N **d'ORDRE : 26/2009-M/MT**

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

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

En : MATHÉMATIQUES

Spécialité : Recherche Opérationnelle :Mathématiques de Gestion

Par : AHMIA Ibtissam

Thème

Méthodes coopératives appliquées aux problèmes d'optimisation combinatoire

Soutenu le : 27/06/2009, devant le jury composé de :

M^r	AIDER Méziane	Professeur	USTHB	Président
M^r	CHAABANE Djamel	Maître de Conférences	USTHB	Directeur de Thèse
M^{me}	MERAZKA Fatiha	Maître de Conférences	USTHB	Examinatrice
M^r	SEMRI Ahmed	Maître de Conférences	USTHB	Examinateur

Remerciements

Je tiens à remercier en tout premier lieu Dieu de m'avoir procuré tout ce dont j'ai besoin pour réussir dans la vie.

Je remercie énormément mon directeur de thèse monsieur CHAABANE Djamel pour m'avoir suivi soutenu et encadré tout au long de la création de ce mémoire de magister, ses conseils et sa disponibilité m'ont permis d'avancer dans le bon sens.

Je tiens également à remercier vivement monsieur AIDER Méziane, de me faire l'honneur de présider le Jury de soutenance de ce mémoire de magister. Je remercie chaleureusement Monsieur SEMRI Ahmed et Madame MERAZKA Fatiha d'avoir accepté d'examiner ce travail de recherche.

Je ne pourrais oublier mes parents et mon frère qui m'ont épaulé et encouragé durant toutes mes années d'études. Je leur adresse toute ma reconnaissance pour leur soutien moral, leur présence ainsi que leur écoute.

TABLE DES MATIÈRES

TABLE DES FIGURES

Introduction

L'optimisation combinatoire multi-objectifs est une branche d'optimisation combinatoire. Les problèmes d'optimisation combinatoire multi-objectifs "Multiple objective combinatorial optimization" (MOCO) sont des problèmes ardus car ils combinent les difficultés des problèmes combinatoires classiques avec ceux des problèmes multi-objectifs. C'est cependant sous cette forme que se présente la plupart des problèmes industriels reéls.

Dans notre travail on s'est intéréssé à la résolution exacte du problème d'affectation bi-objectifs. En effet les premiers articles qui ont traité le problème d'affectation multi-objectifs n'ont pas pris en considération les solutions non supportées. La recherche de ces solutions est la cause de la difficulté théorique de ce problème car leur génération pose un problème vu que ces solutions ne se trouvent pas sur la frontière efficace mais à l'intérieur de l'enveloppe convexe.

Les chercheurs qui se sont rendu compte de la présence des solutions non supportées ont par la suite développé des méthodes pour la résolution de ce type de problème mais aucune méthode affirmant avoir trouvé l'ensemble de toutes les solutions efficaces sans connaissance a priori du problème n'existe.

La méthode en deux phase est un cadre de résolution general qui a été popularisé par Ulungu en 1993. Elle a depuis été appliquée sur un grand nombre de problèmes, en se limitant toutefois au contexte bi-objectifs.
Le but de notre travail a été la détermination d'un ensemble complet de solutions efficaces pour le problème d'affectation bi-objectifs sans connaissance a priori du problème et ceci a été fait en coopérant cette méthode avec une méthode lexicographique qu'on a introduit

en deuxième phase.

Ce présent mémoire de magister est organisé comme suit :
Le premier chapitre contient un état de l'art de l'optimisation multi-objectifs, dans le deuxième nous nous sommes focalisés sur l'optimisation multi-objectifs combinatoire, dans le troisième chapitre nous avons proposé une méthode de résolution pour le problème d'affectation bi-objectifs, le chapitre 4 quant à lui comporte une étude expérimentale de notre méthode avec une explication du code écrit en MATLAB. Enfin, nous terminons par une conclusion qui décrit les avantages de notre méthode basés sur des expérimentations effectuées avec le code MATLAB et nous citons quelques perspectives.

CHAPITRE 1

Optimisation multi-objectifs

L'optimisation multi-objectifs fut abordée pour la première fois dans les travaux de Edgeworth et Pareto au 19ième siècle. Elle a été utilisée initialement en économie et dans les sciences de management puis progressivement dans les sciences pour l'ingénieur.
En effet, la résolution de la plupart des problèmes rencontrés dans la vie pratique nécessite de prendre en considération plusieurs critères souvent contradictoires. L'optimisation multi-objectifs s'est avéré indispensable pour la résolution de ce genre de problèmes. Les premières méthodes utilisées dans l'optimisation multi-objectifs faisaient appel aux méthodes existantes pour l'optimisation mono-objectif en résolvant une succession de problèmes mono-objectif ou en optimisant une agrégation linéaire des différents objectifs mais ceci n'est faisable que si un ordre d'importance sur les objectifs (critères) peut être donné. Malheureusement dans la plupart des problèmes multi-objectifs rencontrés on ne peut pas trouver cet ordre d'importance sur les différents critères. Cependant une recherche de solutions de meilleur compromis est indispensable.

1.1 Formalisation mathématique d'un problème multi-objectifs [1]

Un problème multi-objectifs peut être défini de la manière suivante :

$$(MOP)\begin{cases} \text{“opt”}Z(x) = & (Z_1(x), Z_2(x), \ldots, Z_k(x)) \\ \text{t.q} & x \in X \end{cases} \tag{1.1}$$

Où $k \geqslant 2$ est le nombre de critères (fonctions objectifs).
Le symbole " " signifie qu'il n'est généralement pas possible de trouver dans X une solution qui optimise simultanément les k critères.
$x = (x_1, \ldots, x_n)$ est le vecteur représentant les variables de décision.
X est un sous ensemble de $\mathbb{R}^n$ c'est l'ensemble des solutions admissibles (réalisables) associés à des contraintes d'égalité, d'inégalité .
L'espace $\mathbb{R}^n$ est l'espace de décision dans lequel se situe X.
$Z(x) = (Z_1(x), Z_2(x), \ldots, Z_k(x))$ est le vecteur de critères à optimiser.
Ces problèmes peuvent être résolus dans le domaines de décision ou dans le domaine de critères (voir figure1.1).
L'ensemble Z_X représente les points réalisables dans l'espace de critères (espace objectifs), avec $z = (z_1, \ldots, z_k)$ un point de cet ensemble. On entend par espace de critères l'espace $\mathbb{R}^k$ dans lequel se situe Z_X.

1.1.1 Remarques

- Si de plus les k critères et les contraintes qui définissent X sont des fonctions linéaires,on obtient un programme linéaire multi-objectifs en variables continues qui se formule mathématiquement comme suit :

$$(MOLP)\begin{cases} \text{“ opt”} & Z_j(x) = C_j x, \quad j = \{1, \ldots, k\} \\ \text{t.q} & x \in X \\ & X = \{x \in \mathbb{R}^n : Ax \leqslant b, x \geqslant 0\} \end{cases} \tag{1.2}$$

  avec $A(m \times n), C_j(1 \times n), x(n \times 1), b(m \times 1)$
- On appelle un programme linéaire multi-objectifs en variables entières tout programme qui se formule comme suit :

$$(MOILP)\begin{cases} \text{“opt”} & Z_j(x) = C_j x, \quad j = \{1, \ldots, k\} \\ \text{t.q} & x \in X \\ & X = \{x \in RL/x \in \mathbb{N}\} \end{cases} \tag{1.3}$$

  Où $RL = \{x \in \mathbb{R}^n : Ax \leqslant b, x \geqslant 0\}$ est l'ensemble des solutions de la relaxation

linéaire du problème ($MOLIP$) qui se présente comme suit :

$$\text{“opt”}_{x \in RL} Z_j(x) = C_j x, \quad j = \{1, \ldots, k\} \tag{1.4}$$

– Sachant que l'ensemble $Z_X = Z(X)$ représente les points réalisables dans l'espace des critères, l'ensemble convexe de Z_X se définit comme suit :

$$conv(Z_X) = \{y / y = \sum_{i=1}^{k} \alpha_i y^i \in Z_X, \alpha > 0, \sum_{i=1}^{k} \alpha_i = 1\} \tag{1.5}$$

Définition 1 *On appelle “face” d'un convexe C toute partie F de C ayant la propriété suivante*

$$\left.\begin{array}{l} x \in F \\ \overline{x}, \overline{\overline{x}} \in C \\ x = (\overline{x} + \overline{\overline{x}})/2 \end{array}\right\} \Longrightarrow \overline{x}, \overline{\overline{x}} \in F$$

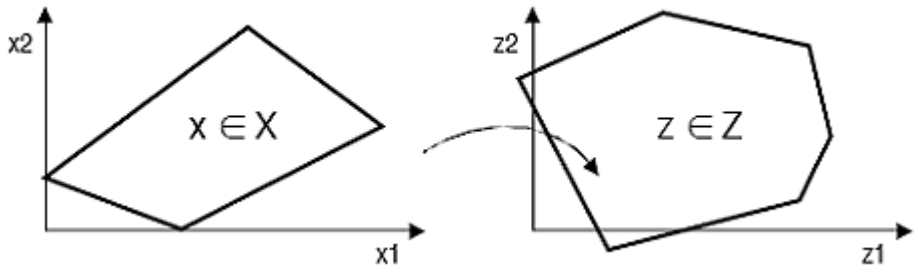

FIG. 1.1 – Correspondance espace désicion-espace des objectifs

1.2 Définitions relatives à la programmation linéaire multi-objectifs [1–8]

1.2.1 Relation d'ordre et de dominance

Dans le cas des problèmes d'optimisation multi-objectifs, les relations d'ordre sont appelées relations de dominance. Plusieurs relations de dominance ont déjà été présentées : la a-dominance [4], la dominance au sens de Geoffrion [7], la cône-dominance [1], ... Mais la plus célèbre et la plus utilisée est la dominance au sens de Pareto. C'est cette relation de dominance que nous allons définir et utiliser dans cette thèse. De manière à définir

clairement et formellement cette notion, les relations $=$, $\leqslant$ et $<$ usuelles sont étendues aux vecteurs.

Remarque 1 *Soit u et v deux vecteurs de même dimension,*
$u = v$ si et seulement si $\forall i \in 1, \ldots, n, \quad u_i = v_i$
$u \leqslant v$ si et seulement si $\forall i \in \{1, \ldots, n\}, \quad u_i \leqslant v_i$
$u < v$ si et seulement si $\forall i \in \{1, \ldots, n\}, \quad u \leqslant v \wedge u \neq v$
Les relations $\geqslant$ et $>$ sont définies de manière analogue.
On remarque que pour un $u = (2, 3)$ et un $v = (3, 2)$, on ne peut pas faire une comparaison à l'aide de ces relations. Contrairement aux problèmes mono-objectif où les relations usuelles $<$, $\leqslant$,... suffisent pour comparer les solutions, elles sont insuffisantes pour comparer les solutions des problèmes multi-objectifs. D'où la nécessité d'une autre relation qui remédie à ce problème. C'est pourquoi la relation de dominance au sens de Pareto s'est apparue permettant de prendre en compte tous les cas de figure rencontrés lors de la comparaison de deux solutions.

1.2.2 La dominance au sens de Pareto (Vilfredo Pareto 1896)

Soit deux vecteurs $z(x), z(x^*) \in \mathbb{R}^k$, on dit que le vecteur $z(x)$ domine $z(x^*)$ dans le cas d'une minimisation d'objectifs, si et seulement si, $\forall j \in \{1, \ldots, k\} z_j(x) \leqslant z_j(x^*)$ et $\exists i \in \{1, \ldots, k\}$ t.q $z_i(x) < z_i(x^*)$

1.2.3 Solution optimale au sens de Pareto (efficace)

Une solution réalisable $x \in X$ est dite **Pareto optimale ou efficace** s'il n'existe pas de solution $x^* \in X$ telle que $z(x^*)$ domine $z(x)$. Ainsi $z(x)$ est dit vecteur **non dominé** (voir figure1.2).

Toute solution de l'ensemble Pareto peut être considérée comme optimale puisque aucune amélioration ne peut être faite sur un objectif sans dégrader la valeur relative à un autre objectif.

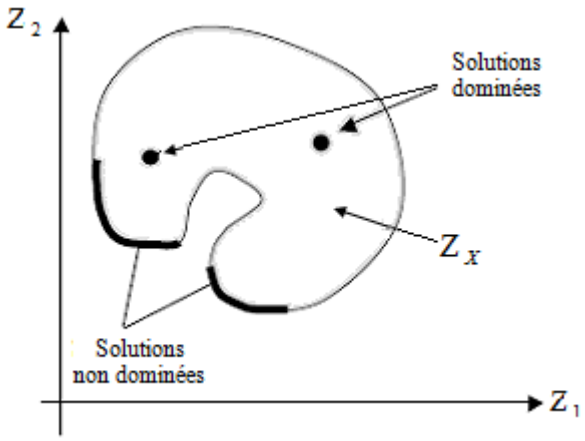

FIG. 1.2 – Optimisation globale au sens de Pareto

1.2.4 Voisinage d'une solution

Le voisinage d'une solution x_0 peut être défini comme une restriction Z_0 de l'espace complet Z_X des objectifs.
Une solution $z \in Z_X$ est dite non dominée par rapport à un ensemble $Z_a \subseteq Z_X$, si et seulement si : $\nexists z_a \in Z_a$, z_a domine z.

1.2.5 Solution localement optimale au sens de Pareto

Sachant que le voisinage d'une solution x est composé des solutions pouvant être atteintes depuis x à l'aide d'une transformation élémentaire, appelée opérateur de voisinage. Attachée à la notion de voisinage, la notion de **Pareto localement optimale** qualifie une solution qui n'est dominée par aucune solution de son voisinage. Autrement dit une solution x est **localement optimale au sens de Pareto** si et seulement si quelque soit x^* une solution appartenant au voisinage de x, x^* ne domine pas x (voir figure1.3).

1.2.6 Solution faiblement efficace (faiblement Pareto optimale)

Une solution x est faiblement efficace si et seulement si

$$\nexists x^* \in X; \quad \forall j \in \{1, \ldots, k\} \quad z_j(x^*) < z_j(x).$$

Ceci et dans le cas où les fonctions objectifs $Z_j, j \in \{1, \ldots, k\}$ sont à minimiser.
La condition de la faible efficacité est plus facile à vérifier que la dominance. Une solution efficace doit être faiblement efficace mais l'inverse n'est pas toujours vrai.

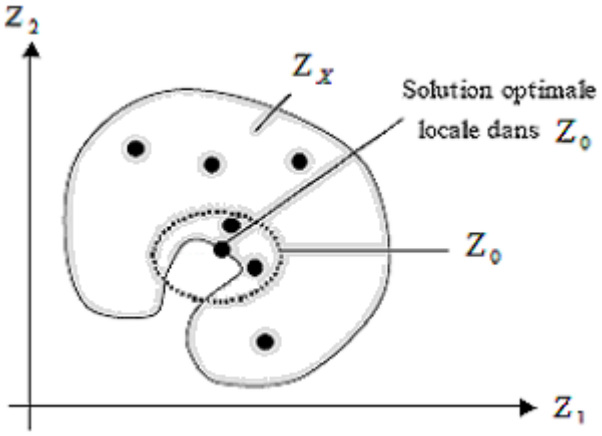

FIG. 1.3 – Optimisation locale au sens de Pareto

1.2.7 Solution fortement efficace

Une solution x est fortement efficace si et seulement si

$$\nexists x^* \in X; \quad \forall j \in \{1, \ldots, k\} \quad z_j(x^*) \leqslant z_j(x).$$

Une solution x est fortement efficace s'il n'existe pas une autre solution x^* telle que le vecteur critère (objectif) qui lui est associé, soit aussi bon que celui de x. On remarque que l'efficacité forte implique l'efficacité qui implique à son tour l'efficacité faible.

1.2.8 Ensemble des solutions efficaces

La collection de toutes les solutions efficaces (paréto optimales) du problème est appelée **l'ensemble efficace**. On le note X_E, l'image de cet ensemble par la fonction Z est appelée **ensemble non dominé** noté Z_{X_E}.

1.2.9 Solutions équivalentes

$x^1, x^2 \in X_E$ sont dites équivalentes si $Z(x^1) = Z(x^2)$.

1.2.10 matrice des gains

Soit $\widehat{x}_j$ une solution optimale du critère Z_j. La matrice $(k \times k)$ formée des éléments $z_{ij} = z_i(\widehat{x}_j)$ est dite matrice des gains (pay-off matrix).

$$\begin{pmatrix} \widetilde{z_1} & z_{12} & \dots & \dots & z_{1k} \\ z_{21} & \widetilde{z_2} & \dots & \dots & z_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{k1} & z_{k2} & \dots & \dots & \widetilde{z_k} \end{pmatrix}$$

La matrice des gains n'est pas toujours univoquement déterminée. En effet, si la solution optimale $\widehat{x}_j$ n'est pas unique pour un critère j, la matrice des gains dépendra de la solution choisie. La colonne correspondante de la matrice des gains dépendra de la solution choisie. Supposons par exemple que le premier critère Z_1 admet deux solutions optimales x^* et x^{**}. Deux matrices des gains peuvent être définies l'une dépend de x^* et l'autre de x^{**}.

$$G_1 = \begin{pmatrix} \widetilde{z_1}(x^*) & z_{12}(x^*) & \dots & \dots & z_{1k}(x^*) \\ z_{21} & \widetilde{z_2} & \dots & \dots & z_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{k1} & z_{k2} & \dots & \dots & \widetilde{z_k} \end{pmatrix}$$

$$G_2 = \begin{pmatrix} \widetilde{z_1}(x^{**}) & z_{12}(x^{**}) & \dots & \dots & z_{1k}(x^{**}) \\ z_{21} & \widetilde{z_2} & \dots & \dots & z_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{k1} & z_{k2} & \dots & \dots & \widetilde{z_k} \end{pmatrix}$$

1.2.11 Points particuliers

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

1. **Point idéal**

   Les coordonnées du point idéal $z^* = (z_1^*; z_2^*, \dots, z_k^*)$ correspondent aux meilleurs valeurs de chaque objectif des points du front Pareto. Les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objectif Z_i

séparément c-à-d : pour des critères à minimiser le point idéal se définit comme suit :

$$z_i^* = \underset{x \in X}{min}(z_i(x)) \quad i = \{1, \ldots, k\}. \tag{1.6}$$

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

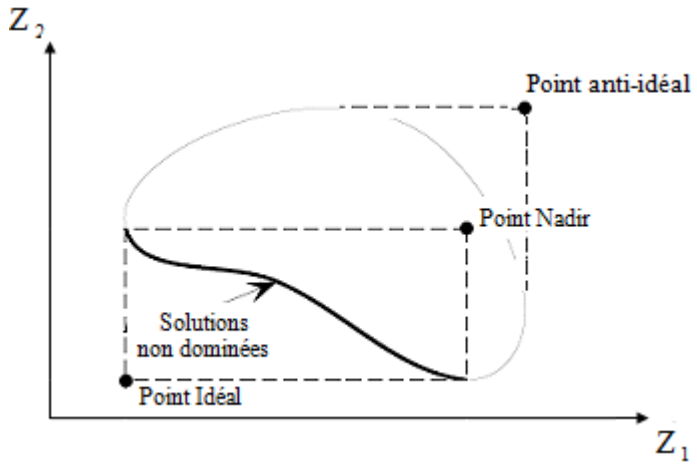

FIG. 1.4 – Points particuliers

2. **Point anti-idéal**

   Le point anti idéal est un point de $\mathbb{R}^k$, pour des critères à minimiser. Le point anti-idéal se définit comme suit :

$$z_i^* = \underset{x \in X}{max}(z_i(x)) \quad i = \{1, \ldots, k\} \tag{1.7}$$

3. **Point Nadir**

   Les coordonnées du point Nadir correspondent aux pires valeurs de chaque objectif des points du front Pareto.

   Pour des critères à minimiser le point nadir est défini comme suit :

$$n_i = \underset{j=\{1,\ldots,k\}}{max}(z_{ij}) \quad i = \{1, \ldots, k\} \tag{1.8}$$

   Où z_{ij} est un élément de la matrice des gains.

   Le point idéal (resp. le point Nadir) domine (resp. est dominé par) tous les autres points de la surface de compromis. Bien que ces points ne soient pas forcément

compris dans la zone réalisable, ils servent souvent de pôle d'attraction (resp. de répulsion) lors de la résolution du problème. On a tendance à s'approcher le plus du point ideal et de s'éloigner du point nadir (voir figure1.4).

1.2.12 Classification des solutions efficaces

Pour le développement des algorithmes de résolution pour les problèmes MOCO (Multiple objective combinatorial optimization), la classification suivante est importante.

Ensemble complet de solutions efficaces

Un ensemble de solution efficaces X_E est dit ensemble complet si pour tout point non dominé on trouve au moins une solution efficace associée à ce point.

Ensemble complet minimal de solutions efficaces

Un ensemble complet minimal de solutions efficaces X_{E_m} est un ensemble complet sans les solutions équivalentes. N'importe quel ensemble complet contient un ensemble minimal complet.

Ensemble complet maximal de solutions efficaces

L'ensemble complet maximal de solutions efficaces X_{E_M} est l'ensemble complet contenant toutes les solutions équivalentes.

1.2.13 Pourquoi cette classification de l'ensemble des solutions efficaces est importante ?

Lorsque deux solutions ont exactement les mêmes valeurs pour l'ensemble des objectifs, elles sont équivalentes dans l'espace objectif, mais peuvent correspondre à deux solutions différentes dans l'espace décisionnel. Une question importante est de savoir s'il est intéressant de garder ces deux différentes solutions. La réponse peut dépendre du contexte (type de problème étudié) en plus de la volonté des décideurs : lors de la résolution d'un problème comportant énormément de solutions efficaces, il est peut étre préférable de privilégier une bonne approximation de cet ensemble et donc favoriser la diversité (du coté objectif) des solutions retenues. Dans ce cas nous parlerons alors de recherche de X_{E_m}.
Au contraire, lorsque l'ensemble des solutions efficaces comporte peu de solutions, afin d'avoir une bonne représentation de l'ensemble des solutions non dominées, il sera intéressant de rechercher les solutions de même valeur. Dans ce cas nous parlerons alors de recherche de X_{E_M}.

1.2.14 Théorème de Geoffrion [9]

Soit le problème

$$(P_\lambda)\begin{cases} \underset{z\in Z_X}{min} Z_\lambda = \sum_{j=1}^{k} \lambda_j Z_j \\ avec \\ \lambda \in \Lambda = \{\lambda \in \mathbb{R}^k, \lambda_k > 0, \quad \sum_{k=1}^{n} \lambda_k = 1\} \end{cases} \tag{1.9}$$

Alors z^* est une solution optimale du problème paramétrique si et seulement si z^* est un vecteur non dominé.
L'implication dans l'autre sens c-à-d pour pouvoir affirmer que si z^* est un vecteur non dominé, il existe $\lambda \in \Lambda$ tel que z^* est solution optimale de (P_λ) n'est vrai que si Z_X soit convexe ce qui est le cas en programmation linéaire multi-objectifs en variables continues.

