$$

où $z_k(x) = c^k x$, c^k est un $1 \times n$- vecteur. $X = \{x \in \mathbb{R}^n, Ax \leq b, x \geq 0\}$ est l'ensemble des solutions réalisables pour (P). $\mathbb{R}^n$ est appelé espace de décisions. Chaque solution $x \in X$ a une image $(z_1(x), z_2(x), ..., z_p(x))$ dans $\mathbb{R}^p$ (espace des critères).

Remarque 1.2.1. *Sans perte de généralité, nous supposerons que toutes les fonctions objectifs sont à maximiser.*

1.2.1 Concepts de base

Notion de dominance et d'efficacité

Dans le contexte multi-objectif, pour comparer des solutions il est courant d'utiliser la relation de dominance de Pareto. Plus formellement, les deux notions caractérisant les solutions d'un problème multi-objectif sont données par les définitions suivantes (70), (75) :

Définition 1.2.1. *Un vecteur objectif $Z = (z_k)_{k=1,...,p} \in \mathbb{R}^p$ domine un autre vecteur objectif $Z'(z'_k)_{k=1,...,p} \in \mathbb{R}^p$, si et seulement si $\forall k \in \{1, ..., p\}$, $z_k \geq z'_k$ et $\exists k \in \{1, ..., p\}$ tel que $z_k > z'_k$. On dit que Z est un point non dominé s'il n'existe aucun point Z' qui le domine.*

Définition 1.2.2. *Une solution $x \in X$ est efficace ou Pareto optimale si son image $Z(x)$ est un point non dominé. On note par E l'ensemble des solutions efficaces du problème $(MOLP)$.*

Définition 1.2.3. *Le front Pareto, dans l'espace des critères, est l'ensemble correspondant des points non dominés. L'ensemble des solutions efficaces, dans l'espace de décisions, est appelé l'ensemble Pareto optimal. En pratique, ce dernier est généralement de grande cardinalité.*
On note par $Z(E)$ l'ensemble des vecteurs non dominés.

La résolution d'un problème multi-objectif, revient soit à trouver l'ensemble des points non dominés dans l'espace des critères ou l'ensemble des solutions efficaces dans l'espace de décision.

Point Idéal, point nadir et matrice des gains

Définition 1.2.4. *Le point idéal I est un point dans l'espace des critères de cordonnées $(z_1^*, ..., z_p^*)$, où z_k^* est la valeur maximale de la fonction objectif k, $k \in \{1, ..., p\}$.*

Définition 1.2.5. *Le point nadir $N \in \mathbb{R}^p$ de coordonnées $(N_1, ..., N_p)$, $z_k(x) = c^k x$, $\forall k \in \{1, ..., p\}$,*
où $N_k = min\{c^k x / x$ est une solution efficacede$(P)\}$.

Définition 1.2.6. *La matrice de gains M associée à un problème multi-objectif peut être représentée par une matrice carrée de dimension p (critère, solution optimale correspondante) comme suit :*

Soit x_k^* une solution optimale obtenue en optimisant le critère z_k sur X, $\forall k \in \{1, ..., p\}$,.

$$M = \begin{pmatrix} z_1^* & z_{12} & z_{13} & \dots & z_{1p} \\ z_{21} & z_2^* & z_{23} & \dots & z_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_{p1} & z_{p2} & z_{p3} & \dots & z_p^* \end{pmatrix}$$

avec $z_k^* = max\{z^k / x \in X\} = c^k x_k^*$ $\forall k \in \{1, ..., p\}$, $z_{kj} = c^j x_k^*$, $\forall k \in \{1, ..., p\}$, $\forall j \in \{1, ..., p\}$, $(z_{kk} = z_k^*)$.

Cette matrice n'est pas unique, si un critère admet plus d'une solution optimale.

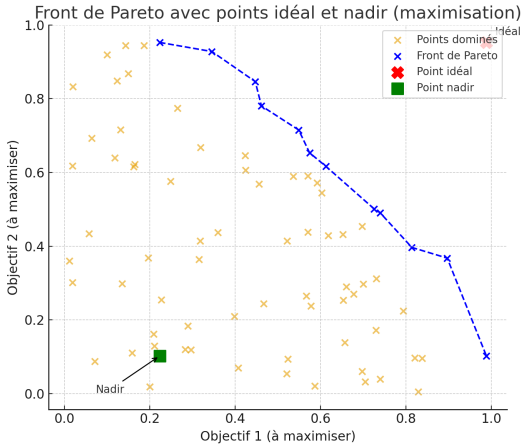


FIGURE 1.1 – Représentation du front Pareto et les points particuliers

Convexité

Définition 1.2.7. *Un ensemble S est dit convexe si, pour tout deux points A et B distincts quelconques de cet ensemble, le segment $[A, B]$ est contenu dans l'ensemble S.(78)*

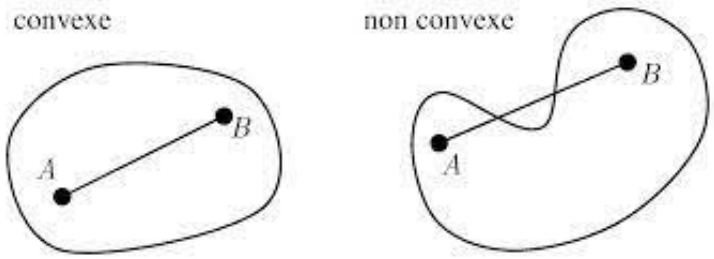


FIGURE 1.2 – Exemple

1.2.2 Caractérisation des solutions efficaces

Soit le problème d'optimisation linéaire multi-objectif en nombres entiers $MOILP$ suivant :

$$(MOILP)\begin{cases} Maximizer\ z^k(x) = c^k x, & k \in \{1, ..., p\} \\ sous \quad x \in D. \end{cases}$$

$X = \{x \in \mathbb{R}^n | Ax = b,\ x \geq 0\}$ avec $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $C = (c^k)_{k \in \{1,...,p\}} \in \mathbb{Z}^{p \times n}$, et $D = X \cap \mathbb{Z}^n$.

Il peut arriver dans la programmation multi-objectif que l'ensemble des solutions efficaces soit très vaste et même infini dans le cas continu. Dans de telles situations, il est souvent impossible d'énumérer toutes les solutions efficaces et même lorsqu'il est possible, il est nécessaire d'aider le décideur à faire un choix d'une solution parmi les solutions efficaces. Pour désigner une solution efficace spécifique comme candidate de meilleur compromis, il est indispensable d'avoir une information supplémentaire sur la structure de préférence du décideur. Elle peut parfois se traduire en termes de paramètres de préférences, appelés les poids $\lambda_k,\ k \in \{1, ..., p\}$ qui reflètent l'importance de chaque critère k.

Étant donné un ensemble de paramètres de préférences

$$\Omega = \{\lambda = (\lambda_k) \in \mathbb{R}^p | \sum_{k=1}^{p} \lambda_k = 1,\ \lambda_k > 0,\ \forall k \in \{1, ...p\}$$

.

Théorème 1.2.1. *(Geoffrion (31)) Soit (P) le programme multi-objectif ayant p fonctions à optimiser et $\lambda \in \mathbb{R}^p$. Le problème pondéré (P_λ) est défini par :*

$$(P_\lambda) : \ max\{\lambda^t Cx | x \in D\}$$

.
avec $\lambda \in \Omega$ et $Z_D = \{Z(x) \in \mathbb{Z}^p | x \in D\}$.

a) Si x est une solution optimale de (P_λ), x est une solution efficace.
b) Si x est une solution efficace et que Z_D est un ensemble convexe, il existe $\lambda \in \Omega$ tel que x est une solution optimale de (P_λ).

Cette méthode de calcul, populaire grâce à sa simplicité de mise en œuvre, a conduit à la distinction des solutions efficaces et des points non dominés en deux catégories :

- les points non dominés supportés, se trouvant sur la frontière de l'enveloppe convexe de Z_D. Les ensembles de points non dominés supportés et des solutions efficaces supportées correspondantes sont notés Y_{SN} et D_{SE}. Ces points sont obtenus par la résolution de sommes pondérées.
- les points non dominés non supportés, constitués des points de Z_D situés à l'intérieur de son enveloppe convexe. Les ensembles de points non dominés non supportés et des solutions efficaces non supportées correspondantes sont notés Y_{NN} et D_{NE}. Par ailleurs, on peut encore distinguer deux types de solutions parmi D_{SE} :
- les solutions efficaces supportées extrêmes dont l'image se situe sur un sommet de $conv(Z_D)$. Ces solutions forment D_{SE_1} et $Y_{SN_1} = Z(D_{SE_1})$ est l'ensemble des points non dominés supportés extrêmes ;
- les solutions efficaces supportées non extrêmes dont l'image n'est pas sur un sommet de $conv(Z_D)$. Ces solutions forment D_{SE_2} et $Y_{SN_2} = Z(D_{SE_2})$ est l'ensemble des points non dominés supportés non extrêmes. (15), (78).

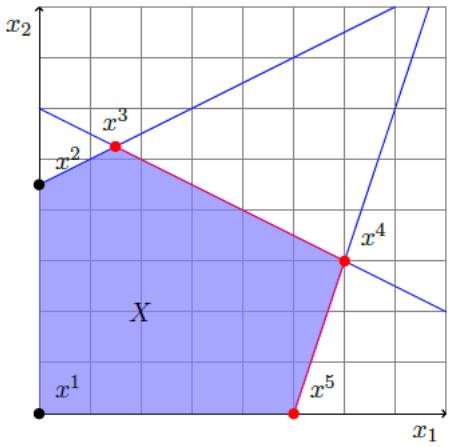
La principale difficulté réside dans la détermination des solutions non supportées puisqu'on ne dispose pas de méthode algorithmiquement simple pour les calculer, comme c'est le cas pour les solutions supportées (24).

Exemple 1.2.1. *Soit le programme linéaire bi-objectif suivant :*

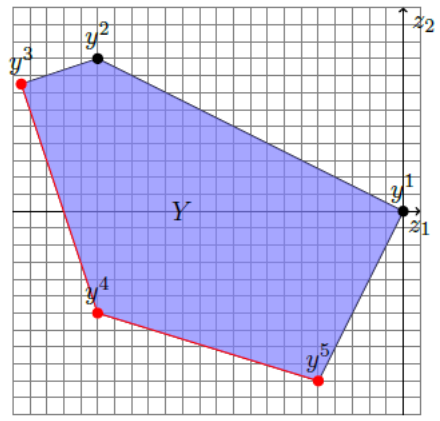
$$(MOLP1) \begin{cases} Min \ Z(x) = \ (-x_1 - 4x_2, -2x_1 + 2x_2) \\ -x_1 + 2x_2 \leq 9 \\ 6x_1 - 2x_2 \leq 30 \\ x_1 + 2x_2 \leq 12 \\ x_i \geq 0 \quad i \in \{1, 2\}, \end{cases}$$

L'ensemble des solutions réalisables X de $MOLP1$ et de son image sont représentés sur la figure 1.3 . En particulier, le polytope X est délimité par cinq solutions extrêmes :

- $x^1 = (0; 0)$ *dont l'image est* $y^1 = (0; 0)$;
- $x^2 = (0; 4.5)$ *dont l'image est* $y^2 = (-18; 9)$;
- $x^3 = (1.5; 5.25)$ *dont l'image est* $y^3 = (-22.5; 7.5)$;
- $x^4 = (6; 3)$ *dont l'image est* $y^4 = (-18; -6)$;
- $x^5 = (5; 0)$ *dont l'image est* $y^5 = (-5; -10)$.



(a) Ensemble réalisable de (MOPL1) et solutions efficaces.



(b) Image de l'ensemble réalisable de (MOPL1) et points non dominés.

FIGURE 1.3 – L'ensemble des solutions réalisables du $MOLP1$ dans l'espace de décision et celui des critères.

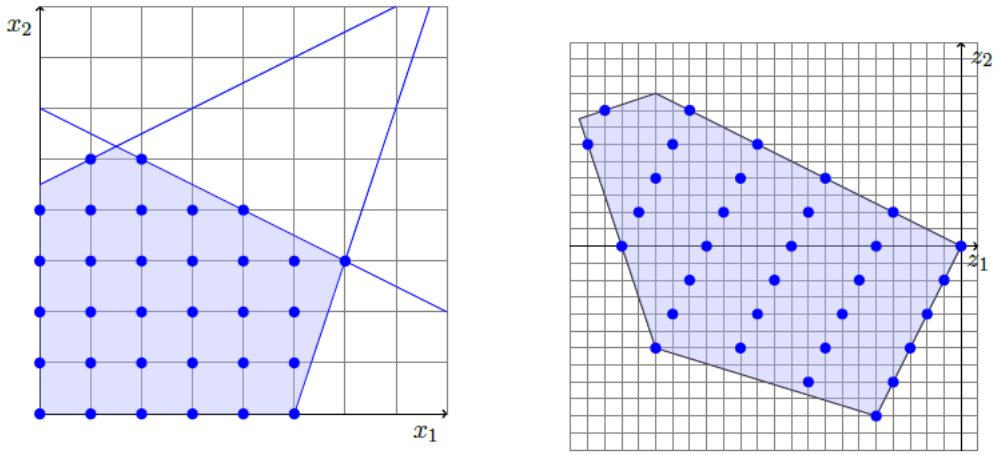
Ici, une infinité de solutions sont efficaces : les segments $[x^3x^4]$ et $[x^4x^5]$. On peut remarquer que ces solutions peuvent être décrites en utilisant uniquement les trois solutions x^3, x^4, x^5. On dit que ces solutions sont efficaces extrêmes. Généralement, elles sont utilisées pour décrire implicitement l'ensemble des solutions efficaces et l'ensemble des points non dominés, et leur nombre peut augmenter exponentiellement avec la taille du problème (70).

Théorème 1.2.2 (Isermann 1974)**.** *Toutes les solutions efficaces d'un MOLP sont supportées.*

Exemple 1.2.2. *Soit le programme linéaire bi-objectif en nombres entiers suivant :*

$$(MOILP1)\begin{cases} Min\ Z(x) = \ (-x_1 - 4x2, -2x1 + 2x_2) \\ -x_1 + 2x_2 \leq 9 \\ 6x_1 - 2x - 2 \leq 30 \\ x_1 + 2x_2 \leq 12 \\ x_i \in \mathbb{N} \quad i \in \{1, 2\}, \end{cases}$$

L'ensemble admissible D de $MOILP1$ et son image sont représentés sur la figure 1.4 :



(a) Ensemble réalisable de (MOIPL1)

(b) Comparaison des valeurs des solutions extrêmes de (MOIPL1)

FIGURE 1.4 – L'ensemble admissible D de $MOILP1$ et son image

Dans cet exemple :

- $D_E = \{(2,5),(4,4),(6,3),(5,0),(5,1)\}$ *et* $Y_N = \{(-22,6),(-20,0),(-18,-6);(-5,-10);(-9;-8)\}$.
- $D_{SE} = \{(2,5);(4;4);(6;3);(5;0)\}$ *et* $Y_{SN} = \{(-22,6),(-20,0),(-18,-6);(-5,-10)\}$.
- $D_{NE} = \{(5,1)\}$ *et* $Y_{NN} = \{(-9,-8)\}$.
- $D_{SE_1} = f(2,5),(6,3),(5,0)\}$ *et* $YSN1 = \{(-22,6),(-18,-6),(-5,-10)\}$.
- $D_{SE_2} = \{(4,4)\}$ *et* $Y_{SN_2} = \{(-20,0)\}$.

On remarque donc, qu'il existe des solutions non supportées dans les problèmes $MOILP$.

Exemple 1.2.3. *Considérons le problème bi-objectif en variables bivalentes suivant :*

$$\begin{cases} Max\ Z(x) = \ (6x_1 + 3x_2 + x_3, x_1 + 3x_2 + 6x_3) \\ \quad x_1 + x_2 + x_3 \leq 1 \\ \quad x_i \in \{0,1\} \quad i \in \{1,2,3\}, \end{cases} \tag{1.1}$$

Par l'application du théorème de Geoffrion, on obtient le modèle mathématique suivant :

$$\begin{cases} Max\ z(x) = \ \lambda(6x_1 + 3x_2 + x_3) + (1-\lambda)(x_1 + 3x_2 + 6x_3) \\ \quad x_1 + x_2 + x_3 \leq 1 \\ \quad x_i \in \{0,1\} \quad i \in \{1,2,3\} \\ 0 < \lambda < 1 \end{cases}$$

Les solutions optimales de ce problème (s^1, s^2) *sont :*

$$\begin{cases} s^1 = (1,0,0) \Rightarrow Z(s^1) = (6,1) \\ s^2 = (0,0,1) \Rightarrow Z(s^2) = (1,6) \end{cases}$$

Nous savons que s^1 *et* s^2 *sont efficaces d'après le théorème. Or, on remarque que la solution* $(0,1,0)$ *est efficace, mais elle n'est pas une solution optimale du problème paramétrique (voir la figure 1.5).*

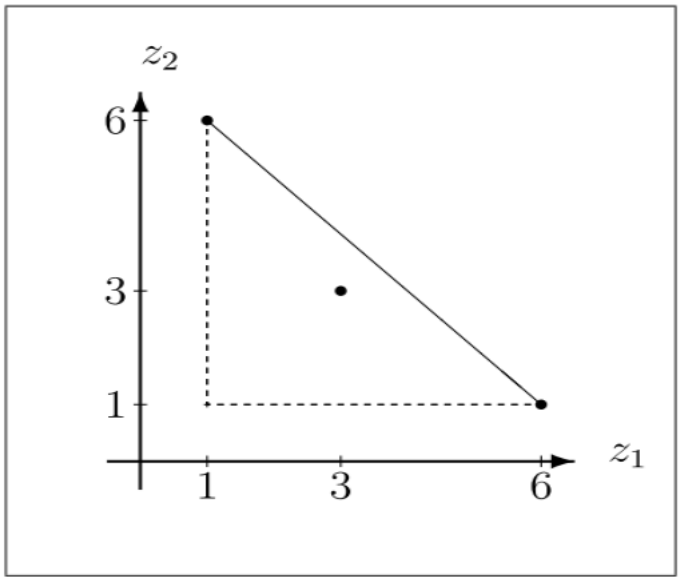


FIGURE 1.5 – L'ensemble des solutions supportés et non supportés du problème 1.1 dans l'espace des critères

La raison est que le domaine des solutions admissibles n'est pas convexe.

1.3 Approches de résolution

Dans la littérature, nous trouvons deux classifications différentes des méthodes de résolution des $MOLP$. Le premier classement adopte un point de vue décideur. Le deuxième classe les méthodes suivant leur façon de traiter les fonctions objectifs. Le premier classement divise ces méthodes en trois familles suivant la coopération entre la méthode d'optimisation et le décideur :

- Les méthodes a priori : dans ces méthodes, le décideur définit le compromis qu'il désire réaliser. Ainsi, il est supposé connaître a priori le poids de chaque objectif, ce qui revient à transformer le problème $MOLP$ en un problème mono-objectif. On retrouve dans cette classe la plupart des méthodes par agrégation.
- Les méthodes interactives : dans ce cas, il y a coopération progressive entre la méthode d'optimisation et le décideur. Le décideur affine son choix de compromis au fur et à mesure du déroulement du processus de résolution à partir des connaissances acquises à chaque étape du processus.
- Les méthodes a posteriori : dans ce cas, le décideur choisit une solution parmi toutes les solutions de l'ensemble Pareto optimales fourni par la méthode d'optimisation. Cette méthode est utilisable lorsque la cardinalité de l'ensemble Pareto optimal est réduite.

On peut trouver des méthodes d'optimisation multi-objectif qui n'entrent pas exclusivement dans une famille. On peut utiliser, par exemple, une méthode a priori en lui fournissant des préférences choisies aléatoirement. Le résultat sera alors un ensemble Pareto à cardinalité élevée qui sera présenté au décideur pour qu'il choisisse une solution. Cette combinaison forme alors une méthode a posteriori.

La deuxième classification divise les méthodes de résolution de $MOLP$ en deux classes :

- La classe de méthodes transformant le problème initial en un problème mono-objectif : ainsi ces méthodes retournent une solution unique, elles utilisent généralement des méthodes d'agrégation.
- La classe de méthodes fournissant un ensemble de solution efficaces : cette classe est aussi divisée en deux sous-classes :

  - Les méthodes Pareto : ces méthodes utilisent directement la notion de dominance au sens de Pareto dans la sélection des solutions générées. Cette idée a été initialement introduite par Goldberg (32) pour résoudre problèmes proposés par Schaffer (65).

  - Les méthodes non Pareto : ces méthodes optimisent séparément les objectifs.

1.4 Méthodes basées sur la séparation et évaluation progressive multi-objectif

Le principe de résolution basé sur la séparation et l'évaluation progressive (branch-and-bound) est générique, bien connu pour le calcul d'une solution optimale d'un problème d'optimisation combinatoire à objectif unique. Basé sur le principe de « diviser pour régner », il repose sur l'énumération implicite, vue comme une recherche dans un arbre. Bien que la méthode branch-and-bound ait été initialement proposée par Land et Doig en 1960 (49), le premier algorithme complet présenté comme un branch-and-bound multi-objectif que nous avons identifié, a été proposé par Kiziltan et Yucaoglu (47) seulement en 1983. À ce jour, peu d'algorithmes basés sur le principe branch-and-bound en contexte combinatoire multi-objectif ont été développés.

Cette situation n'est pas surprenante, car les contributions portant sur les extensions des composants du principe branch-and-bound pour l'optimisation multi-objectif sont récentes.

Adaptation des composants du branch-and-bound multi-objectif

Ensemble bornant inférieurement :
L'ensemble L, qui constitue une borne supérieure, s'applique au problème global et permet ainsi de borner l'ensemble Y_N. Au début de l'algorithme, L est initialisé généralement avec des points réalisables filtrés par dominance obtenus au moyen d'une heuristique ou métaheuristique. Les éléments de L sont potentiellement non dominés pour le problème multi-objectif, et est mis à jour lors de l'obtention d'un nouveau point réalisable. Il est égal à Y_N à la fin de l'algorithme. Dans notre cas, on considère deux ensembles (l'ensemble des solutions potentiellement efficaces et son image dominés). En effet, les éléments de L sont des points réalisables pour lesquels on n'a pas encore trouvé de points réalisables les dominant. Il est important d'obtenir de bonnes solutions dès que possible afin d'améliorer le sondage par dominance.

Ensemble bornant supérieurement :
Un ensemble de bornes supérieures U est calculé pour chaque nœud examiné. Cet ensemble permet de borner l'ensemble non dominé Y_N du sous-problème associé et sert à déterminer si le nœud courant peut être sondé, soit par optimalité, soit par dominance.

Sondage par dominance :
Sonder un nœud par dominance consiste à comparer son ensemble de bornes inférieures L à l'ensemble bornant supérieur actuel U. S'il existe $l \in L$ tel que, pour tout $u \in U$, l domine u, alors tous les points réalisables du sous-problème correspondant à ce nœud sont dominés. Par conséquent, aucun nouveau point non dominé ne pourra être généré à partir de cette branche, et son exploration peut être interrompue.

Sondage par optimalité :
Ce type de sondage devient inopérant dans un contexte multi-objectif. En effet, même si l'on parvient à obtenir une ou plusieurs solutions réalisables, cela permet uniquement d'affirmer qu'elles sont efficaces pour le sous-problème concerné, sans garantir qu'elles constituent l'ensemble non dominé de ce dernier. La seule exception possible survient lorsque l'ensemble bornant supérieurement est réduit à un unique point (idéal) réalisable.

Sondage par irréalisabilité :
Enfin, le sondage par irréalisabilité n'est pas affecté par l'ajout d'objectifs puisqu'il ne concerne que l'espace des décisions. Celui-ci consiste en effet à sonder les nœuds dont les sous-problèmes ne sont pas réalisables.

Choix du nœud actif :
Bien que le choix du nœud actif de l'arbre de recherche s'applique à l'espace des décisions, il peut dépendre des résultats obtenus dans l'espace des objectifs dans le cas d'une exploration dynamique. Néanmoins, les méthodes proposées dans la littérature utilisent presque exclusivement un parcours statique et plus particulièrement un parcours en profondeur de l'arbre de recherche.

Procédure de séparation :
La procédure de séparation, dont le but est de partitionner l'ensemble admissible, ne subit pas de modification fondamentale par rapport à sa version mono-objectif. Les variables binaires n'ayant pas encore été assignées à une valeur sont appelées variables libres. Dans le cadre des programmes en variables binaires, la procédure de séparation généralement adoptée dans la littérature consiste à générer deux sous-problèmes à partir du problème courant, en fixant une variable libre x_i à 0 dans le premier et à 1 dans le second. La fixation d'une variable binaire est une opération relativement simple ; la principale difficulté réside donc, dans le choix de la variable libre à fixer. À cet effet, une stratégie de séparation peut être définie de manière statique (avant le début de la résolution) ou dynamique (au cours du processus de résolution)(78).

Pour plus de détails, quelques algorithmes de branch-and-bound de la littérature sont discutés dans l'état de l'art de Przybylski et Gandibleux(61).

1.5 Quelques méthodes exactes de résolution pour $MOILP$

Plusieurs méthodes exactes sont disponibles pour résoudre le problème $MOILP$(25). Ces algorithmes générant soit l'ensemble des points non dominés ou l'ensemble de solutions efficaces, se différencient par la caractérisation de la manipulation de l'espace de recherche ainsi que par la procédure utilisée pour l'explorer.

Dans cette partie, nous présentons quelques méthodes de résolution du problème $MOILP$.

1.5.1 Méthode de Abbas, Chergui et Ait Mehdi

Cette approche (2) est une méthode branch-and-bound qui génère l'ensemble des points non dominés du problème $MOILP$. En effet, elle fait appel aux coupes efficaces pour passer d'un vecteur entier à un autre. A l'étape l de l'algorithme, la méthode fait appel à la méthode du simplexe pour la résolution d'un problème (P_l). Dans ce cas, Les vecteurs des critères sont adjoints au tableau du simplexe augmenté et manipulés de la même façon que le critère z de (P_l). Une solution optimale est alors obtenue, si cette solution n'est pas entière, un processus de branchement est effectué pour obtenir une solution entière. Dans le cas contraire, lors de l'obtention d'une solution entière, une mise à jour de l'ensemble des solutions potentiellement non dominées est effectuée en considérant le vecteur critère correspondant à cette solution. Afin de rechercher une autre solution entière, les directions d'amélioration des critères sont utilisées pour construire des coupes efficaces permettant d'éviter l'exploration de domaines qui ne contiennent pas de solutions non dominées.

On note :

$$(P_l)\begin{cases} Maximizer\ z(x) = \sum_{i=1}^{p} c^i x \\ x \in X_l. \end{cases}$$

Où $X_0 = X$ et $X_{l+1} = \bigcup_{k\in K_l}\{x \in X_l | c^i x \geq c^i x^l + \sum_{j\in N_l\setminus H_l^i} \lfloor \widehat{c_j^i} \rfloor x_j + max\{1, \lfloor \widehat{c_{j0}^i} \rfloor\}\}$

- x^l : une solution du problème (P_l).
- B^l : Ensemble des indices des variables de base de x^l.
- N^l : Ensemble des indices des variables de hors base de x^l
- $\widehat{c^i}$: vecteur critère réduit relatif au critère $i \in \{1, ..., p\}$.
- $\widehat{c_j^i}$: La $j^{éme}$ composante du coût réduit du vecteur critére i.
- H_l^i : l'ensemble définit comme suit : $H_l^i = \{j \in N_l | \widehat{c_j^i} > 0\}$.
- K_l : $K_l = \{i \in \{1, ..., p\} | H_l^i \neq \emptyset\}$ est l'esemble des critères pouvant être améliorés à partir de la solution x^l.
- On définit la coupe efficace pour chaque $i \in K_l$:

$$\{c^i x \geq c^i x^l + \sum_{j\in N_l\setminus H_l^i} \lfloor \widehat{c_{j0}^i} \rfloor x_j + max\{1, \lfloor \widehat{c_j^i} \rfloor\}$$

où $\widehat{c_{j0}} = min_{j\in H_l}\{\widehat{c_j^i}\}$, pour $i \in K_l$.

L'algorithme 1 détaille les étapes de la méthode.

Algorithm 1: Algorithme de Abbas, Chergui et Ait Mehdi

Étape 0 : (Initialisation)
$l = 0$, $SND = \emptyset$ (L'ensemble des point non dominés).
- Résoudre Le problème linéaire (P_0) au nœud 0 et soit x^0 la solution obtenue.
- Si la solution x^0 n'est pas entière alors aller à l'étape 1.1 sinon aller à l'étape 1.2.

Étape 1 : (Étape générale)
Tant que il y'a des nœuds non sondés faire :
- Choisir un nœud l et résoudre le problème (P_l). Si (P_l) n'a pas de solutions réalisables alors, le nœud l est sondé. Sinon, soit x^l la solution obtenue.
- Si x^l n'est pas entière, aller à l'étape 1.1. Sinon aller au 1.2.

Étape 1.1 : (Processus de séparation)
- Séparer le nœud l en deux nouveaux nœud : ajouter la contrainte $x \leq \lfloor x^j \rfloor$ dans le premier nœud, $x \geq \lceil x^j \rceil$la contrainte au deuxième nœud. Aller à l'étape 1.

Étape 1.2 : (Processus d'évaluation)
- Si x^l est entière. Si Cx^l n'est pas dominée par tous point de SND alors, $SND := SND \cup \{Cx^l\}$.
- Si il existe un vecteur Cy tel que Cx^l le domine alors, $SND := SND \setminus \{Cy\} \cup \{Cx^l\}$. - Déterminer N^l, H_l^i et K^l. -Si $K^l = \emptyset$, alors le nœud l est sondé, aller à l'étape 1.
-Sinon pour chaque $i \in K^l$, ajouter la contrainte au (P_l) :

$$\{c^i x \geq c^i x^l + \sum_{j \in N_l \setminus H_l^i} \lfloor \widehat{c_{j0}^i} \rfloor x_j + max\{1, \lfloor \widehat{c_j^i} \rfloor\}$$

Aller à l'étape 1.
Fin

1.5.2 Méthode ϵ-contrainte

Cette méthode est proposée par Haïmes et *al.* en 1971 (35). Elle consiste à optimiser l'une des fonction objectifs en considérant les autres objectifs comme contraintes, en les intégrant dans la partie contrainte du modèle mathématique comme ci-dessous :

$$(P) \begin{cases} Min\ f_i(x)\} \\ \\ x \in D \\ \\ f_j(x) \leq \epsilon_j \quad j \in \{1, ..., p\} \setminus \{i\}. \end{cases}$$

où ϵ_j est la borne supérieure de la fonction objectif j.

Sur un problème bi-objectif, le problème est de trouver une borne supérieure au départ de la fonction objectif f_2, soit ϵ_0, qui est respectée par toutes les solutions. On obtient alors une solution Pareto optimale. On borne alors cet objectif f_2 par ϵ_1 qui est la valeur pour l'objectif directement inférieure à celle de la solution trouvée. On procède ainsi itérativement en fixant à l'itération i la valeur ϵ_i de telle manière à exclure la solution Pareto optimale trouvée à l'étape précédente sans exclure d'autres solutions Pareto optimales non trouvées. L'approche est illustrée dans la figure 1.6.

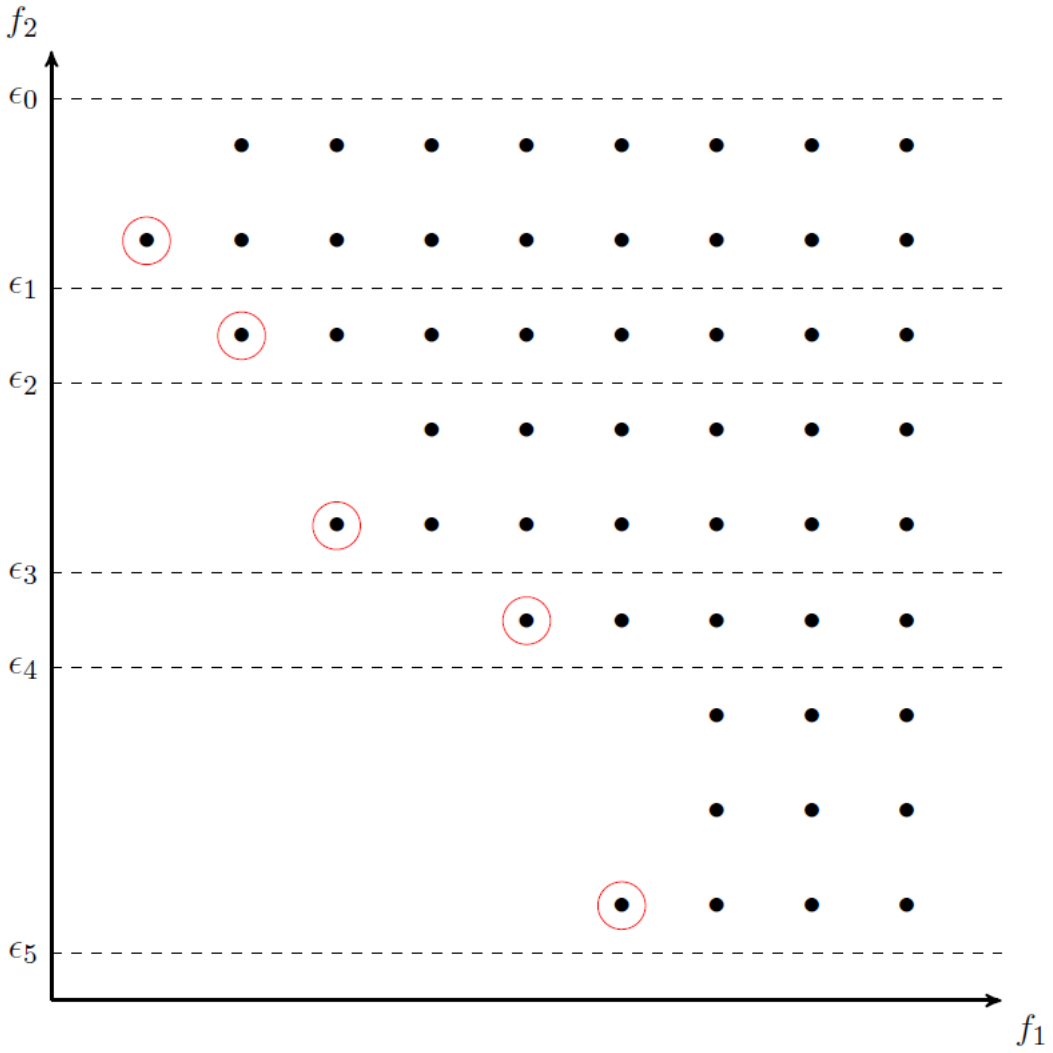


FIGURE 1.6 – Illustration de l'utilisation de la méthode ϵ-contrainte pour le cas bi-objectif.

La méthode ϵ-contrainte est facile à manipuler. Par contre, son inconvénient principal en présence de plus de deux objectifs, est de trouver les bornes de variation des objectifs transformés en contraintes.

1.5.3 Méthode de Özlen et Azizoglu

Cette méthode est une amélioration de la méthode ϵ-contrainte. En effet, dans la méthode ϵ-contrainte les valeur de ϵ_j sont diminuées d'une unité dans le cas de minimisation (toutes les données sont entières), la méthode décrite en (55) diminue ces valeurs en tenant de compte du point non dominé actuelle. Plus précisément, la fonction objectif n'est plus l'un des critères, mais une combinaison pondérée appropriée des critères de problème (P) qui assure l'obtention des solution efficaces.

Dans cette méthode, on montre l'équivalence au sens des solutions efficaces entre le problème (P) et le problème suivant en transformant le $k^{ème}$ critère en contrainte :

$$\begin{cases} Min\ f_1(x) + \epsilon_k f_k(x) \\ . \\ Min\ f_{k-1}(x) + \epsilon_k f_k(x) \\ Min\ f_{k+1}(x) + \epsilon_k f_k(x) \\ . \\ . \\ . \\ Min\ f_p(x) + \epsilon_k f_k(x) \\ \quad f_k(x) \leq l_k \end{cases}$$

où l_k est une borne supérieure de la fonction objectif f_k. On continue de la même façon de transformer le critère $k-1$ en contrainte jusqu'à arriver à la résolution du problème linéaire mono-objectif suivant :

$$\begin{cases} Min\ f_1(x) + \sum_{k=2}^{p} \omega_k f_k(x) \\ \quad f_k(x) \leq l_k \qquad k \in \{2, ..., p\} \\ \quad x \in S. \end{cases} \tag{1.2}$$

où les poids ω_k sont calculés en fonction des bornes supérieures et inférieures de fonction objectif f_k. En résumé, la méthode ramène un problème de p objectifs à un problème de $p-1$ objectif en utilisant à chaque fois la transformation ϵ-contrainte, par exemple, pour le cas biobjectif, la méthode le ramène à un problème mono-objectif.

1.5.4 Méthode amélorée de Özlen et $al.$

Cette méthode (56) consiste à améliorer la version récursive de la méthode de Özlen et Azizoglu pour la génération des points non dominés en utilisant l'ensemble

de sous-problèmes déjà traités et leurs solution pour éviter de résoudre un grand nombre de programmes en nombres entiers.

L'idée principale est, lors de la résolution d'un nouveau problème 1.2, pour rechercher une relaxation de ce problème qui a été résolu auparavant, permettant d'ignorer complètement le nouveau problème obtenu. La méthode n'autorise que la relaxation des contraintes sur les objectifs individuels. Les auteurs ont montré montrent comment une relaxation à un problème 1.2 peut être utilisée pour éviter de le résoudre.

Le pseudo code de la méthode est résumé dans l'algorithme 2.

Algorithm 2: Algorithme de Özlen et *al.*

Étape 0 :
$l_k = \infty$. $ND_k = \emptyset$ (L'ensemble des points non dominés)
Étape 1 :
Répéter :
- Consultez la liste des problèmes précédemment résolus pour trouver une relaxation du problème actuel.
- Si tous les vecteurs non dominés pour la relaxation sont réalisables pour le modèle actuel alors soit ND^*_{k-1} cet ensemble des points non dominé et aller à l'étape 3.
- Si la relaxation est impossible alors stop.

Jusqu'à ce qu'il n'y ait plus d'autres relaxations du problème actuel.
Étape 2 :
- Résoudre le problème 1.2, en utilisant l'Algorithme 2 pour optimiser les $k-1$ premiers objectifs.
- Si le problème est irréalisable alors stop.
- Soit ND^*_{k-1} l'ensemble des points non dominé obtenu.

Étape 3 :
$ND_k := ND_k \cup ND_{k-1}$.
$l_k = max\{f_k | f \in ND_{k-1}\} - 1$.
aller à l'étape 1.
Fin

1.5.5 Méthode en deux phases

La méthode en deux phases a été introduite par Ulungu et Teghem (1995) (77). Dans la première phase, l'ensemble de tous les points supportés non dominés est calculé. Il sert à délimiter une partie de l'espace objectif qui est explorée dans la deuxième phase afin de générer tous les points non supportés non dominés. L'ensemble efficace (ou un ensemble efficace réduit) est éventuellement obtenu comme l'union des ensembles de solutions de la phase 1 et de la phase 2.

La méthode a été instanciée sur des versions bi-objectif de plusieurs problèmes classiques de recherche, incluant entre autres, le problème d'affectation (Przybylski et *al.* (58) ; Delort et Spanjaard (17)), le problème du sac à dos (Visée et *al.* (79) ; Delort et Spanjaard (18)) et le problème de l'arbre bi-objectif (Ramos et *al.* (62) ; Steiner et Radzik (68)). Cet algorithme a également été adapté à plus de deux objectifs par Przybylski et *al.* (60) et appliqué au problème d'affectation et aussi au problème du sac à dos dans Jorge (44) (à la fois pour le cas de trois objectifs). Nous détaillons par la suite les deux phases.

Première phase

La première phase de la méthode en deux phases consiste généralement à calculer l'ensemble de tous les points supportés non dominés. Les points restants se trouvent tous à la fois dans l'enveloppe convexe notée par conv($Yextr$) et dans la partie de l'espace objectif qui n'est dominée par aucun point de l'ensemble des points supportés. L'intersection de ces ensembles définit une zone de recherche. Dans le cas bi-objectif, la zone de recherche correspond à l'union de triangles (voir Figure 1.7).

Deuxième phase

La deuxième phase consiste à l'exploration des zones définies par la première phase. Deux stratégies d'exploration, énumération implicite et classement, se trouvent dans la littérature. La première stratégie consiste à appliquer un schéma d'énumération implicite (généralement procédure branch-and-bound) pour explorer les zones ou toute la zone de recherche.

La deuxième stratégie repose sur un algorithme de classement pour la version mono-objectif du problème étudié. Un algorithme de classement ou k-best pour un problème d'optimisation combinatoire multi-objectif détermine les k meilleures solutions en ordre décroissant de leurs valeurs des fonctions objectifs pour un nombre k positif donné. L'utilisation d'algorithmes de classement avec la la technique des fonctions scalarisantes est habituelle en optimisation combinatoire multiobjectif.

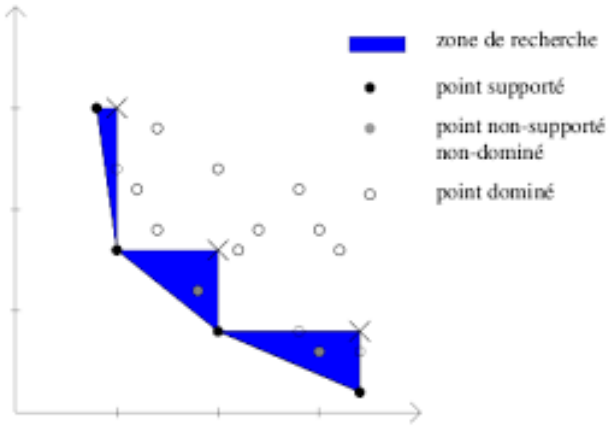


FIGURE 1.7 – Illustration de l'utilisation de la méthode en deux phases pour le cas biobjectif.

1.6 Conclusion

Ce chapitre a présenté le domaine de l'optimisation multi-objectif et plus particulièrement les programmes linéaires multi-objectif en variables entières. Nous avons cité d'une part quelques résultats théoriques et notions de base liés à ces problèmes. D'autre parts, nous avons énumérés quelques méthodes exactes de résolution des problèmes linéaires en variables discrètes.

Chapitre 2

Probléme d'affectation multi-objectif

2.1 Introduction

Le problème d'affectation (AP) vise à trouver une affectation dans laquelle n tâches doivent être exécutées par n ouvriers de sorte que chaque tâche soit réalisée par exactement un ouvrier. L'objectif est de minimiser le coût total de la mission. Le problème AP est un problème d'optimisation combinatoire polynomial qui peut être résolu avec la méthode Hongroise(48). Dans la pratique, cependant, la plupart des problèmes AP nécessitent l'optimisation de deux ou plusieurs fonctions objectives en même temps. Tout problème d'affectation impliquant au moins deux objectifs est appelé problème d'affectation multi-objectif ($MOAP$). Le problème $MOAP$ appartient à la classe des problèmes NP-difficiles (70).

Dans notre étude, nous nous intéressons à la résolution du problème $MOAP$ qui consiste à déterminer soit l'ensemble des affectations efficaces dans l'espace des décisions, soit l'ensemble des affectations non dominés dans l'espace des critères. A notre connaissance, peu d'approches de résolution ont été implémentées, particulièrement, pour le cas bi-objectif (58) et tri-objectif (60).

2.2 Le problème d'affectation mono-objectif

C'est le problème (AP) qui consiste à affecter $n \in \mathbb{N}$ tâches au même nombre d'ouvriers à un coût minimum, de façon que chaque tâche est associée à un et un seul ouvrier. Cette spécificité implique deux particularités à ce programme :

- La fonction objectif correspond à une matrice carrée $C = (c_{ij})_{i,j\in\{1,\dots,n\}}$, où c_{ij} est le coût d'affectation de la tâche i à l'ouvrier j.
- Une solution optimale (ou n'importe quelle solution réalisable du problème AP) est telle qu'il n'y a qu'une seule affectation dans chaque ligne et colonne.

2.2.1 Modèle Mathématique

La formulation mathématique du problème AP est la suivante (41) :

$$(P)\begin{cases} Min\ Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}, \\ \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1,...,n\} \\ \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,...,n\} \\ x_{ij} \in \{0,1\} \qquad \forall i,j \in \{1,...,n\}, \end{cases}$$

où x_{ij} est définie comme suit :

$$x_{ij} = \begin{cases} 1 \text{ si la } i^{\text{ème}} \text{ tâche est affectée au } j^{\text{ème}} \text{ ouvrier} \\ 0 \text{ sinon,} \qquad \forall i,j \in \{1,...,n\}. \end{cases}$$

- Une tâche i est affectée à un et un seul ouvrier j :

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,...,n\}$$

- Un ouvrier j est affectée à une et une seul tâche i :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1,...,n\}$$

Une façon naïve de résoudre le problème d'affectation est d'énumérer toute les solutions possibles et de choisir celle de coût minimum. Ce qui revient à énumérer $n!$ solutions réalisables, et comme le factoriel de n croît très vite avec n, l'énumération totale est presque impossible (63).

2.2.2 Méthode Hongroise

Le mathématicien Hongrois Egervary (19) était le premier à proposer un algorithme implicite pour le problème d'affectation, ce qui inspire Kuhn (48) pour développer la méthode Hongroise (Hungarian method), qui est une une procédure polynomiale de complexité $O(n^4)$. Cette complexité est réduite à $O(n^3)$ plus tard en utilisant les algorithmes de plus court chemin (73). Et depuis, plusieurs approche pour le problème AP ont été développés, le lecteur peut se référer à l'état de l'art de Dell Amico et Martello (16).

Principe de la méthode Soit la matrice C d'éléments (c_{ij}), avec $i,j \in \{1,...,n\}$.

$$C = \begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{pmatrix}$$

Phase 1 :

1) Trouver le minimum de chaque ligne $l_i = min_{1\leq j\leq n}\{c_{ij}\}$.
2) Pour chaque ligne i, on soustrait le minimum correspondant.
3) Trouver le minimum de chaque colonne $w_j = min_{1\leq i\leq n}\{c_{ij}\}$.
4) Pour chaque colonne j, on soustrait le minimum correspondant.
5) Trouver une solution réalisable en affectant un seul zéro par ligne et par colonne.
   Si cette solution existe, alors elle correspond à la solution optimale du problème.
   Sinon, aller à la phase 2.

Phase 2

a) Marquer les lignes ayant un zéro non affecté.
b) Marquer les colonnes ayant un zéro non affecté sur une ligne marquée.
c) Marquer les lignes non encore marquées ayant un zéro affecté dans une colonne marquée.
d) Barrer les lignes non marquées et les colonnes marquées.
e) Trouver le minimum des éléments non barrés, le retrancher des éléments non barrés et le rajouter aux éléments doublement barrés.
f) trouver une solution admissible en affectant un seul zéro par ligne et par colonne. En cas d'échec, reprendre la procédure à partir de la deuxième étape. Si une telle solution existe alors, c'est une solution optimale du problème AP.