1.2.14.1 Exemple introduit par Bowman [10]

Considérons l'exemple suivant :

$$\begin{cases} \max Z_1(x) = & 6x_1 + 3x_2 + x_3 \\ \max Z_2(x) = & x_1 + 3x_2 + 6x_3 \\ \text{t.q} & \\ & x_1 + x_2 + x_3 \leqslant 1 \\ & x_j = (0,1) \quad j = \{1,2,3\} \end{cases} \tag{1.10}$$

Par application du théorème de Geoffrion nous avons que les solutions optimales du problème paramétrique sont des solutions efficaces.

$$\begin{cases} \max Z_\lambda(x) = & \lambda(6x_1 + 3x_2 + x_3) + (1-\lambda)(x_1 + 3x_2 + 6x_3) \\ \text{t.q} & \\ & x_1 + x_2 + x_3 \leqslant 1 \\ & x_j = (0,1) \quad j = \{1,2,3\} \\ & 0 < \lambda < 1 \quad j = \{1,2,3\} \end{cases} \tag{1.11}$$

Ces solutions optimales sont
$x^1 = (x_1 = 1, x_2 = 0, x_3 = 0), \quad z(x^1) = (6,1)$
$x^2 = (x_1 = 0, x_2 = 0, x_3 = 1), \quad z(x^2) = (1,6)$
Or il est facile de constater que la solution

$x^3 = (x_1 = 0, x_2 = 1, x_3 = 0)$, $z(x^3) = (3, 3)$ est également efficace mais n'est pas solution optimale du problème paramétrique. La raison est que l'ensemble des solutions admissibles n'est pas un ensemble convexe.

1.3 Quelques difficultés se posant à la présence des variables entières ou bivalentes

En effet en programmation multi-objectifs en variables entières et en optimisation combinatoire multi-objectifs, l'ensemble des solutions admissibles n'est pas un ensemble convexe ceci provoque l'existence de deux type de solutions efficaces, solutions efficaces supportées et solutions efficaces non supportées.

1.3.1 Ensemble des solutions efficaces supportées

C'est l'ensemble des solutions efficaces générées par la résolution du problème paramétrique :

$$(P_\lambda) \begin{cases} \min\limits_{z \in Z_X} Z_\lambda = \sum\limits_{j=1}^{k} \lambda_j Z_j \\ avec \\ \lambda \in \Lambda = \{\lambda \in \mathbb{R}^k, \lambda_j > 0, \sum\limits_{j=1}^{k} \lambda_j = 1\} \end{cases} \tag{1.12}$$

Tous les points non dominées supportées existent sur la frontière du convexe de Z. En variant λ toutes les solutions supportées peuvent être trouvées. X_{SE} représente l'ensemble des solutions efficaces supportées. Z_{SN} est l'ensemble des points supportés non-dominées.

Théorème 1 *[2] Toutes les solutions efficaces des problèmes de programmation linéaire multi-objectifs sont supportées.*

1.3.2 Solutions efficaces supportées extrêmes

Les solutions efficaces supportées x qui ont comme vecteurs objectifs $z(x)$ l'un des sommets de $conv(Z_X)$ sont appelées solutions efficaces supportées extrêmes et leur ensemble est noté X_{SE1}. $Z_{SN_1} = Z(X_{SE_1})$ est ainsi l'ensemble des points non dominés supportés extrêmes.

1.3.3 Solutions efficaces supportées non-extrêmes

Les solutions supportées x telles que $z(x)$ n'est pas un sommet de $conv(Z_X)$ et qui se trouvent sur l'une des faces de $conv(Z_X)$ sont appelées solutions supportées non extrêmes. Ces solutions forment l'ensemble noté X_{SE_2} et son image est Z_{SN_2}.
Les deux ensembles X_{SE_1} et X_{SE_2} peuvent être trouvés en résolvant (P_λ). Neanmoins trouver X_{SE_2} est généralement plus difficile que de trouver X_{SE_1} et ceci revient au fait que pour toutes les solutions efficaces x avec $Z(x)$ existant sur une même arête (facette) du convexe de Z_X $(conv(Z_X))$ le vecteur λ pour lequel x minimise (P_λ) est le même.
C'est pour quoi trouver X_{SE2} nécessite une énumération de toutes les solutions optimales de (P_λ).

1.3.4 Ensemble des solutions efficaces non supportées

Les premiers articles traitant les problèmes multi-objectifs en variables entières ou bivalentes comme dans le cas de l'optimisation combinatoire ne prenaient pas en considération les solutions non supportées.
La recherche de ces solutions est la cause de la difficulté théorique de ces problèmes. En effet, leur génération pose un problème vu que ces solutions ne ce trouvent pas sur la frontière efficace mais à l'intérieur de l'enveloppe convexe.
Les solutions efficaces non supportées sont des solutions efficaces qui ne sont pas optimales pour (P_λ) pour n'importe vecteur λ. Les points non dominés non supportés existent à l'interieur de conv Z. X_{NE} représente l'ensemble des solutions efficaces non supportées. Z_{NN} représente l'ensemble des points non dominés non supportés.

Une visualisation de la classification de l'ensemble des solutions efficaces pour un problème d'optimisation multi-objectifs dans le cas d'existence de variables entières ou bivalentes est donnée sur la figure 1.6.

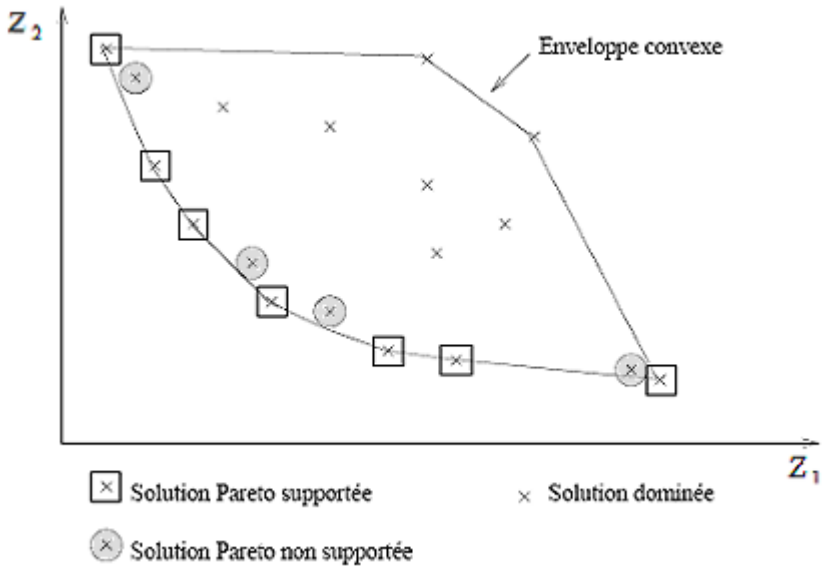

FIG. 1.5 – Solutions supportées et non supportées

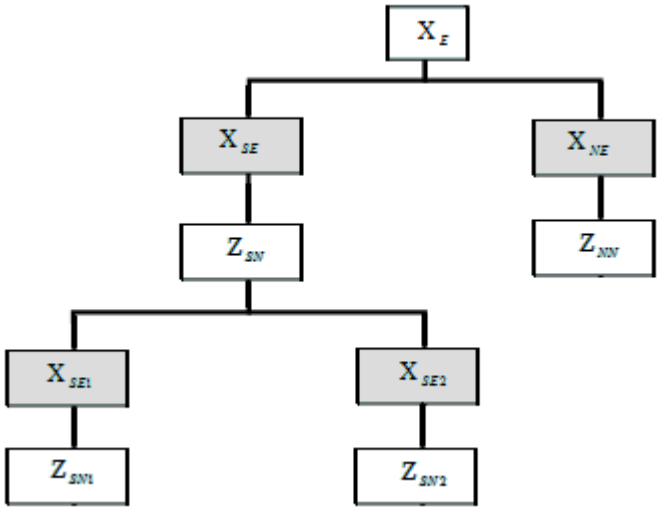

FIG. 1.6 – Classifications des solutions efficaces

1.4 Approches de résolution des problèmes multi-objectifs

Le choix de la méthode de résolution d'un problème multi-objectifs est d'une importance capitale et constitue une tache délicate. C'est pourquoi avant de se lancer dans la résolution il faut bien choisir la méthode qui convient le mieux. En effet, les méthodes de résolution des problèmes multi-objectifs se répartissent en trois familles en fonction du

moment où intervient le décideur :

1. **Les méthodes de résolution a priori**
   Ces méthodes sont rapides vu qu'elles permettent de trouver la solution recherchée en une seul exécution. En effet, le compromis désiré entre les critères se fait avant l'exécution de la méthode. Le premier inconvénient est que le décideur peut ne pas être satisfait de la solution trouvée et donc la recherche se relance avec un autre compromis et le deuxième inconvénient de cette méthode est qu'elle nécessite une modélisation des préference du décideur et ceci n'est pas evident. On retrouve dans cette famille la plupart des méthodes d'agrégation.
2. **Les méthodes de résolution progressives**
   Le décideur dans cette méthode est présent et intervient tout au long du processus de recherche de solutions en orientant la recherche. L'avantage de cette méthode est de permettre de bien prendre en compte les préférences du décideur. On retrouve dans cette famille les méthodes intéractives.
3. **Les méthodes de résolution a postériori**
   Ces méthodes cherchent à fournir au décideur un ensemble de bonne solutions bien réparties et c'est à lui de choisir parmi cet ensemble la solution qui lui convient. Ces méthodes alors ne nécessite ni la présence du décideur ni une modélisation de ses préférences mais trouver un ensemble de solutions bien réparties n'est pas une tache simple et nécessite un temps de calcul important ceci dit le procédé d'optimisation doit étre puissant.
   Dans ce type de méthodes deux phases sont à considérer :
   – La recherche des solutions de meilleurs compromis (solutions de Pareto optimale). C'est la phase de résolution multi-objectifs.
   – Le choix de la solution à retenir.
     C'est la tâche du décideur qui, parmi l'ensemble des solutions de compromis, doit extraire celle(s) qu'il utilisera. Une optimisation sous un ensemble de solutions efficaces ce fait dans ce cas.

Il existe des méthodes de résolution multi-objectifs qui n'appartient pas exclusivement à une des familles définies ci-dessus. Cette classification permet seulement de se faire une idée sur la démarche que l'on devra suivre pour obtenir notre résultat.
Dans ce mémoire nous nous intéressons aux méthodes de résolution a postériori et plus précisément à la recherche des solutions de meilleurs compromis.

1.5 Exposition de quelques méthodes de résolution pour les problèmes multi-objectifs

1.5.1 Méthode d'agrégation (de pondération)

Elle consiste à combiner les fonctions objectifs Z_i en une seul fonction Z de façon linéaire c-à-d : transformer le problème $(MOLP)$ en un problème paramétré $(MOLP)_\lambda$

$$(MOLP)_\lambda \begin{cases} Z(x) = & \sum_{i=1}^{k} \lambda_i Z_i(x) \\ \text{t.q} & x \in X \\ & \lambda_i > 0 \\ & \sum_{i=1}^{k} \lambda_i = 1 \end{cases} \tag{1.13}$$

Où $x \in \mathbb{R}^n$ et X est un sous ensemble de $\mathbb{R}^n$ c'est l'ensemble des solutions admissibles (réalisables) associés à des contraintes d'égalité ou d'inégalité .
Les solutions fournis par la résolution de ce problème sont dites **efficaces supportées**. Différents poids fournissent différentes **solutions efficaces supportées**. Les résultats obtenus dans la résolution du problème $(MOLP)_\lambda$ dépendent fortement des paramètres choisis pour le vecteur de poids λ. Les poids λ_i doivent aussi être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate. Ainsi, une approche généralement utilisée est de résoudre le problème $(MOLP)_\lambda$ avec différentes valeurs de λ.

1.5.2 Méthode ϵ-contrainte

1. **Principe**
   Une autre façon de transformer un problème d'optimisation multi-objectifs en un problème simple objectif est de convertir $k-1$ des k objectifs du problème en contraintes et d'optimiser séparèment l'objectif restant.
   Le problème peut être reformulé de la manière suivante :

$$(MOP_r(\epsilon)) \begin{cases} Min & Z_r(x) \\ \text{t.q} & x \in X \\ & Z_j \leqslant \epsilon_j, \quad j = \{1, \ldots, k\}, \quad j \neq r \end{cases} \tag{1.14}$$

Où $\epsilon = (\epsilon_1, \ldots, \epsilon_{r-1}, \epsilon_{r+1}, \ldots, \epsilon_k)$

Différentes valeurs de ϵ_j peuvent être données pour pouvoir générer différentes solutions Pareto optimales.
La connaissance a priori des intervalles appropriés pour les valeurs ϵ_j est requise pour tous les objectifs.
Pour pouvoir définir les valeurs adéquates pour ϵ_j le vecteur idéal doit être calculé pour déterminer les bornes inférieures. On aura donc :

$$\epsilon_j \geqslant z_j(x^*), \quad j = \{1, 2, \ldots, r-1, r+1, \ldots, k\}$$

2. **Fonctionnement de la méthode ϵ-contrainte**
   Aprés avoir effectuer un choix du vecteur $\epsilon = (\epsilon_1, \ldots, \epsilon_{r-1}, \epsilon_{r+1}, \ldots, \epsilon_k)$, on commence par optimiser le premier objectif selon l'ordre choisi des fonctions objectifs Z_r, les $r-1$ autres objectifs seront borné par les ϵ_i tel que $i \in \{1, \ldots, k\}$, $i \neq r$ et formerons $k-1$ autres contraintes en plus des contraintes du problème initial. Ainsi en résolvant ce problème on trouve la première solution efficace et la valeur optimale de Z_r noté Opt_r trouvée en l'optimisant détermine la borne Br sur l'objectif Z_{r-1} et ainsi de suite,...
   Le principe de la méthode ϵ-contrainte est intéressant lorsque l'on cherche à énumérer toutes les solutions d'un front Pareto. En effet, en utilisant cette méthode itérativement, en repartant à chaque fois de la solution trouvée pour définir la borne suivante, il est possible en utilisant une méthode exacte mono-objectif de générer, pour des problèmes combinatoires, l'ensemble des solutions Pareto optimales.

1.5.3 Méthode de Corley

Il existe une relation entre le modèle basé sur l'agrégation et le modèle basé sur la méthode ϵ-contrainte.

Théorème 2 (Chankong et Haimes 1983) *Supposons que les ensembles X et Z sont convexes. Si, pour un certain r, x^* est solution de $MOP_r(\epsilon)$, il existe alors λ tel que x^* est solution de MOP_λ, et inversement.*

Plusieurs hybridations de ces deux modèles ont été proposées. Ci-dessous un exemple de modèle combinant le modèle MOP_λ et le modèle $MOP_r(\epsilon)$.

$$MOP(\lambda,\epsilon)\begin{cases} Min & Z(x)=\sum_{i=1}^{n}\lambda_i Z_i(x) \\ \text{t.q} & x\in X \\ & Z_j(x)\leqslant \epsilon_j, j=\{1,\ldots,n\} \end{cases} \tag{1.15}$$

Cette méthode est connue sous le nom de méthode de Corley elle permet de combiner les avantages des deux méthodes citées précédemment (méthode d'agrégation des objectifs et la méthode $\epsilon-$contraintes). Elle a montré son efficacité sur différents type de problèmes qu'ils soient convexes ou non convexes. Cependant, une difficulté survient : le nombre de paramètres à déterminer a été multiplié par deux. Il sera beaucoup plus difficile au décideur d'exprimer ses préférences en jouant sur l'ensemble des paramètres.

1.5.4 Méthode lexicographique

1. **Principe**
   Cette méthode est très intuitive. En effet, elle consiste à considérer les fonctions objectifs les unes après les autres et à minimiser un problème d'optimisation mono-objectif, en complétant au fur et à mesure l'ensemble des contraintes.
2. **Présentation de la méthode**
   On part du problème P.
   On procède ensuite en k étapes (autant d'étapes qu'il y a de fonctions objectifs).
   Commençons avec la première fonction objectif. On résout :

$$\begin{cases} min & Z_1(x) \\ \text{t.q} & g(x)\leqslant 0 \\ & h(x)=0 \end{cases} \tag{1.16}$$

   g et h étant des matrices de contraintes, et x est le vecteur de decision.
   On note z_1^* la solution à ce problème.
   Ensuite, on transforme la première fonction objectif en contrainte d'égalité puis on prend la seconde fonction objectif et on résout le problème suivant :

$$\begin{cases} min & Z_2(x) \\ \text{t.q} & Z_1(x) = z_1^* \\ & g(x) \leqslant 0 \\ & h(x) = 0 \end{cases} \tag{1.17}$$

On répète cette démarche jusqu'à la fonction objectif k. Alors pour finir on aura :

$$\begin{cases} min & Z_{k-1}(x) \\ \text{t.q} & Z_1(x) = z_1^* \\ & Z_2(x) = z_2^* \\ & \vdots \\ & Z_{k-1}(x) = z_{k-1}^* \\ & g(x) \leqslant 0 \\ & h(x) = 0 \end{cases} \tag{1.18}$$

La solution de ce dernier programme est une solution efficace.

3. **Discussion**
   L'inconvénient de cette méthode est le choix de la séquence des objectifs à minimiser qu'elle requiert.

1.6 L'optimisation multi-objectifs Discrète

En effet certaines variables de décision dans la modélisation des programmes multi-objectifs ne peuvent prendre que des valeurs entières Si de tels variables existent alors le programme multi-objectifs est soit Mixte dans le cas où des variables continu existent aussi, soit un programme multi-objectifs en nombre entier est ce dans le cas où toutes les variables du problème sont entières.
Un trés grand nombre de problème réel se formule sous la forme d'un programme multi-objectifs en variable entière (discète), parmi les problèmes connu on peux citer les problèmes d'optimisation de réseaux de télécommunications, des problèmes de planification de production, des problèmes d'emploi du temps, de gestion d'équipages,...
Comme c'est déjà mentionner auparavant un programme linéaire multi-objectifs en variables entières se formule comme suit :

$$(MOILP)\begin{cases} \text{“opt”} & Z_j(x) = c_j x, \quad j = \{1, \ldots, k\} \\ \text{t.q} & x \in X \\ & X = \{x \in RL / x \in \mathbf{N}\} \end{cases} \tag{1.19}$$

Où $RL = \{x \in \mathbb{R}^n : \quad Ax \leqslant b, \quad x \geqslant 0\}$ est l'ensemble des solutions de la relaxation linéaire du problème $(MOLIP)$ qui se présente comme suit :

$$\underset{x \in RL}{\text{“opt”}} Z_j(x) = c_j x, \quad j = \{1, \ldots, k\} \tag{1.20}$$

Plusieurs méthodes ont été développer pour la résolution de ce type de problèmes, elles ont été classer suivant l'espace de recherche en deux catégories : méthodes de résolution dans l'espace des critères et dans l'espace de décision.

1.6.1 Méthodes de résolution d'un programme linéaire mono-objectif en nombres entiers

Un problème de programmation linéaire mono-objectif en nombre entier se formule comme suit :

$$(ILP)\begin{cases} opt & Z(x) = cx \\ \text{t.q} & x \in X \end{cases} \tag{1.21}$$

Où $X = S \cap \mathbb{Z}^n, \mathbb{Z}$ est l'ensemble des entiers relatifs.
$S = \{x \in \mathbb{R}^n | Ax = b, x \geqslant 0\}; A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^m, c \in \mathbb{R}^{1 \times n}$.
Si toutes les variables sont restreintes à 0 ou 1, le problème se formule comme suit :

$$(LPBIL)\begin{cases} opt & Z(x) = cx \\ \text{t.q} & Ax = b \\ & x \in \{0, 1\}^n \end{cases} \tag{1.22}$$

Les techniques qui sont souvent utilisées pour résoudre les problèmes linéaires mono-objectif en nombres entiers sont la méthode des coupes et la méthode de séparation et évaluation connue sous le nom de “Branch and Bound”, dans ce qui suit nous donnerons un aperçu sur des deux méthodes.

1. **Algorithme de Coupe fractionnaire de Gomory [16]**
   Soit le problème (ILP') le problème linéaire obtenu à partir de (ILP) en relâchant les contraintes d'intégrité sur les variables.

$$(ILP') \begin{cases} opt & Z(x) = cx \\ & Ax = b \\ & x \geqslant 0 \end{cases} \tag{1.23}$$

Définition 2 *J une base dans $\mathbb{R}^m$ est tout ensemble $\{j^1, j^2, \ldots, j^m\}$ tel que $j^i \in \mathbb{R}^m$ de m vecteurs linéairement indépendents. Rappelons qu'un ensemble de vecteurs est linéairement indépendent si et seulement si l'unique vecteur $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ pour lequel $\alpha_1 j^1 + \alpha_2 j^2 + \cdots + \alpha_m j^m = 0 \in \mathbb{R}^m$ est le vecteur où $\alpha_1 = \alpha_2 = \cdots = \alpha_m = 0$*

Le problème suivant est un problème équivalent à (ILP) et représente son écriture sous forme canonique par rapport à une base J.

$$(ILPC)' \begin{cases} \widetilde{c}^{\overline{J}} x_{\overline{J}} = Z(opt) - \pi b \\ x_J + (A^J)A^{\overline{J}} x_{\overline{J}} = (A^J)^{-1} b \\ x \geqslant 0 \end{cases} \tag{1.24}$$

où :
$\overline{J} = \{\{1, 2, ..., n\} \backslash J\}$ ces les indices hors base
$\pi = c^J (A^J)^{-1}$ est dit vecteur muliplicateur relative à la base J
$\widetilde{c} = c - \pi A$ est dit vecteur coût relatif à la base J

- A une base J du problème linéaire (ILP') on associe la solution de base $\overline{x}$ correspondant à J, définie par
  $\overline{x}_j = 0$ pour $j \in \overline{J}$.
- Une base J est dite réalisable si la solution de base associée est réalisable pour (ILP'), c'est à dire si $\overline{x}_J = (A^J)^{-1} b \geqslant 0$.
- Si le vecteur coût $\widetilde{c}$ relatif à une base réalisable J est négatif ou nul, la solution de base correspondante est solution optimale de (ILP'). La base J est alors dite "base optimale".

Soit le problème $(ILPC)'$. On sait que si les trois conditions suivantes sont satisfaites alors $\overline{x}$, solution de base relative à la base J, est solution optimale de (ILP).

(a) $c \leqslant 0$

(b) $b \geqslant 0$

(c) b entier

Etant donné le programme en nombres entiers (ILP) on dit que l'inéquation $ax \leqslant \alpha$ est valide si elle est satisfaite par toute solution réalisable de (ILP).
Une **Coupe** est une inéquation valide qui n'est pas satisfaite pour tout point de D', domaine des solutions réalisables de la relaxation (ILP') de (ILP).

Théorème 3 *[17]*
Si α est un scalaire quelconque, on désigne par :
– $\lfloor \alpha \rfloor$ *représente la partie entière inférieure.*
– $\lceil \alpha \rceil$ *représente la partie entière supérieure.*
– $\langle \alpha \rangle = \alpha - \lfloor \alpha \rfloor$.
Pour toute valeur du scalaire α, l'inéquation
$\sum_{j \notin J}(\lfloor \alpha \rfloor A_i^j - \lfloor \alpha A_i^j \rfloor)x_j \geqslant \lfloor \alpha \rfloor b_i - \lfloor \alpha b_i \rfloor$ *est valide. Pour certaines valeurs de α c'est une coupe et en particulier pour $\alpha = 1$ on a la coupe de Gomory*
$\sum_{j \notin J} \langle A_i^j \rangle x_j \geqslant \langle b_i \rangle$

2. **Méthode de séparation et d'évaluation (branch and bound) [17]**

Cette méthode utilise deux concepts :

(a) **Le branchement** : qui consiste à diviser un ensemble de solutions en sous-ensembles.