Remarque 2.2.1. *La solution optimale donnée par la méthode hongroise n'est pas unique.*

2.3 Définitions et notations

Tout d'abord, le modèle mathématique du problème $MOAP$ se présente comme suit :

Étant donné n tâches et n ouvriers, de sorte que chaque tâche soit réalisée par un ouvrier à la fois, et p matrices de coûts $C^k = c_{ij}^k$ telles que c_{ij}^k est le $k^{\text{éme}}$ poids de l'attribution d'une tâche i à l'ouvrier j, $k \in \{1, ..., p\}$, $p \geq 2$ avec c_{ij}^k sont des entiers non négatifs.

$$(P)\begin{cases} Min\ Z(x) = \quad (Z_k(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij}),\ k \in \{1, ..., p\} \\ \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1, ..., n\} \\ \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1, ..., n\} \\ x_{ij} \in \{0, 1\} \qquad \forall i, j \in \{1, ..., n\}, \end{cases}$$

où x_{ij} est définie comme suit :

$$x_{ij} = \begin{cases} 1 \text{ si la } i^{\text{éme}} \text{ tâche est affectée au } j^{\text{éme}} \text{ ouvrier,} \\ 0 \text{ sinon,} \qquad i, j \in \{1, ..., n\}. \end{cases}$$

Le problème d'affectation multi-objectif est intraitable. Une famille d'instances du problème d'affectation avec deux critères qui le prouve peut être consultée dans le livre de Ehrgott (2005) (23). Tous les points réalisables de chaque instance de cette famille sont non-dominés, par exemple $C^1_{ij} = (j-1)n^{i-1} et\ C^2_{ij} = n^n - C^1_{ij}$, les coûts d'affectation de la tâche i à l'ouvrier j selon le premier et le second objectif, pour tout $i, j \in \{1, ..., n\}$. Les points réalisables de cette instance sont tous non-dominés, i.e. pour toute paire de solutions différentes $x1$, $x2$, $z(x1)$ ne domine pas $z(x2)$ et $z(x2)$ ne domine pas $z(x1)$(7). Aussi, le problème $MOAP$ est NP-difficile(23).

D'autre côté, Plusieurs procédures ont été proposées pour résoudre le problème d'affectation multi-objectif. Les plus efficaces ont été développées par Przybylski et al.(58) pour les cas à deux objectifs, et par Przybylski et al. (60) pour ceux à trois objectifs. On va les détailler dans la section suivante.

2.4 État de l'art

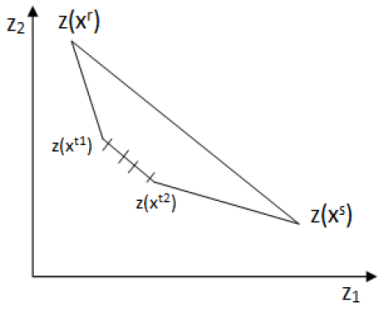
Les travaux de Przybylski et al. ((58), (60)) se concentrent sur le développement de procédures efficaces pour les cas à deux et trois objectifs.

2.4.1 Méthode en deux phases appliquée au cas bi-objectif

La méthode en deux phases est un cadre général pour résoudre les problèmes d'optimisation combinatoire multi-objectif (MOCO) comme on a cité dans le chapitre précédent. Elle a été adaptée par Przybylski et al. (58) pour le problème d'affectation bi-objectif (BAP) avec des contributions importantes sur les bornes et l'exploration des solutions non supportées.
Nous noterons le problème d'affectation mono-objectif pondéré obtenu à partir de (BAP) par (BAP_λ).

Phase 1 : Détermination de solution efficaces supportées

La phase 1 (voir les algorithmes 3 et 4) est une variation du schéma dichotomique proposé par Aneja et Nair (47). Elle détermine l'ensemble minimal des solutions efficaces supportées X_{SEM}. Soit S un ensemble de solutions efficaces trouvées par l'algorithme. $S = \{x^1, x^2\}$ est initialisé avec les deux solutions optimales lexicographiques correspondant à $lexmin_{x \in X}(z_1(x), z_2(x))$ et $lexmin_{x \in X}(z_2(x), z_1(x))$, respectivement.



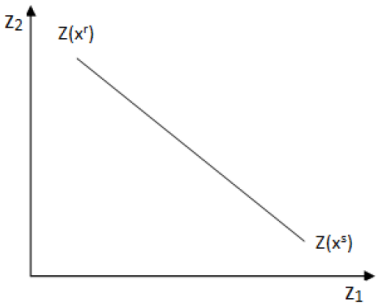


FIGURE 2.1 – Cas a) de la phase 1 cas b)de la phase 1

Algorithm 3: Pseudo code de la phase 1.

Entrées : Les matrices des coûts C_1, C_2.
Sortie : L'ensemble X_{SEM}.
– Calculer une solution optimale lexicographique x_1 pour (z_1, z_2).
-Résoudre le problème d'affectation mono-objectif en considérant la matrice des coûts C_1. Soit x'_1 la solution optimale trouvée et $\bar{C}$ la matrice réduite obtenue.
- Considérer (BAP_λ), le problème d'optimisation à un seul objectif avec la matrice de coûts C'_1, avec $C'_1 = k^1 c^1_{ij} + k^2 c^2_{ij}$.
avec : $k_1 = z_1(x'_1) + 1$ et $k_2 = 1$.
- Résoudre le problème d'affectation mono-objectif en considérant la matrice des coûts C'_1. Soit x^1 la solution optimale trouvée et $\bar{C}$ la matrice réduite obtenue.
- Calculer la solution optimale lexicographique x_2 pour (z_2, z_1).
-Résoudre le problème d'affectation mono-objectif en considérant la matrice des coûts C_2. Soit x'_2 la solution optimale trouvée et $\bar{C}$ la matrice réduite obtenue.
-Considérer (BAP_λ), le problème d'optimisation à un seul objectif avec la matrice de coûts C'_2, avec $C'_2 = k^1 c^1_{ij} + k^2 c^2_{ij}$.
avec : $k_1 = 1$ et $k_2 = z_1(x'_2) + 1$.
- Résoudre le problème d'affectation mono-objectif en considérant la matrice des coûts C'_2. Soit x^2 la solution optimale trouvée et $\bar{C}$ la matrice réduite obtenue.
- Trouver l'ensemble X_{SEM}, comme suit :
$S := \{x^1, x^2\}$. $S = solveRecursion(x^1, x^2, S)$, $X_{SEM} = S$.

Algorithm 4: Pseudo code de la procédure *solveRecursion* de la phase 1.

Entrées : x^r, x^s, S.
Sortie : S.
- Trouver l'ensemble R des solutions optimales de (BAP_k) :
- $min\{k_1 z_1(x) + k_2 z_2(x) : x \in X$, où : $k_1 = z_2(x^r) - z_2(x^s)$ et $k_2 = z_1(x^s) - z_1(x^r)$.
- $C^k = c^k_{ij}$, $c^k_{ij} = k_1 c^1_{ij} + k_2 c^2_{ij}$, $i, j \in \{1, ..., n\}$.
- Résoudre le problème d'affectation mono-objectif en considérant la matrice des coûts C^k, soit x la solution optimale obtenue et $\bar{C^k}$ la matrice réduite obtenue, Énumérer toutes les solutions alternatives de x, Soit R : l'ensemble des solutions trouvées.
- $S := S \cup R$.
- Si $\{z^k(x) : x \in R\} \bigcap \overline{z(x^r)z(x^S)} = \emptyset$ alors,
cas a) : soit x^{t_1} et x^{t_2} les solutions de R, la valeur min et max de z_1 respectivement.
$S := solveRecursion(x^r, x^{t_1}, S)$; $S := solveRecursion(x^{t_1}, x^s, S)$.
Fin Si.
Si $\{z^k(x) : x \in R\} \subset \overline{z(x^r)z(x^S)}$ alors, cas b) : Rien à faire ;

La phase 1 s'arrête s'il n'y a plus aucun problème d'affectation mono-objectif pondéré (BAP_k) à résoudre, on a alors $S = X_{SEM}$ (58).

Phase 2 : Détermination des solutions non supportées

La phase 2 consiste à déterminer les solutions $x \in X$ telle $z(x)$ se situe à l'intérieur de triangle défini par deux points supportés consécutifs $z(x^r)$ et $z(x^s)$ dans l'espace des critères. Ainsi, des bornes supérieures et inférieures ont été proposées par Ulung et Teghem (77), Tuyttens et *al.* (74) afin de limiter l'exploration des triangles construites.

Bornes inférieures de Teghem Ulungu et Teghem (77) ont proposé deux bornes inférieures différentes, selon que x est optimal pour le problème à objectif unique ou non. Soit $\alpha_1(C, x, (i, j))$ la borne inférieure de la valeur de la fonction objectif avec la matrice des coefficients de la fonction objectif C, où x est une solution optimale pour cet objectif, et avec (i, j) l'affectation imposée (c'est-à-dire que $x_{i,j} = 1$). Par analogie, soit $\alpha_2(C, x, (i, j))$ la borne inférieure de la valeur de la fonction objectif si n'est pas une solution optimale pour l'objectif C.

Bornes supérieures de Tuyttens Chaque triangle est exploré séparément, il est défini par deux points supportés consécutifs. Soient x^r et x^s deux solutions supportées consécutives et $\lambda = (\lambda_1, \lambda_2)$ le poids tel que : x^r et x^s sont deux solutions optimales pour (BAP_λ). Ulungu et Teghem (77) et Tuyttens et al (74) ont établi des bornes supérieures sur la valeur de la fonction objectif z_λ définie par $\lambda_1 z_1(x) + \lambda_2 z_2(x)$ pour les solutions efficaces x avec $z(x)$ dans le triangle défini par $z(x^r)$ et $z(x^s)$, soit

$\delta(x^r, x^s)$ l'intérieur de ce triangle. Dans cette phase, tous les points réalisables dans une bande du triangle $\delta(x^r, x^s)$ doivent être énumérés. Une bande est une zone dans le triangle entre la droite $(z(x^r), z(x^s))$ et une droite parallèle à celle-ci. la droite parallèle contient le point $(z(x^r), z(x^s))$ au pire des cas , donc toute solution x telle que $z(x)$ se situe dans le triangle doit être énumérée. L'utilisation de bornes supérieures permet de rapprocher la droite parallèle de $(z(x^r), z(x^s))$. Tuyttens et *al.*(74) améliorent la borne supérieure d'Ulungu et Teghem (77) en utilisant les informations de tous les points déjà explorés pour réduire cette bande.

Algorithme de Ranking La méthode de ranking a pour but de calculer les K meilleures solutions d'un problème mono-objectif. L'objectif est donc d'obtenir K solutions $x^1, x^2, ..., x^K$, telles que :

$$z(x^1) \leq z(x^2) \leq ...z(x^K) \leq z(x), \ \forall x \in X \setminus \{x^1, x^2, ..., x^K\}$$

Cheggiredy et Hamacher (13) ont proposé plusieurs variantes d'algorithmes pour déterminer les K meilleurs couplage parfaits dans un graphe pondéré. En particulier, une des variante peut être utilisée pour énumérer les K meilleures solutions d'un problème d'affectation. L'algorithme est basé sur le principe d'exploration d'un arbre binaire de recherche proposé par Hamacher et Queyranne (37) comme indiqué dans la figure 2.2. Il est nécessaire de calculer la première et la seconde meilleure solution pour chaque nœud de l'arborescence.

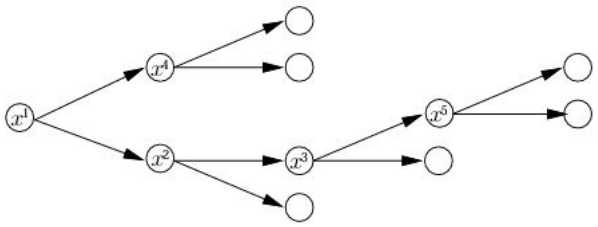


FIGURE 2.2 – Schéma de séparation proposé par Hamacher et Queyranne (37)

Dans un premier temps, il faut déterminer la meilleure solution x^1 et la deuxième meilleure solution x^2 du problème. Ensuite, l'ensemble admissible X du problème est partitionné en deux sous-ensembles X^1 et X^2 tel que x^1 et la meilleure solution dans X^1 et x^2 est la meilleure solution dans X^2. Il est nécessaire de trouver la deuxième meilleure solution x_2^1, x_2^2 dans chacun de ces nouveaux sous-ensembles X^1 et X^2. A la K^{eme} itération de l'algorithme, l'ensemble admissible X est partitionné en K ensembles $X^1; ...; X^K$, la meilleure et deuxième meilleure solution dans chacun de ces sous-ensembles sont connues. La comparaison de chacune de ces deuxièmes meilleures solutions $x_1^2; ...; x2_K$ permet l'obtention de la $(K+1)^{eme}$ meilleure solution dans X.

Description de la méthode Soient x^r et x^s deux solutions efficaces consécutives supportées qui ne sont pas équivalentes, et soit λ tel que x_r et x_s soient toutes deux des solutions optimales de (BAP_λ). La phase 2 trouve des solutions faisables x avec : $z_1(x^r) < z_1(x) < z_1(x^s)$ *et* $z_2(x^r) > z_2(x) > z_2(x^s)$.
Évidemment, toutes ces solutions se trouvent dans le triangle $\delta(x^r, x^s)$. Ulungu et Teghem (33) proposent une méthode d'exploration énumérative pour rechercher ces solutions. L'idée est de trouver des solutions efficaces non supportées en imposant des affectations, c'est-à-dire en fixant les variables $x_{ij} = 1$ dans (BAP_λ), puis en résolvant le problème d'affectation réduit qui en résulte. En d'autres termes, la phase 2 cherche des solutions faisables de (BAP_λ), qui ne sont pas optimales, en partant de et en modifiant ces solutions. Ulungu et Teghem (77) proposent d'utiliser la liste d'affectations L :

$$L = \{\bar{c_{ij}}^{\lambda} > 0\}$$

où : $\bar{C_\lambda}$ désigne la matrice des coûts réduits obtenue par la méthode hongroise pour le problème (BAP_λ) pour lequel x^r et x^s sont optimaux. En imposant une affectation de L, on obtient des solutions avec des points correspondants situés au-dessus de la ligne droite reliant $z(x^r)$ et $z(x^s)$. Cependant, il est possible de supprimer certaines affectations de L en testant si leur imposition ne produit que des points en dehors du triangle $\delta(x^r, x^s)$ en utilisant les bornes inférieures proposées par Ulungu et Teghem (77). Le pseudo code de la méthode est présente dans l'algorithme 5 .

Algorithm 5: Procédure de Phase 2. (60)

Entrées : Les matrices des coûts C_1, C_2.
Sortie : L'ensemble des solutions non supportées S. - Pour tout x_r, x_s solutions consécutives dans X_{SEM}, faire :
-Calculer $C^\lambda = [\lambda c^1_{ij} + \lambda_2 c^2_{ij}]$ avec :

$$\lambda_1 = z_2(x^r) - z_2(x^s), \quad \lambda_2 = z_1(x^s) - z_1(x^r), \quad y_N = (y^{s1}, y^{r2}).$$

- Résoudre le problème d'affectation mono-objectif relatif à la matrice des coûts C^λ, soit $\bar{x}$: la solution obtenue,
- $\bar{C}^\lambda$: la matrice des coûts réduits pour z^λ.
-$L = \{(i,j) \mid c^\lambda_{ij} > 0\}$.(Liste des affectations)
- $R = \emptyset$ (liste des solutions efficaces du triangle $\Delta(y_r, y_s)$).
- $BS = \lambda_1 y^{s1} + \lambda_2 y^{r2}$ (valeur initiale pour la borne supérieure).
- Si $BS > z^\lambda(x^r)$, alors :
  - Pour tout $(i,j) \in L$, faire :
    - Tester si la présence de l'affectation (i,j) donne une solution à l'extérieur du triangle.
    - Si oui, alors $\bar{c}^\lambda_{ij} \leftarrow +\infty$.

Fin Si.
- Si $L \neq \emptyset$, alors :
  - $K \leftarrow 0$.
  - Tant que $z^\lambda(x^K) \leq BS$, faire :
    -$K \leftarrow K + 1$.
    - Appliquer K-best (K) pour calculer la prochaine meilleure solution x^K et la matrice réduite $\bar{C}^\lambda$.
    - Si x^K est dans le triangle $\Delta(y^r, y^s)$ et x^K est non dominé (par tous les éléments de R), alors :
      - $R \leftarrow R \cup \{x^K\}$.
      - Calculer la nouvelle borne supérieure BS.

      Fin Si

  Fin Tant que.
- $S \leftarrow S \cup R$.

2.4.2 Amélioration de la méthode en deux phases

La contribution principale développée par Delort (17) est de montrer l'apport d'un prétraitement (le shaving (52)) dans la méthode en deux phases. Ce prétraitement consiste à fixer une variable booléenne du problème (rendre obligatoire une affectation d'une tâche à un ouvrier). Un calcul de borne qui permettra éventuellement de supprimer cette variable du problème est réalise ensuite (dans le cas où qu'il ne peut exister de solution efficace pour cette valeur de la variable) comme montre la figure 2.3. Cette opération est répétée pour chaque variable booléenne du problème. Le shaving, qui intervient préalablement à l'exploration de chaque zone de recherche

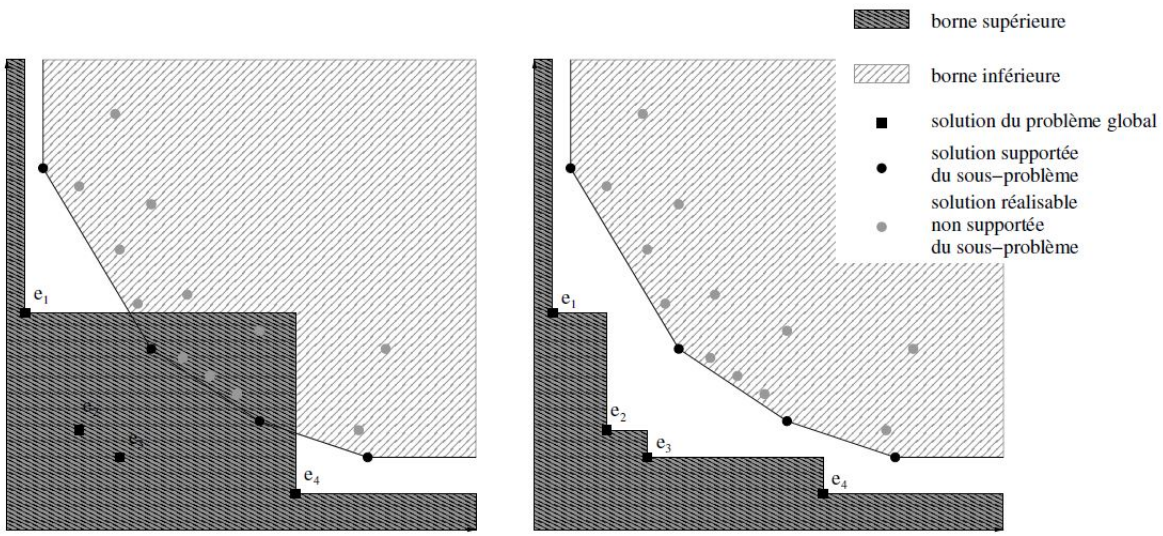


FIGURE 2.3 – Borne supérieure et borne inférieure (39)

(Triangle) lors de la deuxième phase (zone où se trouvent les solutions efficaces non supportées), exploite le même genre de bornes que celles développées pour le branch-and-bound bi-objectif (67), et permet de réduire la taille du problème. Il permet ainsi d'accélérer significativement l'exploration des zones de recherche lors de la deuxième phase. méthode apparaît très compétitive pour résoudre ce problème sur certaines instances.

2.4.3 Méthode en deux phases appliquée au cas tri-objectif

L'extension de la méthode en deux phases pour les problèmes tri-objectifs est un défi méthodologique important, car elle nécessite des adaptations spécifiques pour prendre en compte la complexité supplémentaire introduite par un troisième objectif.

Dans la première phase, les auteurs (60) proposent d'utiliser un nouveau espace des W^0, $W^0 = \{\lambda \in \mathbb{R}^{\mathbb{P}} : \lambda_p = 1 - \sum_{i=1}^{p-1} \lambda_i\}$, ils montrent qu'il y a =une correspondance entre les points non dominés extrêmes et les polytopes convexes de W^0. Une méthode alors calculant la décomposition de l'espace des poids et, simultanément, les points non dominés extrêmes est proposée. Les nouveaux poids à considérer dans cette méthode étant obtenus en décomposant l'espace de poids, tous les poids considérés ont des composantes strictement positives.

La phase 1 qui génère l'ensemble des points Y_{SN} est synthétisée dans l'algorithme suivant :

Deuxième phase : La détermination des zones de recherche à visiter est une tâche difficile dans ce type de méthode, car la notion de solutions successives devient

Algorithm 6: Algorithme Compute Y_{SN} (8)

Sortie : l'ensemble Y_{SN}.
- Initialiser S avec les points non dominés extrêmes des p problèmes à $p-1$ objectifs.
- Répéter :
  - Mettre à jour $conv(S)$.
  - Choisir une facette non explorée ayant un poids $\lambda \in \mathbb{R}^p_+$.
  - Résoudre le problème de somme pondérée.
- Jusqu'à Toutes les facettes non dominées de $conv(S)$ soient explorées.
- $Y_{SN} := S$.

effectivement obsolète dans un contexte multi-objectif, rendant ainsi la définition de triangles à visiter bien moins évidente. Afin de remédier à ce problème, Przybylski et *al.* (60) propose de remplacer la notion de point nadir local par celle de point délimitant qui est définie comme suit :

Définition 2.4.1. *(78) Soit U un ensemble de points ne se dominant pas les uns les autres. d est un point délimitant pour U si et seulement si :*
(i) Pour tout $\epsilon \in \mathbb{R}^p_+$, $d - \epsilon$ n'est pas dominé par un point de U.
(ii) Il n'existe pas de point v vérifiant la condition (i) avec $d \leq v$.

L'ensemble des points délimitants est donc est une généralisation de l'ensemble des points nadirs locaux utilisée dans les triangles au cas bi-objectif. Ils déterminent la zone dans laquelle peuvent se trouver des points non dominés et qu'il faut donc explorer. Cet ensemble doit être mis à jour, quand un point non dominé y est trouvé.