(b) **L'évaluation** : qui consiste à borner ou minorer les solutions.

Dans le cas des programmes linéaires en nombre entier (ILP), la méthode du "Branch and Bound" commence par résoudre la relaxation linéaire du programme linéaire, c'est à dire en enlevant les conditions d'intégralité sur les variables. Cette évaluation est toujours inférieure à la valeur de la solution optimale du (ILP), puisque toute solution du (ILP) est une solution de la relaxation linéaire. Par conséquent si la solution optimale de la relaxation linéaire est entière, alors c'est une solution optimale du (ILP). Sinon on effectue une opération de branchement en choisissant une variable x_i^* non entière dans la solution optimale x^* du programme linéaire que l'on a résolu. L'ensemble des solutions se divise alors en deux, celles pour lesquelles $x_i \leqslant \lfloor x_i^* \rfloor$ et celles pour lesquelles $x_i \geqslant \lfloor x_i^* \rfloor + 1$. On évalue ensuite les nouveaux noeuds et éventuellement on élague ceux qui sont inutiles. Un noeud peut être élagué dans trois cas possibles :

(a) Le programme linéaire n'est pas réalisable.

(b) La valeur de la solution est supérieure à la valeur de la meilleure solution réalisable trouvée, dans ce cas il est inutile de continuer la recherche dans la sous-arborescence enracinée en ce noeud, puisque les bornes inférieures obtenues sont strictement croissantes suivant la profondeur de l'arbre du "Branch and Bound".

(c) La solution est entière, donc réalisable.

L'algorithme s'arrête quand on n'a plus de noeud à évaluer. L'arborescence obtenue par les branchements appliqués au problème d'optimisation P est appelée arbre du "Branch and Bound", qu'on note $T = (N, E)$ où N représente l'ensemble des noeuds de l'arbre et E les arcs correspondant au processus de branchement. Il y a trois types de noeuds dans l'arbre du "Branch and Bound" pendant le déroulement de l'algorithme, le noeud courant qui est en train d'être évalué, des noeuds actifs qui sont dans la liste des noeuds qui doivent être traités, et des noeuds inactifs qui ont été élagués au cours du calcul. Quand la liste des noeuds actifs est vide, la meilleure solution réalisable obtenue est la solution optimale du problème. Notons que le choix de la stratégie de la sélection du noeud actif à traiter influe lourdement sur la taille de l'arborescence du "Branch and Bound" visitée. De même pour le choix de la variable de branchement.

1.6.2 Méthodes de résolution d'un programme linéaire multi-objectifs en nombre entier

1. **Méthode de Klein-Hannan [18]**

Cette méthode consiste à résoudre progressivement des programmes linéaires mono-objectif avec des contraintes ajoutées à chaque itération.

– Etape1 (Initialisation)

Choisir arbitrairement un critère $i \in \{1, 2, \ldots, k\}$ et résoudre le problème mono-objectif suivant :

$$(P_{r_0}) \begin{cases} max & c^i x \\ \text{t.q} & x \in X \end{cases} \tag{1.25}$$

Si la solution du problème (P_{r_0}) est unique, alors elle est efficace et elle est l'unique élément dans la liste initiale des solutions efficaces $IE_0(P)$. Sinon, soit $\zeta(P_{r_0})$

l'ensemble des solutions optimales de (P_{r_0}) et $IE_0(P)$ l'ensemble des solutions efficaces correspondant à $\zeta(P_{r_0})$.

– Etape j $(j \geqslant 1)$

On résout le problème (P_{r_j}) défini par :

$$(P_{r_j}) \begin{cases} \max \quad Z_i = c^i x \\ \text{t.q} \quad x \in X \\ \bigwedge_{l=1}^{r} (\bigvee_{s=1, s\neq i}^{p} (c^s x \geqslant c^s \tilde{x}_l + \epsilon^s)) \end{cases} \tag{1.26}$$

Où $\bigvee$ est l'opérateur d'addition logique et $\bigwedge$ est l'opérateur du produit logique. $0 < \epsilon^s \leqslant 1$ pour $\epsilon^s < 1$, la méthode produit un sous ensemble de l'ensemble des solutions efficaces, mais si $\epsilon^s = 1$, la procedure donne toutes les solutions efficaces, $\tilde{x}_l; \quad l = \{1, 2, \ldots, r\}$ sont les solutions efficaces obtenues dans les itérations $\{0, 1, \ldots, j-1\}$ respectivement.

Les contraintes supplémentaires assurent que les solutions optimales du problème (P_{r_j}), si elle existent, seront meilleures que toutes les solutions efficaces $\{\widetilde{x}_l | l = 1, 2, \ldots, r\}$ sur au moins un critère $s \neq i$.

La liste des solutions efficaces obtenue à l'étapes j est :

$$IE_j(P) = \bigcup_{l=0}^{j-1} IE_l(P) \tag{1.27}$$

La procedure s'arrête dès que la problème devient irréalisable.

2. **Méthode de Villareal-Karwan [19–21]**

Dans cette méthode les éléments de la matrice des contraintes (A) et de la matrice des critères (C) du problème $(MOILP)$, sont supposés entiers non-négatifs et les variables de décision $x_j, 0 \leqslant x_j \leqslant k_j, \quad k_j$ entier positif pour tout $j = \{1, \ldots, n\}$. On résout progressivement tous les programmes $P_i(Y^i)$ définis comme suit pour $i = \{1, \ldots, n\}$ et $Y^i = \{1, \ldots, b\}$

$$P_i(Y^i) \begin{cases} \text{“opt”} \sum_{j=1}^{i} c_j x_j \\ \sum_{j=1}^{i} a_j x_j \leqslant Y^i \\ 0 \leqslant x_j \leqslant k_j, \quad j = \{1, \ldots, i\} \end{cases} \tag{1.28}$$

Où c_j est le jème vecteur colonne de C, a_j est le jème vecteur colonne de la matrice des contrainte A, Y^i est le second membre défini par $Y^i = 1, ..., b$ avec b le second membre des contraintes du problème $(MOILP)$.
Puis on détermine à chaque fois $H_i(Y^i)$, ensemble des solutions efficaces de l'étape i, avec comme second membre Y^i. Pour $i = n$ et $Y^n = b$, on obtient l'ensemble des solutions efficaces $IE(P) = H_n(Y^n)$.

1.7 Problèmes multi-objectifs à variables binaires

C'est des problèmes qui se modélise sous forme de programme ayant des variables de décision binaire c'est à dire qu'ils ne peuvent prendre que deux valeurs 0 ou 1. Ces problèmes sont appelés $MOBLP$ (Multiple Objective Binary Linear Programs). Un grand nombre de problème rencontré dans la réalité se formule sous cette forme citons comme exemple les problèmes d'optimisation combinatoire multi-objectifs tel que le problème de sac à dos multi-objectifs, d'affectation multi-objectifs, le problème de voyageur de commerce,...
Les problèmes de cette classe se formulent comme suit :

$$(MOBLP) \begin{cases} \text{“max”} & Z_k = c^k x, \quad k = \{1, 2, \ldots, p\} \\ & x \in S' \end{cases} \tag{1.29}$$

Où $S' = \{x \in \{0,1\}^n | Ax \leqslant b\}$

1.7.1 Quelques Méthodes de résolution

1. **Méthode de Bitran [22, 23]**
   Soit le problème :

$$(P_{B_1}) \begin{cases} \text{“}\max_{x \in S_1}\text{”} & Z_k = c^k x, \quad k = \{1, 2, \ldots, p\} \end{cases} \tag{1.30}$$

   avec $S_1 = \{0,1\}^n$.
   Toute solution efficace pour le problème (P_{B_1}) et réalisable et efficace pour le problème $(MOBLP)$

Une solution qui n'est pas efficace pour le problème (P_{B_1}) peut par contre être efficace pour le problème $(MOBLP)$

(a) Caractérisation de $P(B_1)$:
Considérons l'ensemble V dit ensemble de directions de préférences, où T est un ensemble d'indices.

$$V = \{v^t \in \mathbb{R}^n, \quad t \in T| \quad Cv^t \geqslant 0, \quad v_j^t = 0, 1 \quad \text{ou} - 1, \quad \forall j\}$$

On dit que x^* domine x dans la direction v^t si et seulement si $x^* = x + v^t$.
Donc $Cx^* \geqslant Cx$
Soit $M(v^t)$ l'ensemble des points de S_1 dominés dans la direction v^t par un autre point de S_1. On a $M(v^t) = \{x^{t,r}|r = \{1, 2, \ldots, R\}\}$ avec :

$$x^{t,r} = \begin{cases} 0 & & & \text{si} & v_j^t = 1 \\ 1 & & & \text{si} & v_j^t = -1 \\ 0 & \text{ou} & 1 & \text{si} & v_j^t = 0 \end{cases}$$

Bitran a montré que cet ensemble peut être déterminé par l'ensemble des solutions optimales du problème suivant :

$$\begin{cases} \underset{x \in S_1'}{min} & v^t x \end{cases} \tag{1.31}$$

Où $S_1' = \{x \in \mathbb{R}^n| \quad 0 \leqslant x_j \leqslant 1, \quad \forall j\}$ est la relaxation linéaire de S_1.
L'ensemble $E(P_{B_1})$ des solutions efficaces de (P_{B_1}) est déterminé à partir de la résolution successive des problèmes relaxés, notés (RP_{B_1}), pour différentes directions $v^t, \quad t \in T$.
Ce dernier permet de trouver des ensembles $M(v^t)$ puis $E(P_{B_1}) = \overline{\bigcup_{t \in T} M(v^t)}$ notons que quelques directions seulement sont examinées.

(b) Détermination de l'ensemble $E_1 = E(P_{B_1}) \cap S'$, c-à-d tester l'admissibilité des solutions de $E(P_{B_1})$.

(c) Caractérisation de l'ensemble $E(MOBLP)$ des solutions efficaces du problème $(MOBLP)$: Pour déterminer l'ensemble des solutions non efficaces dans (P_{B_1})

et efficaces dans ($MOBLP$), ensemble noté E_2. La propriété suivante est utilisé :

$x \in E_2 \Rightarrow x + v^t \notin S, \quad \forall v^t, \quad t \in T \quad \text{t.q} \quad x \in M(v^t)$

$$E(MOBLP) = E_1 \cup E_2$$

2. **Méthode de J.Sylva et A.Crema [24]**

Soit le problème (P_{02}) suivant :

$$(P_{02}) \begin{cases} \text{“max”} & Z = Cx \\ \text{t.q} & x \in X = \{Ax = b; x \in \mathbb{Z}^n, x \geqslant 0\} \end{cases} \tag{1.32}$$

Où $C \in \mathbb{Z}^{k \times n}, \quad A \in \mathbb{R}^{m \times n}$ et $b \in \mathbb{R}^m$. k est le nombre de critères(objectifs), m nombres de critères, n nombre de variables.

La proposition suivante sert à justifier l'approche.

Propriété 1 *Soient $x^1, x^2, \ldots, x^m$ des solutions efficaces du problèmes (P_{02}) et $\triangle_l = \{x \in \mathbb{Z}^n | Cx \leqslant Cx^l, \quad l = \{1, 2, \ldots, m\}\}$ Si x^* une solution efficace du problème (P_{02}^m) définit comme suit :*

$$(P_{02}^m) \begin{cases} \text{“max”} & Z = Cx \\ t.q & x \in (X - \bigcup_{l=1}^m \Delta_l) \end{cases} \tag{1.33}$$

Alors x^* est une solution efficace pour le problème (P_{02}^m).En plus, si le problème (P_{02}^m) devient impossible, alors $\{Cx^l, \quad l = \{1, \ldots, m-1\}\}$ est l'ensemble de tous les points non dominés dans l'espace des critère.

Corollaire 1 *Soient $x^1, x^2, ..., x^m$ des solutions efficaces du problème (P_{02}) et $\Delta_l = \{x \in \mathbb{Z}^n | Cx \leqslant Cx^l, \quad l = \{1, \ldots, k\}\}$. Si x^* est une solution optimale pour le problème unicritère (P_λ^m).*

$$(P_\lambda^m) \left\{ max \quad \lambda' Cx | x \in (X - \bigcup_{l=1}^m \Delta_l) \right. \tag{1.34}$$

Pour certain vecteur $\lambda \in \mathbb{R}^k, \lambda > 0$, alors x^* est une solution efficace pour le problème (P_{02}).

Déscription de la méthode

(a) **Etape1**

Après avoir choisi un paramètre $\lambda, \quad \lambda > 0$, la résolution du problème $(P^{(0)})$ de programmation linéaire en variables entières mono-objectif se fait :

$$(P^{(0)}) \left\{ \; max \quad \lambda' Cx | Ax = b, \quad x \geqslant 0, \quad x \in \mathbb{Z}^n \; \right\}. \tag{1.35}$$

Si ce problème est impossible, alors le problème (P_{02}) est aussi impossible. Sinon, par le corollaire une solution optimale x^1 trouvée est efficace pour le problème (P_{02}).

Une suite de problèmes mono-objectif est résolue séquentiellement en ajoutant à chaque itération des contraintes éliminant les solutions déjà trouvées précédemment. Après m itérations du processus, si le problème $(P^{(m-1)})$ est irréalisable, alors l'algorithme s'arrête. Sinon, une nouvelle solution efficace est génerêée et un nouveau problème $(P^{(m)})$ est défini en éliminant de l'ensemble d'admissibilité de $(P^{(m-1)})$ toutes les solutions vérifiant $Cx \leqslant Cx^m$.

$(Cx)_l \geqslant ((Cx^m)_l + 1)y_l^m - M_l(1 - y_l^m), \quad l = \{1, 2, \ldots, p\}$

et $\sum_{l=1}^{k} y_l^m \geqslant 1, \quad y_l^m \in \{0, 1\}, \quad l = \{1, 2, \ldots, k\}.$

Où $-M_k$ est la borne inférieure pour toute valeur réalisable de kème fonction objectif.

(b) **Etape m**

Résoudre le problème :

$$(P^m) \begin{cases} max & \lambda' Cx \\ \text{t.q} & Ax = b \\ & (Cx)_l \geqslant ((Cx^i)_l + 1)y_l^i - M_l(1 - y_l^i), \\ & i = \{1, 2, ..., m\}, \quad l = \{1, 2, \ldots, k\} \\ & \sum_{l=1}^{p} y_l^i \geqslant 1, \quad y_l^m \in \{0, 1\}, \\ & i = \{1, 2, \ldots, m\}, \quad l = \{1, 2, \ldots, k\} \\ & x \in \mathbb{N} \end{cases} \tag{1.36}$$

Remarque 2 *Pour des problèmes de grandes dimensions, l'énumération de*

toutes les solutions efficaces est généralement impossible et ne semble pas intéréssante, l'intérêt se porte sur une partie seulement des solutions efficaces. Dans ce cas le problème (P^m) devient :

$$(P'^m)\begin{cases} max & \lambda' Cx \\ t.q & Ax = b \\ & (Cx)_l \geqslant ((Cx^i)_l + f_l)y_l^i - M_l(1 - y_l^i), \\ & i = \{1, 2, \ldots, m\}, \quad l = \{1, 2, \ldots, k\} \\ & \sum_{l=1}^{p} y_l^i \geqslant 1, \quad y_l^m \in \{0, 1\}, \\ & i = \{1, 2, \ldots, m\}, \quad l = \{1, 2, \ldots, k\} \\ & x \in \mathbb{N} \end{cases} \tag{1.37}$$

Où f_l représente l'amélioration minimale dans la kième fonction objectif.
La procédure continue jusqu'à ce que le problème (P^m) devienne impossible à résoudre.
A la fin de l'algorithme, Soit l'ensemble de toutes les solutions efficaces est obtenu, soit une partie seulement qui intéresse le décideur.

Cette méthode est très astucieuse et coûteuse en terme de temps de calcul vu qu'elle permet de traiter et déterminer les solutions efficaces supportées. Pour des applications de grande dimensions, les auteurs ont seulement implémenté la méthode pour les problèmes de sac à dos avec trois et deux fonctions objectifs, dix contraintes et jusqu'à 30 variables.

CHAPITRE 2

Optimisation combinatoire multi-objectifs

2.1 Les problèmes d'optimisation combinatoires

Parmi les problèmes posés par la recherche opérationnelle les problèmes d'optimisation combinatoire sont fréquemment rencontrés. Ces problèmes consistent à trouver la meilleure solution suivant un critère donné parmi un ensemble discret fini de solutions possibles. L'optimisation combinatoire comprend des problèmes très divers tels que le problème de voyageur de commerce, les problèmes d'affectation et d'ordonnancement ... Certain de ces domaines présentent de forts enjeux économiques et pratiques.

2.1.1 Complexité

Supposons que l'on dispose d'une mesure de la taille du problème. Le nombre d'opérations élémentaires effectuées par un algorithme de résolution de ce dernier est donc une fonction qui dépond de n la taille des données. Cette fonction est appelée la complexité. Le but de cette fonction est de comparer la vitesse des différents algorithmes destinés à la résolution d'un même problème. La complexité d'un problème est la complexité de son meilleur algorithme connu.

2.1.2 Algorithme non déterministe

Le concept d'algorithme non déterministe s'agit d'un outil imaginaire mais théorique trés utile. Un algorithme non déterministe ressemble à un algorithme ordinaire, sauf qu'il est équipé d'une instruction supplémentaire appelée instruction de choix.

2.2 Les classes de problèmes

On peut classer les problèmes suivant leur complexité. On distingue deux familles de problèmes.

- **Les problèmes de classe** $\mathbb{P}$: Ce sont les problèmes qui posséde une complexité polynomiale c'est à dire qu'un algorithme polynomial a été mis en evidence pour leurs résolutions.
- **Les problème de classe** $\mathbb{NP}$: Un problème appartient à la classe $\mathbb{NP}$ s'il existe un algorithme non déterministe polynomial pour le resoudre.
  autrement dit pour un problème de classe $\mathbb{NP}$ on ne sait pas encore trouver un algorithme ordinaire qui peut résoudre toutes les instances du problème de façon optimale avec un temps d'excécution qui augmente de façon polynomiale par rapport à la taille de l'instance.

2.2.1 Les problèmes $\mathbb{NP}$-complets

Pour beaucoup de problèmes de la classe $\mathbb{NP}$, on ne sait pas dire s'ils peuvent ou non admettre un algorithme polynomial. Il a été montré l'existance d'une classe d'équivalence particulière de problèmes dits "$\mathbb{NP}$-complets". Avant de la définir, il convient de rappeler le théorème suivant :

Théorème 4 *Soient A et B deux problèmes de décision. Si A se réduit polynomialement à B et B est dans* $\mathbb{P}$*, alors A est aussi dans* $\mathbb{P}$*.*
Un problème de decision A est dit $\mathbb{NP}$*-complet si :*

1. *A appartient à la classe* $\mathbb{NP}$*.*
2. *Tout problème de cette classe peut se réduire polynomialement à A.*

2.2.2 problème $\mathbb{NP}$-difficile

On dira qu'un problème est $\mathbb{NP}$-difficile si on peut montrer qu'il ne peut pas se résoudre polynomialement.
Les algorithmes polynomiaux sont considérés comme "efficace". En effet le temps de leurs exécutions augmente de manière raisonnable avec la taille du du problème.
Les algorithmes non-polynomiaux sont dit "inefficaces" car leur temps d'exécution augmente de façon spectaculaire avec la taille de l'instance traitée.
Dans le cas des problèmes $\mathbb{NP}$-difficiles, deux possibilités sont offertes. Si le problème est de petite taille, alors un algorithme exact permettant de trouver la solution optimale peut

être utilisé (procédure de séparation et évaluation (Branch and Bound), programmation dynamique ...). Malheureusement, ces algorithmes par nature énumératifs, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles.
Dans ce cas, il est nécessaire de faire appel à des heuristiques permettant de trouver de bonnes solutions approchées. Parmi ces heuristiques, on trouve les métaheuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes.

2.3 Problèmes combinatoires multi-objectifs

L'optimisation combinatoire multi-objectifs est une branche d'optimisation combinatoire. Les problèmes d'optimisation combinatoire multi-objectifs (MOCO) sont des problèmes ardus car ils combinent les difficultés des problèmes combinatoires classiques avec des préférences complexes sur leur solutions. C'est cependant sous cette forme que se présentent la plupart des problèmes industriels réels.

2.3.1 Formalisation mathématique

Un problème $MOCO$ peut être formulé comme suit :

$$(MOCO)\begin{cases} \text{“opt”} & Z(x) = & (Z_1(x), Z_2(x), \ldots, Z_k(x)) \\ \text{t.q} & & x \in X \end{cases} \tag{2.1}$$

Où $k \geqslant 2$ est le nombre de fonctions objectifs.
$x = (x_1, \ldots, x_n)$ est le vecteur représentant les variables de decision.
X est dit espace de décision c'est l'ensemble fini discret des solutions admissibles (réalisables) associés à des contraintes d'égalité, d'inégalité et des bornes explicites. $Z(x) = (Z_1(x), Z_2(x), \ldots, Z_k(x))$ est le vecteur des critères à optimiser.

Remarque 3 *Un problème MOCO est dit $\mathbb{NP}$-problème si la vérification q'une solution est ou pas non-dominé ne peut pas se faire en un temps polynomial pour toute les instances du problème.*
Un problème MOCO est $\mathbb{NP}$-problème si le problème mono-objectif qui lui correspond c'est à dire le problème formé par l'un des objectifs et le même ensemble des solutions réalisable est un problème $\mathbb{NP}$.

Serafini(1986) [25] a prouvé que quelques problèmes mono-objectif qui appartiennent à la classe $\mathbb{P}$ se transforment en problèmes $\mathbb{NP}$-complet en leurs formes bi-objectifs. Ceci est le cas du problème du plus court chemin et le problème d'affectation qui sera l'objet de ce mémoire.

2.4 Quelques problèmes d'optimisation combinatoire multi-objectifs

2.4.1 Problème de localisation multi-objectifs

C'est un problème relevant de la classe $\mathbb{NP}$, connu sous le nom MOLOP (Multiple Objective LOcation Problem) et peut se formuler comme suit :

$$(MOLOP)\begin{cases} \text{“min”}Z_k(x) = & \displaystyle\sum_{i=1}^{m} f_i^k y_i + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij} \quad k = \{1,\ldots,K\} \\ & \displaystyle\sum_{i=1}^{m} x_{ij} = 1 \quad j = \{1,\ldots,n\} \\ & a_i y_i \leqslant \displaystyle\sum_{j=1}^{n} d_{ij} x_{ij} \leqslant b_i y_i \quad i = \{1,\ldots,m\} \\ & x_{ij} \in \{0,1\} \quad \forall i = \{1,\ldots,m\}, \forall j = \{1,\ldots,n\} \\ & y_i \in \{0,1\} \quad \forall i = \{1,\ldots,m\} \end{cases} \tag{2.2}$$

$y_i = 1$ si un centre de service est installé en i, x_{ij} égale à 1 si le client j est affecté au centre installé en i (il est donc supposé qu'un client n'est affecté qu'un seul centre), d_{ij} est un certain usage du centre i par le client j lorsqu'il est affecté à ce centre ; a_i et b_i sont les bornes inférieures et supérieures d'usage au centre i ; c_{ij}^k est un coût, une distance ou une durée pour le critère k lorsque le client j est affecté au centre i, f_i^k est, pour le critère k, le coût fixe associé au centre situé en i. Dans certaines études le nombre de sites à choisir est fixé à l'avance ($\sum_{i=1}^{m} y_i = p$).