La procédure de mise à jour des points déliminants est décrite dans l'algorithme suivant :

Algorithm 7: Algorithme Compute UBS

Entrée : Ensemble de points réalisables U, point nadir N, point idéal I.
Sortie : Ensemble $D(U)$
- Étape 1 : (Initialisation).
- À chaque étape, Q est un sous ensemble de U.
  - $Q := \emptyset$;
  - $D(Q) := \{N\}$;
- Étape 2 :(Détermination de $D(U)$ en considérant les points de U un par un)

Pour chaque $y \in U$ faire :
  - $D(Q \cup \{y\})$=Update UBS $(y; D(Q)))$
  - $Q := Q \cup \{y\}$

Algorithm 8: Algorithme Update UBS (60)

Entrée : Point réalisable $y \in U \setminus Q$, ensemble $D(Q)$.
Sortie : Ensemble $D(Q \cup \{y\})$.
- W est l'ensemble des points de $D(Q) \setminus D(Q \cup \{y\})$.
- N est l'ensemble des points de $D(Q \cup \{y\}) \setminus D(Q)$.
- $W := \emptyset$;
- $N := \emptyset$;
(Comparaison de y avec chaque point $u \in D(Q)$).
- pour chaque $u \in D(Q)$ faire
  Si $y < u$ alors
  $W := W \cup \{u\}$
  Par la suite, $ui = (u_1; ...; u_{i-1}; y_i; u_{i+1}; ...; up)$
  Pour chaque $i \in \{1; ...; p\}$ faire
  Si ($u_i = y_i$) ou (il existe $v \neq u \in D(Q)$ tel que $u_i \leq v$) alors
  Rien à faire ici
  Sinon
  $N := N \cup \{u_i\}$
  fin si
  fin pour
  fin si
- fin pour

En résumé, la première phase consiste à déterminer l'ensemble des points déliminants, il reste alors à explorer les régions qu'ils délimitent, et de la même façon qu'il faut traiter les triangles dans le cas bi-objectif. Comme plusieurs points délimitant peuvent définir des zones de recherches communes, les auteurs propose d'organiser l'exploration de ces zones comme suit. Parmi les hyperplans générés par les faces obtenu à l'issue de la première phase de ($convY_{SN}$), l'hyperplan $h(u)$ le plus proche du point u est déterminé, pour chaque point u de l'ensemble des points qui délimitent la zone qu'il reste à explorer notée : F . l'ensemble des hyperplans les plus proches d'au moins un point délimitant H_p est formé par les hyperplans $h(u)$, Cet ensemble n'est pas nécessairement égal à l'ensemble H de tous les hyperplans. De plus, on associe à chaque hyperplan h l'ensemble $U(h) := \{u \in F : h(u) = h\}$ et $Val(h) := max_{u \in U(h)} dist(u, h)$. Afin d'explorer $\bigcup_{u \in U(h^*)}(u - \mathbb{R}^p_{\geq})$, à chaque étape, on cherche h tel que $Val(h^*) = min_{h \in H_p} Val(h)$. Cette exploration se fait en utilisant la bande définie par l'hyperplan h^* et l'hyperplan qui y est parallèle et passe par le point de $U(h)$ le plus éloigné de h. Lors de cette exploration, les nouveaux points trouvés sont utilisés pour mettre à jour l'ensemble F et ce processus se répète jusqu'à obtenir $F = \emptyset$.

2.4.4 Les métaheuristiques adaptées pour le problème $MOAP$

Ces dernières années, l'application des techniques métaheuristiques pour résoudre des problèmes d'optimisation multi-objectif est devenue un domaine de recherche

actif. Résoudre ce type de problèmes implique d'obtenir un ensemble de solutions Pareto optimales de manière à ce que le front de Pareto correspondant réponde aux exigences de convergence vers le véritable front de Pareto et à une diversité uniforme.

Tuyttens et *al.* (74) ont proposé une méthode de recuit simulé multi-objectif (MOSA) pour améliorer l'efficacité de l'affectation bi-critère (AP) avec une approche gloutonne. Les résultats obtenus ont également été comparés à ceux d'une méthode exacte. Aussi, Ibrahim et al.(33) ont proposé un modèle de programmation linéaire par objectifs 0–1 pour transformer le problème d'affectation multi critère (MCAP) en un problème d'affectation classique (AP), dont la solution est obtenue à l'aide de l'algorithme hongrois. Adiche et Aïder (3) ont développé une méthode méta heuristique pour résoudre les problèmes d'affectation multi-objectif, dérivés de la variante du coût de dominance du MOSA, en hybridant des procédures de recherche de voisinage impliquant une recherche arborescente multi-objectif (branch-and-bound). Hammadi (53) a proposé la méthode de recherche tabou multi-objectif pourle problème $MOAP$ afin de générer des alternatives non dominées. Belhoul et *al.* (6) ont élaboré une méthode efficace pour trouver les meilleures solutions de compromis pour le problème $MOAP$. Huang et Lim (40) ont présenté un algorithme génétique hybride pour résoudre le problème d'affectation à trois objectifs. Pour trouver la solution du $MOAP$, celui-ci est transformé en un problème à objectif unique. Récemment, Tailor et Dhodiya (71) proposent une approche basée sur un algorithme génétique (GA) pour trouver une solution au problème $MOAP$. Afin d'élaborer un plan d'affectation efficace, le décideur (DM) doit spécifier différents niveaux d'aspiration (ALs) en fonction de ses préférences.

2.5 Conclusion

Nous avons présenté dans ce chapitre le problème d'affectation multi-objectif, sa formulation mathématique ainsi que les notions d'efficacité, non dominance et le point idéal. Nous avons présenté aussi les deux axes principaux de l'approche de résolution des problèmes multi-objectif : Exactes, Approchées. Parmi les méthodes exactes, nous avons rappelé la méthode en deux phases appliquée aux cas : bi-objectif et tri objectif du problème.

Chapitre 3

Méthode exacte pour le problème d'affectation multi-objectif

Dans ce chapitre, nous présentons une technique basée sur la méthode de séparation et évaluation progressive (branch-and-bound) pour générer l'ensemble des points non dominé pour le problème d'affectation multi-objectif. L'approche utilise l'algorithme hongrois pour résoudre un problème d'affectation à objectif unique à chaque nœud de l'arbre de recherche. Elle sélectionne ensuite l'ensemble des variables candidates en utilisant des directions de descente des fonctions objectifs, ce qui permet de déclencher le processus de branchement. L'algorithme est amélioré grâce à des règles de réduction pour accélérer la convergence de la méthode. Des expériences de calcul ont été menées sur un grand nombre d'instances, en considérant plus de trois objectifs. Ces expériences démontrent l'efficacité de la méthode et sa compétitivité par rapport aux travaux antérieurs.

3.1 Introduction

Le problème d'affectation (AP) vise à trouver une affectation dans laquelle n tâches doivent être effectuées par n ouvriers, en s'assurant que chaque tâche est attribuée à un seul ouvrier. L'objectif est de minimiser le coût total de l'affectation. Le problème d'affectation est un problème d'optimisation combinatoire polynomial qui peut être résolu à l'aide de l'algorithme hongrois (2.2.2).

Dans la réalité, cependant, la plupart des problèmes d'affectation nécessitent l'optimisation simultanée de deux ou plusieurs fonctions objectifs, telles que le temps, le coût, la sécurité ou la qualité. Voir les références (72), (39) pour plus de détails. Un problème d'affectation impliquant au moins deux objectifs est appelé problème d'affectation multi-objectif ($MOAP$). Le problème $MOAP$ appartient à la classe des problèmes NP-difficiles (76), (69).

Nous avons commencé par mener une revue approfondie de littérature, qui a révélé seulement deux études prenant en compte les cas bi-objectifs (58) et tri-objectifs

(60) du problème $MOAP$. Dans (58), les auteurs ont proposé un algorithme en deux phases pour résoudre le problème d'affectation bi-objectifs, basé sur l'utilisation d'une procédure de classement pour identifier tous les points non dominés. Cet algorithme a ensuite été étendu au cas tri-objectifs (60) en intégrant des bornes supérieures locales dans des dimensions arbitraires, et en proposant un algorithme en ligne pour mettre à jour cette borne lorsqu'on découvre de nouveaux points faisables. Dans (59), la méthode proposée a été avantageusement comparée à trois autres méthodes. La méthode en deux phases pour le problème d'affectation bi-objectifs a été initialement introduite par Ulungu et Teghem (77). La première phase génère toutes les solutions supportées à l'aide d'une scalarisation par somme pondérée, tandis que la deuxième phase identifie les solutions non supportées en utilisant une méthode spécifique au problème basée sur les solutions supportées. Pederson et *al.* (57) ont proposé une méthode en deux phases pour résoudre le problème d'affectation multi-modal bi-objectifs, qui est une généralisation du problème d'affectation linéaire. Delort et Spaanjard (17) ont amélioré la deuxième étape de l'algorithme en deux phases de (58) en intégrant un prétraitement des données. En outre, plusieurs méthodes générales dédiées au problème de programmation linéaire entière multi-objectif ((56), (46), (5)) ont été testées sur des instances de certains cas du problème $MOAP$.

Cette étude vise à développer une méthode exacte pour la résolution du problème $MOAP$ comportant un nombre arbitraire d'objectifs p (avec $p \geq 2$), permettant l'identification exhaustive de l'ensemble des solutions non dominées. Notre investigation se concentre principalement sur l'amélio-ration du processus de séparation et évaluation progressive (branch-and-bound) en intégrant diverses techniques d'amélioration pour accélérer ses performances.

Bien que l'algorithme de séparation et évaluation progressive ait été initialement proposé par (49) en 1960, le premier algorithme complet introduit comme une méthode de séparation et évaluation progressive multi-objectif a été proposé par Przybylski et Gandibleux (61) en 2017. Cependant, il est important de noter que Kiziltan & Yucaoglu (47) avaient introduit le concept de séparation et évaluation progressive multi-objectif dès 1983. La méthode de séparation et évaluation progressive est une technique qui divise le domaine de recherche en sous-domaines de dimensions inférieures, permettant ainsi de générer tous les points non dominés dans les problèmes d'optimisation multi-objectif. Toutefois, cette méthode n'a pas encore été appliquée au problème $MOAP$.

À notre connaissance, cette étude est la seule à considérer le problème d'affectation multi-objectif avec plus de trois objectifs. Ce chapitre présente une nouvelle méthode basée sur la séparation et évaluation progressive pour générer tous les points non dominés pour le problème $MOAP$. La méthode est conçue pour traiter un nombre quelconque d'objectifs, p (où $p \geq 2$).

Initialement, la méthode de recherche à voisinage variable multi-objectif ($MOVNS$) est adaptée pour obtenir un ensemble initial de points potentiellement non dominés. Un point est considéré comme potentiellement non dominé s'il reste non dominé à l'étape actuelle de l'algorithme. L'ensemble des points potentiellement non dominés est mis à jour à chaque étape, et l'ensemble des points non dominés est déterminé uniquement à la fin de l'algorithme. Par la suite, la méthode résout un problème d'affectation à objectif unique en utilisant la méthode hongroise pour la première matrice de coûts. Les mêmes opérations sont ensuite appliquées aux autres matrices de coûts à chaque itération, mettant en évidence l'ensemble des coûts réduits qui génèrent des sous-domaines contenant des points potentiellement non dominés. Cette approche élimine des points dominés sans qu'il soit nécessaire de les calculer explicitement. Ces opérations sont gérées au sein d'un arbre de recherche, où le processus de branchement produit des points potentiellement non dominés. De plus, une évaluation vectorielle est initiée à chaque nœud de l'arbre de recherche en calculant deux bornes vectorielles afin de réduire l'espace de recherche. L'ensemble des points potentiellement non dominés généré par l'algorithme est considéré comme une borne supérieure, tandis que le point idéal correspondant à chaque nœud sert de borne inférieure. En outre, une procédure de prétraitement est utilisée pour éliminer les sous-problèmes infaisables.

Le chapitre est organisé comme suit : tout d'abord, nous présentons les définitions et notations utilisées tout au long du chapitre et décrivons les différentes procédures employées dans l'algorithme. Un exemple illustratif est également présenté dans la Section 3. Dans la Section 4, des résultats théoriques sont démontrés pour justifier les étapes suivies et la convergence de l'algorithme. Les résultats expérimentaux sont présentés dans la Section 5. Enfin, la conclusion est donnée dans la Section 6.

3.2 Définitions et notations

Tout d'abord, le modèle mathématique du problème d'affectation multi-objectif ($MOAP$) est présenté comme suit :
Étant donné un ensemble de n tâches et n ouvriers, où chaque tâche est attribuée à un ouvrier à la fois, ainsi que p matrices de coûts $C^k = c_{ij}^k$, c_{ij}^k représente le poids de l'attribution de la tâche i à l'ouvrier j pour l'objectif k, $k \in \{1, ..., p\}$ et $p \geq 2$. Les valeurs de c_{ij}^k, sont des entiers non négatifs. L'objectif du problème est de déterminer l'ensemble des points non dominés pour le problème $MOAP$:

$$(P)\begin{cases} Min \;\; Z(x) = \;\; (z_k(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij}), \quad k \in \{1, ..., p\} \\ \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1, ..., n\} \\ \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1, ..., n\} \\ x_{ij} \in \{0, 1\} \qquad \forall i, j \in \{1, ..., n\}, \end{cases}$$

où x_{ij} est défini par :

$$x_{ij} = \begin{cases} 1 \text{ Si La tâche } i \text{ est assignée à l'ouvrier } j \\ 0 \text{ Sinon,} \qquad i, j \in \{1, ..., n\}. \end{cases}$$

Soit D l'ensemble de toutes les solutions faisables pour le problème $MOAP$, où D appartient à l'ensemble $\{0, 1\}^{n^2}$. Ici, x représente la matrice des variables de décision notée $(x_{ij})_{i,j\in\{1,...,n\}}$.

Définition 3.2.1. *Le point idéal I pour (P) est un point de l'espace des objectifs ayant pour coordonnées $(Z_1^*, ..., Z_p^*)$, où Z_k^* représente le poids minimum du problème d'affectation pour l'objectif k, $k \in \{1, ..., p\}$. En général, le point idéal I n'est pas une solution réalisable (41). Cependant, pour l'objectif k, $k \in \{1, ..., p\}$, le calcul de Z_k^* est un problème polynomial (48).*

Afin de générer l'ensemble des points non dominés pour le problème d'affectation multi-objectif $MOAP$, nous introduisons les notations suivantes, qui seront utilisées tout au long de l'article.

- Le problème d'affectation multi-objectif $MOAP$ et la méthode de recherche à voisinage variable multi-objectif $(MOVNS)$.
- ND représente l'ensemble de tous les points non dominés, et E représente l'ensemble des solutions efficaces correspondantes.
- $SPND$ désigne l'ensemble des points potentiellement non dominés, qui fait référence aux points qui ne sont dominés par aucun point trouvé précédemment. SPE représente l'ensemble associé des solutions potentiellement efficaces.
- À chaque nœud l de l'arbre de recherche dédié au problème $MOAP$, les notations suivantes sont utilisées :

— Les matrices de coûts réduits, notées $\widehat{C}_l^k, k \in \{1, ..., p\}$, sont obtenues récursivement en appliquant la méthode hongroise à la première matrice de coûts du nœud parent. Les mêmes opérations sont ensuite appliquées pour mettre à jour les autres matrices.
— Soit x_l^* la solution obtenue au nœud l.
— N_l représente l'ensemble des indices des variables hors base dans la solution x_l^*.
— H_l désigne l'ensemble des directions de descente des objectifs en x_l^*. Ces directions de descente permettent d'améliorer les valeurs des objectifs :

$$H_l = \{(i, j) \in N_l, \exists k \in \{2, ..., p\} \ ; \ (\widehat{c}_{ij})_l^k < 0\}$$

où $(\widehat{c}ij)_l^k$ désigne l'élément de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne de la matrice de coûts réduits, notée $\widehat{C}_l^k$, pour $k \in \{1, ..., p\}$.

– Les coordonnées du point idéal **local** I_l sont les valeurs optimales pour chaque objectif, mais elles prennent également en compte les contraintes considérées jusqu'au nœud l. Chaque coordonnée est la somme de deux valeurs : la première représente la contribution des variables fixées le long de la branche, avec le nœud l agissant comme feuille. La seconde valeur est calculée localement au nœud l en utilisant la définition 3.2.1, qui prend en compte les conditions et contraintes spécifiques à ce nœud. En résumé, les coordonnées du point idéal local I_l sont obtenues en combinant les variables fixées le long de la branche et les valeurs calculées localement au nœud l, tout en prenant en considération les objectifs et contraintes jusqu'à ce nœud.

3.3 Algorithme pour générer l'ensemble des affectations non dominées

Cette section présente un algorithme permettant d'obtenir l'ensemble de tous les points non dominés pour le problème $MOAP$. Dans un premier temps, nous générons quelques points potentiellement non dominés en adaptant la métaheuristique $MOVNS$ au problème $MOAP$, comme décrit dans la sous-section 3.3.1. À chaque nœud l de l'arbre de recherche, nous résolvons un problème d'affectation mono-objectif relatif au premier objectif en utilisant la méthode hongroise. Les opérations appliquées à la matrice de coûts C_l^1 sont également appliquées à toutes les autres matrices C_l^k, $k \in \{2, ..., p\}$. Nous construisons l'ensemble H_0 à partir de la première solution obtenue pour le problème $MOAP$ et effectuons le processus de branchement sur chaque élément de H_0. Lorsqu'une branche est créée en relation avec un élément (i, j) de H_0, nous fixons la variable correspondante x_{ij} à 1. En conséquence, toutes les matrices de coûts sont réduites en supprimant la ligne i et la colonne j. Le problème $MOAP$ réduit est ensuite résolu de la même manière. En récupérant toutes les valeurs des variables fixées le long de la branche qui se termine au nœud l et en les ajoutant à celles de la solution partielle courante au nœud l, nous obtenons une solution réalisable x_l pour le problème $MOAP$. Nous fixons également la borne inférieure au point idéal I_l et la borne supérieure à l'ensemble $SPND$.

Si le point idéal local I_l est dominé par un élément de l'ensemble $SPND$, le nœud l correspondant est sondé pour éviter d'explorer des sous-domaines qui ne contiennent pas de points non dominés. Sinon, les ensembles $SPND$ et SPE sont mis à jour en conséquence. Diverses règles de sondage peuvent être utilisées pour accélérer l'algorithme.

Si l'ensemble H_l est vide, il n'existe aucune direction de descente, ce qui signifie qu'aucune amélioration des objectifs n'est possible. Dans ce cas, la solution actuelle correspond à un point idéal local au nœud l. L'ensemble H_l est conçu pour ne visiter que les sous-domaines les plus prometteurs. Sinon, une procédure de prétraitement

est mise en œuvre pour réduire la taille du problème. Les sous-sections suivantes détailleront les procédures de notre méthode.

La vitesse de la méthode peut être affectée par la taille des ensembles H_l et $SPND$. Ainsi, nous avons suggéré un processus de branchement pour éviter que les solutions visitées par la méthode soient redondantes. De plus, l'ensemble $SPND$ représente une borne supérieure solide pour restreindre le domaine de recherche.

3.3.1 Génération de l'ensemble initial des points potentiellement non dominé *SPND*

Pour améliorer la convergence de la méthode, il est recommandé de commencer avec un ensemble initial $SPND$ non vide. A cette fin, nous générons des solutions supportées en optimisant une somme pondérée des fonctions objectif, ainsi que des points potentiellement non dominés en utilisant l'algorithme $MOVNS$.

En substance, la métaheuristique $MOVNS$ est une procédure de recherche locale décrite dans (30), qui modifie systématiquement la structure de voisinage pour explorer l'espace des solutions. Nous considérons quatre types différents de structures de voisinage, à savoir :

1. Insertion optimale : Sélectionner une tâche et l'assigner à l'ouvrier avec le coût le plus faible en fonction de chaque vecteur de poids.
2. 2-opt : Choisir deux ouvriers et alterner les affectations des tâches qui leur sont attribuées.
3. 3-opt : Sélectionner trois ouvriers et alterner les affectations des tâches qui leur sont attribuées.
4. 3-swap : Choisir aléatoirement trois tâches et permuter leurs affectations.

Le pseudo-code de l'algorithme $MOVNS$ pour le problème $MOAP$ est présenté dans l'Algorithme 9 ci-dessous. Tout d'abord, nous générons quelques solutions supportées en utilisant la méthode hongroise, en considérant la fonction objectif de somme pondérée avec des paramètres générés aléatoirement $\lambda_k > 0$, $k \in \{1, ..., p\}$, tels que $\sum_{k=1}^{p} \lambda_k = 1$. Soit SPP l'ensemble représentant cette collection de solutions. Les solutions de l'ensemble SSP sont ensuite soumises à une technique de perturbation qui sélectionne aléatoirement une structure de voisinage pour l'algorithme $MOVNS$, afin de produire davantage de solutions pour le problème $MOAP$. Ensuite, l'ensemble $SPND$ des points potentiellement non dominés obtenues, est mis à jour en fonction des solutions découvertes, et la procédure est répétée jusqu'à ce que la condition d'arrêt soit remplie. Nous notons que la solution pondérée minimale est définie en générant, à chaque itération, de nouveaux paramètres λ_k, $k \in \{1, ..., p\}$.

Algorithm 9: Pseudo-code de l'adaptation de $MOVNS$ pour l'algorithme $MOAP$.

Data: C^k, $k \in \{1, ..., p\}$: les matrices de coûts initiales, $Kmax$, Critère d'arrêt.
Result: $SPND$, SPE : l'ensemble initial de points potentiellement non dominés et l'ensemble correspondant de solutions potentiellement efficaces respectivement.
Générer un ensemble SSP de certaines affectations supportées en résolvant : $min\{\sum_{k=1}^{p} \lambda_k z_k(x), x \in D\}$;
for $t \leftarrow 1$ *à* $Kmax$ **do**
    Choisir une structure de voisinage NS aléatoirement ;
    Appliquer (NS) à la solution pondérée minimale de l'ensemble SSP et la mettre à jour ;
end
while *le critère d'arrêt n'est pas vérifiée* **do**
    Choisir la solution pondérée minimale S appartenant à SSP ;
    Appliquer toutes les structures de voisinage à S ;
    Mettre à jour l'ensemble SSP avec les solutions trouvées ;
end
$SPND \leftarrow SSP$;

3.3.2 Étape de branchement

Dans l'étape de branchement du problème $MOAP$ au nœud l, le nombre de nœuds enfants créés est égal au nombre d'éléments dans l'ensemble H_l. À chaque nœud enfant correspondant à (i, j) dans H_l, la variable x_{ij} devient certaine ($x_{ij} = 1$). En conséquence, la ligne et la colonne correspondantes sont éliminées des matrices de coûts actuelles. Pour créer une partition de l'ensemble des solutions réalisables au nœud l, toute variable qui a été rendue certaine ($x_{ij} = 1$) sera interdite (fixée à 0) pour les autres nœuds enfants. Cela garantit que la même variable n'est pas sélectionnée comme certaine dans différents nœuds enfants, permettant ainsi d'explorer des solutions distinctes.

3.3.3 Procédure de prétraitement

La technique de prétraitement des nœuds a été introduite par Kiziltan & Yucaoglu (47) comme un moyen de résoudre des problèmes linéaires zéro-un multi-objectif. Cette procédure est appliquée à chaque nœud de l'arbre de recherche avant de commencer un nouveau branchement, afin de déterminer si une solution potentielle aboutira à une solution dominée. Elle consiste à fixer une variable libre x_{ij} à 1 ou 0 et à examiner le résultat.

Dans notre cas spécifique, la procédure de prétraitement proposée est appliquée à certains nœuds l, où au moins une matrice de coûts C_l^k, $k \in \{2, ..., p\}$, contient des

éléments positifs ou nuls dans une ligne ou une colonne. Le point idéal local I_l est calculé pour ce nouveau sous-problème obtenu, et chaque variable dans la ligne ou la colonne correspondante est évaluée pour déterminer si elle peut être fixée à 1. Si l'une des règles de sondage est satisfaite, la variable est jugée irréalisable et le coût correspondant est fixé à $+\infty$ dans toutes les matrices de coûts. Si aucun des éléments de la ligne ou de la colonne ne peut être fixé à 1, le sous-problème est considéré comme irréalisable, et le nœud l est sondé. Cette technique de prétraitement permet d'éliminer certaines branches dans l'arbre de recherche, réduisant potentiellement l'effort de calcul en écartant les solutions irréalisables dès le départ.