Parmi les articles qui existe dans la littérature pour la résolution de ce type de problème nous trouvons l'article de Lee S.M, Green G.I and Kim C.S [26],l'article de Pelegrin P. and Fernandez F.R [27] et l'article de Ross T. and Soland R [28].

2.4.2 problème de Voyageurs de commerce multi-objectifs

C'est un problème relevant de la classe $\mathbb{NP}$ connu sous le nom MOTSP qui veut dire Multiple Objective Traveling Salesman Problem. Un voyageur de commerce doit réaliser une tourne en visitant une et une seule fois n villes tout en minimisant les divers frais et risques qui résultent de ces déplacements.

Le problème MOTSP se formule mathématiquement comme suit :

$$(MOTSP)\begin{cases} \text{“min”}Z_k(x) = \displaystyle\sum_{i=1}^{n} c_{ij}^k x_{ij} \quad k = \{1,\ldots,K\} \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \\ \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\ \displaystyle\sum_{i\in T}\sum_{j\in \overline{T}} x_{ij} \geqslant 1 \quad \forall T \subset N, \overline{T} = N\backslash T \\ x_{ij} \in \{0,1\} \quad \forall i,j \end{cases} \tag{2.3}$$

Où $N = \{1,2,\ldots,n\}$ l'ensemble des villes à visiter. $x_{ij} = 1$ si l'on effectue le trajet allant de i à j et 0 sinon.
Il n'existe pas beaucoup d'articles dans la littérature qui traitent ce problème dans son contexte multi-objectifs, parmi ces articles nous trouvons l'article de Fisher et Richter [29], l'article de Gupta et Warburton [30] et l'article de Keller et Goodchild [31].

2.4.3 Problème de sac à dos multi-objectifs

C'est un problème relevant de la classe $\mathbb{NP}$, son problème mono-objectif est $\mathbb{NP}$-complet c-à-d on ne sait pas encore résoudre le problème de sac à dos mono-objectif par un algorithme polynomial.

Supposons qu'on dispose de n objets ayant des poids (ou volumes) w_j et W la capacité totale (en poids ou volume) d'un sac à dos (ou chargement). A chaque objet on associe K valeurs $c_j^k, \quad k = \{1,2,\ldots,K\}$ mesurant l'importance des objets selon K critères.
Le problème de sac à dos multi-objectifs consiste à remplir le sac d'objets de manière à maximiser les différentes valeurs des objets choisis, sans dépasser la capacité W. Il peut se formuler comme suit :

$$(MOKP)\begin{cases} \text{“max”}Z_k(x) = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_i^k x_i \quad k=\{1,\ldots,K\} \\ \displaystyle\sum_{i=1}^{n} x_i w_i \leqslant W \\ x_{ij} \in \{0,1\} \quad i,j=\{1,\ldots,n\} \end{cases} \tag{2.4}$$

Où c_i^k, w_i sont des entiers positifs, W est une constante positive et $x = (x_1, \ldots, x_n)$ est la matrice des variables de decision.
Pour la résolution de ce type de problèmes plusieurs algorithmes de résolution ont été proposés citons comme exemple l'article de M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu [32], article de V. Barichard and J.K. Hao qui ont proposé une méthode approchée pour sa résolution [33].

2.4.4 Le problème d'affectation multi-objectifs

C'est un problème relevant de la classe $\mathbb{P}$ bien évidement parce que l'algorithme de Khun connu sous le nom de la méthode Hongroise [34–36] représente un algorithme polynomial pour la résolution du problème d'affectation mono-objectif.
Le problème d'affectation consiste à affecter n personnes à n tâches. Dans le cas mono-objectif, un coût est associé à l'affectation d'une personne à une tâche et l'idée est de déterminer la permutation des n objets ayant le coût total minimal. Dans le cas multi-objectifs MOAP (Multi-Objective Assignment Problem), K coûts sont associés à l'affectation de chaque personne à chaque tâche et le problème revient à déterminer l'ensemble des solutions efficaces.
Soit i l'indice des personnes, j l'indice des objets et c_{ij}^k le coût de l'affectation de i à j par rapport au critère k. Ce problème se formule mathématiquement comme suit :

$$(MOAP)\begin{cases} \text{“min”}Z_k(x) = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij} \quad k=\{1,\ldots,K\} \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad j=\{1,\ldots,n\} \\ \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad i=\{1,\ldots,n\} \\ x_{ij} \in \{0,1\} \quad i,j=\{1,\ldots,n\} \end{cases} \tag{2.5}$$

Où c_{ij}^k sont des entiers positifs et $x = (x_{11}, \ldots, x_{nn})$ est la matrice des variables de decision telle que $x_{ij} = 1$ si la personne i est affecté à l'objet j et est égale à 0 sinon.

Pour la résolution de ce problème plusieurs méthodes se sont élaborées notamment celles de Malhotra [37, 38], les méthodes qui reposent sur la méthode en deux phases [11, 12], le goal programming [39], etc. Dans ce mémoire le problème d'affectation bi-objectifs a été le centre de notre intêret, nous citons dans ce qui suit quelques méthodes qui existent dans la littérature pour sa résolution.

Théorème 5 *Le problème d'affectation multi-objectifs est* $\mathbb{NP}$*-complet.*
La preuve du caractère $\mathbb{NP}$*-complet de ce problème a été donnée par [Seraffini 86] [25]*

2.4.5 Méthodes de résolution exacte pour le problème d'affectation multi-objectifs

Le but des méthodes exactes est de trouver l'ensemble des solutions efficaces exactes pour cela elles doivent parcourir l'ensemble de l'espace de recherche ou au moins avoir l'assurance de n'écarter aucune solution qui peut être meilleure que la solution déjà trouvée par l'algorithme.
Les premiers papiers qui ont traité le problème d'affectation multi-objectifs n'ont pas pris en considération les solutions non supportées, les méthodes de résolution utilisées se basaient soit sur des techniques de programmation par objectif ou bien sur l'utilisation des combinaisons convexes des objectifs.
Les auteurs qui se sont rendu compte de l'existance des solutions non supportées pour ce type de problèmes ont proposés des algorithmes ayant pour but la détermination d'un ensemble complet de solutions efficaces dans le contexte bi-objectifs.
Malhotra et al. [37, 38] ont proposé un algorithme pour la résolution du problème d'affectation bi-objectifs mais Ulungu [11] a proposé une alternative meilleure.
Ulungu et Teghem [12, 40] ont proposé ensuite une méthode en deux phases avec une deuxième phase spécifique au problème d'affectation bi-objectifs, Tuyttens et Teghem [13] ont réalisé une implémentation de cette méthode les temps de résolution se sont montrés élévés donc ceci à conduit une limitation de la taille des instances traités.
Dans ce qui suit nous donnerons un apercu sur les méthodes de résolution exacte les plus répondues pour la résolution du problème d'affectation bi-objectifs noté (BAP).

2.4.6 Méthode en deux phases originale [11]

La méthode en deux phases présente un schéma de résolution exacte trés intéréssant elle a montré son efficacité pour la résolution de différent problèmes combinatoire bi-objectifs notamment le problème d'affectation bi-objectifs.
L'avantage de cette méthode est qu'elle permet l'utilisation des méthodes mono-objectif tout au long de son algorithme ceci nécessite alors avoir une méthode efficace et si possible polynomiale pour le problème mono-objectif. Cependant si pour le problème mono-objectif on ne procure pas d'une méthode puissante pour sa résolution l'application de la méthode en deux phases s'avère pas intéréssante vu que son temps d'exécution devient trés grand.
Comme son nom l'indique cette méthode se fait en phase :

1. **Phase1 : Détermination des solutions supportées**
   L'objectif de la première phase est d'obtenir l'ensemble des solutions efficaces supportées X_{SE}.
   Comme nous l'avons vu précédemment, ces solutions ont l'avantage d'étre relativement faciles à trouver puisqu'elles optimisent une combinaison linéaire des objectifs (la recherche de ces solutions est basée sur le théorème de Geoffrion [9] une solution optimale pour (BAP_λ) avec un vecteur λ positif est efficace).
   On note S l'ensemble des solutions efficaces supportées. $\widetilde{S}$ est l'ensemble des solutions efficaces supportées extrêmes. Cet ensemble est initialisé par les deux solutions optimales (et efficaces en cas d'unicité de la solution optimale) x^1 et x^2 respectivement des deux critères, donnant lieu, dans le plan des critères, aux valeurs $(\widehat{z_1}, z_2), (z_1, \widehat{z_2})$. Les solutions de $\widetilde{S}$ sont rangées par valeur croissante du critère Z_1. Deux solutions consécutives qui ne sont pas équivalentes x^r et x^s telles que $z_1(x^r) < z_1(x^s)$ et $z_2(x^r) > z_2(x^s)$, sont séléctionnées.
   Un problème d'optimisation (BAP_λ) de matrice $C^{(t)} = [\lambda_1^{(t)} c_{ij}^{(1)} + \lambda_2^{(t)} c_{ij}^{(2)}]$ avec $\lambda_1 = z_2(x^r) - z_2(x^s)$ et $\lambda_2 = z_1(x^s) - z_1(x^r)$ est résolu et toutes les solutions optimales sont énumérées. Soit $\{x^t : t \in T\}$ l'ensemble des solutions optimales de (BAP_λ) obtenu par la méthode hongroise alors :
   (a) Soit $\{x^r, x^s\} \cap \{x^t : t \in T\} = \emptyset$
   Dès lors les solutions x^t sont des nouvelles solutions supportées et elles sont inclues dans l'ensemble $\widetilde{S}$ ensemble des solutions efficaces supportées extrêmes (voir figure2.1).
   (b) Soit $\{x^r, x^s\} \subset \{x^t : t \in T\}$ alors
   – Si x^r et x^s sont les deux seules solutions x^t, dans ce cas il n'y a pas d'autres

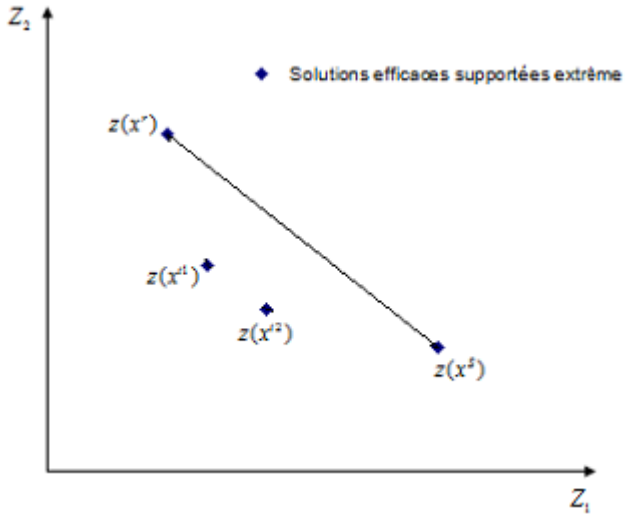

FIG. 2.1 – Phase1 Cas a)

solutions supportées entre x^r et x^s.

– Dans le cas contraire c-à-d l'ensemble $\{x^t : t \in T\}$ contient les deux solutions x^r et x^s et d'autres solutions différentes de ces dernières à ce moment là les autres solutions x^t sont des nouvelles solutions supportées mais dont la representation z^t dans le plan des critères est situé sur la droite engendrée par les points z^r et z^s c'est des solutions qu'on appelle efficaces supportées non extrêmes et qu'on place dans l'ensemble $\widetilde{S}'$ (voir figure2.2).

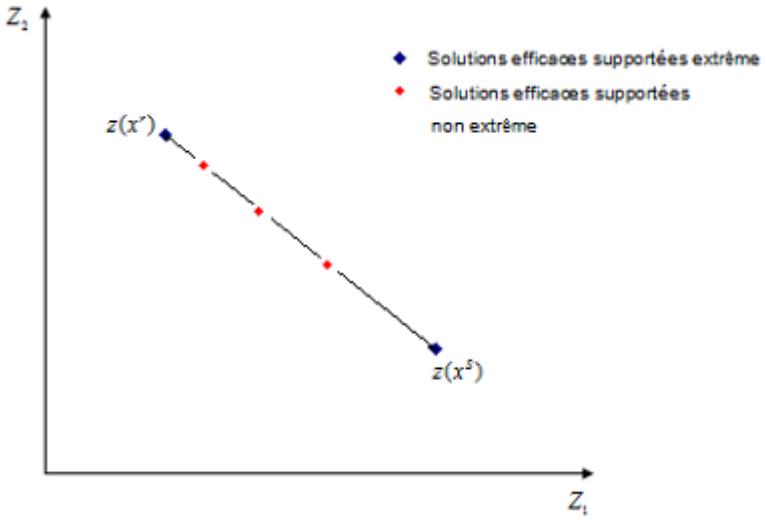

FIG. 2.2 – Phase1 Cas b)

Cette première phase est poursuivie tant que toutes les paires (x^r, x^s) de l'ensemble $\widetilde{S}$ n'ont pas été examinées, en particulier dans le cas a) ci-dessus, il

convient d'examiner les paires (x^r, x^t) et (x^t, x^s).

A la fin de la première phase, on obtient :

$S = \widetilde{S} \cup \widetilde{S}'$.

2. **Phase2 : Détermination des solutions non supportées**

La deuxième phase consiste alors en la recherche des solutions non supportées efficaces. Ces solutions ne peuvent pas être obtenues par combinaison d'objectifs. Ulungu et Teghem proposent alors d'utiliser les solutions suppotées trouvées afin de réduire l'espace de recherche en argumentant que les solutions Pareto non supportées restantes sont forcément dans les triangles basés sur deux solutions supportées consécutives.

L'espace de recherche des solutions non-dominées est réduit par l'utilisation des solutions supportées utilisées comme des bornes.

L'objectif est de rechercher, pour chaque paire (x^r, x^s) de solution de S s'il existe des solutions qui vérifient

$$z_1(x^r) < z_1(x^u) < z_1(x^s) \tag{2.6}$$

$$z_2(x^r) > z_2(x^u) > z_2(x^s) \tag{2.7}$$

de semblables solutions seraient situées à l'intérieur du triangle $\Delta\,[z(x^s)Yz(x^r)]$ tel que $Y = (z_1(x^s), z_2(x^r))$.

Le problème paramétrique (BAP_λ) associé au couple (x^r, x^s) est optimisé par la méthode Hongroise nous considérons les affectations (i, j) de coût réduit strictement positif.

x^r, x^s sont solutions optimales du problème (BAP_λ) et tous les coûts réduits sont donc non négatifs.

Soit $L = \left\{(i,j), \overline{c}_{ij}^{(t)} > 0\right\}$ les affectations étant classées dans l'ordre croissant des valeurs $\overline{c}_{ij}^{(t)}$ l'imposition d'une de ces affectations conduit à réoptimiser le problème (BAP_λ) réduit de dimension $(n-1) \times (n-1)$.

Géométriquement l'imposition d'une affectation de L imposera, lors de la réoptimisation, de quitter la droite $[z(x^r), z(x^s)]$.

Cependant toutes les affectations de L ne sont pas candidates à être imposées ; certaines peuvent être éliminées d'office de la liste L, étant donné qu'elles conduiraient certainement à des solutions dominées situées dans les zones décrites par les figures 2.3,2.4,2.5. L'idée de la détermination d'une affectation pouvant être éliminée de la liste L est de calculer une borne inférieure de l'augmentation, respectivement pour

les trois cas :

– Du critère correspondant à la matrice $C^{(t)}$ par rapport à la valeur prise par les points $z(x^r)$ et $z(x^s)$ (voir figure2.3).
– Du critère Z_1 par rapport à la valeur $z_1(x^s)$ (voir figure2.4)
– Du critère Z_2 par rapport à la valeur $z_2(x^r)$ (voir figure2.5)

engendrés par l'imposition de cette affectation. Ces bornes inférieures sont calculées à partir des matrices de coût réduit.

(a) **Test1 : Sur le problème** (BAP_λ)

Soit (i^*j^*) une affectation de L et soit i_r, j_r,i_s,j_s les indices tels que
$x_{i_rj^*} = x_{i^*j_r} = x_{i_sj^*} = x_{i^*j_s} = 1$
i_r,i_s d'une part, j_r,j_s d'autre part, pouvant désigner le même indice.
Par rapport à la solution x^r l'imposition de i^*j^* impose au minimum de créer un autre zéro dans la ligne i_r et dans la colonne j_r que ceux correspondant aux affectations (i_r, j^*) et (i^*, j_r).
Si ce nouveau zéro est créé en (i_r, j_r) la borne inférieure de l'augmentation de $\tilde{z}^{(t)}$ sera $\bar{c}^{(t)}_{i_rj_r}$, si deux nouveaux zéros sont crés, la borne inférieure de l'augmentation sera

$$\Upsilon_r^{(t)} \equiv \underset{j\neq j^*}{min}\bar{c}^{(t)}_{i_rj} + \underset{i\neq i^*}{min}\bar{c}^{(t)}_{ij_r} \tag{2.8}$$

Le même raisonnement peut être fait en référence x^s. Aussi la borne inférieure de l'augmentation de $\tilde{z}^{(t)}$ sera :

$$l_t = \bar{c}^{(t)}_{i^*j^*} + min(\bar{c}^{(t)}_{i_rj_r}; \bar{c}^{(t)}_{i_sj_s}; \Upsilon_r^{(t)}; \Upsilon_s^{(t)}) \tag{2.9}$$

L'affectation (i^*, j^*) peut être élimine de la liste L si

$$l_t \geqslant a_1^{(t)} z_1(x^s) + a_2^{(t)} z_2(x^r) - \tilde{z}^{(t)} \tag{2.10}$$

Etant donné que

$$\tilde{z}^{(t)} = a_1^{(t)} z_1(x^s) + a_2^{(t)} z_2(x^s) = a_1^{(t)} z_1(x^r) + a_2^{(t)} z_2(x^r) \tag{2.11}$$

Le test s'écrit

$$l_t \geqslant a_1^{(t)}.a_2^{(t)} \tag{2.12}$$

Bien évidemment, le membre de gauche $a_1^{(t)}.a_2^{(t)}$ n'est qu'une borne inférieure de l'augmentation de $\tilde{z}^{(t)}$. Par ailleurs, même si l'augmentation réelle de $\tilde{z}^{(t)}$

est inférieure à $a_1^{(t)}.a_2^{(t)}$, les solutions correspondantes peuvent néanmoins être situées à l'extérieur du triangle $\Delta[z(x^s)Yz(x^r)]$. Les test 2,3 essaient de détecter ces cas.

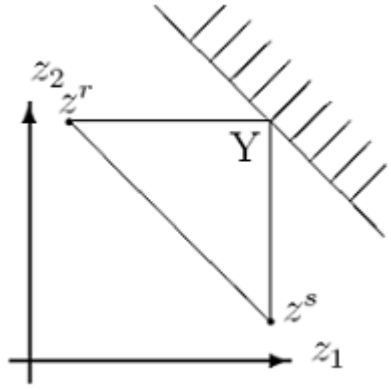

FIG. 2.3 – Test1

(b) **Test2 : Test d'augmentation de** $z_1(x^r)$

– $x^r = \tilde{x}^1$

Dans ce cas comme au test 1 la matrice des coûts réduits de z_1 relativement à x^r ne contient que des éléments non négatifs. Aussi le même type de raisonnement que celui du test1 peut être appliqué.

L'affectation (i^*, j^*) peut être éliminée de la liste L si

$$l_1 \geqslant z_1(x^s) - z_1(x^r) = a_2^{(t)} \tag{2.13}$$

Où l_1, la borne inférieure de l'augmentation de $z_1(x^r)$ engendrée par l'imposition de l'affectation (i^*, j^*) est

$$l_1 = \bar{c}_{i^*j^*}^{(1)} + min(\bar{c}_{i_1j_1}^{(1)}; \underset{j\neq j^*}{min}\bar{c}_{i_1j}^{(1)} + \underset{i\neq i^*}{min}\bar{c}_{ij_1}^{(1)}) \tag{2.14}$$

avec i_1 et j_1, indices tels que $\tilde{x}_{i_1j^*} = \tilde{x}_{i_*j_1} = 1$

– $x^r \neq \tilde{x}_1$

Dans ce cas la matrice des coûts réduits de z_1 relativement à x^r contient des éléments négatifs qui peuvent être utilisés lors d'une réoptimisation. Aussi la borne infrieure $z_1^{(r)}$ est calculée sur base des éléments minimaux situés dans les de l'augmentation de (n - 1) lignes (colonnes) qui peuvent être réaffectées.

L'affectation (i^*, j^*) peut être éliminée de la liste L si

$$l_1' = \bar{c}^{(1)}_{i^*j^*} + max(\sum_{i \neq i^*} \underset{j \neq j^*}{min} \bar{c}^{(1)}_{ij}; \sum_{j \neq j^*} \underset{i \neq i^*}{min} \bar{c}^{(1)}_{ij}) \geqslant a_2^{(t)} \tag{2.15}$$

Ce test se révelera de moins en moins efficace au fur et mesure que la valeur de z_1^r sera plus grande, c'est à dire que la solution x^r est éloignée de $\tilde{x}_1$. La présence de davantage de coûts réduits négatifs indique que l'imposition de (i^*, j^*) peut conduire des diminutions de $z_1^{(r)}$.

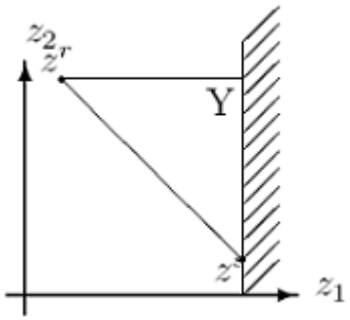

FIG. 2.4 – Test2

(c) **Text3 : Test d'augmentation de** $z_2(x^s)$

– $x^s = \tilde{x}^2$

L'affectation (i^*, j^*) peut être éliminé de la liste L si :

$$l_2 = \bar{c}^{(1)}_{i^*j^*} + min(\bar{c}^{(2)}_{i_2j_2}, \underset{j \neq j^*}{min} \bar{c}^{(2)}_{i_2j}; \underset{i \neq i^*}{min} \bar{c}^{(1)}_{ij_2}) \geqslant a_1^{(t)} \tag{2.16}$$

Avec $\tilde{x}^2_{i_2j^*} = \tilde{x}^2_{i^*j} = 1$ et où les coûts réduits $\bar{c}^2_{ij}$ sont ceux du critère z_2 relativement à la solution x^s.