3.3.4 Étape de l'évaluation

Dans l'étape d'évaluation, le problème $MOAP$ réduit au nœud l est résolu. Si le sous-problème est jugé infaisable, le nœud est sondé et supprimé. En revanche, si une solution faisable x_l est obtenue, l'ensemble H_l est construit à partir de x_l. Si H_l est vide, indiquant qu'aucun autre branchement n'est possible, le nœud l est sondé. Ensuite, le point idéal local I_l est calculé. Si I_l est dominé par au moins un élément de l'ensemble $SPND$, le nœud l est également sondé. Cela signifie que la solution obtenue n'est pas prometteuse par rapport aux solutions non dominées existantes. En revanche, si I_l n'est dominé par aucun élément de l'ensemble $SPND$, un test de dominance est effectué entre le vecteur de coût de la solution x_l^* et les ensembles $SPND$ et SPE. En fonction des résultats du test de dominance, les ensembles $SPND$ et SPE sont mis à jour en conséquence, intégrant les nouveaux points non dominées trouvées.

L'approche décrite pour résoudre le problème $MOAP$, incluant ces étapes, est appelée algorithme $BBMOAP$ tout au long de ce chapitre.

Algorithm 10: Algorithme $BBMOAP$ pour le problème $MOAP$

Data: C^k, $k \in \{1, ..., p\}$: matrices de coûts initiales.
Result: ND : l'ensemble des points non dominés pour le problème $MOAP$, et E : l'ensemble correspondant des solutions efficaces.

Étape 1 : **Initialisation**
Initialiser $SPND$ et SPE : ensembles de points potentiellement non dominés et solutions potentiellement efficaces, respectivement, en utilisant la procédure $MOVNS$. Fixer $l \leftarrow 0$ et créer le nœud racine 0 en initialisant $\widehat{C}_0^k \leftarrow C^k$, pour tout $k \in \{1, ..., p\}$.

Étape 2 : **Étape Générale**
Soit Q l'ensemble des nœuds non explorés dans l'arbre de recherche.
while *Q n'est pas vide* **do**
- Sélectionner un nœud l selon la stratégie de recherche en profondeur (DFS). Résoudre le problème d'affectation simple pour la première matrice de coûts $\widehat{C}_l^1$, en appliquant les mêmes opérations à toutes les matrices $\widehat{C}_l^k$, $k \in \{2, ..., p\}$. Soit x_l^* la solution obtenue.

Étape 2.1 : **Étape d'évaluation**
- Calculer le point idéal local I_l et construire l'ensemble H_l.

if *I_l est dominé par un élément quelconque de $SPND$, ou si $H_l = \emptyset$, ou si le sous-problème est infaisable* **then**
- Sonder et supprimer le nœud l.

end
else
if *$Z(x_l^*)$ n'est dominé par aucun élément de $SPND$* **then**
- Mettre à jour $SPND := SPND \cup \{Z(x_l^*)\}$, et $SPE := SPE \cup \{x_l^*\}$.

end
else
if *il existe $Z(v) \in SPND$ tel que $Z(v)$ est dominé par $Z(x_l^*)$* **then**
- Remplacer $Z(v)$ dans $SPND$ par $Z(x_l^*)$, et remplacer v dans SPE par x_l^*.

end
end

Étape 2.2 : **Procédure de prétraitement**
if *il existe $\widehat{C}_l^k$, $k \in \{2, ..., p\}$, contenant des éléments positifs dans une ligne ou une colonne* **then**
- Appliquer la procédure de prétraitement aux matrices de coûts $\widehat{C}_l^k$, $k \in \{1, ..., p\}$.

if *$\widehat{C}_l^1$ contient une ligne ou une colonne dont tous les éléments sont égaux à $+\infty$* **then**
- Sonder le nœud l.

end
end

Étape 2.3 : **Étape de branchement**
- Partitionner le nœud l en $|H_l|$ sous-nœuds. Pour chaque $(i, j) \in H_l$, fixer $x_{ij} = 1$ et interdire tous les autres éléments de H_l. - Mettre à jour Q. - Retourner à l'étape 2.

end
end

3.4 Exemple illustratif

Considérons le problème d'affectation bi-objectif, défini par :

$$C^1=\begin{pmatrix}1&5&1&4\\9&7&6&9\\5&8&11&5\\8&7&8&7\end{pmatrix}, \qquad C^2=\begin{pmatrix}9&6&4&8\\7&5&5&6\\5&8&7&11\\6&3&4&10\end{pmatrix}.$$

Step 0 : $l := 0$, $ND := \emptyset$.
Les solutions suivantes sont générées par la méta heuristique $MOVNS$:

$$S_1 = \{(1,1),(2,3),(3,4),(4,2)\} \text{ avec } Z(S_1) = (19,28),$$
$$S_2 = \{(1,3),(2,4),(3,1),(4,2)\} \text{ avec } Z(S_2) = (22,18),$$

Aucune des deux solutions ne domine l'autre, donc :
$SPND := \{Z(S_1), Z(S_2)\}.$; $SPE := \{S_1, S_2\}$.
Nous commençons par appliquer la méthode hongroise à la première matrice de coûts et mettons à jour les autres matrices en effectuant les mêmes opérations comme suit :

$$\widehat{C}_0^1=\begin{pmatrix}\underline{1}&5&1&4\\9&7&\underline{6}&9\\\underline{5}&8&11&5\\8&\underline{7}&8&7\end{pmatrix} \longrightarrow \begin{pmatrix}\boxed{0}&4&\emptyset&3\\3&1&\boxed{0}&3\\\emptyset&3&6&\boxed{0}\\1&\boxed{0}&1&\emptyset\end{pmatrix}$$

$$\widehat{C}_0^2=\begin{pmatrix}\underline{9}&6&4&8\\7&5&\underline{5}&6\\\underline{5}&8&7&11\\6&\underline{3}&4&10\end{pmatrix} \longrightarrow \begin{pmatrix}0&-3&-5&-1\\2&0&0&1\\0&3&2&6\\3&0&1&7\end{pmatrix} \longrightarrow \begin{pmatrix}0&-3&-5&-7\\2&0&0&-5\\0&3&2&0\\3&0&1&1\end{pmatrix}$$

Une nouvelle solution S_3 est obtenue : $S_3 = \{(1,1),(2,3),(3,4),(4,2)\}$ avec $Z(S_3) = (19,28)$. Ce point existe dans l'ensemble $SPND$. Pour la solution actuelle S_3, l'ensemble H_0 est défini comme suit : $H_0 = \{(1,2),(1,3),(1,4),(2,4)\}$.
Étapes de séparation et d'évaluation Le nœud 0 est divisé en 4 nœuds en ajoutant les contraintes $x_{ij} = 1$, $(i,j) \in H_0$, en appliquant les modifications aux matrices $\widehat{C}_0^k$, $k \in \{1,2\}$ comme suit :
Nœud 1 ($x_{12} = 1$) : on supprime la ligne 1 et la colonne 2 de $\widehat{C}_0^k$, $k \in \{1,2\}$. Le point idéal local $I_1 = (23,19)$ est dominé par $Z(S_2)$. Par conséquent, le nœud 1 est sondé.
Nœud 2 ($x_{13} = 1, x_{12} = 0$) : on supprime la ligne 1 et la colonne 3 de $\widehat{C}_0^k$, $k \in \{1,2\}$ et on fixe $\widehat{c_{120}}^1 = +\infty$ et $\widehat{c_{120}}^2 = +\infty$. Le point idéal local $I_2 = (20,18)$ n'est dominé par aucun élément de l'ensemble $SPND$. L'application de la méthode hongroise se déroule comme suit :

$$\widehat{C}_2^1 = \left(\begin{array}{cc|cc}\cancel{0}&\cancel{+\infty}&\cancel{0}&\cancel{3}\\3&1&0&3\\0&3&6&0\\1&0&1&0\end{array}\right) \longrightarrow \begin{pmatrix}3&\underline{1}&3\\0&3&0\\1&0&0\end{pmatrix} \longrightarrow \begin{pmatrix}2&\boxed{0}&2\\\boxed{0}&3&\emptyset\\1&\emptyset&\boxed{0}\end{pmatrix}$$

$$\widehat{C}_2^2 = \left(\begin{array}{cc|cc} \cancel{0} & \cancel{+\infty} & \cancel{5} & \cancel{7} \\ 2 & 0 & 0 & -5 \\ 0 & 3 & 2 & 0 \\ 3 & 0 & 1 & 1 \end{array}\right) \longrightarrow \begin{pmatrix} 2 & \underline{0} & -5 \\ 0 & 3 & 0 \\ 3 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 2 & 0 & -6 \\ 0 & 3 & -1 \\ 2 & 0 & 0 \end{pmatrix}$$

Nous obtenons la solution : $S_4 = \{(1,3),(2,2),(3,1),(4,4)\}$ avec $Z(S_4) = (20,24)$. Les ensembles $SPND$ et SPE sont mis à jour :

$$SPND := \{Z(S_1), Z(S_2), Z(S_4)\}$$

$$SPE := \{S_1, S_2, S_4\}$$

L'ensemble correspondant H_2 est défini par : $H_2 = \{(2,4),(3,4)\}$. L'application de la procédure de prétraitement à ce nœud se déroule comme suit :

- La variable $x_{41} = 1$: le point idéal local correspondant $(21,24)$ est dominé par $Z(S_1)$. Donc, $c_{41} = (+\infty, +\infty)$
- La variable $x_{42} = 1$: le point idéal local correspondant $(22,18)$ est dominé par $Z(S_2)$. Donc, $c_{42} = (+\infty, +\infty)$
- La variable $x_{44} = 1$: le point idéal local correspondant $(20,24)$ est dominé par $Z(S_4)$. Donc, $c_{44} = (+\infty, +\infty)$

Le sous-problème correspondant est infaisable, car il contient une ligne avec des valeurs infinies pour tous les éléments. Par conséquent, ce nœud est sondé.

Nœud 3 $(x_{14} = 1, x_{12} = 0, x_{13} = 0)$: on supprime la ligne 1 et la colonne 4 de $\widehat{C}_0^k$, $k \in \{1,2\}$ et on fixe : $\widehat{c_{120}}^1 = +\infty$, $\widehat{c_{120}}^2 = +\infty$ et $\widehat{c_{130}}^1 = +\infty$, $\widehat{c_{130}}^2 = +\infty$. Le point idéal local $I_4 = (22,20)$ est dominé par $Z(S_2)$. Par conséquent, le nœud 3 est sondé.

Nœud 4 $(x_{24} = 1, x_{12} = 0, x_{13} = 0, x_{14} = 0)$: on supprime la ligne 2 et la colonne 4 de $\widehat{C}_0^k$, $k \in \{1,2\}$ et on fixe : $\widehat{c_{120}}^1 = +\infty$, $\widehat{c_{120}}^2 = +\infty$; $\widehat{c_{130}}^1 = +\infty$, $\widehat{c_{130}}^2 = +\infty$ et $\widehat{c_{140}}^1 = +\infty$, $\widehat{c_{140}}^2 = +\infty$. Le point idéal local $I_5 = (22,18)$ est dominé par $Z(S_2)$. Par conséquent, le nœud 4 est sondé. Ainsi, l'ensemble final des points non dominés est $ND := SPND$ et l'ensemble associé des solutions efficaces E : $E := SPE = \{S_1, S_2, S_4\}$.

3.5 Résultats théoriques

Les résultats suivants sont démontrés pour justifier les étapes de l'algorithme $BBMOAP$:

Proposition 3.5.1. *À chaque nœud l, une fois qu'une variable x_{ij} est fixée à 1 dans un nœud fils, $(i,j) \in H_l$, elle doit prendre la valeur zéro pour tous les autres nœuds fils du nœud l afin de créer une partition de l'ensemble des solutions faisables du nœud l.*

Démonstration. Pour obtenir une partition de l'ensemble S_l des solutions faisables au nœud l, il ne doit pas y avoir de solutions faisables identiques dans deux sous-ensembles S_p et S_q de S_l, où p et q sont deux nœuds fils différents du nœud l. Ainsi,

en fixant une variable x_{ij} à la valeur 1 dans un nœud fils du nœud l, toutes les autres variables, non encore fixées, sont libres et peuvent prendre la valeur 1 ou 0. Cependant, pour tous les autres nœuds fils du nœud l, nous ne devons pas trouver de solutions faisables avec $x_{ij} = 1$. □

Proposition 3.5.2. *Si le point idéal local I_l est dominé par au moins un élément de $SPND$, alors le nœud l est exploré et éliminé.*

Démonstration. Il est évident que le point idéal local I_l domine toutes les solutions du sous-domaine correspondant au nœud l. Si au moins un élément de $SPND$ domine I_l, alors par transitivité, il domine toutes les solutions correspondant au nœud l. Par conséquent, l'exploration de ce nœud est inutile. □

Proposition 3.5.3. *S'il existe un $k \in \{2, \ldots, p\}$ tel que la matrice des coûts réduits $\widehat{C}_l^k$ ait tous les éléments d'une ligne ou d'une colonne égaux à $+\infty$, alors le sous-problème correspondant est infaisable.*

Démonstration. Une matrice des coûts réduits $\widehat{C}^k$ avec tous les éléments d'une ligne i' ou d'une colonne j' égaux à $+\infty$ signifie que la tâche i' ou l'ouvrier j' ne peut pas être assigné à un ouvrier j ou une tâche i, respectivement. Cela implique que la contrainte d'affectation est violée, rendant le sous-problème infaisable. □

Théorème 3.5.1. *Si $H_l = \emptyset$, alors le sous-domaine correspondant ne contient aucun point non dominé.*

Démonstration. Si l'ensemble H_l est vide, alors aucun objectif ne peut être amélioré davantage en x^l, par définition de l'ensemble H_l. Le point correspondant $Z(x^l)$ représente alors un point idéal local, et le nœud l actuel doit être sondé. □

Théorème 3.5.2. *L'algorithme $BBMOAP$ génère tous les points non dominés pour le problème $MOAP$ et converge en un nombre fini d'itérations.*

Démonstration. Tout d'abord, l'étape de partitionnement à chaque nœud l dans l'arbre de recherche garantit une division des sous-domaines obtenus dans les nœuds fils, conformément à la Proposition 1. Ce partitionnement assure que chaque nœud fils explore un sous-ensemble distinct de l'espace des solutions, favorisant ainsi la diversité et évitant les redondances. De plus, lors de la recherche de points potentiellement non dominés, l'algorithme $BBMOAP$ s'assure qu'aucun point non dominé n'est omis, comme le confirment les Propositions 2 et 3, ainsi que le Théorème 1. Ces propositions et théorème justifient théoriquement l'efficacité de l'algorithme pour capturer tous les points non dominés. Il est également important de noter que le nombre de nœuds créés dans l'arbre de recherche est fini. Par conséquent, l'algorithme $BBMOAP$ s'arrête après un nombre fini d'itérations. Cette condition de terminaison garantit que l'algorithme explore de manière exhaustive tout l'espace des solutions, aboutissant à l'identification de tous les points non dominés.

En résumé, l'algorithme $BBMOAP$ garantit à la fois le partitionnement des sous-domaines via le partitionnement et la découverte de tous les points non dominés. Avec un nombre fini d'itérations, l'algorithme atteint la terminaison et fournit un ensemble complet de points non dominés pour le problème donné. □

3.6 Résultats Numériques

L'algorithme $BBMOAP$, spécifiquement conçu pour le $MOAP$, ainsi que la méthode en deux phases pour les cas bi-objectif (58) et tri-objectif (60) du problème $MOAP$, et la méthode proposée par Özlen et al. (56) , ont été implémentés dans un programme Python. Ce programme a été exécuté sur un ordinateur Dell équipé d'un processeur Intel Core $i5-7300U$ et de 8 GB de RAM. Afin d'évaluer la performance de ces méthodes, une série de 10 instances avec différentes tailles de matrices de coûts $(n \times n)$ et un nombre d'objectifs (p) variant de 2 à 5 ont été utilisées. Les coefficients des fonctions objectifs ont été générés aléatoirement dans l'intervalle $[0, 20]$ selon une distribution uniforme. Lors des expériences, les calculs ont été interrompus pour les instances dont le temps d'exécution dépassait une limite de trois heures. Cette contrainte visait à garantir que l'exécution du programme reste dans un délai raisonnable. Ces expériences avaient pour but d'évaluer l'efficacité et la performance des méthodes implémentées pour résoudre le $MOAP$, en tenant compte de diverses tailles de problèmes et du nombre d'objectifs.

Les résultats présentés dans toutes les tables montrent clairement que le nombre de points non dominés $nbSND$ augmente très rapidement avec la taille des instances. De plus, il est à noter que les résultats des Tables 3.1 et 3.2 indiquent que les méthodes citées dans (58) et (60) sont très compétitives dans les cas bi-objectif et tri-objectif pour $n \leq 30$. Cependant, notre méthode est plus rapide pour $n \geq 40$. Par ailleurs, tous les résultats obtenus montrent que l'algorithme $BBMOAP$ surpasse la méthode décrite dans (56). La figure 3.1 illustre les résultats obtenus. Cela s'explique par le fait que l'algorithme $BBMOAP$ exploite la structure du problème en ne visitant que les régions prometteuses à l'aide de la définition des directions de descente des fonctions objectifs dans le processus de branchement, ainsi que le rôle des règles de sonder. En outre, la taille des sous-problèmes est progressivement réduite à chaque niveau de l'arbre de recherche en définissant les variables de branchement par la valeur 1 et en interdisant les variables déjà visitées dans le même parent. Toutes ces opérations, ayant un temps polynomial, servent également à réduire la taille de l'ensemble des variables de descente.
Par ailleurs, la contribution d'un ensemble $SPND$ de points potentiellement non dominés, utilisé comme borne supérieure vectorielle pour le $MOAP$ dès le début de l'algorithme, évoluant dynamiquement et mis à jour en temps polynomial au cours de la recherche, a eu un effet positif sur l'amélioration du temps CPU en permettant une sondage tôt des nœuds créés.
La comparaison entre les trois méthodes est réalisée comme suit

TABLE 3.1 – Résultats de comparaison entre les méthodes de Przybylski et *al.* (60), Özlen et *al.* (56) et l'algorithme $BBMOAP$ avec le temps CPU en secondes. ($p = 2$)

n	$nbSND$	Przybylski et *al.* (58)	Özlen et *al.* (56)	$BBMOAP$
5	8.2	10^{-3}	10^{-3}	10^{-3}
10	18.4	10^{-3}	0.662	0.124
15	28.7	0.207	8.365	4.203
20	49.9	1.012	48.875	6.268
30	71.3	11.512	152.362	12.136
40	98.6	25.178	288.515	23.833
50	123.5	50.367	396.208	48.737

TABLE 3.2 – Résultats de comparaison entre les méthodes de Przybylski et *al.* (60), Özlen et *al.* (56) et l'algorithme $BBMOAP$ avec le temps CPU en secondes. ($p = 3$)

n	$nbSND$	Przybylski et *al.* (60)	Özlen et *al.* (56)	$BBMOAP$
5	23.4	10^{-3}	0.215	0.119
10	187.1	0.278	3.663	1.085
15	311.6	1.052	24.393	2.261
20	475.3	10.633	125.205	16.424
30	802.7	23.225	279.775	24.663
40	1044.8	51.351	625.362	45.912
50	1620.2	126.305	907.243	103.610

TABLE 3.3 – Résultats expérimentaux pour la méthode de Özlen et *al.* (56) et la méthode $BBMOAP$ avec le temps CPU en secondes. ($p = 4$)

n	$nbSND$	Özlen et *al.* (56)	$BBMOAP$
5	45.1	3.634	0.305
10	314.7	150.557	8.804
15	468.2	449.321	22.356
20	814.4	1114.419	81.992
30	1031.6	3271.254	184.361
40	1992.7	×	292.665
50	3134.3	×	409.749

TABLE 3.4 – Résultats de comparaison entre la méthode de Özlen et *al.* (56) et la méthode $BBMOAP$ avec le temps CPU en secondes. ($p = 5$)

n	$nbSND$	Özlen et *al.* (56)	$BBMOAP$
5	89.3	8.2	0.675
10	592.9	1721.4	15.236
15	827.1	3576.8	40.492
20	1549.5	8052.3	153.330
30	2742.1	×	245.215
40	3217.9	×	413.787
50	4693.2	×	756.469

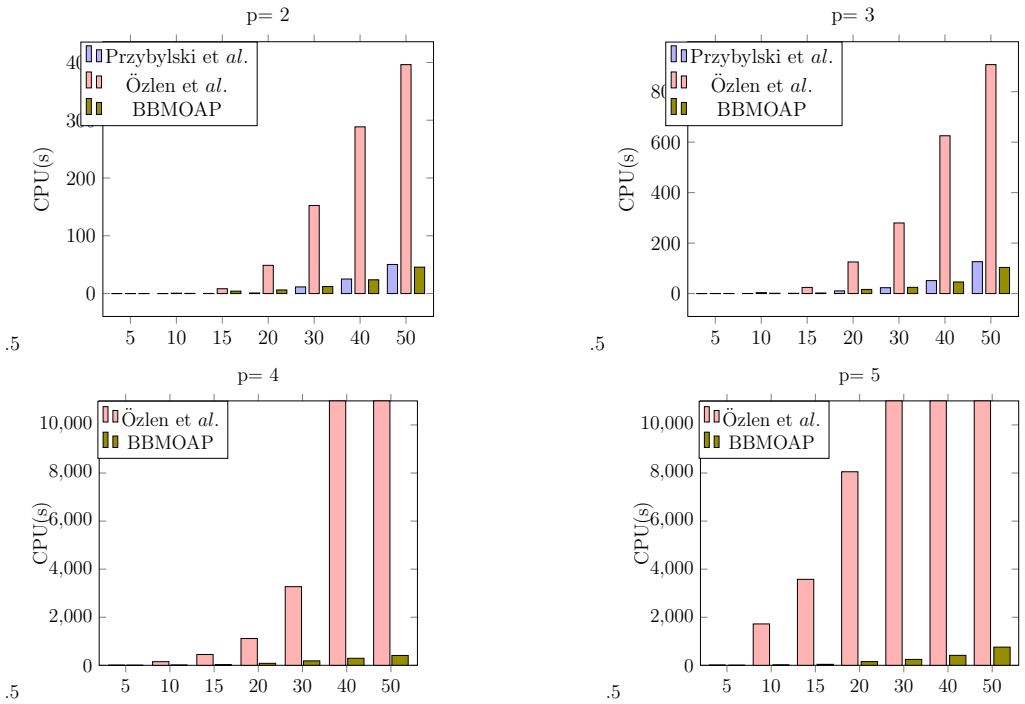


FIGURE 3.1 – Comparaison du temps CPU moyen entre les méthodes en fonction de la taille des instances n et pour le nombre de critères $p = 2, 3, 4, 5$.

3.7 Conclusion

Ce chapitre présente une nouvelle méthode exacte pour résoudre le problème d'affectation multi-objectif en combinant le principe de la méthode de séparation et évaluation progressive avec la méthode hongroise. Le problème d'affectation multi-objectif ($p \geq 2$) est connu pour être NP-difficile, et les études précédentes se sont principalement concentrées sur les cas bi-objectif et tri-objectif. Cependant, ce travail présenté dans ce chapitre se distingue comme étant la seule étude qui aborde le problème d'affectation multi-objectif avec plus de trois fonctions objectifs. En prenant en considération la structure spécifique du problème, il intègre l'adaptation de la recherche à voisinage variable multi-objectif pour générer de bons vecteurs de bornes supérieures initiales. Ceci est couplé avec l'évaluation exacte du point idéal local, qui sert de vecteur de borne inférieure. Cette combinaison s'avère efficace pour éviter la recherche dans les zones non prometteuses dès les premières étapes du processus de recherche. De plus, les directions de descente des fonctions objectifs permettent d'éviter la recherche dans les zones qui ne contiennent pas de points non dominés, améliorant ainsi l'efficacité de l'algorithme.