– $x^s \neq \tilde{x}^2$

L'affectation (i^*, j^*) peut être éliminée de la liste L si :

$$l_2' = \bar{c}^{(2)}_{i^*j^*} + max(\sum_{i \neq i^*} \underset{j \neq j^*}{min} \bar{c}^{(2)}_{ij}; \sum_{j \neq j^*} \underset{i \neq i^*}{min} \bar{c}^{(2)}_{ij}) \geqslant a_1^{(t)} \tag{2.17}$$

Ce test (3') n'a une chance d'être efficace que pour les faibles valeurs de $z_2(x^s)$.

Les affectations restantes de L sont imposées tour à tour et le même problème (BAP_λ) est réoptimisé ; les solutions obtenues sont triées,seules les solutions effi-

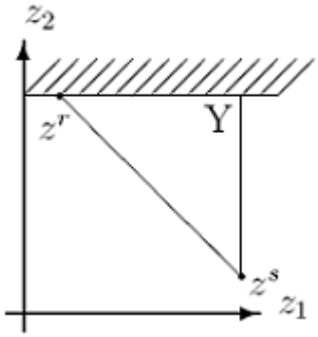

FIG. 2.5 – Test3

caces sont conservées.

Il n'est toutefois pas nécessaire de considérer toutes les affectations de L pour déterminer si une nouvelle solution x^u déterminée dans le triangle $\Delta[z(x^s)Yz(x^r)]$ est efficace.

En effet, si x^u correspond à la valeur $z(x^u)$ pour l'optimisation du critère (BAP_λ) et que toutes les bornes inférieures l_t des autres affectations de L indiquent que leur imposition conduira à une valeur supérieure à $z(x^u)$, ceci signifie que x^u est une solution efficace non supportée (voir figure2.6).

Une fois une nouvelle solution non supportée est trouvée le test1 est actualisé avec une nouvelle borne égale à $max(a_1^{(t)}(z_1(x^u) - z_1(x^r)), a_2^{(t)}(z_2(x^u) - z_2(x^s)))$.

Cette deuxime phase est répétée pour chaque paire (x^r, x^s) de solutions de $\widetilde{S}$. Une réoptimisation est arrétée dès que l'on constate par application de la méthode hongroise que la valeur réoptimisée $z(x^u)$ vérifie :

$$z(x^u) \geqslant \tilde{z}^{(t)} + a_1^{(t)} a_2^{(t)} \tag{2.18}$$

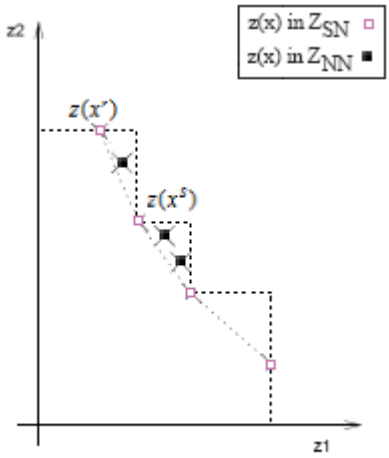

FIG. 2.6 – Deuxième phase

2.4.7 Méthode en deux phases Ulungu, Teghem 1995 [12], Tuyttens et Teghem 2000 [13]

1. **Phase 1 :Détermination des solutions efficaces supportées**

   A la différence de la phase 1 proposé par Ulungu en 1993 [11], La phase une dans cette version commence par les deux solutions x_1 et x_2 optimales pour Z_1 et Z_2 et détérminées d'une manière lexicographique.

   Le but de cette phase comme c'est déjà mentionner est de determiner l'ensemble X_{SE} ensemble des solutions efficaces supportées. On note S l'ensemble des solutions efficaces obtenues par l'algorithme, S est initialisé avec les deux solutions optimales pour Z_1 et Z_2 déterminées lexicographiquement et qui correspondent respectivement à $lexmin_{x\in X}(Z_1(x), Z_2(x))$ et $lexmin_{x\in X}(Z_2(x), Z_1(x))$.

   Les solutions de S sont triées par valeurs croissantes de Z_1.

   Deux solutions consécutives qui ne sont pas équivalentes x^r et x^s telles que $z_1(x^r) < z_1(x^s)$ et $z_2(x^r) < z_2(x^s)$(car si $z_2(x^r) > z_2(x^s)$ la solution x^s domine x^r).

   Un problème d'optimisation (BAP_λ) avec $\lambda_1 = z_2(x^r) - z_2(x^s)$ et $\lambda_2 = z_1(x^s) - z_1(x^r)$ est résolu et toutes les solutions optimales sont énumérées.

   La recherche est initialisée avec $x^r = x^1$ et $x^s = x^2$.

   Soit $R = \{x^t : t \in T\}$ l'ensemble des solutions optimales de (BAP_λ), où T est un ensemble d'indices tel que $card(T)$ est le nombre de solutions optimales de (BAP_λ).

   Pour deux points y^r et y^s dans $\mathbb{R}^2$, nous notons $[y^r y^s]$ le segment joignant les points y^r et y^s. Il ya deux cas possibles :

(a) Si $z(R) \cap [z(x^r)z(x^s)] = \emptyset$, alors toutes les solutions x^t sont de nouvelles solutions supportées et sont ajoutées à S.

Ensuite les solutions de $\{x^t : t \in T\}$ avec les valeurs minimales et maximales pour z_1 sont calculées. Soient x^{t_1} et x^{t_2} les solutions où le minimum et le maximum sont atteints. Pour poursuivre la recherche, deux nouveaux problèmes définis par une somme pondérée sont considérés : L'un défini par les solutions x^r et x^{t_1} et l'autre défini par les solutions x^{t_2} et x^s. Il n'est pas nécessaire de considérer un problème pondéré défini par deux solutions dans R parce que le poids λ obtenu est alors le même que celui défini par x^r et x^s et que par conséquent aucune nouvelle solution ne peut être obtenue.

(b) $z(R) \subset [z(x^r)z(x^s)]$, alors toutes les solutions x^t sont des solutions supportées et les nouvelles solutions éventuelles sont ajoutées à S mais aucun nouveau problème pondéré n'est généré.

La phase1 s'arrête s'il n'y a plus aucun problème pondéré (BAP_λ) à résoudre, on aura alors $S = X_{SE}$.

2. **Phase 2 :Détermination des solutions efficaces non-supportées**

Dans la deuxième phase, on determine des solutions $x \in X$ telles que $z(x)$ se situe dans le triangle défini par deux points supportées consécutifs $z(x^r)$ et $z(x^s)$ dans l'espace des objectifs.

Dans le but de limiter l'exploration dans le triangle considéré, des bornes inférieures et supérieures sur la valeur des fonctions objectifs ont été proposées par (Ulungu et Teghem)et (Tuyttens et Teghem).

(a) **Bornes inférieures**

Les bornes inférieures évitent l'exploration de solutions qui ne peuvent pas être efficaces. Soient $x \in X$ et C la matrice des coefficients de la fonction objectif d'un problème d'affectation mono-objectif.

ils s'intéressent ici à la valeur de la fonction objectif qui resulte de la fixation d'une variable x_{ij}=1. Dans les algorithmes, C sera soit C^1, soit C^2 soit $C\lambda = \lambda_1 C^1 + \lambda_2 C^2$ avec $\lambda_1 > 0$ et $\lambda_1 > 0$, ou une sous matrice carrée de celles-ci. Ces bornes inférieures sont calculées à partir d'une solution x dont on connaît la valeur pour la fonction objectif.

Ulungu et Teghem [12] ont proposé deux bornes inférieures différentes selon les cas où x est optimale pour le problème mono-objectif ou pas.

Supposons que x soit optimale pour le problème mono-objectif $min_{x \in X} Cx$. Alors la résolution du problème correspondant permet l'obtention d'une ma-

trice de coûts réduits $\bar{C}$ avec des coéfficients tous positifs ou nuls. On suppose qu'on impose $x_{i^*j^*} = 1$, mais dans la solution il ya deux indices i_t et j_t tels que $x_{i_tj^*} = x_{i^*j_t} = 1$. Par conséquent, dans la solution optimale du problème avec la variable fixée, la valeur d'au moins une des variables dans la ligne i_t et dans la colonne j_t, autre que $x_{i_tj^*}$ et $x_{i^*j_t} = 1$, sera égale à 1. Deux cas sont possibles :

– Si ces deux variables coïncident, on a $x_{i_tj_t} = 1$ et la variation(l'augmentation) de la fonction objectif sera au moins $\bar{c}_{i_tj_t}$.
– Sinon l'augmentation est au moins

$$\gamma = \underset{j\neq j^*}{min}\bar{c}_{i_tj} + \underset{i\neq i^*}{min}\bar{c}_{ij_t} \tag{2.19}$$

Alors, une borne inférieure sur la fonction objectif avec la fixation de la variable $x_{i^*j^*} = 1$ est

$$\alpha_1 = Cx + \bar{c}_{i^*j^*} + min\{\bar{c}_{i^tj^t}, \gamma\} \tag{2.20}$$

S'il y a plus d'une solution optimale pour le problème mono-objectif considéré, on peut calculer une borne inférieure pour chacune de ces solutions, et la meilleure sera donnée par la plus grande valeur parmi les α_1 calculés.

Supposons maintenant que x ne soit pas optimale pour le problème mono-objectif $min_{x\in X}Cx$. Soit $\bar{C}$ une matrice de coûts réduits correspondant à une solution de base, ou obtenue à l'aide d'une solution duale, qui vérifie $\bar{c}_{ij} = c_{ij} - u_i - v_j = 0$ pour tout (i, j) tel que $x_{ij} = 1$. Ici, toute matrice de coûts réduits contient des coefficients négatifs.

Puisque la variation de la fonction objectif provoquée par la fixation d'une variable peut ici être négative, la borne inférieure sur la variation est calculée en utilisant les éléments minimaux de toutes les lignes et toutes les colonnes de $\bar{C}$. Par conséquent, une borne inférieure sur la valeur de la fonction objectif après la fixation d'une variable est donnée par

$$\alpha_2 = Cx + \bar{c}_{i^*j^*} + max\{\sum_{i\neq i^*} \underset{j\neq j^*}{min}\bar{c}_{ij}, \sum_{j\neq j^*} \underset{i\neq i^*}{min}\bar{c}_{ij}\} \tag{2.21}$$

(b) **Borne supérieure de Tuyttens et Teghem [13]**

Dans la phase2, chaque triangle défini par deux points supportées consécutifs doit être exploré. Soient x^r et x^s deux solutions supportées consécutives dans X_{SE_m} et λ le poids pour lequel x^r et x^s sont deux solutions optimales de

(BAP_λ). Ulungu et Teghem [12] ainsi que Tuyttens et Teghem [13] ont proposé des bornes supérieures sur la valeur de la fonction objectif z^λ définie par :

$$Z^\lambda = \lambda_1 Z_1(x) + \lambda_2 Z_2(x) \tag{2.22}$$

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 la phase 2, 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. Dans le pire des cas, la droite parallèlle contient le point $(z_1(x^s), z_2(x^r))$, donc toute 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 $(z(x^r), z(x^s))$.

Tuttens et al ont amélioré la borne supérieure proposée par Ulungu et Teghem en utilisant tous les points réalisables déjà explorés dans le triangle.

L'algorithme utilise un ensemble de solutions potentiellement efficaces X_{PE} obtenues durant l'exploration. Soit $\{x^i : 0 \leqslant i \leqslant q\}$l'ensemble X_{PE} trié par valeur croissante de z_1.

Soit $\gamma = max_{i=1}^{q}\{\lambda_1 z_1(x^i) + \lambda_2 z_2(x^{i-1})\}$. Une borne supérieure sur les points non-dominés du triangle est donnée par

$$\beta_0 = max\{\gamma, \lambda_1 z_1(x^0) + \lambda_2 z_2(x^r), \lambda_1 z_1(x^s) + \lambda_2 z_2(x^q)\} \tag{2.23}$$

Comme toutes les solutions $x \in X$ telles que $z(x) \in \Delta(x^r, x^s)$ avec $\lambda_1 z_1 + \lambda_2 z_2 \geqslant \beta_0$ sont dominées, l'énumération de toutes les solutions $x \in X$ avec $z(x)$ entre les deux droites permet l'obtention de toutes les solutions non-supportées avec $z(x)$ dans le triangle.

(c) **Exploration des triangles**

Soient x^r et x^s deux solutions consécutives dans X_{SE_m} et λ le poids tel que x^r et x^s sont des solutions optimales de (BAP_λ). La phase 2 détermine des solutions admissibles x telles que $z_1(x^r) < z_1(x^s)$ et $z_2(x^r) > z_2(x) > z_2(x^s)$.

Les points correspondants à toutes ces solutions se situent dans le triangle $\Delta(x^r, x^s)$(voir figure2.7). Ulungu et Teghem proposent une méthode d'exploration énumérative pour chercher ces solutions. Le principe est de trouver des solutions non-supportées en imposant des affectations, c'est à dire en fixant des variables x_{ij} dans (BAP_λ), et en résolvant (où plutôt en réoptimisant) le

problème d'affectation de taille réduite qui en résulte. Autrement dit, la phase 2 recherche des solutions admissibles x qui ne sont pas optimales pour (BAP_λ) en partant de x^r et x^s et en modifiant ces solutions.

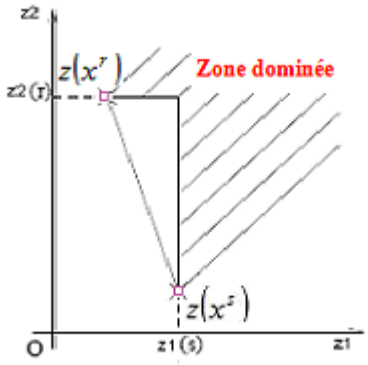

FIG. 2.7 – Zone dominée

On peut ici parler de réoptimisation car Ulungu et Teghem proposent d'utiliser la matrice de coûts réduits obtenue lors de la résolution du problème sans variable fixée, cela permet de réutiliser les zéros présents dans cette matrice pour une optimisation à moindre coût en pratique.
Pour la fixation des variables, Ulungu et Teghem proposent d'utiliser la liste d'affectation

$$L = \{(i,j) : \bar{c}_{ij}^{\lambda} > 0\}, \tag{2.24}$$

Où $\bar{C}^{\lambda}$ est la matrice de coûts réduits obtenue par la résolution du problème (BAP_λ) pour lequel x^r et x^s sont optimales. En imposant une affectation de L, des solutions dont les points correspondants se situent au-dessus de la droite $(z(x^r)z(x^s))$ sont obtenus. Il n'est heureusement pas nécessaire d'imposer toutes les affectations de L. En effet, il est possible de supprimer des affectations de L en testant si l'imposition de ces affectations ne génère que des points qui se situent à l'extérieur de $\Delta(x^r, x^s)$. Pour cela, les bornes inférieures de Ulungu et Teghem sont utilisées. Il y a 3 tests :

– **Test 1 :**
  Utilisation de la borne inférieure l_λ sur z^λ par comparaison avec $z^\lambda(x^r) = z^\lambda(x^s)$. l_λ est donnée par la plus grande des valeurs α_1 obtenues en utilisant l'equation (2.19) de la section 2.4.6 avec les solutions x^r et x^s et la matrice de coûts réduits obtenue par la résolution du problème.

L'affectation peut être supprimée de L, si l_λ montre que toute solution contenant cette affectation $x_{ij} = 1$ est située au-dessus de la droite parallèle à $(z(x^r)z(x^s))$ contenant le point $y = (z_1(x^s), z_2(x^r))$. L'augmentation de z^λ par rapport aux solutions supportées est :

$$\lambda_1 z_1(x^s) + \lambda_2 z_2(x^r) - \lambda_1 z_1(x^r) + \lambda_2 z_2(x^r) = \lambda_1 \lambda_2. \tag{2.25}$$

par conséquent, l'affectation peut être supprimée si

$$l_\lambda \geqslant z_\lambda(x^r) + \lambda_1 \lambda_2. \tag{2.26}$$

– **Test 2 :**
Utilisation d'une borne inférieure l_1 sur z_1 par comparaison avec $z_1(x^s)$. l_1 est obtenue soit par la valeur α_1 en utilisant l'équation (2.19) de la section 2.4.6 et x^r soit par la valeur α_2 en utilisant l'equation (2.20) de la section 2.4.6 suivant les cas où x^r est optimale pour z^1 ou pas.
L'affectation peut être supprimée de L, si l_1 montre que toute solution contenant cette affectation se situe à droite de la droite verticale passant par $z(x^s)$, c'est à dire si

$$l_1 \geqslant z_1(x^s). \tag{2.27}$$

– **Test 3 :**
Utilisation d'une borne inférieure l_2 sur z_2 par comparaison avec $z_2(x^r)$.
l_2 est obtenue soit par valeur α_1 en utilisant l'équation (2.19) de la section 2.4.6 et x^s soit par valeur α_2 en utilisant l'équation (2.20) de la section 2.4.6 suivant le cas où x^s est optimale pour z^2 ou pas. L'affectation peut être supprimée de L si l_2 montre que toute solution contenant cette affectation se situe au-dessus de la droite horizontale passant par $z(x^r)$, c'est à dire si

$$l_2 \geqslant z_2(x^r). \tag{2.28}$$

La liste réduite d'affectation L est ensuite utilisée pour l'exploration du triangle $\Delta(x^r, x^s)$. Pendant cette recherche, les solutions admissibles générées sont stockées dans une liste de solutions potentiellement efficaces X_{PE}. Chaque nouvelle solution x est comparée aux autres solutions de X_{PE} et X_{PE} est mis à jour si nécessaire. A la fin de cette exploration, c'est à dire si on a la garantie que toutes les solutions x telles que $z(x) \in \Delta(x^r, x^s)$ pas encore énumérées, vérifient

l'inéquation $\lambda_1 z_1(x) + \lambda_2 z_2(x) \geqslant \beta_0$ (borne supérieure de Tuyttens et Teghem, X_{PE} est un ensemble complet de solutions non-supportées du triangle.
Cependant, en imposant chaque affectation de L individuellement, on n'explore que partiellement le triangle. Ulungu et Teghem terminent la description de leur methode en précisant que l'imposition simultanée de plusieurs affectations est nécessaire pour l'exploration complète de chaque triangle.
Cela s'explique par le fait que la fixation d'une variable supplémentaire dégrade la fonction objectif z^λ, mais n'implique pas nécessairement une dégradation simulanée de z_1 et z_2(voir figure ci-dessous).

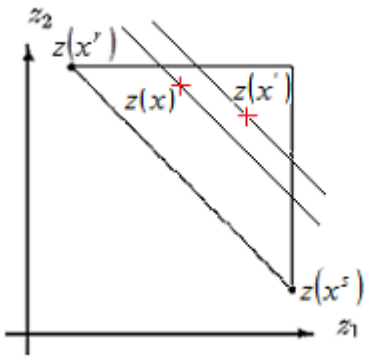

FIG. 2.8 – $z^\lambda(x') > z^\lambda(x) \quad$ mais $\quad z^2(x') < z^2(x)$

La méthode pour fixer plusieurs affectations simulatanément n'est pas décrite avec plus de détails dans l'article d'Ulungu et Teghem [12] ni dans l'article de Tuyttens et Teghem [13]. Ces auteurs précisent juste que comme pour la phase 1, il est nécessaire d'énumérer toutes les solutions optimales de chaque problème considéré dans la phase 2 et que c'est une difficulté majeure, puisque le nombre de solutions énumérées peut se montrer trés large en pratique.

2.5 Méthode en deux phases dévelopée par Xavier Gandibleux, Ehrgott Matthias et Anthony Przybylski [14]

2.5.1 Première phase

La phase1 est initialisée par les solutions optimales détérminées de manière lexicographique x^1 et x^2 qui représentent deux solutions efficaces supportées extrêmes, pour les déterminer la procedure suivante est proposée on commence par calculer d'abord $x^{1'}$ et $x^{2'}$ les deux solutions optimales pour chacun des objectifs.
x^1 peut être trouver en résolvant le problème $min\{Z_2 : x \in X, Z_1(x) \leqslant z_1(x^{1'})\}$ ou en résolvant un problème paramètrique (BAP_λ) avec le vecteur poids défini par $\lambda_1 = z_2(x^{1'})+1$ et $\lambda_2 = 1$ et x^2 peut être trouver en résolvant le problème $min\{Z_1 : x \in X, Z_2(x) \leqslant z_2(x^{2'})\}$ ou en résolvant un problème paramètrique (BAP_λ) avec le vecteur poids défini par $\lambda_1 = 1$ et $\lambda_2 = z_1(x^{2'}) + 1$ Une fois x^1 et x^2 trouvées la recherche des solutions supportées se fait de la même façon que celle introduite dans la version en deux phases décrite auparavant.

2.5.2 Bornes supérieure

- **Première amélioration**
  Soit $\delta_1 = max_{i=0}^{q}\{\lambda_1 z_1(x^i) + \lambda_2 z_2(x^i)\}$ la valeur maximale pour la somme pondérée des deux objectifs pour les solutions potentiellement efficaces et $\delta_2 = max_{i=1}^{q}\{\lambda_1 z_1((x^i)-1) + \lambda_2 z_2((x^{i-1})-1)\}$ la valeur maximale pour les points situés une unité en dessous et à gauche des points nadirs locaux. Alors la borne supérieure améliorée est définie par : $\beta_1 = max\{\delta_1, \delta_2, \lambda_1(z_1(x^0)-1)+\lambda_2(z_2(x^r)-1), \lambda_1(z_1(x^s)-1)+\lambda_2(z_2(x^q)-1)\}$
  Toute solution $x \in X$ telle que $z(x) \in \Delta(x^r, x^s)$ avec $\lambda_1 z_1 + \lambda_2 z_2 > \beta_1$ est dominée. Donc l'énumération de toutes les solutions admissibles dont le point correspondant $(z_1(x), z_2(x))$ est situé entre les droites $(z(x^r)z(x^s))$ et $\{z : \lambda_1 z_1 + \lambda_2 z_2 = \beta_1\}$ permet d'obtenir toutes les solutions non-supportées du triangles.y compris les solutions équivalentes. Par conséquent en utilisant cette borne supérieure, l'exploration de tous les triangles permet l'obtention de l'ensemble X_{E_M}.
- **Deuxième amélioration**
  Dans le cas ou on ne s'intéresse pas à un ensemble complet maximum mais juste à un ensemble complet, la borne supérieure peut être amélioré une nouvelle fois. La

borne supérieure est donc définie par :

$$\beta_2 = max\{\delta_2, \lambda_1(z_1(x^0) - 1) + \lambda_2(z_2(x^r) - 1), \lambda_1(z_1(x^s) - 1) + \lambda_2(z_2(x^q) - 1)\}$$

Toute solution $x \in X$ telle que $z(x) \in \Delta(x^r, x^s)$ avec $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_2$ est dominée ou équivalente à une solution potentiellement efficace.

2.5.3 Deuxième phase

C'est une stratégie différente pour l'exploration des triangles $\Delta(x^r, x^s)$ définis par deux solutions consécutives x^r,x^s dans X_{SE_m}. Dans la phase 2 on cherche des solutions par valeur croissante de z^λ jusqu'à ce qu'une des bornes supérieures $\beta_i, (i = 1, 2)$ soit atteinte. Cela peut être réalisé avec un algorithme de ranking, c'est à dire un algorithme qui détermine les solutions par ordre croissant par rapport à leurs valeurs pour z^λ.
C'est un choix naturel dans la méthode en deux des phases. En effet, ce choix n'implique aucune modification de la structure du problème. Par rapport à une stratégie de fixation de variables, l'utilisation d'un algorithme de ranking a deux avantages naturels : il n'y aucune redondance et l'exploration est ordonnée, grâce à la monotonie de l'énumération par rapport à z^λ.
L'application de cette stratégie d'exploration requiert un algorithme de ranking efficace pour le problème d'affectation.
Déscription de l'algorithme :
On considére le problème (BAP_λ) pour lequel les solutions supportées consécutives x^r et x^s sont optimales et nous appliquons un algorithme pour déterminer les k meilleures solutions. Comme dans les autres variantes de la phase2, pour chaque solution obtenue pendant l'exploration on vérifie s'il est nécéssaire de mettre à jour X_{PE} et par conséquent la borne supérieure.
La procédure s'arrête dès l'obtention d'une première solution x telle que $z^\lambda(x) > \beta_i$ pour $i = 1$ ou 2. Il est aussi possible d'utiliser les bornes inférieures de Ulungu et Teghem [12] afin d'éviter d'énumérer des solutions dont le point correspondant est situé à l'extérieur du triangle $\Delta(x^r, x^s)$. Dans un algorithme de ranking, il n'est pas nécéssaire de fixer les variables à 1, cependant on peut interdire les variables x_{ij} supprimées à l'aide des bornes inférieures en remplaçant le coût correspondant c_{ij}^λ par un nombre moralement grand.
Tandis que les tests 2 et 3 permettent d'éviter d'explorer à l'extérieur du triangle, le test 1 ne permet aucune réduction du nombre de solutions énumérées. Il n'y a plus qu'une condition d'arrêt avec cette variante de phase 2, donnée par la borne supérieure.

2.6 Méthodes de résolution Approchées Heuristiques et Méta-Heuristiques

En l'absence d'approches exactes garantissant un temps d'exécution raisonnable capable de résoudre certain problèmes d'optimisation combinatoire Les chercheurs ont dévellopé des heuristiques et des méta-heuristiques qui représente des techniques de résolution approchées et qui se sont montrés comme une alternative prometteuse. La pratique nous fait observer que les désideurs semblent se satisfaire largement et généralement d'une bonne approximation des solutions efficaces du problème.
Les méta-heuristique représente un chéma de résolution approché générale c'est à dire pouvant étre appliqué sur divers problème et garantissant des solutions approximative tout en respectant des délais de réponse raisonnables.

2.6.1 Le Recuit Simulé

Le recuit simulé a été introduit par Kirkpatrick et al en 1983 [41] et est inspiré du processus de recuit physique. Son processus répéte une procédure itérative qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût.

Voir Algorithme Recuit simulé ci-dessous, T_0 est la température initiale, Seuil est le seuil minimal que la température peut atteindre, α la fonction diminuant la température à certains paliers, it_{palier} le nombre d'itérations à effectuer dans un palier, N la fonction de voisinage, f la fonction d'évaluation, et x_0 la configuration initiale servant de point de départ à l'algorithme. Dans l'algorithme que nous présentons, le nombre d'itérations (it_{palier}) devant être atteint pour effectuer un changement de palier est fixe. Dans la pratique, les algorithmes de recuit simulé font varier ce paramètre en fonction de la température actuelle.
Nous allons maintenant décrire deux implémentations du recuit simulé multi-objectifs.

1. **La méthode PASA Pareto Archived Simulated Annealing**
   (a) **Principe**
   cette méthode,developpée par Engrand et al en 1998 [42], utilise une fonction d'agrégation des fonctions objectif couplée avec un système d'archivage de solutions non dominées.
   (b) **Présentation de la méthode**
   On suppose que les fonctions objectifs du problème d'optimisation multi-objectifs

Algorithme 1 : Algorithme Recuit simulé

Paramètres d'entrée : T_0 ; Seuil ; α ; it_{palier} ;N ; f ; x_0
Paramètres de sortie : x
$x \leftarrow x_0$
$T \leftarrow T_0$
Tant que $T > Seuil$ faire
    nombre-itérations $\leftarrow 0$
    Tant que nombre-itérations $< it_{palier}$ faire
        nombre-itérations $\leftarrow$ nombre-itérations + 1
        Choisir $x^{'} \in N(x)$
        $\Delta f \leftarrow f(x^{'}) - f(x)$
        Si $\Delta f < 0$ Alors
            $x \leftarrow x^{'}$
        Sinon
            Tirer r de manière aléatoire dans l'intervalle $[0; 1]$
            Si $r < e^{-\Delta f/T}$ Alors
                $x \leftarrow x^{'}$
            FinSi
        FinSi
    FinTantQue
$T \leftarrow \alpha(T)$
FinTantQue
Renvoyer x

sont positives. Cette hypothèse permet d'utiliser la fonction d'agrégation suivante, pour se ramener à un problème de minimisation mono-objectif :

$$G(x) = \sum_{i=1}^{n} \ln(f_i)(x), \quad x \in \mathbb{R}^m \tag{2.29}$$

De cette manière, l'expression suivante :

$$\Delta G = G(x^{'}) - G(x) = \sum_{i=1}^{n} (\frac{f_i(x^{'})}{f_i(x)}), \quad x, x^{'} \in \mathbb{R}^m \tag{2.30}$$

représente la variation relative moyenne des fonctions objectif entre le point courant x et le point à tester $x^{'}$:

- Si $\Delta G > 0$, la nouvelle solution $x^{'}$ détériore, en moyenne relative, l'ensemble des fonctions objectif ;
- Si $\Delta G < 0$, la nouvelle solution $x^{'}$ améliore, en moyenne relative, l'ensemble des fonctions objectif.

Ensuite, comme pour le recuit simulé original, on définit une probabilité d'acceptation p :

$$p = \exp(-\frac{\Delta G}{T}) \tag{2.31}$$

Où T désigne la température de l'algorithme de recuit. Cette température a la même signification que dans la méthode du recuit simulé mono-objectif.

Pour l'archivage des solutions non dominées, on utilise une "archive" de taille variable(entre 0 et N_{max} solutions dans l'archive). La gestion de l'archivage se conforme aux regles suivantes :

- Si la solution $x^{'}$ est dominée par au moins une solution de l'archive, alors la solution $x^{'}$.
- Si la solution $x^{'}$ domine une solution y de l'archive, alors la solution $x^{'}$ remplace la solution y dans l'archive.
- Si la solution $x^{'}$ n'est pas dominée par l'archive, alors on place cette solution dans l'archive et on retire de l'archive les solutions qui sont dominées par la solution $x^{'}$.

Pour que la recherche se fasse sur toute la surface de compromis, il est nécessaire de relancer régulièrement la recherche à partir d'un point choisi au hasard, au sein de l'archive.

(c) **Avantage et inconvenient de cette méthode :**

La qualité des solutions initialement présentes dans l'archive influencera la qualité du resultat obtenu à la fin de l'optimisation courante.

Un avantage de cette méthode est la facilité avec la quelle on peut inclure des régles heuristiques pour la gestion de l'archive. Le défaut de cette méthode est que la forme d'agrégation de la fonction objectif utilisée ne supporte que les valeurs positive de fonctions objectif.

2. **La méthode MOSA (Multiple Objective Simulated Annealing)**

(a) **Principe :**

Cette méthode a été proposé par Ulungu et al en 1999 [43]. Elle utilise le recuit simulé pour rechercher la surface de compromis.

(b) **Présentation de la méthode :** On commence par définir une suite de fonctions donnant la probabilité d'acceptation d'une mauvaise solution, pour chaque fonction objectif :

$$\pi_k \begin{cases} \exp(-\frac{\Delta f_k}{T_n}) & si \quad \Delta f_k > 0 \\ 1 & si \quad \Delta f_k \leqslant 0 \end{cases} \tag{2.32}$$

Avec, T_n la température du recuit simulé à l'itération n.

f_k kème fonction objectif.

$x_n \in \mathbb{R}^m$ solution obtenu à l'itération n.

y point voisin de x_n considéré lors de l'itération n.

$\Delta f_k = f_k(y) - f_k(x_n)$ avec $x_n, y \in \mathbb{R}^m$

Ensuite, une fois que toutes ces probabilités ont été calculées, on les agrége. Pour cela il existe plusieurs façons :

– **On effectue le produit des probabilités :**

$$p = t(\Pi, \lambda) = \prod_{k=1}^{N} (\pi_k)^{\lambda_k} \tag{2.33}$$

avec

Π ensemble des π_k, $k = \{1, \ldots, N\}$

λ ensemble des λ_k, $k = \{1, \ldots, N\}$

– **On prend la plus petite probabilité :**

$$p = t(\Pi, \lambda) = \underset{k=\{1,\ldots,N\}}{min} (\pi_k)^{\lambda_k} \tag{2.34}$$

ici λ_k correspond à un coefficient de pondération relatif à une fonction objectif. Ce coefficient permet de prendre en compte un certain ordre entre les diffrents objectifs.

Sélection d'une solution

Cette phase s'effectue avec une condition d'acceptation d'une solution, qui est alors la suivante :

Tout d'abord on a :

$$f(x) = \sum_{i=1}^{N} w_i.f_i(x), \quad x \in \mathbb{R}^m \tag{2.35}$$

et $\Delta f_{eq} = f_{eq}(x_{n+1}) - f_{eq}(x_n)$

– Si $\Delta f_{eq} \leqslant 0$ alors $x_{n+1} = y$

– Si $\Delta f_{eq} > 0$ alors :

$x_{n+1} = y$ avec la probabilité p

$x_{n+1} = x_n$ avec la probabilité $1 - p$

(c) **Discussion**

Ulungu et al 1999 [43], on introduit la méthode MOSA pour la résolution d'un problème d'optimisation combinatoire bi-objectifs.

Elle a été utilisée de la manière suivante :

- Deux couples de coefficients de pondération sont calculés.
- Pour chaque couple,une population de solutions est calculée. Chaque couple va “remplir” une parti de la surface de compromis.
- Les deux populations sont alors réunies, puis les individus non dominants de la population totale sont éliminés.

Cette méthode fonctionne bien car, à haute température, le recuit simulé répartit les indivudus sur toute une surface, et non pas sur un point.

2.6.2 Recherche tabou

La recherche tabou a été introduite par Glover en 1986 [44, 45], c'est une méta-heuristique de recherche réputée pour son habilité à echapper aux minima locaux. Elle a été utilisé avec succès en coopération avec d'autres heuristiques et méthodes pour la résolution des problèmes d'ordonnancement,d'allocation de resources et de télécommunication.

A partir d'une solution initiale quelconque, tabou engendre une succession de solutions.A chaque itération le mécanisme de passage d'une solution x à une autre y est le suivant :

- On construit l'ensemble des voisin de x c'est à dire l'ensemble des solutions accessibles en un seul ”mouvement” élémentaire à partir de x. Soit Voisinage(x) l'ensemble envisagé ;
- On évalue la fonction objectif f du problème pour chacune des solutions appartenant à Voisinage(x). La solution y qui succède à la solution x dans la chaine de markov construite par Tabou, est la solution de Voisinage(x) en laquelle f prend sa valeur minimale.

  Notons que la solution y est adoptée même si $f(y) > f(x)$: C'est grace à cette particularité que Tabou permet d'éviter les minima locaux de f.

Cependant la procedure telle quelle est décrite ne fonctionne généralement pas,car il ya un risque important de retrouver une solution déjà retenue lors d'une itération précédente, ce qui provoque l'apparition d'un cycle. Pour éviter ce phénomène on tient à jour, à chaque itération, une ”liste Tabou” de mouvements interdits.

La procedure peut étre stopée dès que l'on a effectué un nombre donné d'itérations,sans améliorer la meilleure solution atteinte jusqu'ici.

2.7 Méthodes Coopératives

Afin de rendre les algorithmes plus efficaces, de nombreux travaux proposent de combiner les différentes méthodes de résolutions leur but est de combiner les avantages des différentes méthodes d'optimisation. Les méthodes coopératives peuvent être classées en deux classes :

- **Classe des méthodes coopératives exactes :**
  Regroupent les méthodes qui coopérent soit des méthodes exactes entre elles, soit des méthodes excates avec des heuristiques qui sont utilisées afin d'accélérer l'énumération des solutions en offrant de bonne solutions initiales par exemple, malgré l'utilisation d'une heuristique l'approche globale de ce type de coopération reste exacte.
- **Classe des méthodes coopératives approchées :**
  Regroupent les méthodes qui coopérent soit des métaheuristiques entre elles, soit des méthodes exactes et approchées mais cette fois ci l'approche globale est approchée, par exemple inclure une méthode exacte dans une heuristique pour une exploration exacte de l'espace de recherche.

CHAPITRE 3

Notre contribution dans le problème d'affectation bi-objectifs

3.1 Formalisation mathématique du problème d'affectation bi-objectifs

Le problème d'affectation mono-objectif (AP) est un problème de programmation entière qui peut être résolu comme un programme linéaire car sa matrice de contraintes est totalement uni-modulaire. Il appartient à la classe $\mathbb{P}$ vu qu'il existe plusieurs méthodes polynomiales qui le résolvent de manière optimale en un temps qui croit polynomialement tels que l'algorithme de Khun connu sous le nom de la méthode Hongroise [34–36] et les algorithmes du plus court chemin qui sont trés connu [46].
Le programme d'affectation bi-objectifs se formule comme suit :

$$(BAP)\begin{cases} \min \quad Z_k(x) = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{r} x_{ij} \quad k = \{1,2\} \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad j = \{1,\ldots,n\} \\ \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad i = \{1,\ldots,n\} \\ x_{ij} \in \{0,1\} \quad i,j = \{1,\ldots,n\} \end{cases} \tag{3.1}$$

Où c_{ij}^{k} sont des entiers positifs. X l'ensemble des solutions réalisables pour (BAP). On appelle $\mathbb{R}^{n^2}$ espace des decision tel que $X \subset \{0,1\}^{n^2} \subset \mathbb{R}^{n^2}$ et $\mathbb{R}^2$ espace des objectifs (critères) tel que $Z_X = \{z(x) : x \in X\} \subset \mathbb{N}^2 \subset \mathbb{R}^2$.

Le problème paramétrique (BAP_λ) se formule comme suit :

$$(BAP_\lambda)\begin{cases} \min & Z_\lambda = & \lambda_1 Z_1(x) + \lambda_2 Z_2(x) \\ \text{t.q} & & x \in X \\ & & \lambda_1, \lambda_2 > 0 \end{cases} \tag{3.2}$$

Théorème 6 *Le problème d'affectation bi-objectifs est* $\mathbb{NP}$*-complet. La preuve existe dans l'article de Serafini [25]*

3.2 Notre version de la méthode en deux phases

3.2.1 Première phase

L'algorithme de cette phase est le suivant :
On note X_{SE1} l'ensemble des solutions efficaces supportées extrêmes et X_{SE2} l'ensemble des solutions efficaces supportées non extrême.
L'ensemble X_{SE1} est initialisé avec les deux solutions x^1, x^2 optimale pour les deux critère Z_1 et Z_2 trouvées d'une manière lexicographique, on commence par calculer d'abord x^{1^*} et x^{2^*} les deux solutions optimales pour chacun des objectifs. x^1 peut être trouvé en résolvant le problème $min\{Z_2 : x \in X, \quad Z_1(x) \leqslant z_1(x^{1^*})\}$ ou en résolvant un problème paramètrique (BAP_λ) avec le vecteur poids définis par $\lambda_1 = z_2(x^{1^*}) + 1$ et $\lambda_2 = 1$ et x^2 peut être trouver en résolvant le problème $min\{Z_1 : x \in X, \quad Z_2(x) \leqslant z_2(x^{2^*})\}$ ou en résolvant un problème paramètrique (BAP_λ) avec le vecteur poids définis par $\lambda_1 = 1$ et $\lambda_2 = z_1(x^{2^*}) + 1$.
Pour chaque couple de solutions supportées extrêmes on cherche les solutions efficaces supportées qui existent entre elles ces solutions peuvent être extêmes ou non extrêmes.
La recherche des solutions supportées commence, deux solutions consécutives de l'ensemble X_{SE1} qui ne sont pas équivalentes x^r et x^s telles que $z_1(x^r) < z_1(x^s)$ et $z_2(x^r) > z_2(x^s)$, sont considérées (initialement $x^r = x^1$ et $x^s = x^2$).
Le problème paramètrique (BAP_λ) avec $\lambda_1 = z_2(x^r) - z_2(x^s)$ et $\lambda_2 = z_1(x^s) - z_1(x^r)$ est résolu et toutes ses solutions sont énumérées.

L'ensemble R regroupe les solutions de (BAP_λ).

- Si $\{z_\lambda(x), x \in R\} \cap [z(x^r)z(x^s)] = \emptyset$ alors on est dans le cas a), soient x^{t_1} et x^{t_2} les solutions de R avec respectivement les valeurs minimales et maximales pour Z_1. Une nouvelle recherche se fait avec les couple (x^r, x^{t_1}) et (x^{t_2}, x^s).
- Si $\{z_\lambda(x), x \in R\} \subset [z(x^r)z(x^s)]$ alors on est dans le cas b), rien à faire, arrêt de la

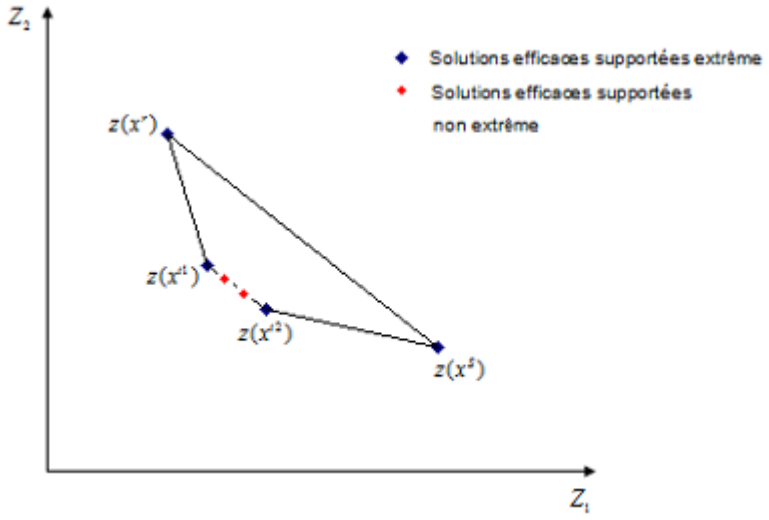

FIG. 3.1 – cas a) de la première phase

recherche.

Soit $R = \{x^t : t \in T\}$ l'ensemble des solutions optimales de (BAP_λ), où T est l'ensemble d'indices tel que $card(T)$ est le nombre de solutions optimales de (BAP_λ). Il y a deux cas possibles :

- cas a) $z(R) \cap [z(x^r)z(x^s)] = \emptyset$ sachant que $[z(x^r)z(x^s)]$ est le segment joignant les points $z(x^r)$ et $z(x^s)$, alors toutes les solutions x^t sont de nouvelles solutions supportées. Ensuite les solutions de $\{x^t : t \in T\}$ avec les valeurs minimales et maximales pour Z_1 sont calculées. Soient x^{t_1} et x^{t_2} les solutions où le minimum et le maximum sont atteints ces deux solutions sont ajoutées à l'ensemble X_{SE1} et les autres solutions de R sont inclues dans X_{SE2}. Pour poursuivre la recherche, deux nouveaux problèmes définis par une somme pondérée sont considérés : l'un défini par les solutions x^r et x^{t_1} et un l'autre défini par les solutions x^{t_2} et x^s (voir figure3.1). Il n'est pas nécessaire de considérer un problème pondéré défini par deux solutions dans R parce que le poids λ obtenu est alors le même que celui défini par x^r et x^s et que par conséquant aucune nouvelle solution ne peut être obtenue.
- cas b) $z(R) \subset [z(x^r)z(x^s)]$, alors toutes les solutions x^t différentes de x^r et x^s sont des solutions efficaces supportées non extrême qui seront placés dans l'ensemble X_{SE2} et aucun nouveau problème pondéré n'est généré (voir figure3.2).

Cette première phase est poursuivie tant qu'il existe des problèmes pondérés à résoudre. A la fin de l'algorithme on aura l'ensemble X_{SE} (ensemble des solutions supportées) qui sera l'union des deux ensembles X_{SE1} (ensemble des solutions supportées extrêmes) et X_{SE2} (ensemble des solutions supportées non extrêmes).

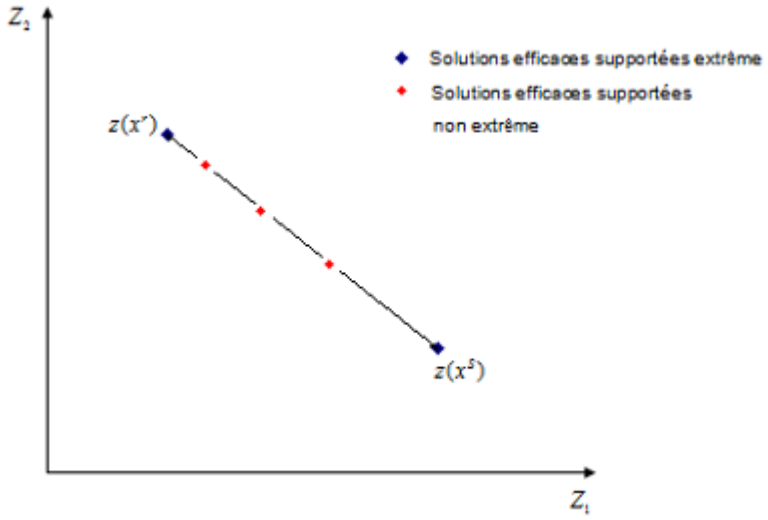

FIG. 3.2 – cas b) de la première phase

3.2.2 Deuxième phase

Cette phase se base sur une méthode lexicographique qu'on a introduit pour explorer les différents triangles. Avant de commencer cette deuxième phase il faut avoir l'ensemble des solutions supportées efficaces extrême générées par la première phase. Une fois l'ensemble X_{SE_1} est généré un tri par ordre croissant de cet ensemble se fait par rapport au premier objectif Z_1.