Les résultats expérimentaux montrent que la méthode proposée surpasse la méthode proposée par Özlen et *al.* ainsi qu'elle est compétitive par rapport aux deux méthodes

proposées par Przybylski et *al*. pour la plupart des instances testées. Ces résultats valident l'efficacité de l'approche et mettent en évidence sa supériorité pour résoudre le problème d'affectation multi-objectif, en particulier lorsqu'il s'agit d'un nombre plus élevé de fonctions objectifs.

Chapitre 4

Optimisation d'un critère linéaire sur l'ensemble des solutions efficaces de $MOAP$

4.1 Introduction

Dans de nombreux cas pratiques, l'énumération exhaustive de l'ensemble des solutions efficaces n'est pas toujours souhaitable. En effet, les fonctions objectifs impliquées sont généralement conflictuelles, ce qui engendre un ensemble de solutions efficaces (ou points non dominés) de grande taille, rendant le choix final, basé sur les préférences du décideur, particulièrement complexe. Pour pallier cette difficulté, on formule un problème d'optimisation globale visant à maximiser une fonction représentant les préférences du décideur au sein de l'ensemble des solutions efficaces. Toutefois, ce problème s'avère complexe sur le plan computationnel (9), (80), (29), en raison notamment de la structure potentiellement non convexe de l'ensemble des solutions réalisables (solutions efficaces dans ce cas).

Les problème d'optimisation d'un critère sur l'ensemble des solution efficaces (points non dominées) et ses applications pratiques ont été discuté dans la littérature (8), (41), (66), (53). En particulier, Benson (9) montre que dans certains problèmes, le problème d'optimisation d'un critère sur l'ensemble de solutions efficaces est plus réaliste que les problèmes d'optimisation multi-objectif. En outre, la résolution du problème d'optimisation d'une fonction linéaire sur l'ensemble des solutions efficaces évite l'énumération de toutes les solutions (l'explosion combinatoire). On s'intéresse au problème d'optimisation d'une fonction linéaire sur l'ensemble des solutions efficaces des problèmes $MOILP$. La première étude de ce problème a été réalisé par N.C.Nguyen (53) qui a proposé juste une borne supérieure de la valeur optimale du critère à optimiser. Ensuite, plusieurs méthodes ont été proposer pour résoudre ce problème en évitant l'énumération explicite des solutions efficaces ou bien les points non dominés. Abbas et Chaabane (1) présentent une méthode itérative dans l'espace de décision, où la valeur du critère à optimiser est améliorée à chaque étape en impo-

sant différentes coupes. Jorge (43) a proposé une méthode, basée sur la résolution de problèmes mono-objectif en nombres entiers, où de nouvelles variables et contraintes sont ajoutées à chaque itération, afin de générer une solution non dominée par les solutions déjà exploré, jusqu'à l'obtention d'une solution efficace optimale. Une autre méthode est proposée par Chaabane et Pirlot (12) dans l'espace des critères, basée sur l'optimisation progressive d'une norme pondérée de Tchebychev (11) augmenté, en ajoutant des contraintes successivement pour réduire le domaine des solutions réalisables et améliorer la valeur du critère à optimiser. Boland et *al.* (10) modifient l'algorithme de Jorge (43) en utilisant une procédure de décomposition et de recherche. L'algorithme est également appliqué pour trouver le point nadir composé des pires valeurs de chaque critère sur l'ensemble non dominé. Les résultats de l'expérimentation montrent que l'algorithme de Boland et *al.* surpasse l'algorithme de Jorge. L'algorithme présenté par Lokman (51) constitue une autre variante de celui proposé par Jorge. L'auteur introduit deux algorithmes de décomposition et de recherche, appelés DSA et DSAm, conçus pour optimiser une fonction linéaire f sur l'ensemble des points non dominés d'un problème d'optimisation multiobjectif (MOP), où la fonction f est définie comme une combinaison linéaire des critères du MOP. Afin de réduire les calculs et d'éviter de résoudre des problèmes de plus en plus contraints à chaque itération, ces algorithmes décomposent l'espace des critères en sous-problèmes plus petits. Ces sous-problèmes sont construits en imposant des bornes sur chaque critère à partir des points non dominés déjà identifiés. Après avoir déterminé, dans chaque sous-problème, le point qui maximise la fonction f, l'algorithme sélectionne celui ayant la valeur la plus élevée de f, puis évalue son efficacité. Lokman a introduit l'algorithme DSA, fondé sur une version antérieure proposée par Jorge, en y intégrant plusieurs mécanismes d'amélioration. Toutefois, lors de l'optimisation de la fonction objectif, qui est une combinaison linéaire des critères du programme MOP, l'apparition fréquente de points dominés au cours des itérations devient plus marquée lorsque certains critères possèdent des coûts négatifs. Cette situation entraîne une complexité computationnelle accrue. Conscient de cette limitation, l'auteur a développé un second algorithme, nommé DSA1, qui se concentre sur la maximisation du premier critère de MOP ayant le plus grand coefficient positif. Une étude comparative sur des instances du problème $MOAP$ a montré que DSA1 surpasse systématiquement DSA en termes de performance.

4.2 Présentation du problème

Le problème d'optimisation d'un critère Φ sur l'ensemble des solutions efficaces d'un problème $MOILP$ qui se formule par :

$$(MOILP) \begin{cases} Max\ Cx \\ \\ x \in D \end{cases} \tag{4.1}$$

où $D = S \cap \mathbb{Z}^n$, $\mathbb{Z}$ étant l'ensemble des entiers relatifs. S est borné et convexe $S = \{x \in \mathbb{R}^n | Ax \leq b,\ x \geq 0\}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, C est une matrice de dimension

$p \times n$ d'élément réels, ses vecteurs lignes sont $c^k \in \mathbb{R}^n$, $k \in \{1, ..., p\}$. Le problème principal est donné par le modèle mathématique suivant :

$$(OP) \begin{cases} Max\ \Phi(x) = dx \\ \\ x \in E \end{cases} \tag{4.2}$$

où E est l'ensemble des solutions efficaces du problème 4.1, $d = (d_j)_{1 \leq j \leq n}$ est un vecteur ligne de dimension n.
Soit le problème relaxé de 4.2 :

$$(OP_r) \begin{cases} Max\ \Phi(x) = dx \\ \\ x \in D \end{cases} \tag{4.3}$$

Le problème 4.2 est un problème d'aide à la décision, est devenu l'un des plus importants et des plus intéressants domaines de la programmation multi-objectif et de l'optimisation globale. Pour le cas discret, peu de travaux existent dans la littérature.

Naïvement, ce problème peut être résolu en énumérant la liste des solutions efficace du problème 4.1 et on choisit la solution qui correspond à la valeur maximale de Φ. Cette méthode n'est pas appropriée due à la difficulté de générer l'ensemble E qui peut être de taille exponentielle. Le problème 4.2 est difficile à traiter (27),(81). Les difficultés particulières rencontrées dans sa résolution sont dues à :

- L'optimisation sur un ensemble inconnue a priori.
- La nature du domaine de décision : non convexe, discret.
- La présence des solutions supportées et non supportées sur le front Pareto.

4.2.1 Résultats fondamentaux

Théorème 4.2.1 (Isermann 1974)**.** *Soit x^* un point arbitraire de S, x^* est une solution efficace du problème $MOLP$ si et seulement si la valeur optimale θ^* est nulle dans le programme linéaire suivant :*

$$(T_{x^*}) \begin{cases} Max\ \theta = \sum_{k=1}^{p} \psi_k \\ c^k x - \psi_k = c^k x^* \\ \\ x \in S \\ \\ \psi_k \geq 0,\ \forall k \in \{1, ..., p\} \end{cases} \tag{4.4}$$

Démonstration. Raisonnons par absurde pour montrer les deux implications.
Si x^* est une solution efficace x^* de $MOILP$, alors $\theta = 0$.
Supposons que $\theta \neq 0$, alors $\exists k \in 1, ..., p$, tel $\psi_k > 0$ et $\exists y \in S$ tel que Cy domine

Cx^*, ce qui contredit l'hypothèse de x^* est efficace.
Si $\theta = 0$, alors x^* est une solution efficace x^* de $MOLP$.
On suppose maintenant que x^* n'est pas efficace, alors $\exists y \in S$ tel que $Cy \geq Cx^*$ et $Cy \neq Cx^*$, donc $\exists k \in \{1, ..., p\}$, tel $\psi_k > 0$, ce qui est contradictoire avec l'hypothèse que $\theta = 0$. □

Le problème 4.4 est souvent utilisé pour tester l'efficacité d'une solution réalisable donnée.

Théorème 4.2.2 (Eker and Kouada 1975 (20)). *Si T_{x^*} possède une valeur maximale finie non nulle en un point réalisable $\tilde{x}$, alors $\tilde{x}$ est efficace.*

Ce théorème montre que T_{x^*} génère une solution efficace même si x^* ne l'est pas.

Théorème 4.2.3 (Eker and Kouada 1975 (20)). *Si T_{x^*} n'admet pas une solution optimale finie, alors l'ensemble E des solutions efficaces du problème $(MOLP)$ est vide.*

Ces théorèmes ont été énoncé dans le cas des problèmes $(MOLP)$, et il restent valables pour le cas des problèmes $MOILP$. Dans les algorithmes qui seront présentés dans ce chapitre, la résolution du problème 4.4 nous permet de tester l'efficacité d'une solution réalisable x^*, appelée souvent "Test d'efficacité". Pour le cas discret, il faut juste changer le domaine S (continu) par D (discret) dans le problème 4.4.

4.3 Méthodes exactes pour la résolution de (OP)

4.3.1 Méthode de Jorge

Dans l'espace de décision, la méthode (44) résout le problème 4.2 et produit une solution optimale sans avoir à énumérer explicitement toutes les solutions efficaces. Initialement, la méthode commence par la résolution du problème relaxé 4.3, la solution optimale trouvée est soumise au test efficacité. Dans le cas générale, la première solution obtenue n'est pas efficace, donc une nouvelle solution efficace $\hat{x}^1$ est générée par le test d'efficacité, dont l'image domine l'image de la solution courante.

Il peut y avoir des solutions efficaces qui ont le même vecteur critère dans l'espace des critères. Pour cela, le problème 4.5 est résolu pour optimiser le critère principal sur toutes les solutions alternées à $\hat{x}^1$.

$$(T_1') : \ max\{dx | Cx = C\hat{x}^1, x \in D\}. \tag{4.5}$$

À chaque itération, le problème (R_l) est résolu pour optimiser le critère principal sur le domaine réduit, en utilisant des contraintes supplémentaires sur des variables entières, pour éliminer les points dominées par le point courant et fournir un autre point qui est non dominée par les points précédemment visitées. Le processus se poursuit jusqu'à l'obtention d'une solution efficace.

(R_l) : $max\{dx|Cx = C\hat{x}^1, x \in D - \bigcup_{s=1}^{l} D_s\}$.
où $D_s = \{x \in \mathbb{Z}^n | Cx \leq C\hat{x}^s\}$, avec $\hat{x}^s$, $s = \{1, ..., l\}$ les solutions obtenues jusqu'à l'étape l.

Le pseudo code de la méthode est donné par 11.

Algorithm 11: Algorithme de Jorge (43)

Entrées :
$A_{(n\times m)}$: matrice des contraintes.
$b_{(m)}$: vecteur du second membre.
$d_{(n)}$: vecteur du critère principal.
$C_{(p\times n)}$: matrice des critères.
Sorties :
x_{opt}, Φ_{opt} : la solution (respectivement la valeur du critère principal) optimale du problème.
Étape 0
$\Phi_{inf} = -\infty$, $\Phi_{sup} = +\infty$, $l = 1$.
Résoudre le problème (OP_r) relaxé. - Si (OP_r) n'est pas réalisable alors Terminer, le problème (OP) n'a pas de solutions.
- Soit x^l la solution optimale de (OP_r).
Étape 1
- Tester l'efficacité de x^l.
- Si x^l est alors terminer et retourner $x_{opt} = x^l$, $\Phi_{opt} = dx^l$.
- Sinon poser $\phi_{sup} = dx^l$ et aller à l'étape 2.
Étape 2
soit $\hat{x}^l$ la solution optimale du test d'efficacité.
- Résoudre le problème (T'_l), soit $\tilde{x}^l$ la solution optimale obtenue.
- Si $d\tilde{x}^l > \Phi_{inf}$ alors $\Phi_{inf} = d\tilde{x}^l$, $x_{opt} = \tilde{x}^l$ et aller à l'étape 3.
- Sinon $\Phi_{inf} = \Phi_{sup}$, terminer x_{opt} est la solution optimale de (OP).
Étape 3
- Résoudre le problème (R_l).
- Si R_l n'est pas réalisable alors terminer x_{opt} est la solution optimale de (OP).
- Sinon soit x^{l+1} la solution optimale de (R_l).
  - Si $dx^{l+1} \leq \Phi_{inf}$ alors terminer x_{opt} est la solution optimale de (OP).
  - Sinon poser $l = l + 1$ et aller à l'étape 1.

Fin.

4.3.2 Méthode de Ouaïl et *al.*

Cette méthode (54) est basée sur le principe du branch-and-bound pour résoudre (OP) sur l'ensemble des solutions efficaces de $MOILP$, sans avoir à énumérer tous

les élément de cet ensemble. Avant de donner la description de cette méthode, il est nécessaire de définir :

- Le domaine S, $S = \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0\}$
- Le programme linéaire correspond au nœud l :

$$(OP_l) \begin{cases} Max\ \Phi(x) \\ \\ x \in S_l \end{cases} \tag{4.6}$$

avec $S_0 = S$ et S_{l+1} est obtenu à partir de S_l en ajoutant des coupes efficaces décrites ci-dessous. Pour ce faire, soit x_l^* la première solution entière obtenue en résolvant (OP_l) en utilisant, éventuellement, le processus de branchement bien connu dans la méthode branch-and-bound.

- $B_l(N_l)$: l'ensemble des indices des variables de base (hors base) correspondant à x_l^*.
- $\bar{c}_i^j$: la $j^{ème}$ composante du vecteur de coût réduit de c_i pour chaque $i \in \{1, ..., p\}$ dans le dernier tableau du simplexe.
- Le premier type de coupe efficace est définit comme suit :

$$\sum_{j \in H_l} x_j \geq 1$$

avec $H_l = \{j \in N_l | \exists i \in \{1, ..., p\}; \bar{c}_i^j < 0\} \cup \{j \in N_l | \forall i \in \{1, ..., p\}; \bar{c}_i^j = 0\}$ Cette coupe permet de supprimer les solution non efficaces sans avoir les visiter.

- Le deuxième type de coupe efficace est définit comme suit :

$$\Phi(x) \geq \Phi_{opt}$$

Cette coupe permet de supprimer les solution qui ont des profits faibles.

- Deux types d'ensembles (notés S_{l+1}^1 et S_{l+1}^2) au nœud l comme suit :

$$S_{l+1}^1 = \{x \in S_l | \sum_{j \in H_l} x_j \geq 1\}$$

$$S_{l+1}^2 = \{x \in S_l | \Phi(x) \geq \Phi_{opt}\}$$

- Eff_l est l'ensemble des solutions potentiellement efficaces obtenus jusqu'à l'étape l. Cet ensemble est mis à jour en considérant le vecteur $z(x_l^*)$; si $z(x_l^*)$ est non dominé par aucun point $z(x),\ x \in eff_{l-1}$ alors $eff_l = eff_{l-1} \cup \{x_l^*\}$ et supprimer toutes les solutions dont les vecteurs image sont dominés par $z(x_l^*)$ ou par $z(y_l)$, y_l est la solution du problème (T_{x^*}).

Le pseudo code de la méthode est donné par 12.

Algorithm 12: Algorithme de Ouaïl et *al.* (54)

Entrées :
$A_{(n\times m)}$: matrice des contraintes.
$b_{(m)}$: vecteur du second membre.
$d_{(n)}$: vecteur du critère principal.
$C_{(p\times n)}$: matrice des critères.
Sorties :
x_{opt}, Φ_{opt} : la solution (respectivement la valeur du critère principal) optimale du problème.
Étape 0
$\Phi_{opt} = +\infty$, $l = 0$, $eff_l = \emptyset$.
Résoudre le problème (OP_r) relaxé. - Si (OP_r) n'est pas réalisable alors Terminer, le problème (OP) n'a pas de solutions.
- Soit x^l la solution optimale de (OP_r).
Étape 1
Tant qu'il y a des nœuds non sondés, résoudre le problème (OP_l) par la méthode du simplexe ou simplexe dual tout dépend du signe du vecteur b. Aller à l'étape 2.1.
Étape 2 (tests)
2.1 **test de réalisable** Si (OP_l) est irréalisable alors terminer et le nœud l est sondé. Sinon soit x_l^* la solution obtenue, si $\Phi_{opt} \geq \Phi(x_l^*)$, le nœud l est sondé sinon aller à l'étape 2.3.
2.2 **test de** Si la solution x_l^* est entière, mettre à jour l'ensemble eff_l et aller à l'étape 2.3, sinon à l'étape 3.
2.3 **test d'efficacité** Si x_l^* ne peux être ajoutée à l'ensemble eff_l, alors x_l^* n'est pas efficace et aller à l'étape 4. sinon résoudre le problème (T_{x^*}), si x_l^* est efficace alors mettre à jour x_{opt} et Φ_{opt} ; le nœud l est sondé jusqu'à le critère Φ ne peut pas être amélioré, sinon (x_l^* n'est pas efficace) soit y_l la solution de (T_{x^*}), mettre à jour x_{opt}, Φ_{opt} et eff_l. Aller à l'étape 4.
Étape 3 (Processus de branchement)
- Choisir une cordonnée x_j de x_l^*, telle que $x_j = \alpha_j$ avec α_j la partie fractionnaire. Ensuite, (OP_l) est subdivisé en deux sous problèmes ; ajouter la contrainte $x_j \leq \lfloor \alpha_j \rfloor$ pour obtenir le premier programme OP_{l_1} et la contrainte $x_j \geq \lfloor \alpha_j \rfloor + 1$ et construire S_{l+1}^2 pour avoir le deuxième programme OP_{l_2}, tel que $l_1 > l + 1$, $l_2 > l + 1$ et $l_1 \neq l_2$. aller à l'étape 1. En effet, comme l'arbre de recherche est manipulé suivant la stratégie recherche en profondeur d'abord, la coupe $\Phi(x) \geq \Phi_{opt}$ est ajoutée au branche l_2.
Étape 4 (Coupes efficaces)
Construire l'ensemble H_l. Si $H_l = \emptyset$ alors le nœud l est sondé, sinon déterminer l'ensemble S_{l+1}^1 et aller à l'étape 1.
Fin.

4.3.3 Méthode de Lokman

L'auteur (51) a présenté deux méthodes qui optimisent une fonction linéaire donnée sur l'ensembles des points non-dominée d'un $MOIP$. Les algorithmes génèrent itérativement des points non dominés et réduisent l'ensemble des solutions réalisables en excluant non seulement les régions dominées mais aussi les régions inférieures par rapport à la fonction linéaire. A chaque itération, l'ensemble des solutions réalisables réduit est décomposé en sous-ensembles et une recherche est effectuée sur tous ces sous-ensembles pour trouver un nouveau point. Alors que le premier algorithme recherche le point qui maximise la fonction linéaire comme dans les algorithmes de Jorge(43) et Boland et al.(10), le deuxième algorithme choisit l'un des critères basés sur la fonction linéaire et trouve le point maximisant ce critère. La procédure de décomposition et de recherche est accompagnée de mécanismes afin d'explorer efficacement l'espace des critères. De plus, les algorithmes sont conçus comme des algorithmes d'approximation avec garantie de performance sur la qualité de la solution.

Le premier algorithme DSA qui partitionne l'ensemble des solutions possibles de $[P^n]$ en sous-ensembles et cherche une solution optimale sur tous ces sous-ensembles. A chaque itération $n > 1$, $[P^n]$ génère un point qui maximise la fonction Φ et aussi non dominé par les points précédemment générés. Dans DSA, l'espace des critères réalisables est divisé en sous-ensembles qui sont formé en imposant des bornes à chaque critère. Dans chaque sous-ensemble, on trouve le point non dominé (le cas échéant) qui maximise la fonction Φ. Après avoir considéré tous les sous-ensembles, le point qui correspond à la valeur maximale de Φ est le point optimal de $[P^n]$.

Chaque sous-ensemble est caractérisé à l'aide d'un vecteur $k = (k_1, ..., k_p)$ où $0 \leq k_i \leq n-1$ pour tout $i \in \{1, ..., p\}$. Si $k_i = 0$, il n'y a pas de borne inférieure supplémentaire pour le critère i et donc la borne inférieure M_i (la borne inférieure initiale du critère i) est définie. Sinon, on utilise la $i^{ème}$ valeur de critère de $z_i^k \in \mathbb{Z}^n$ pour imposer une borne inférieure pour le critère i. Si $b^k = (b_1^k, ..., b_p^k)$ désigne le vecteur de borne inférieure caractérisé par k, alors nous définissons le sous-modèle comme suit :

$$[P^k] \begin{cases} Max\ \Phi(x) = \sum_{i=1}^{p} v_i z_i \\ \quad z_i(x) \geq b_i^k \qquad i \in \{1, ..., p\} \\ \quad x \in D \end{cases}$$

où

$$b_i^k = \begin{cases} M_i & k_i = 0 \\ z_i^{k_i} + \epsilon & k_i \neq 0 \end{cases}$$

Un ensemble de k vecteurs, K^n, est identifié à chaque itération de sorte que l'union de les ensembles des possibles des sous-modèles correspondants, $Z[P^k]$, seront équivalents à l'ensemble des possibles de $[P^n]$ noté par $Z_{[P^n]}$, c-à-dire $Z_{[P^n]} = \bigcup_{k \in K^n} Z_{[P^k]}$. Afin de générer tous les $k \in K^n$, le schéma de décomposition de Lokman et Koksalan (51) est utilisé.

L'algorithme DSA est conçu de manière à pouvoir se rapprocher de la solution optimale Φ_{opt} avec un niveau de précision prédéfini, $g^* \geq 0$. Pour ce faire, il conserve une borne inférieure Φ_l et une borne supérieure Φ_u pour Φ pendant l'algorithme et s'arrête lorsque $g = \dfrac{|\Phi_u - \Phi_l|}{\Phi_l} \leq g^*$. Différent des algorithmes existants, l'algorithme DSA incorpore ces informations dans le processus de résolution et révise $[P^k]$ pour chaque $k \in K^n$ comme suit :

$$[S^k] \begin{cases} Max\ \Phi(x) = \sum_{i=1}^{p} v_i z_i \\ z_i(x) \geq b_i^k \qquad i \in \{1, ..., p\} \\ \sum_{i=1}^{p} v_i z_i \geq \Phi_l + g^* |\Phi_l| \\ \sum_{i=1}^{p} v_i z_i \leq \Phi_u \\ x \in D \end{cases}$$

A l'itération n, le meilleur point non dominé rencontré est appelé titulaire, noté par z^{inc}. Soit z'^n le point optimal de $[p^n]$, $\Phi_l = z^{inc}$, $\Phi_u = z'^n$.