1. Soit X_{SE_1} l'ensemble des solutions supportées extrêmes générées à la première phase. Soit x^r, x^s deux solution supportées extrêmes successives tirées de l'ensemble X_{SE_1}. On explore le triangle formé par ces deux solutions dans le but de trouver les solutions non supportées existantes dans ce triangle.
2. **Initialisation**
   $obj_1 = Z_1(x^r)$;
   $obj_2 = Z_2(x^r)$;

3. **Etape1 : Résolution du problème par rapport au premier objectif**

$$(P_1)\begin{cases} \min Z_1(x) = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^1 x_{ij} \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad j = \{1,\ldots,n\} \\ \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad i = \{1,\ldots,n\} \\ Z^1 \geqslant obj1 + 1 \\ Z^2 \leqslant obj2 - 1 \\ x_{ij} \in 0,1 \quad i,j = \{1,\ldots,n\} \end{cases}$$

A la fin de cette étape nous aurons une solution optimale x^* du programme (P_1) (voir figure3.3)

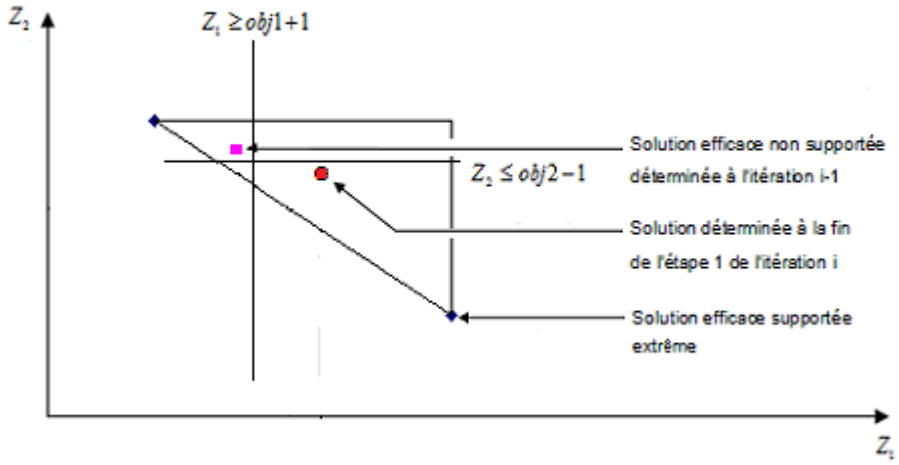

FIG. 3.3 – Etape1 de la recherche lexicographique dans un triangle donné

4. **Etape2 : Résolution du problème par rapport au deuxième objectif**
   $obj_1 = Z_1(x^*)$;
   $obj_2 = Z_2(x^*)$;

$$(P2)\begin{cases} \min Z_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^2 x_{ij} \\ \sum_{i=1}^{n} x_{ij} = 1 \quad j = \{1,\ldots,n\} \\ \sum_{j=1}^{n} x_{ij} = 1 \quad i = \{1,\ldots,n\} \\ Z^1 = obj1 \\ Z^2 \leqslant obj2 \\ x_{ij} \in 0,1 \quad i,j = \{1,\ldots,n\} \end{cases}$$

Cette étape nous permet d'obtenir une nouvelle solution potentiellement efficace x^n (voir figure3.4).

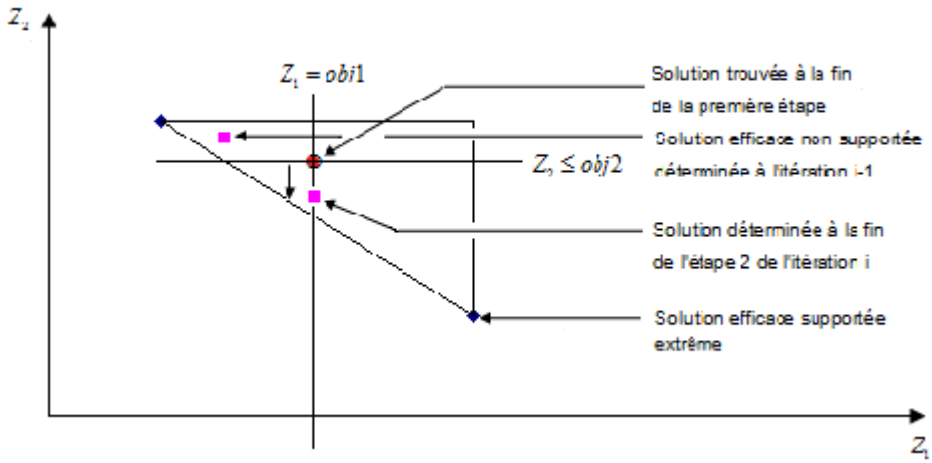

FIG. 3.4 – Etape2 de la recherche lexicographique dans un triangle donné

5. **Etape3 : Processus itératif**
   Mettre à jour les valeurs d'$obj1$ et $obj2$.
   $obj1 = Z_1(x^n)$;
   $obj2 = Z_2(x^n)$;
   Recommencer l'algorithme depuis l'etape1.
6. L'exploration du triangle s'arrête lorsqu'on retrouve la solution efficace x^s ou lorsque l'un des programme soit irréalisable.
7. **Condition d'arrêt de l'algorithme**
   L'algorithme s'arrête lorsqu'on aura exploré tous les triangles formés à chaque fois

de deux solutions supportées extrêmes consécutives ou si le programme (P_1) ou (P_2) est irréalisable.

8. **Finitude de l'algorithme**
   L'algorithme est fini car le nombre de triangle à explorer est fini et égale à $card(X_{SE_1}) - 1$.

9. A la fin de l'algorithme on vérifie s'il n'existe pas des points dominés dans l'ensemble généré en deuxième phase. Une fonction suppdominé a été éléboré dans le but de supprimer une solution potentiellement efficace si le vecteur objectifs relative à cette solution est dominé par un autre. L'ensemble fourni par cette deuxième phase est l'ensemble des solutions efficaces non supportées.

Remarque 4 *L'ensemble des solutions efficaces générées par cet version de la deuxième phase est un ensemble complet minimal c'est à dire que pour chaque point non dominé on retrouve une solution efficace associée à ce point.*

3.3 Application sur un exemple didactique

Soit le problème d'affectation bi-objectifs suivant :

$$C^{(1)} = \begin{pmatrix} 7 & 5 & 2 \\ 8 & 3 & 4 \\ 2 & 9 & 5 \end{pmatrix} \quad C^{(2)} = \begin{pmatrix} 5 & 2 & 6 \\ 3 & 7 & 8 \\ 9 & 4 & 5 \end{pmatrix}$$

– **Résolution par application de la méthode en deux phases originale**

1. **Détermination de X_{SE}**
   Tableaux optimaux trouvés après application de la méthode hongroise :

$$\overline{C}^{(1)} = \begin{pmatrix} 5 & 3 & 0 \\ 5 & 0 & 1 \\ 0 & 7 & 3 \end{pmatrix} \quad \overline{C}^{(2)} = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 4 & 4 \\ 5 & 0 & 0 \end{pmatrix}$$

   $x^1 \equiv \{x_{13} = x_{22} = x_{31} = 1\}$ avec $(z_1, z_2) = (7, 22)$
   $x^2 \equiv \{x_{12} = x_{21} = x_{33} = 1\}$ avec $(z_1, z_2) = (18, 10)$
   Donc les deux solutions de depart sont x^1, x^2.
   Calcul des paramètres : $\lambda_1 = 22 - 10 = 12$ et $\lambda_2 = 18 - 7 = 11$

Donc

$$C^t = \begin{pmatrix} 139 & 82 & 90 \\ 129 & 113 & 136 \\ 123 & 152 & 115 \end{pmatrix}$$

Les deux seules solutions optimales sont x^1 et x^2. situation b) donc pas d'autre solutions supportées.

2. **Détermination de X_{NS} :**
   $L = \{(2,3),(1,1),(3,2)\}$
   L'imposition de $(2,3)$ donne la solution efficace $x^3 \equiv \{x_{12}, x_{23}, x_{31} = 1\}$ avec $(z_1^3, z_2^3) = (11, 19)$
   L'imposition de $(1,1)$ donne la solution efficace $x^4 \equiv \{x_{11}, x_{22}, x_{33} = 1\}$ avec $(z_1^4, z_2^4) = (17, 15)$
   Le test 2 sur l'affectation $(3,2)$ permet de l'éliminer car $7 + 5 > 11$

– **Résolution par application de Notre version de la méthode en deux phases :**

1. Détermination de l'ensemble des solutions efficaces supportées
   Deux solutions optimales x^{1*} et x^{2*} pour les deux critères Z_1 et Z_2 respectivement, sont trouvées par application de la méthode hongroise :
   $x^{1*} \equiv \{x_{13} = x_{22} = x_{31} = 1\}$ avec $(z_1, z_2) = (7, 22)$
   $x^{2*} \equiv \{x_{12} = x_{21} = x_{33} = 1\}$ avec $(z_1, z_2) = (18, 10)$
   En résolvant les problèmes $min\{Z_2 : x \in X, \quad Z_1(x) \leqslant z_1(x^{1^*})\}$ et
   $min\{Z_1 : x \in X, \quad Z_2(x) \leqslant z_2(x^{2^*})\}$ on retrouve les mêmes solutions x^{1*}, x^{2*}.
   Donc les deux solutions de depart sont $x^1 = x^{1*}, x^2 = x^{2*}$.
   Calcul des paramètres : $\lambda_1 = 22 - 10 = 12$ et $\lambda_2 = 18 - 7 = 11$
   Donc

$$C^t = \begin{pmatrix} 139 & 82 & 90 \\ 129 & 113 & 136 \\ 123 & 152 & 115 \end{pmatrix}$$

   Les deux seules solutions optimales sont x^1 et x^2. situation b) donc pas d'autre solutions supportées.

2. Determination des solutions non supportées.
   L'ensemble des solutions efficaces supportées extrêmes est $\{x^1, x^2\}$.
   On explore le triangle $\Delta(x^1, x^2)$

Etape1 : $obj1 = Z_1(x^1) = 7$, $obj2 = Z_2(x^1) = 22$.

$$(P1)\begin{cases} \min Z_1(x) = & \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^1 x_{ij} \\ & \sum_{i=1}^{3} x_{ij} = 1 \quad j = \{1,2,3\} \\ & \sum_{j=1}^{3} x_{ij} = 1 \quad i = \{1,2,3\} \\ & \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^1 x_{ij} \geqslant 8 \\ & \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^2 x_{ij} \leqslant 21 \\ & x_{ij} \in \{0,1\} \quad i,j = \{1,2,3\} \end{cases}$$

Sa résolution donne $x^3 \equiv \{x_{12}, x_{23}, x_{31} = 1\}$ avec $(z_1^3, z_2^3) = (11, 19)$
Etape2 : $obj1 = 11$, $obj2 = 19$.

$$(P2)\begin{cases} min Z_2(x) = & \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^2 x_{ij} \\ & \sum_{i=1}^{3} x_{ij} = 1 \quad j = \{1,2,3\} \\ & \sum_{j=1}^{3} x_{ij} = 1 \quad i = \{1,2,3\} \\ & \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^1 x_{ij} = 11 \\ & \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^2 x_{ij} \leqslant 19 \\ & x_{ij} \in \{0,1\} \quad i,j = \{1,2,3\} \end{cases}$$

Sa résolution donne la première solution potentiellement efficace $x^3 \equiv \{x_{12}, x_{23}, x_{31} = 1\}$ avec $(z_1^3, z_2^3) = (11, 19)$.
Etape1 : $obj1 = 11$, $obj2 = 19$.

$$(P1)\begin{cases} minZ_1(x) = \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^1 x_{ij} \\ \displaystyle\sum_{i=1}^{3} x_{ij} = 1 \quad j = \{1,2,3\} \\ \displaystyle\sum_{j=1}^{3} x_{ij} = 1 \quad i = \{1,2,3\} \\ \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^1 x_{ij} \geqslant 12 \\ \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^2 x_{ij} \leqslant 18 \\ x_{ij} \in \{0,1\} \quad i,j = \{1,2,3\} \end{cases}$$

Sa résolution donne $x^4 \equiv \{x_{11}, x_{22}, x_{33} = 1\}$ avec $(z_1^4, z_2^4) = (17, 15)$.
Etape2 : $obj1 = 17$,$obj2 = 15$.

$$(P2)\begin{cases} minZ_2(x) = \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^2 x_{ij} \\ \displaystyle\sum_{i=1}^{3} x_{ij} = 1 \quad j = \{1,2,3\} \\ \displaystyle\sum_{j=1}^{3} x_{ij} = 1 \quad i = \{1,2,3\} \\ \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^1 x_{ij} = 15 \\ \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^2 x_{ij} \leqslant 17 \\ x_{ij} \in \{0,1\} \quad i,j = \{1,2,3\} \end{cases}$$

Sa résolution donne la deuxième solution potentiellement efficace $x^4 \equiv \{x_{11}, x_{22}, x_{33} = 1\}$ avec $(z_1^4, z_2^4) = (17, 15)$. Arrêt de l'algorithme car l'application de l'étape1 redonne la solution efficace $\widetilde{x}^2$.

3. les vecteurs objectifs relatives aux solutions potentiellement efficaces x^3, x^4 sont respectivement $(11, 19)$, $(17, 15)$, aucun ne domine l'autre ceci dit ces deux vecteurs objectifs sont non dominés et x^3, x^4 sont non supportées efficaces.

Donc l'ensemble des solutions efficaces est $\{x^1,x^2,x^3,x^4\}$.

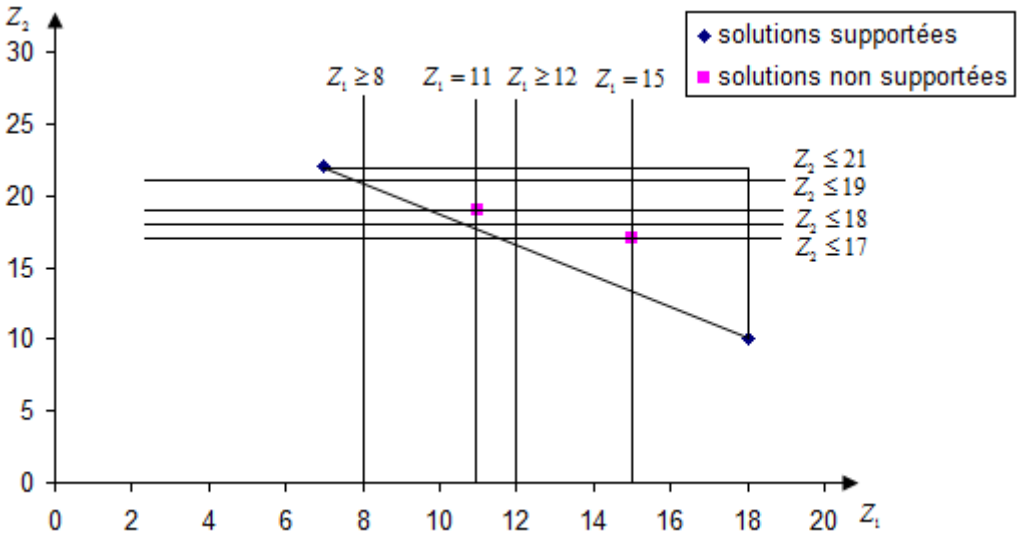

FIG. 3.5 – illustration graphique du déroulement de la méthode pour cet exemple

CHAPITRE 4

Implémentation et Expérimentation des résultats

L'implémentation à été faite en MATLAB7, Le choix du language de programmation n'a pas été au hasard. En effet MATLAB est un language de programmation trés puissant. Le nom Matlab dérive de Matrix laboratory qui veut dire laboratoire des matrices vu que l'implémentation des matrices dans ce language est trés facile. Enregistré marque déposé de MathWorks, il a été à l'origine développé par John Little et Cleve Molaire de MathWorks en 1970 pour des applications impliquant des matrices et de l'analyse numérique. MATLAB est un language de programmation à haut niveau, basé sur un environnement interactif pour le développement d'algorithme, la visualisation de données, l'analyse de données et le calcul numérique qui ont tous des données représentées par des matrices. MATLAB est trés populaire parcqu'il a été testé et examiné tout au long de ces années et a montré et prouvé son efficacité. Il est devenu la première platforme pour les calculs scientifiques dans les instituts d'éducactions et les établissements de recherche.
Le plus important avantage d'un système intéractif est que les programmes peuvent être testés et corrigé rapidement. L'inconvienient qui peux s'avérer gênant est qu'un programme matlab ne peut être éxecuté que dans un ordinateur ayant déjà le matlab instalé.

4.1 Implémentation et description des codes

Le but principale des tests est de valider les algorithmes. Avant d'énoncer les différents tests effectués et leurs résultats, une déscription des différentes Algorithmes est faite.

4.1.1 Génération et description des données

Les jeux de données utilisés se composent notamment des problèmes existants sur le site d'MCDM (Multiple Criteria Decision Making) et autres construits de manière aléatoire. Une fonction notée **random** a été développé pour une génération aléatoire des données, dès qu'on lui définit la dimension n de la matrice carrée souhaitée elle nous génère une matrice d'entiers positifs.

4.1.2 Description du code de la méthode en deux phases écrit en Matlab

Deux appels pour la fonction random se font pour générer les deux matrices de coût c^1 et c^2. La fonction **deuxphase** est la fonction principale en entrée elle a besoin de c^1 et c^2 et en sortie elle donne l'ensemble des solutions efficaces supportées noté XS ainsi que l'ensemble des solutions efficaces non supportées noté XNS.
La fonction **deuxphase** fait appel à deux fonctions **BAPhase1, BAPphase2**.
La fonction BAPhase1 a pour rôle la génération de l'ensemble des solutions efficaces supportées et la fonction BAPphase2 quant a elle génère l'ensemble des solutions efficaces non-supportées.
Avant de faire appel à la fonction BAPphase2 un appel a la fonction TriZ1 se fait dans le but de trier l'ensemble X_{Sext} ensemble des solutions efficaces supportées extrêmes selon un ordre croissant par rapport au premier objectif.

Algorithme 2 : Fonction Deuxphase

Paramètres $\downarrow$: c^1, c^2
Paramètres $\uparrow$: XS, XNS
BAPphase1 ($\downarrow c^1, c^2, \uparrow XS$)
Triz1(X_{Sext})
BAPphase2 ($\downarrow X_{Sext}, \uparrow XNS$)
Suppdominé ($\downarrow XNS, \uparrow XNS$)
➙ La fonction Suppdominé supprime les solutions dominées en cas d'existence.
$XE := XS \cup XNS$
Findedeuxphase.

Algorithme 3 : Algorithme de BAPphase1

Paramètres ↑: XS
lexicosol ($\downarrow c^1, c^2, \uparrow x^1, x^2$)
$x^r := x^1$
$x^s := x^2$
solve ($\downarrow x^r, x^s, \uparrow X_{Sext}, X_{Snonext}$)
$XS = X_{Sext} \cup X_{Snonext}$
FindeBAPphase1

Algorithme 4 : Algorithme de lexicosol

Paramètres ↑: x^1, x^2
Hongroise($\downarrow c^1, \uparrow x^{1^*}$)
$\lambda_1 = z_2(x^{1^*}) + 1$, $\lambda_2 = 1$
$c^\lambda_{ij} = \lambda_1 c^1_{ij} + \lambda_2 c^2_{ij}$
Hongroise($\downarrow c^\lambda, \uparrow x^1$)
Hongroise($\downarrow c^2, \uparrow x^{2^*}$)
$\lambda_1 = 1$, $\lambda_2 = z^2(x^{1^*}) + 1$
Hongroise($\downarrow c^\lambda, \uparrow x^2$)
Finlexicosol

Algorithme 5 : Algorithme de solve

Paramètres ↓: x^r, x^s
Paramètres ↑: $X_{Sext}, X_{Snonext}$
pbparamétrique ($\downarrow x^r, x^s, \uparrow R$)
si $\{x^t, t \in T\} \cap \{x^r, x^s\} = \emptyset$
alors
calculminz1 ($\downarrow R, \uparrow x^{t_1}$)
calculmaxz1 ($\downarrow R, \uparrow x^{t_2}$)
$X_{Sext} = X_{Sext} \cup \{x^{t_1}, x^{t_2}\}$
$X_{Snonext} = \{X_{Snonext} \cup \{x^t, t \in T\}\}/\{x^{t_1}, x^{t_2}\}$
solve ($\downarrow x^r, x^{t_1}, \uparrow X_{ext}, X_{Snonext}$)
solve ($\downarrow x^{t_2}, x^s, \uparrow X_{ext}, X_{Snonext}$)
sinon
si $\{x^t, t \in T\} \neq \{x^r, x^s\}$
alors
$X_{Snonext} = \{X_{Snonext} \cup \{x^t, t \in T\}\}/\{x^r, x^s\}$
sinon
Ne rien faire
finsi
finsi
Finsolve

Algorithme 6 : Algorithme de la fonction pbparamétrique

Paramètres : $\downarrow x^r, x^s$
Paramètres : $\uparrow R$
$\lambda_1 = z_2(x^r) - z_2(x^s)$
$\lambda_2 = z_1(x^s) - z_1(x^r)$
$c_{ij}^{\lambda} = \lambda_1 c_{ij}^1 + \lambda_2 c_{ij}^2$
➙ La fonction Hongroise permet de trouver une solution du (BAP_λ).
Hongroise $(\downarrow c^{\lambda}, x)$
➙ La fonction optsolutions génére les autres solutions optimales pour le problème (BAP_λ) qu'on regroupe dans l'ensemble R.
optsolutions $(\downarrow x, \uparrow R)$
Finpbparamétrique

Algorithme 7 : Algorithme de la fonction BAPphase2

Paramètres $\downarrow$: X_{Sext}
Paramètres $\uparrow$: XNS
①couple $(\downarrow X_{Sext}, \uparrow x^r, x^s)$
Tantqu'il existe x^r, x^s adjacents dans X_{Sext}
condition$(\downarrow x^r, x^s, \uparrow obj1, obj2, stop1, stop2)$
➙La fonction condition définit les conditions d'arrêt d'exploration des triangles ainsi que les deux scalaire obj1 et obj2 utilisés dans l'algorithme de la méthode lexicographique introduite dans la fonction explore.
explore $(\downarrow obj1, obj2, stop1, stop2 \uparrow Sol)$
$XNS = XNS \cup Sol$
retour à ①
fintantque
FinBAPphase2

Algorithme 8 : Algorithme de la fonction Explore

Paramètres ↓: $obj1, obj2, stop1, stop2$
Paramètres ↑: Sol
➙La fonction explore permet d'explorer le triangle formé par le couple donné par la fonction couple dans le but de détecter les solutions efficaces non supportées existantes dans ce triangle.
Tantque ($obj1 \neq stop1$) et ($obj2 \neq stop2$) et (arrêt=0)
➙Etape1 de la méthde lexicographique.
①resolution $(P_1)(\downarrow obj1, obj2, \uparrow x)$
actualiser $(\downarrow x, \uparrow obj1, obj2)$
➙Etape2 de la méthde lexicographique.
②resolution $(P_2)(\downarrow obj1, obj2, \uparrow x^*)$
actualiser $(\downarrow x^*, \uparrow obj1, obj2)$
➙P_1, P_2 sont les problèmes définis dans le chapitre précédant à travers la méthode lexicographique, la fonction resolution utilise pour leur résolution l'algorithme de Branch and Bound ainsi que l'algorithme du Simplexe.
$Sol = Sol \cup x^*$
si P_1 ou P_2 est irréalisable
alors arrêt :=1
sinon retours à ①
finsi
fintantque
Finexplore

Algorithme 9 : Algorithme de la fonction condition

Paramètres ↓: x^r, x^s
Paramètres ↑: $obj1, obj2, stop1, stop2$
$obj1 = z^1(x^r)$
$obj2 = z^2(x^r)$
$stop1 = z^1(x^s)$
$stop2 = z^2(x^s)$
Fincondition

4.2 Tests sur des exemples traités par quelques auteurs

4.2.1 Tests sur les exemples traités par Ulungu [11]

Le tableau suivant présente les exemples traités par Ulungu fondateur de la première version de la méthode en deux phases. Nous avons trouvé les mêmes solutions efficaces.