Pour tester le dominance de point donné z'^n, il est nécessaire de résoudre le modèle mathématique suivant :

$$[E_w^n] \begin{cases} Max\ \sum_{i=1}^{p} w_i z_i \\ z_i(x) \geq z'^n \qquad i \in \{1, ..., p\} \\ x \in D \end{cases}$$

où $w = (w_1, ..., w_p)$, désigne un vecteur de poids satisfaisant $w_i > 0$ pour tout $i \in \{1, ..., p\}$. Si z'^n est un point dominée, $[E_w^n]$ génère un point non dominé, z^n, qui domine z'^n. z^{inc} et Φ_l sont mis à jour pour la prochaine itération.

Le pseudo code de *DSA* est donné par l'algorithme 13.

Algorithm 13: Le pseudo code de DSA.

Étape 1 : Initialisation
$n = 1, w_i = v_i - \min_{i \in \{1,...,p\}} v_i + 1$ $i \in \{1, ..., p\}$ et $z'^1 = \max_{z(x) \in Z} \Phi(x)$.

Étape 2 : Vérification de la non-dominance
Résoudre $[E_w^n]$ et dénotons le point optimal par z^n. Si $z^n = z'^n$, alors on pose $\Phi_u = \Phi_l = \Phi(z^n)$, $z^{inc} = z^n$ et passez à l'étape 5. Sinon, passez à l'étape 3.

Étape 3 : Mise à jour des bornes
Fixer $\Phi_u = \Phi(z'^n)$. Si $\Phi(z^n) \geq \Phi_l$, alors posez $\Phi_l = \Phi(z^n)$ et $z^{inc} = z^n$. Si $g = \dfrac{|\Phi_u - \Phi_l|}{|\Phi_u|} \leq g^*$, passez à l'étape 5.
Sinon, passez à l'étape 4.

Étape 4 : nouvelle génération de points
Fixer $n = n + 1$ et mettre à jour K^n. Résolvez $[S^k]$ correspondant à chaque $k \in K^n$ et notez le point optimal comme z^k. Si $[S^k]$ est irréalisable pour tout $k \in K^n$, passez à l'étape 5. Sinon, trouvez $k = \underset{k \in K^n,\ Z_{[S^k]} \neq \emptyset}{\operatorname{argmax}} \Phi(z^k)$ et $z'^n = z^k$. Passez à l'étape 2.

Étape 5 : Terminer
Arrêter. $z^* = z^{inc}$ maximise (se rapproche) de la fonction Φ sur l'ensemble non-dominé, $\Phi_{opt} = \Phi(z)$ (avec un niveau de précision souhaité, g^*).

Nous donnons ensuite le deuxième algorithme noté par DSA_m comme suit :

Algorithm 14: Le pseudo code de DSA_m

Étape 1 : Initialisation

- $m = \underset{i\in\{1,...,p\}}{\text{argmax}}\, v_i, w_i = v_i - \min\limits_{i\in\{1,...,p\}} v_i + 1\ i \in \{1,...,p\}$,
- $z'^0 = \max\limits_{z(x)\in Z} \Phi(x)$ et $\Phi_u = \Phi(z'^0)$
- Résoudre $[E_w^0]$ et dénotons le point optimal par z^0, poser $z^{inc} = z^n$ et $\Phi_l = \Phi(z^{inc})$.

$z'^1 = \max\limits_{x\in D} z_m(x)$ et $n = 1$.

Étape 2 : Vérification de la non-dominance

Résoudre $[E_w^n]$ et dénotons le point optimal par z^n. **Étape 3 :** Mise à jour des bornes

Si $\Phi(z^n) > \Phi(z^{inc})$ alors, poser $z^{inc} = z^n$. Si $g = \dfrac{|\Phi_u - \Phi(z^{inc})|}{|\Phi(z^{inc})|} \leq g^*$, passez à l'étape 5.

Sinon, passez à l'étape 4.

Étape 4 : nouvelle génération de points

Fixer $n = n + 1$ et mettre à jour K_m^n. Résolvez $[P_m^k]$ correspondant à chaque $k \in K_m^n$. Si $[P_m^k]$ est irréalisable pour tout $k \in K_m^n$, passez à l'étape 5.

Sinon, trouvez $k^* = \underset{k\in K_m^n,\ Z_{[S^k]}\neq\emptyset}{\text{argmax}} \Phi(z_m^k)$ et $z'^n = z^{k^*}$. Passez à l'étape 2.

Étape 5 : Terminer

Arrêter. $z^* = z^{inc}$ maximise (se rapproche) de la fonction Φ sur l'ensemble des points non-dominés (avec un niveau de précision souhaité, g^*).

où le modèle mathématique $[P_m^k]$ est formulé comme suit :

$$[P_m^k] \begin{cases} Max\ z_m(x) \\ \\ z_i(x) \geq b_i^k \qquad i \in \{1,...,p\},\ i \neq m \\ \\ \sum\limits_{i=1}^{p} v_i z_i \geq \Phi_l + g^*|\Phi_l| \\ \\ \sum\limits_{i=1}^{p} v_i z_i \leq \Phi_u \\ \\ x \in D \end{cases}$$

z^k désigne le point non dominé optimal si $[P_m^k]$ est faisable. la borne pour chaque critère i est fixé en fonction de la valeur de k_i. L'ensemble des vecteurs réalisables k, noté K_m^n, est obtenu comme dans DSA mais avec une information a priori de $k_m = 0$ puisque aucune borne inférieure est imposée pour critère m dans

$$DSA_m$$

.

4.3.4 Méthode de Zerfa et Chergui

Cette méthode (82) est basée sur le principe branch-and-bound et utilise des coupes efficaces dans le but de résoudre le problème d'optimisation d'un critère linéaire sur l 'ensemble des points non dominés d'un problème $MOILP$ (83).

Notations

— Le problème d'optimisation d'une fonction linéaire sur l'ensemble non dominé NDP :

$$(OP)\ \{max\ \phi(y) = \rho^t y | y \in NDP\}$$

où $\rho \in \mathbb{R}^p$, $\rho = (\rho_k)_{k=1,...,p}$, et ϕ est une fonction linéaire des fonctions objectifs du programme défini par 4.1 et $y = Cx$, $x \in D$.

— A l'étape l, soit le programme linéaire suivant,

$$(OPR)^l\ \{max\ \phi(y) = \rho^t y | y \in \gamma_{D^l}\}$$

où $\gamma_{D^l} = \{y \in \mathbb{R}^p | y = Cx, x \in D_l\}$ et

$$\begin{cases} D^l = S^l \cap \mathbb{Z}^n,\ l \geq 0,\ entier \\ D^{l+i} = \{x \in D^l | c^k.x \geq c^k.x^l + \epsilon_k\} \subset D^l,\ \forall k = 1, ..., p \end{cases}$$

où ϵ_k est une valeur fixée plus petite que l'erreur qu'on accepte sur les valeurs des c^k.

— Soit x^l une solution optimale du programme $(OPR)^l$. Les auteurs ont proposé un nouveau test d'efficacité, notée $(ZE)^l$ suivant en x^l, qui assure de générer une solution efficace x qui donne la plus grande valeur de la fonction ϕ parmi toutes les solutions alternatives. $(ZE)^l$ est présenté comme suit :

$$(ZE)^l \begin{cases} Max\ \ F(x,v) = M \sum_{k=1}^{p} v_k + \phi(c^k x) \\ c^k x - v_k = c^k x^l \qquad k = 1, ..., p \\ x \in D \\ v_k \geq 0, \qquad k = 1, ..., p \end{cases}$$

où M est une valeur positive très grande, $M \gg max_{x \in D} \phi(x)$.

L'algorithme est détaillé comme suit :

Algorithm 15: Algorithme de Zerfa et Chergui (83)

Données : C et $\phi(f)$ deux fonctions linéaires,
$X := \{x \in \mathbb{R}^n / Ax \leq b, x \geq 0\}$.
Résultats : x_{opt} et ϕ_{opt}.
Initialisation : $x_{opt} := \emptyset$, $\phi_{opt} := -\infty, l := 0$.
Étape générale :
Tant qu'il existe un nœud l non sondé dans l'arborescence faire :
1. Résoudre le programme $(OPR)^l$.
2. Si le $(OPR)l$ est non réalisable, alors le nœud l est sondé.
3. Sinon, Soit x_l une solution optimale.
(a) Si $\phi_{opt} \geq \phi(x_l)$, alors le nœud l est sondé.
(b) Sinon aller à l'étapel_1.
Étape l_1 : Test d'efficacité
Résoudre le programme $(ZE)^l$ en x_l, soit $(\hat{x}, \hat{v})$ une solution optimale obtenue :
1. Si $\phi(\hat{x}) \geq \phi(x_l)$, alors $x_{opt} = \hat{x}$, $\phi opt = \phi(\hat{x})$ et le nœud correspond est sondé.
2. Sinon, si $\phi(\hat{x}) \geq \phi opt$, alors $x_{opt} = \hat{x}$, $\phi opt = \phi(\hat{x})$
3. Si x_l est efficace alors le nœud correspond est sondé.
4. Sinon, aller à l'étape l_2
Étape l_2 : Évaluation
Pour $k := 1 : p$
Rajouter les coupes planes :

$$\begin{cases} c^k x \geq c^k \hat{x} + \epsilon_k \\ \phi(x) \geq \phi_o pt \end{cases}$$

où ϵ_k est une valeur plus petite que l'erreur qu'il accepte sur les valeurs des $c^k x$. Aller à l'étape l.

Remarque 4.3.1. *Si toutes les fonctions c^k, $\forall k = 1, ..., p$ sont à coefficients entiers, les auteurs (82) proposent de prendre $\epsilon_k = 1$, pour tout k, $k = 1, ..., p$.*

La méthode proposée (79) surpasse les approches existantes dans la littérature, en particulier (43), (51), (54)

4.4 Méthode exacte pour le problème d'optimisation d'un critère linéaire sur l'ensemble non dominé du problème $MOAP$

Dans ce chapitre, nous nous intéressons au problème d'optimisation d'une fonction sur l'ensemble efficient d'un problème $MOAP$. Une méthode exacte basée sur le

principe branch-and-bound et utilisant la méthode hongroise, est décrite dans le but de résoudre le problème d'optimisation d'un critère linéaire sur l'ensemble des points non dominés d'un problème $MOAP$. D'abord, la méthode est présentée en détail. Nous terminons le chapitre avec les résultats théoriques qui prouvent la finitude et la convergence de l'algorithme.

Soit $E \subset D$ l'ensemble des solutions efficaces du problème $MOAP$. Le problème d'optimisation d'une fonction linéaire sur l'ensemble E se formule comme suit :

$$(OP) \begin{cases} Min \;\; \Phi(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} x_{ij} \\ \\ x \in E \end{cases}$$

où : $\Phi : D \to \mathbb{Z}$ peut être une combinaison linéaire des fonctions objectifs du problème $MOAP$, ou toute autre fonction linéaire. Elle est désignée par "critère principal".

Nous définissons diverses notations utilisées tout au long de ce travail pour décrire les phases de notre approche d'optimisation d'une fonction linéaire sur l'ensemble des solutions efficaces du problème $MOAP$.
• SPE est l'ensemble des solutions potentiellement efficaces, et $SPND$ l'ensemble associé des points non dominés.
• $F = f_{ij}$ est la matrice des coûts relative à Φ.
À chaque nœud l de l'arborescence de recherche, les notations suivantes sont utilisées :

- $\widehat{C}_l^k, k \in \{1, ..., p\}, \widehat{F}l$: les matrices de coûts réduites sont obtenues récursivement en appliquant la méthode hongroise à la matrice des coûts F, et les matrices de coûts restantes sont mises à jour en utilisant les mêmes opérations.
- Soit x_l^* la solution obtenue.
- x_{opt} est la meilleure solution efficace de (OP) trouvée jusqu'à l'étape l, et Φ_{opt} la valeur correspondante du critère principal.
- N_l : l'ensemble des indices des variables non basiques de x_l^*.
- L'ensemble des directions de descentes des critères en x_l^*, noté H_l, qui peuvent améliorer les valeurs des critères pour $k \in \{1, ..., p\}$:

  $$H_l = \{(i,j) \in N_l, \exists k \in \{1, ..., p\} \; ; \; (\widehat{c}_{ij})_l^k < 0\} \cup \{(i,j) \in N_l, (\widehat{c}_{ij})_l^k = 0, \forall k \in \{1, ..., p\} \}$$

  tel que les matrices de coûts réduites sont notées :

  $$\widehat{C}_l^k = (\widehat{c}_{ij})_l^k, \;\; k \in \{1, ..., p\}$$

  .

— (T_l) le programme linéaire suivant :

$$(T_l)\begin{cases} Max \quad w = \sum_{k=1}^{p} e_k \\ Cx + e = Cx_l^* \\ \sum_{i=1}^{n} x_{ij} = 1 \qquad j \in \{1, ..., n\} \\ \sum_{j=1}^{n} x_{ij} = 1 \qquad i \in \{1, ..., n\} \\ x \in \{0, 1\},\ e_k \geq 0, \qquad k \in \{1, ..., p\} \\ e = (e_k)_{k \in \{1,...,p\}} \end{cases}$$

permettant de tester l'efficacité de la solution x_l^*, comme décrit dans (20).

– Le point idéal local, noté I_l, du problème $MOAP$ avec des variables fixées le long de la branche ayant le nœud l comme feuille.

Développement de la méthode L'algorithme proposé trouve une solution optimale de (OP) sans avoir à énumérer l'ensemble des solutions efficaces de $MOAP$. Pour ce faire, le principe branch-and-bound est utilisé, accompagné de plusieurs règles de sondage et de stratégies de recherche intelligentes. L'idée est d'utiliser la méthode hongroise pour résoudre un problème d'affectation simple lié à la fonction F_l à chaque nœud l de l'arborescence de recherche, en gardant à l'esprit que toutes les matrices de coûts C_l^k, $k \in \{1, ..., p\}$, sont soumises aux mêmes opérations que la matrice de coûts F_l. En conséquence, une solution réalisable x_l^* de (OP) est obtenue. En suivant le test d'efficacité décrit au paragraphe suivant, si x_l est efficace, le nœud l est sondé, et les paramètres suivants sont mis à jour : x_{opt}, Φ_{opt}, SPE et $SPND$.

Dans le cas contraire, l'ensemble H_l est construit et la phase de séparation décrite au paragraphe suivant est activée. Cette procédure est répétée jusqu'à ce que tous les nouveaux nœuds générés soient sondés.

Un nœud l de arborescence de recherche est sondé dans les situations suivantes :

1. Le point idéal est dominé par un élément de l'ensemble $SPND$.
2. La solution réalisable correspondante est efficace.
3. L'ensemble H_l est vide.
4. $\Phi(x_l^*) < \Phi_{opt}$, où Φ_{opt} est la meilleure valeur de Φ rencontrée précédemment.
5. Le sous-problème est infaisable.

Étape de séparation L'ensemble réalisable est successivement divisé en plusieurs sous ensembles, chacun spécifiant un sous problème différent, afin d'énumérer les solutions possibles du problème. Dans notre cas, l'étape de séparation est effectuée en rendant obligatoire chaque variable de l'ensemble H_l, ce qui élimine la ligne et la

colonne correspondantes des matrices de coûts générées. Afin de créer une partition de l'ensemble des solutions possibles à un nœud parent l, toute variable de l'ensemble H_l déclarée obligatoire (fixée à 1) sera interdite (fixée à 0) pour les autres nœuds enfants du même parent.

Test d'efficacité Cette procédure vérifie si une solution obtenue x_l^* est efficace ou non. Dans notre situation, si la solution x_l^* ne peut pas être ajoutée à l'ensemble SPE, alors cette solution n'est pas efficace. Sinon, nous devons résoudre le programme T_l en considérant cette solution. Ensuite, nous mettons à jour Φ_{opt}, x_{opt} et les ensembles SPE et $SPND$ en tenant compte de la solution x_l^*, lorsqu'elle est efficace, ou de la solution générée par T_l, notée y_l.

Quelques résultats théoriques

Proposition 4.4.1. *Si $H_l = \emptyset$, alors le sous-domaine correspondant ne contient pas de solutions efficaces..*

Démonstration. Si l'ensemble H_l est vide, aucun critère dans x_l^* ne peut être amélioré davantage, conformément à la définition de l'ensemble H_l. Alors, le nœud courant l doit être sondé. □

Proposition 4.4.2. *À chaque nœud l, si la variable x_{ij} est fixée à 1 dans un nœud fils, pour (i, j) appartenant à l'ensemble H_l, alors x_{ij} est forcée à prendre la valeur 0 dans tous les autres nœuds fils du nœud l .*

Démonstration. Pour obtenir une partition de l'ensemble S_l des solutions réalisables au nœud l, il ne faut pas retrouver des solutions réalisables identiques dans deux sous-ensembles S_r et S_t de S_l, où r et t sont deux nœuds fils du nœud l. Ainsi, en fixant une variable x_{ab} à la valeur 1 dans un nœud fils du nœud l, cela signifie que toutes les autres variables, non encore fixées, sont libres. elles peuvent prendre la valeur 1 ou la valeur 0. Mais, dans tous les autres nœuds fils du nœud l, il ne faut pas retrouver de solutions réalisables avec $x_{ab} = 1$. □

Proposition 4.4.3. *Si $\phi(x_l^*) \geq \phi_{opt}$, alors le nœud l est sondé.*

Démonstration. Au nœud l, lorsque l'on se déplace depuis x_l^* dans la direction de H_l, la fonction ϕ diminue. Ainsi, il n'existe aucune solution x telle que $\phi(x) < \phi(x_l^*)$ et $\phi_{opt} \leq \phi(x_l^*)$. Le nœud l est donc sondé. □

Théorème 4.4.1. *L'algorithme se termine en un nombre fini d'itérations et retourne la solution optimale du programme (OP).*

Démonstration. Tout d'abord, l'étape de séparation à tout nœud l de l'arbre de recherche assure la partition des sous-domaines obtenus aux nœuds fils, conformément à la Proposition 4.4.2. Ensuite, dans la recherche de la solution optimale de (OP), aucune solution optimale n'est omise selon les Propositions 4.4.2, 4.4.1, 4.4.3 . Le nombre de nœuds créés étant fini, l'algorithme se termine donc en un nombre fini d'itérations et tous les points non dominés sont obtenus. □

Exemple illustratif Considérons l'instance suivante :

$$C^1=\begin{pmatrix}9&3&4&9\\2&7&1&2\\4&7&1&7\\6&6&9&7\end{pmatrix},\quad C^2=\begin{pmatrix}7&4&6&3\\1&9&1&2\\1&7&8&5\\9&1&1&5\end{pmatrix},\quad F=\begin{pmatrix}1&5&6&11\\2&17&10&19\\10&8&4&1\\17&2&10&3\end{pmatrix}.$$

Initialisation $l=0$, $\Phi_{opt}=+\infty$, $SPE=\emptyset$, $SPND=\emptyset$.
Nous commençons par appliquer la méthode hongroise à la matrice de coûts F et mettons à jour les autres matrices en appliquant les mêmes modifications, on obtient :

$$\widehat{F}_0=\begin{pmatrix}\underline{1}&5&6&11\\\underline{2}&17&10&19\\10&8&4&\underline{1}\\17&\underline{2}&10&3\end{pmatrix}\to\begin{pmatrix}0&4&5&10\\9&15&8&17\\9&7&\underline{3}&0\\15&0&8&1\end{pmatrix}\to\begin{pmatrix}0&4&2&10\\0&15&5&17\\9&7&0&0\\15&0&5&1\end{pmatrix}\to\begin{pmatrix}0&2&\boxed{0}&8\\\boxed{0}&13&3&15\\11&7&0&\boxed{0}\\17&\boxed{0}&5&1\end{pmatrix}$$

$$\widehat{C}_0^1=\begin{pmatrix}\underline{9}&3&4&9\\\underline{2}&7&1&2\\4&7&1&\underline{7}\\6&\underline{6}&9&7\end{pmatrix}\to\begin{pmatrix}0&-6&-5&0\\0&5&-1&0\\-3&0&\underline{-6}&0\\0&0&3&1\end{pmatrix}\to\begin{pmatrix}0&-6&1&0\\0&5&5&0\\-3&0&0&0\\0&0&-3&1\end{pmatrix}\to\begin{pmatrix}0&-7&0&-1\\0&4&4&-1\\-2&0&0&0\\1&0&-3&1\end{pmatrix}$$

$$\widehat{C}_0^2=\begin{pmatrix}\underline{7}&4&6&3\\\underline{1}&9&1&2\\1&7&8&\underline{5}\\9&\underline{1}&1&5\end{pmatrix}\to\begin{pmatrix}0&-3&-1&-4\\0&8&0&1\\-4&2&\underline{3}&0\\8&0&0&4\end{pmatrix}\to\begin{pmatrix}0&-3&-4&-4\\0&8&-3&1\\-4&2&0&0\\8&0&-3&4\end{pmatrix}\to\begin{pmatrix}0&1&0&0\\0&12&1&5\\-8&2&0&0\\4&0&-3&4\end{pmatrix}$$

La solution optimale obtenue est $x_0^*=\{(1,3),(2,1),(3,4),(4,2)\}$, avec $Z(x_0^*)=(19,13)$. x_0^* n'est pas efficace selon le test d'efficacité ; une nouvelle solution efficace est obtenue : $y_0=\{(1,3),(2,4),(3,1),(4,2)\}$. Ainsi, $x_{opt}=y_0$, $\Phi_{opt}=37$, $SPE:=\{y_0\}$ et $SPND=\{(16,10)\}$.
L'ensemble $H_0=\{(1,1),(1,2),(1,4),(2,4),(3,1),(3,2),(3,3),(4,3)\}$.

Étape 1

Le nœud 0 est séparé en 8 nœuds comme suit :
Nœud 1 ($x_{11}=1$) : $I_1=(18,14)$ est dominé par $Z(y_0)$. Le nœud 1 est sondé.
Nœud 2 ($x_{11}=0; x_{12}=1$) : $I_2=(12,11)$ n'est dominé par aucun élément de $SPND$. On applique alors la méthode hongroise à ce sous-problème :

$$\widehat{F_2}=\begin{pmatrix}0&3&15\\11&0&0\\17&5&\underline{1}\end{pmatrix}\to\begin{pmatrix}\boxed{0}&3&15\\11&\boxed{0}&0\\16&4&\boxed{0}\end{pmatrix}$$

$$\widehat{C_1^2}=\begin{pmatrix}0&4&-1\\-2&0&0\\1&-3&\underline{1}\end{pmatrix}\to\begin{pmatrix}0&4&-1\\-2&0&0\\0&-4&0\end{pmatrix}$$

$$\widehat{C_2^2}=\begin{pmatrix}0&1&5\\-8&0&0\\4&-3&\underline{4}\end{pmatrix}\to\begin{pmatrix}0&1&5\\-8&0&0\\0&-7&0\end{pmatrix}$$

La solution $x_2^* = \{(1,2),(2,1),(3,3),(4,4)\}$ avec $\phi(x_2^*) = 14$ et $Z(x_2^*) = (13,18)$ est obtenue. Cette solution n'est dominée par aucun élément de l'ensemble $SPND$, il faut donc tester l'efficacité de cette solution en résolvant le programme mathématique T_2. Ainsi, cette solution est efficace et le nœud 2 est sondé. $\phi_{opt} := 14$; $x_{opt} := x_2^*$; $SPE := SPE \cup x_2^*$; $SPND := SPND \cup Z(x_2^*)$

Nœud 3 $(x_{11} = 0; x_{12} = 0; x_{14} = 1)$: En appliquant l'algorithme hongrois, on obtient :

$$\widehat{F_3} = \begin{pmatrix} \boxed{0} & 13 & 3 \\ 11 & 7 & \boxed{0} \\ 17 & \boxed{0} & 5 \end{pmatrix}; \widehat{C_3^1} = \begin{pmatrix} 0 & 4 & 4 \\ -2 & 0 & 0 \\ 1 & 0 & -3 \end{pmatrix}; \widehat{C_3^2} = \begin{pmatrix} 0 & 12 & 1 \\ -8 & 2 & 0 \\ 4 & 0 & -3 \end{pmatrix}.$$

La solution obtenue est : $x_3^* = \{(1,4),(2,1),(3,3),(4,2)\}$, $\phi(x_3^*) = 19 > \phi_{opt}$. Donc, le nœud 3 est sondé.

Nœud 4 $(x_{11} = 0; x_{12} = 0; x_{14} = 0; x_{24} = 1)$: L'application de la méthode hongroise est effectué comme suit :

$$\widehat{F_4} = \begin{pmatrix} +\infty & +\infty & 0 \\ \underline{11} & 7 & 0 \\ 17 & 0 & 5 \end{pmatrix} \to \begin{pmatrix} +\infty & +\infty & \boxed{0} \\ \boxed{0} & 7 & 0 \\ 6 & \boxed{0} & 5 \end{pmatrix}$$

$$\widehat{C_4^1} = \begin{pmatrix} +\infty & +\infty & 0 \\ \underline{-2} & 0 & 0 \\ 1 & 0 & -3 \end{pmatrix} \to \begin{pmatrix} +\infty & +\infty & 0 \\ 0 & 0 & 0 \\ 3 & 0 & -3 \end{pmatrix}$$

$$\widehat{C_4^2} = \begin{pmatrix} +\infty & +\infty & 0 \\ \underline{-8} & 2 & 0 \\ 4 & 0 & -3 \end{pmatrix} \to \begin{pmatrix} +\infty & +\infty & 0 \\ 0 & 2 & 0 \\ 12 & 0 & -3 \end{pmatrix}$$

La solution obtenue est $x_4^* = \{(2,4),(1,3),(3,1),(4,2)\}$, $\phi(x_4^*) = 26 > \phi_{opt}$. Le nœud 4 est donc sondé.

Nœud 5 $(x_{11} = 0; x_{12} = 0; x_{14} = 0; x_{24} = 0; x_{31} = 1)$: nous appliquons la méthode hongroise et la solution obtenue est $x_5^* = \{(3,1),(1,3),(2,2),(4,4)\}$, $\phi(x_5^*) = 36 > \phi_{opt}$. Le nœud 5 est donc sondé.

Nœud 6 $(x_{11} = 0; x_{12} = 0; x_{14} = 0; x_{24} = 0; x_{31} = 0; x_{32} = 1)$: nous résolvons le sous-problème correspondant à ce nœud comme suit :

$$\widehat{F_6} = \begin{pmatrix} +\infty & 0 & +\infty \\ 0 & 3 & +\infty \\ 17 & 5 & \underline{1} \end{pmatrix} \to \begin{pmatrix} +\infty & \boxed{0} & +\infty \\ \boxed{0} & 3 & +\infty \\ 16 & 4 & \boxed{0} \end{pmatrix}$$

$$\widehat{C_6^1} = \begin{pmatrix} +\infty & 0 & +\infty \\ 0 & 4 & +\infty \\ 1 & -3 & \underline{1} \end{pmatrix} \to \begin{pmatrix} +\infty & 0 & +\infty \\ 0 & 4 & +\infty \\ 0 & -4 & 0 \end{pmatrix}$$

$$\widehat{C_6^2} = \begin{pmatrix} +\infty & 0 & +\infty \\ 0 & 1 & +\infty \\ 4 & -3 & \underline{4} \end{pmatrix} \to \begin{pmatrix} +\infty & 0 & +\infty \\ 0 & 1 & +\infty \\ 0 & -7 & 0 \end{pmatrix}$$

La solution obtenue est $x_6^* = \{(3,2),(1,3),(2,1),(4,4)\}$, $\phi(x_6^*) = 19 > \phi_{opt}$. Le nœud 6 est donc sondé.
Nœud 7 $(x_{11} = 0; x_{12} = 0; x_{14} = 0; x_{24} = 0; x_{31} = 0; x_{32} = 0; x_{33} = 1)$: Ce sous-problème est infaisable. Le nœud 7 est donc sondé.
Nœud 8 $(x_{11} = 0; x_{12} = 0; x_{14} = 0; x_{24} = 0; x_{31} = 0; x_{32} = 0; x_{33} = 0; x_{43} = 1)$. Ce sous-problème est infaisable. Ainsi, le nœud 8 est sondé.
Donc, $\phi_{opt} = 14$, $x_{opt} = \{(1,2),(2,1),(3,3),(4,4)\}$.

Expérimentation numérique La méthode proposée, notée CS, dédiée à l'optimisation d'une fonction linéaire sur l'ensemble efficient du problème $MOAP$, ainsi que la méthode DSA_m (51), ont été implémenté en Python à l'aide de la bibliothèque Gurobi 9.1, sur un ordinateur Dell I5-7300U avec 8GB de RAM. Les deux méthodes ont été testées sur une série de 10 instances aléatoires avec des matrices de coûts de taille $n \times n$, avec n variant entre 30, 40 et 50, et un nombre de critères $p \in \{2,3,5\}$. La fonction Φ est une combinaison linéaire des objectifs du $MOAP$, où les coefficients α_k, $k \in \{1,...,p\}$, sont générés aléatoirement dans l'intervalle $[-100,100]$. Les fonctions objectifs, quant à elles, sont générées aléatoirement dans l'intervalle $[1,20]$ selon une distribution uniforme. Le tableau 4.1 présente les résultats numériques obtenus. Nous rapportons le temps moyen CPU pour chaque méthode. Les résultats montrent que notre méthode surpasse la méthode proposée par Lokman (51).

TABLE 4.1 – Résultats de comparaison entre DSA_m (51) et CS.

p	n	Temps CPU (Secondes)	
		DSA_m	CS
2	30	58.303	63.790
	40	308.324	100.132
	50	989.901	181.344
3	30	438.349	294.405
	40	2090.456	538.092
	50	>3600	777.219
5	30	>7200	563.694
	40	>7200	1163.396
	50	>7200	1332.522

4.5 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode pour résoudre le problème d'optimisation d'un critère linéaire sur l'ensemble des points non dominés du problème $MOAP$. L'approche est basée sur le principe branch-and-bound en utilisant la méthode hongroise. A chaque nœud de l'arborescence, le critère principal est optimisé sur des sous domaines obtenus par séparant sur le problème en utilisant

une liste de descentes possibles des critères qu'on génère. Chaque solution trouvée est soumise à un test d'efficacité. L'étude expérimentale que nous avons mené a montré que notre méthode surpasse la méthode proposée par Lokman (51).

Chapitre 5

Conclusion et perspectives

Dans ce présent travail, nous avons passé en revue dans les premiers chapitres, les outils de base nécessaires au développement des notions et concepts de la programmation combinatoire multi-objectif qui a fait l'objet de cette thèse. C'est ainsi que nous nous sommes focalisé sur la mise en œuvre de deux méthodes exactes qui traitent le problème d'affectation multi-objectif. La première méthode génère l'ensemble des points non dominés dans l'espace des critères pour le problème d'affectation multi-objectif, avec les coefficients des fonctions objectives positifs ou nulles. Ce thème n'a pas fait couler beaucoup d'encre, étant donné que les travaux publiés traitent seulement le cas bi-objectif et tri objectif. La méthode proposée exploite la structure du problème d'affectation pour en déduire des variables qui servent à améliorer au moins une valeur des critères à même d'éliminer des sous-ensembles de points dominés réalisables, dans une arborescence de recherche structurée, basée sur le principe branch-and-bound. Le nombre de nœuds enfants créés en chaque nœud parent, correspond au nombre de directions de descente des critères. Les règles mises en œuvre de sondage des nœuds ainsi que les ensembles d'éléments vectoriels d'évaluations des critères considérés, ont eu un effet positif sur la vitesse de convergence de la méthode. De plus, la métaheuristique $MOVNS$ a permis d'enrichir l'ensemble des points potentiellement non dominés et la procédure de prétraitement adoptée, a permis d'éliminer les domaines non prometteurs, entraînant ainsi une accélération de la vitesse de convergence des algorithmes proposés. Une étude comparative est rapportée pour évaluer les performances de notre méthode avec la méthode de Özlen et *al.* et cells proposée par Przybylski et *al.*. L'analyse des résultats montre que notre méthode est compétitive ces méthodes sur plusieurs instances.

Nous avons également étudié le problème de l'optimisation d'un critère linéaire sur l'ensemble des points non dominés d'un problème $MOAP$. À cet effet, nous avons élaboré une méthode exacte, fondée sur le principe du branch-and-bound combiné à la méthode hongroise, permettant d'identifier, dans l'espace des critères, le meilleur point parmi les points non dominées d'un problème $MOAP$. Les résultats expérimentaux obtenus sont satisfaisants, et l'étude comparative menée avec la méthode proposée par Lokman dans (51) a démontré la supériorité de notre ap-

proche.

Toutefois, il demeure nécessaire de comparer notre méthode avec la nouvelle approche proposée par Zerfa et Chergui (83), afin d'évaluer plus précisément son efficacité et de juger de ses performances.

Parmi les perspectives de recherche, nous envisageons d'étendre notre méthodologie à d'autres variantes du problème $MOAP$, telles que le problème d'affectation quadratique multi-objectif et le problème d'affectation multimodal. L'adaptation et l'intégration de métaheuristiques constituent également des axes prometteurs à explorer.

Bibliographie

[1] Abbas, M. and Chaabane, D., 2006. Optimizing a linear function over an integer efficient set. European Journal of Operational Research, 174(2) : 1140-1161.

[2] Abbas, M. ; Chergui, M.E.A. and Mehdi, M.A., 2012. Efficient cuts for generating the non-dominated vectors for multiple objective integer linear programming. International Journal of Mathematics in Operational Research, 4(3), pp.302-316.

[3] Adiche, C., & Aïder, M., 2010. A hybrid method for solving the multi-objective assignment problem. Journal of Mathematical modelling and Algorithms, 9, 149-164.

[4] Aneja Y.P.and Nair K.P.K. , 1979, Bicriteria transportation problem. Management Science, 25(1) :73–78.

[5] Bektas¸ T. 2018. Disjunctive programming for multiobjective discrete optimisation. INFORMS Journal on Computing, 30(4) : 625–633.

[6] Belhoul, L. ; Lucie, G. and Daniel, V., 2014. An efficient procedure for finding best compromise solutions to the multi-objective assignment problem. Comput Oper Res 49 :97–106.

[7] Belhoul Lyes, 2014, Résolution de problèmes d'optimisation combinatoire mono et multi-objectif par énumération ordonnée (Doctoral dissertation, Université Paris Dauphine-Paris IX).

[8] Benson, H.P., 1984. Optimization over the efficient set. Journal of Mathematical Analysis and Applications, 98 : 562-580.

[9] Benson, HP., 1991. An all-linear programming relaxation algorithm for optimizing over the efficient set. J Global Opt 1(1) :83–104.

[10] Boland, N. ; Charkhgard, H. and Savelsbergh, M., 2017. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. European Journal of Operational Research, 260 : 904-919.

[11] Bowman V. J. , 1976 , On the relationship of the Tchebytcheff norm and the efficient frontier of multiple-criteria objectives, Lecture Notes in Economics and Mathematical Systems, 130, 76–85.

[12] Chaabane, D., Pirlot, M., 2010. A method for optimizing over the integer efficient set. Journal of Industrial & Management Optimization, 6(4), 811.

[13] Chegireddy, C.R. and Hamacher, H.W., 1987. Algorithms for finding k-best perfect matchings. Discrete Applieds Mathematics, 18 :155-165.

[14] Chinchuluun, A. ; Pardalos, M., 2007. A survey of recent developments in multiobjective optimization.Ann.Oper. Res. 154 :29-50.

[15] Collecte Y. et Siarry P., 2002. optimisation multiobjectif, Eyrolles, ISBN-2-212-11168-1.

[16] Dell'Amico M. and S. Martello, 1997, Linear assignment. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, Annotated Bibliographies in Combinatorial Optimization, pages 355–371. Wiley, Chichester.

[17] Delort, C. & Spanjaard, O. 2011. Yet another two-phase method for the biobjective assignment problem. In : 21st International Conference on Multiple Criteria Decision Making (MCDM2011).

[18] Delort, C. and Spanjaard, O., 2010. Using bound sets in multiobjective optimization : Application to the biobjective binary knapsack problem. In Experimental Algorithms : 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings 9 (pp. 253-265). Springer Berlin Heidelberg.

[19] Egerváry E., 1931, Matrixok kombinatorius tulajdonságairól [Hungarian with Ger man summary]. Matematikai és Fizikai Lapok, 38 :16–28. [English translation [by H.W. Kuhn] 1931 : On combinatorial properties of matrices, Lo gistics Papers, George Washington University, issue 11 , paper 4, pp. 1–11].

[20] Ecker JG, Kouada IA, 1975, Finding efficient points for linear multiple objective programs. Math Program 8(1) :375–377

[21] Edgeworth, F.Y., 1881. Mathematical psychics : An essay on the application of mathematics to the moral sciences (Vol. 10). Kegan Paul.

[22] Ehrgott M., X. Gandibleux, 2000, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spectrum 22 (4) 425–460.

[23] Ehrgott, M., 2005 : Multicriteria Optimization. Springer.

[24] Ehrgott, M., 2005. Multicriteria optimization. vol. 491. Springer Science & Business Media.

[25] Ehrgott, M., Gandibleux, X. and Przybylski, A., 2016. Exact methods for multiobjective combinatorial optimisation. Multiple criteria decision analysis : State of the art surveys, pp.817-850.

[26] Ehrgott, M. ; Wiecek, M., 2005. Multiobjective programming. In : Figueira, J., Greco, S.,Ehrgott, M.(eds.) Multicriteria Decision Analysis : State of the Art Surveys., pp. 667-722. Springer Science + Business Media, New York.

[27] Evans, J.P. ; Steuer, R.E., 1973. Generating efficient extreme points in linear multiple objective programming : two algorithms and computing experience. Multiple Criteria Decision Making 1, 349–365.

[28] Flux, A.W., 1896. Vilfredo Pareto. Cours d'economie politique.

[29] Fülöp J, Muu LD, 2000, Branch-and-bound variant of an outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. J Optim Theory Appl 105(1) :37–54

[30] Geiger MJ. 2008. Randomised variable neighbourhood search for multiobjective optimisation. ArXiv preprint arXiv :0809.0271.

[31] Geoffrion, A.M., 1968. Proper efficiency and the theory of vector maximization. Journal of mathematical analysis and applications, 22(3), pp.618-630.

[32] Golberg, D.E., 1989. Genetic algorithms in Search, Optimization and Machine learning. Addison-Wesley Publishing Company. Reading, Massachussets.

[33] Gungor, I. and GUNES M., 2000. Fuzzy multiple criteria assignment problems for fusion : the case of Hungarian algorithm. In : Proceedings of the third international conference on information fusion, 2000. FUSION 2000, vol 1, IEEE, pp TUD4–8.

[34] Gutjahr WJ & REITER P. 2010. Bi-objective project portfolio selection and staff assignment under uncertainty. Optimization, 59(3) : 417–445.

[35] Haïmes, Y. ; Ladson, L. and Wismer, D., 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transactions on System, Man and Cybernetics, 1 : 296–297.

[36] Halffmann, P. ; Schäfer, L. E. ; Dächert, K. ; Klamroth, K. ; Ruzika, S., 2022. Exact algorithms for multiobjective linear optimization problems with integer variables : A state of the art survey. Journal of Multi-Criteria Decision Analysis, 29(5-6) : 341–363.

[37] Hamacher, H.W. and Queyranne, M., 1985. K-best solutions to combinatorial optimization problems. Annals of Operations Research, 4 :123-143.

[38] Hammadi AMK, 2017, Solving multi-objective assignment problem using Tabu search algorithm. Global J Pure Appl Math 13(9) :4747–4764

[39] Hamacher. H and M. Queyranne, 1985. k-best solutions to combinatorial optimization problems. Annals of Operations Research, 4 :123-143.

[40] Huang G, Lim A (2006) A hybrid genetic algorithm for the three-index assignment problem. Eur J Oper Res 172(1) :249–257

[41] Isermann, H., & Steuer, R. E, 1988. Computational experience concerning payoff tables and minimum criterion values over the efficient set. European journal of operational research, 33(1), 91-97.

[42] Jacques Teghem, 1996, Programmation Linéaire, Ellipses.

[43] Jorge, J.M, 2009 : An algorithm for optimizing a linear function over an integer efficient set. Eur J Oper Res., 195 :98-103.

[44] Jorge, J.M., 2010. Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires. Thèse de doctorat, Université de Nantes.

[45] Kacprzyk, J., Coello Coello, C.A., Dhaenens, C., Jourdan, L. 2010. Multiobjective combinatorial optimization : Problematic and context, in : C.A. Coello Coello, C. Dhaenens, L. Jourdan (Eds.), Advances in Multi-Objective Nature Inspired Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, p. 272.

[46] Kirlik G & Sayin S. 2014. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. European Journal of Operational Research, 232(3) : 479–488.

[47] Kiziltan, G. & Yucaoglu, E., 1983. An algorithm for multiobjective zero-one linear programming. Management Science, 29(12) : 1444–1453.

[48] Kuhn HW. 1955. The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2) : 83–97.

[49] Land AH & Doig AG. 2010. An automatic method for solving discrete programming problems. Springer.

[50] Liu, Q., Li, X., Liu, H. and Guo, Z., 2020. Multi-objective metaheuristics for discrete optimization problems : A review of the state-of-the-art. Applied Soft Computing, 93, p.106382.

[51] Lokman B., 2021, Optimizing a linear function over the nondominated set of multiobjective integer programs. International Transactions in Operational Research, 28 : 2248-2267.

[52] Martin P.D. & Shmoys D.B. 1996. A new approach to computing optimal schedules for the job shop scheduling problem. In S.T. McCormick, W.H. Curnigham and M. Queyranne, editors, Proceedings of the Fifth international IPCO conference, Vancouver, Canada, pages 389–403. LNCS 1084.

[53] Nguyen, N.C., 1992, An Algorithm for Optimizing a Linear Function over the Integer Efficient Set. Konrad-Zuse-zentrum fur Informationstechnik Berlin.

[54] Ouaïl, F. Z., Chergui, M. E.-A., Moulaï, M. : An exact method for optimizing a linear function over an integer efficient set. WSEAS Transactions on Circuits and Systems,16 :141-148,(2017).

[55] Özlen, M. and Azizoğlu, M., 2009. Multi-objective integer programming : A general approach for generating all non-dominated solutions. European Journal of Operational Research, 199(1), pp.25-35.

[56] Özlen M, Burton BA & Macrae CA. 2014. Multi-objective integer programming : An improved recursive algorithm. Journal of Optimization Theory and Applications, 160 : 470–482.

[57] Pedersen CR, Nielsen LR & Andersen KA. 2008. The bicriterion multimodal assignment problem : Introduction, analysis, and experimental results. INFORMS Journal on Computing, 20(3) : 400–411.

[58] Przybylski, A. ; Gandibleux, X. & Ehrgott, M., 2008. Two phase algorithms for the biobjective assignment problem. European Journal of Operational Research, 185(2) : 509–533.

[59] Przybylski, A. ; Gandibleux, X. & Ehrgott, M., 2009. Computational results for four exact methods to solve the three-objective assignment problem. In : Multiobjective Programming and GoalProgramming : Theoretical Results and Practical Applications. pp. 79–88. Springer.

[60] Przybylski, A. ; Gandibleux, X. & Ehrgott, M., 2010. A two phase method for multiobjective integer programming and its application to the assignment problem with three objectives. Discrete Optimization, 7(3) : 149–165.

[61] Przybylski, A. ; Gandibleux, X. 2017. Multi-objective branch and bound. European Journal of Operational Research, 260(3) : 856–872.

[62] Ramos R., Alonso, S., Scilia, J. et GONZÁLEZ, C. 1998. The problem of the optimal biobjective spanning tree. European Journal of Operational Research, 111 :617–628.

[63] Sakarovitch M. , 1983, Techniques mathématiques de la recherche opérationnelle : Optimisation dans les réseaux, tome 3, ENSIMAG, Université Scientifique et Médicale Institut national Polytechnique de Grenoble.

[64] Satla H & CHERGUI MEA. 2025. An exact method to solve the multi-objective assignment problem. Pesquisa Operacional, 45 : e290556. doi : 10.1590/0101-7438.2025.045.00290556.

[65] Schaffer J.D. 1985, Multiple objective optimisation with vector evaluated genetic algorithms. In J. J. Grefenstette, editor, ICGA Int. Conf on Genetic Algorithms, pages 93–100. Lawrence Erbaum.

[66] Shigeno, M., Takahashi, I., & Yamamoto, Y, 2003. Minimum maximal flow problem : an optimization over the efficient set. Journal of Global Optimization, 25, 425-443.

[67] Sourd, F. and Spanjaard O, 2008. A multi-objective branch-and-bound framework. Application to the bi-objective spanning tree problem. INFORMS Journal of Computing, 20(3) :472–484.

[68] Steiner, S. and Radzik, T., 2008. Computing all efficient solutions of the biobjective minimum spanning tree problem. Computers & Operations Research, 35(1), pp.198-211.

[69] Steuer, R.E. : Multiple criteria optimization : Theory, computation and application. John Wiley, New York, (1986).

[70] Steuer R.E., 1986. Multiple criteria optimization. Theory, computation, and application.

[71] Tailor A.R, & Dhodiya, J. M., 2021. Multi-objective assignment problems and their solutions by genetic algorithm. In Computational Management : Applications of Computational Intelligence in Business Management (pp. 409-428). Cham : Springer International Publishing.

[72] Tailor A.R, Pandit D & Dhodiya JM. 2022. Multi-Objective Interval Assignment Problems and their Solutions Using Genetic Algorithms. In : Industrial Transformation. pp. 143–172. CRC Press.

[73] Tomizawa N., 1971, On some techniques useful for solution of transportation network problems. Networks, 1 :173–194.

[74] Tuyttens, D., Teghem, J., Fortemps, Ph., Van Nieuwenhuyse, K., 2000. Performance of the MOSA method for the bicriteria assignment problem. Journal of Heuristics 6, 295–310.

[75] Ulungu, E.L, 1993. Optimisation combinatoire multicritère : détermination de l'ensemble des solutions efficaces et méthodes interactives. Thesis PhD, Université de MonsHainaut.

[76] Ulungu E.L., Teghem J., 1994, Multi-objective combinatorial optimization problems : A survey, J. Multi-Criteria Decis. Anal. 3 (2), 83–104.

[77] Ulungu E.L & Teghem J. 1995. The two phases method : An efficient procedure to solve biobjective combinatorial optimization problems. Foundations of computing and decision sciences, 20(2) : 149–165.

[78] Vincent, T., 2013. Caractérisation des solutions efficaces et algorithmes d'énumération exacts pour l'optimisation multi-objectif en variables mixtes binaires (Doctoral dissertation, Nantes).

[79] Visée, M., Teghem, J., Pirlot, M. et Ulungu, E.,1998. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. Journal of Global Optimization, 12 :139–155.

[80] Yamamoto Y, 2002, Optimization over the efficient set : overview. J Global Optim 22 :285–317

[81] Yu, P.L., 1985. Multiple-Criteria Decision Making. Plenum, New York.

[82] Zerfa, L., 2022. Description de l'ensemble efficient de problèmes d'optimisation multi-objectif (Doctoral dissertation).

[83] Zerfa, L. and Chergui, M.E.A., 2024. An efficient branch-and-bound algorithm to optimize a function over a nondominated set. International Transactions in Operational Research.