Exemple BAP	n x n	card(XSE)	card(XNS)	card(XE)	durée(s)
1	3x3	2	3	5	0.116000
2	4x4	3	2	5	0,107000
3	4x4	3	4	7	0,264000

4.2.2 Exemple du résultat fourni par matlab pour l'exemple traité par Malhotra [15]

$$c1 = \begin{pmatrix} 5 & 2 & 6 & 3 \\ 2 & 4 & 3 & 1 \\ 3 & 5 & 4 & 8 \\ 6 & 7 & 4 & 5 \end{pmatrix}$$

$$c1 = \begin{pmatrix} 4 & 2 & 3 & 1 \\ 2 & 5 & 3 & 4 \\ 5 & 3 & 4 & 5 \\ 3 & 4 & 2 & 3 \end{pmatrix}$$

Appel de la fonction Twophase avec c1 et c2
$[XS, XNS] = Twophase(c1, c2)$
la1ème solution efficace supportée est

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$(Z1, Z2) = (10, 13)$

la2ème solution efficace supportée est

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$(Z1, Z2) = (14, 8)$
la1ème solution efficace non supportée est

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$(Z1, Z2) = (13, 11)$
Elapsed time is 0.115000 seconds.
Elapsed time is 0.115000 seconds.
Le nombre de solutions supportées est 2
Le nombre de solutions non supportées est 1
Le nombre de solutions efficaces est 3

4.2.3 Tests sur des exemples tités du site MCDM

Le tableau ci-joint présente les résultats des tests sur quelques instances tirées du site MCDM (Multiple Criteria Decision Making).
Le nom de l'instance provient des caractéristiques suivantes : le nombre d'objectifs, le type de problème, sa dimension et l'intervalle contenant les coefficients des matrices coûts. Par exemple, 2AP05-A20 est un problème d'affectation (AP) bi-objectifs(2), avec 5×5 variables (05) ; les coefficients des matrices coûts sont générés aléatoirement (A) dans l'intervalle $[0, 20]$ (20).

$instances$	$card(XSE)$	$card(XNS)$	$card(XE)$	$dure(sec)$
$2AP05 - A20$	3	5	8	0.279
$2AP10 - A20$	6	10	16	23.003
$2AP15 - A20$	12	27	39	253.645

4.2.4 Tests sur des exemples générés aléatoirement

Le tableau ci-joint représente les résultats des tests sur quelques exemples générés aléatoirement à travers la fonction random définie ci-dessus qui génère à chaque appel une matrice coût d'entiers positifs dans l'intervalle [0,20].
Chaque exemple est un problème (BAP), et est défini par deux matrices coûts générées aléatoirement selon une loi uniforme. Une génération d'un échantillon de 10 exemples a été faite pour chaque type de matrice ensuite une moyenne des solutions efficaces trouvées a été calculée ainsi que celle du temps nécessaire pour la résolution.

Exemple 5X5	card(XSE)	card(XNS)	card(XE)	durée(sec)
ex1	4	3	7	0,715
ex2	3	2	5	0,272
ex3	3	1	4	0,221
ex4	2	1	3	0,667
ex5	3	3	6	0,352
ex6	4	5	9	0,618
ex7	3	5	8	0,556
ex8	3	2	5	0,295
ex9	2	1	3	0,120
ex10	3	2	5	0,302
moyenne	3	3	6	0,411

Exemple 10X10	card(XSE)	card(XNS)	card(XE)	durée(sec)
ex1	5	9	14	20,288
ex2	5	4	9	21,077
ex3	5	4	9	21,771
ex4	5	5	10	21,771
ex5	5	7	12	77,304
ex6	5	11	16	118,029
ex7	5	7	12	77,071
ex8	4	10	14	46,611
ex9	5	3	8	41,384
ex10	3	6	9	10,71
moyenne	5	7	12	57,06

Remarque 5 *Les tests expérimentaux nous font remarquer que les temps d'exécutions dépend de la taille des matrices traitées mais aussi des coûts de ces matrices ce qui veut dire que deux problèmes(BAP) ayant des matrices coûts de même dimension peuvent avoir des temps de résolutions totalement différents.*

Le graphique suivant représente les temps d'exécutions pour les différents exemples traités ayant chaqu'un deux matrices coûts de dimension 10×10.

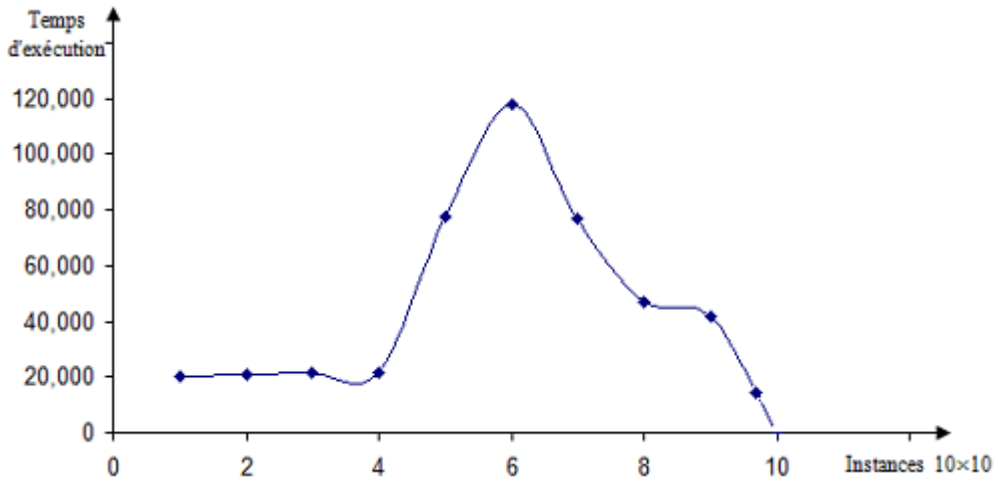

FIG. 4.1 – Temps d'exécutions des différentes instances traitées de dimension 10x10

4.3 Représentations graphiques des solutions efficaces trouvées pour quelques instances traitées

Les graphiques ci-joint représentent les solutions efficaces trouvées par notre méthode pour deux problèmes(BAP) ayant des matrices coûts générées aléatoirement.

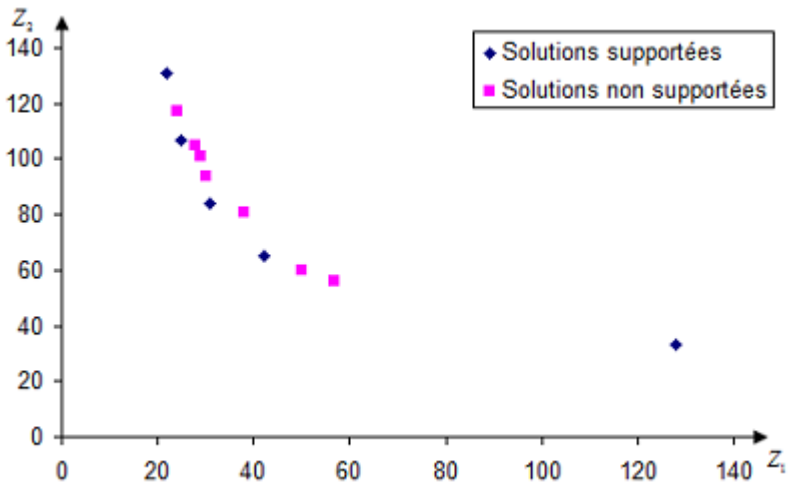

FIG. 4.2 – Solutions efficaces trouvées pour un problème BAP avec deux matrices coûts de dimension 10x10

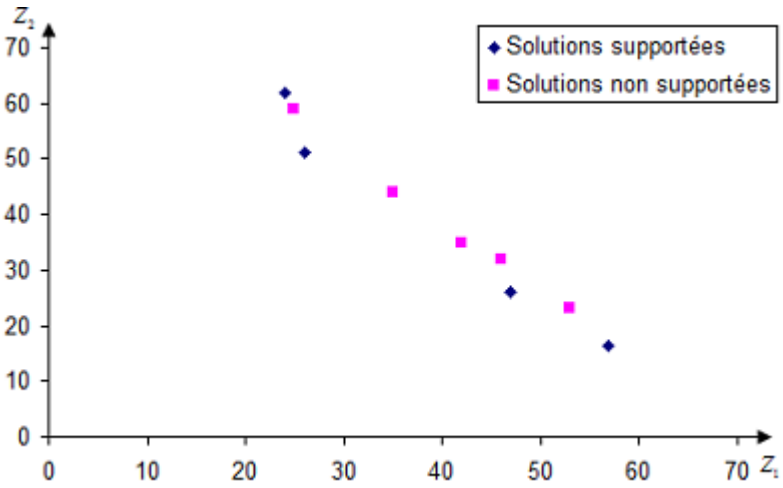

FIG. 4.3 – Solutions efficaces trouvées pour un problème BAP avec deux matrices coûts de dimension 5x5

Les graphiques ci-dessus donnent une visualisation de la position des solutions efficaces supportées et non supportées pour deux problèmes d'affectations bi-objectifs ayant des tailles différentes.
Nous avons remarqué que notre algorithme fonctionne bien et que notre version de la deuxième phase génère un ensemble complet minimal de solutions efficaces non supportées, l'avantage de cet algorithme est qu'il ne nécessite pas une stratégie de fixation de variables [12], ni un algorithme efficace de classement des solution [14], il n'est pas destiné au problème d'affectation bi-objectifs seulement mais peut être utile pour la résolution de d'autres problèmes d'optimisation combinatoire bi-objectifs.

Conclusion générale et perspectives

Pour être réaliste les problèmes d'optimisation combinatoire mono-objectif ont montré leur limite vu que dans la pratique il existe plusieurs objectifs à réaliser simultanément et ces derniers sont souvent conflictuels. La vision multi-objectifs s'impose désormé. Nous nous sommes intéressé cependant au problème d'affectation bi-objectifs (un de ces problèmes) qui est un problème de complexité $\mathbb{NP}$-complet à la différence de son problème mono-objectif qui est polynomial.

Parmi les méthodes existantes pour la résolution de ce problème on retrouve la méthode en deux phases qui a été l'objet de notre travail en la coopérant avec une autre méthode exacte qui est la méthode lexicographique.

Cette méthode a l'avantage d'être exacte bien évidement c'est à dire on n'a pas affaire à des solutions approximatives comme quelques versions de la méthode en deux phases qui existe dans la littérature telle que celle de M.Ehrgot et X.Gandibleux [41], elle a l'avantage aussi de générer un ensemble complet de solutions efficaces sans connaissance a priori du problème, elle ne nécessite ni une stratégie de fixation de variables comme la version introduite par Ulungu et Teghem [12], ni un algorithme efficace de classement comme la version d'A.Przybylski, X.Gandibleux et M.Ehrgott [14].

Une amélioration de cette méthode est envisagée dans le but de déterminer un ensemble complet maximal de solutions efficaces, nous comptons aussi développer un algorithme polynomial pour la deuxième phase pour réduire les temps d'executions.

ANNEXE

Organigrammes de quelques fonctions utilisées

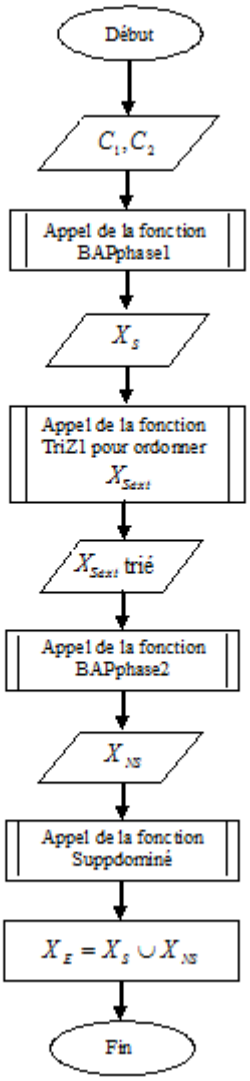

FIG. 4.4 – Organigramme de la méthode en deux phase

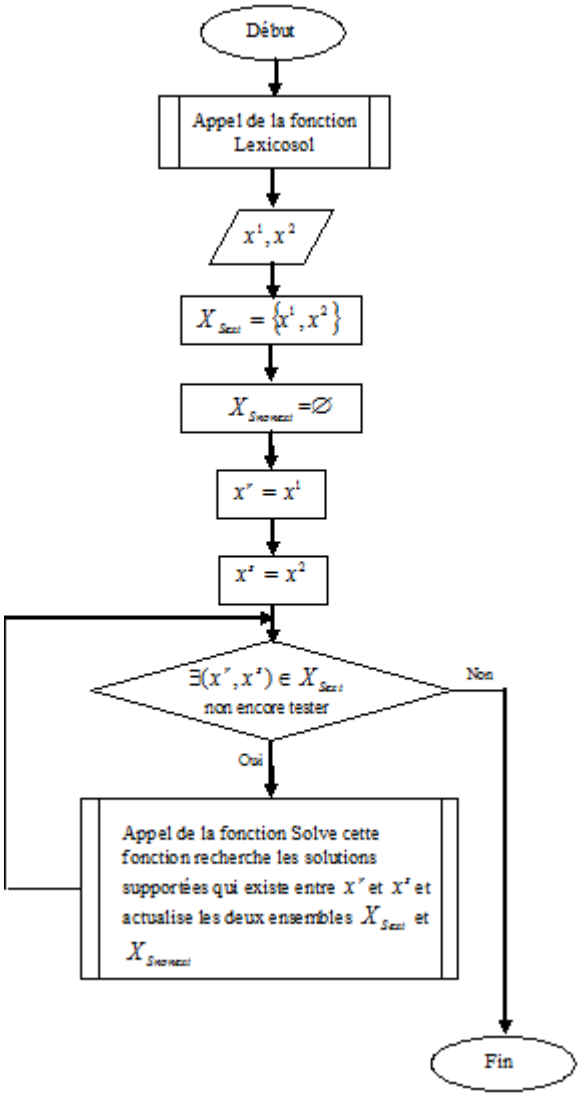

FIG. 4.5 – Organigramme de la première phase

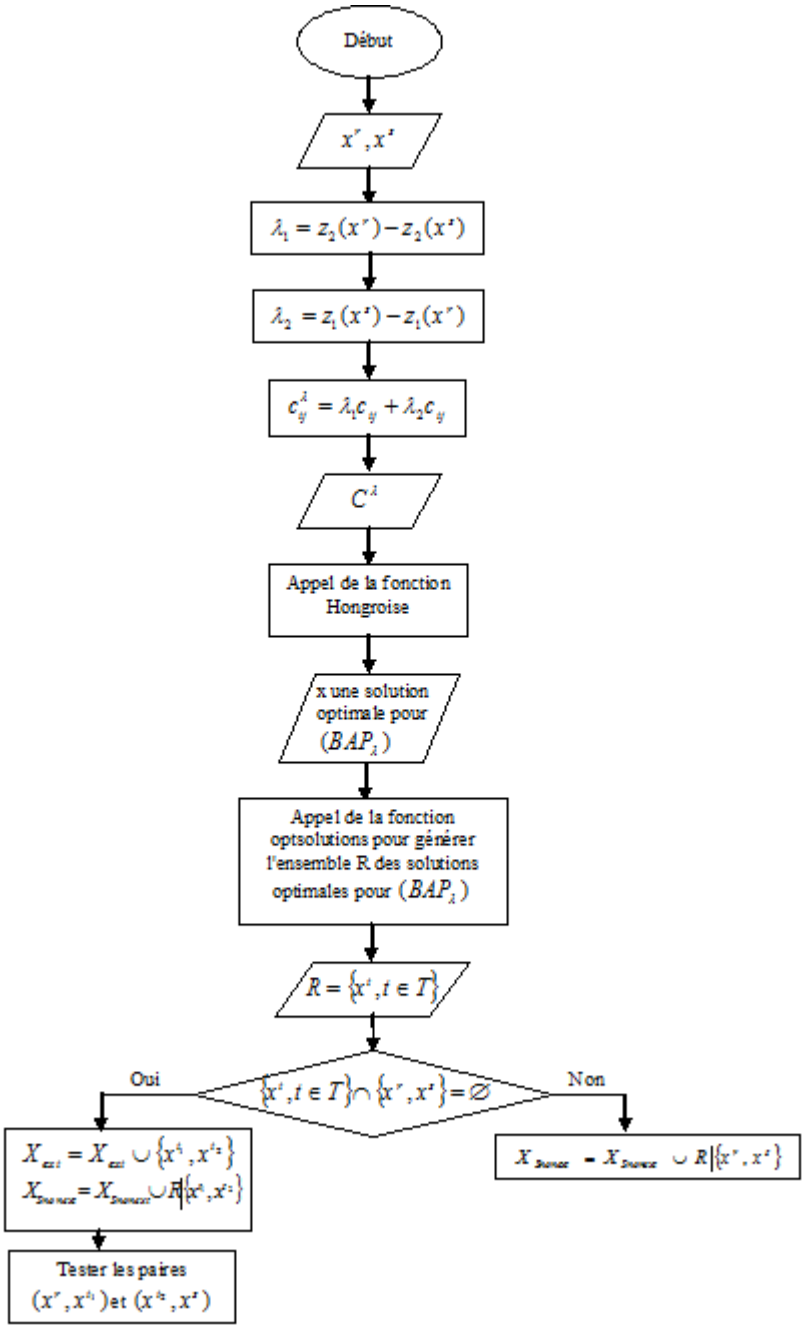

FIG. 4.6 – Organigramme de la fonction solve

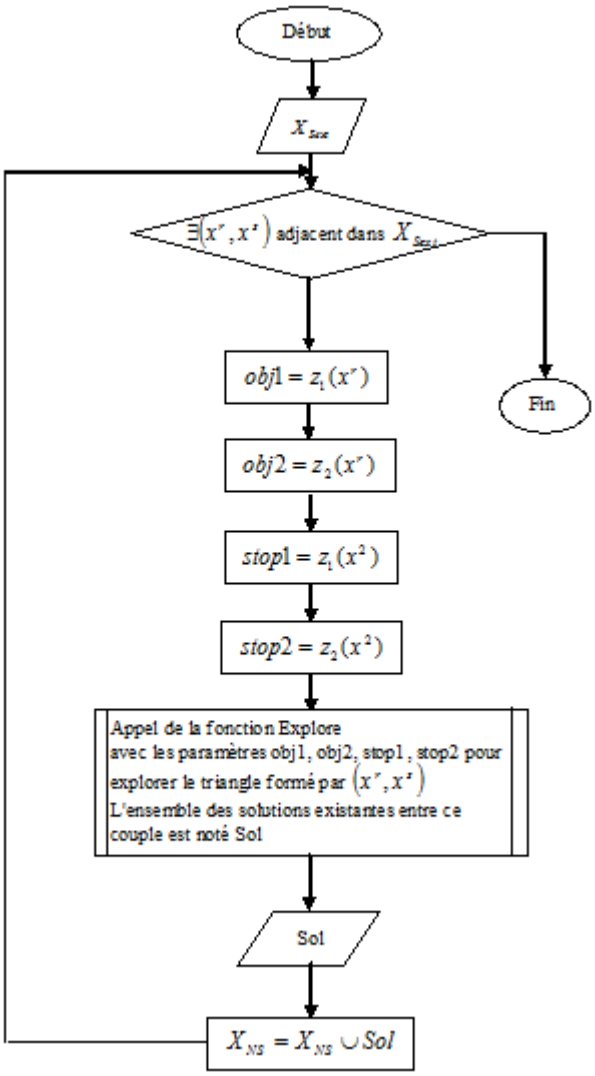

FIG. 4.7 – Organigramme de la fonction BAPphase2

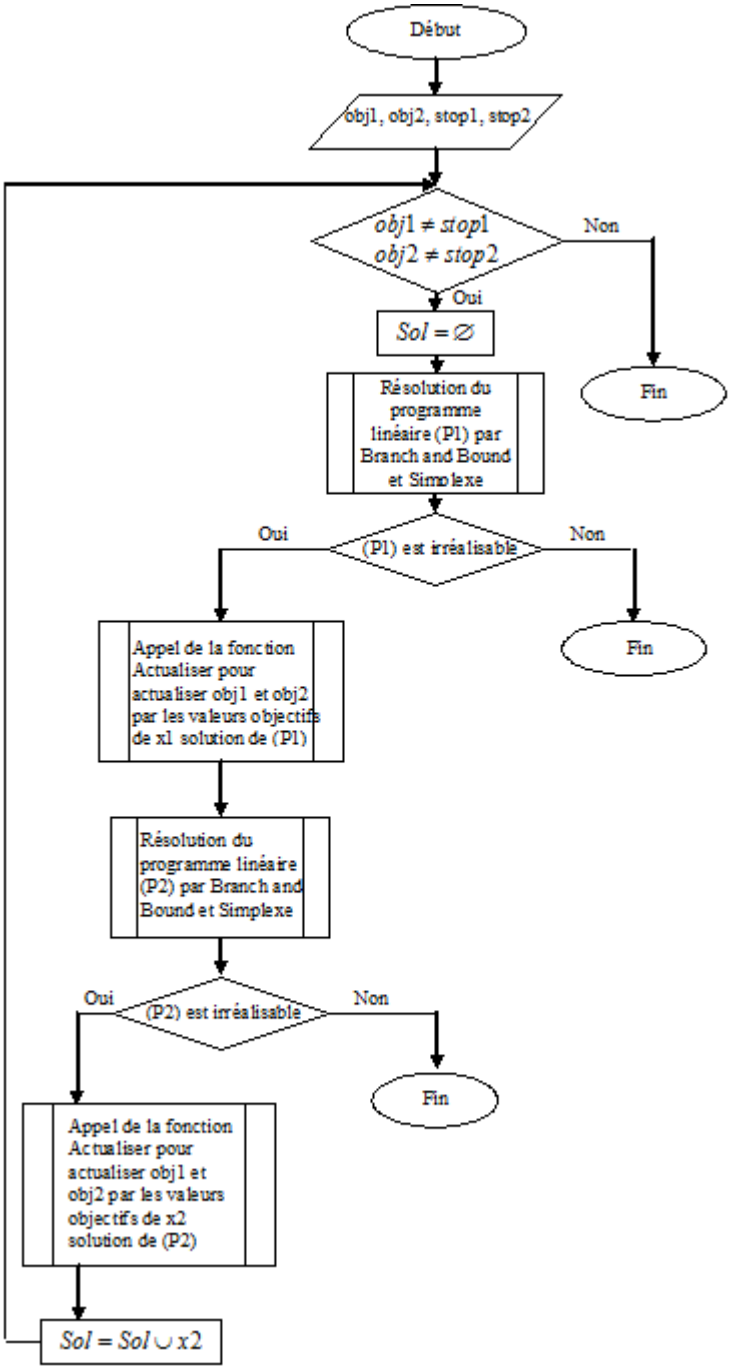

FIG. 4.8 – Organigramme de la fonction Explore

BIBLIOGRAPHIE

[1] Patrick Siarry Yann Collectte. *Optimisation multiobjectif*. 2002.

[2] R.E.Steuer. *Multiple criteria Optimisation*. 1985.

[3] Vincent BARICHARD. *Approches hybrides pour les problèmes multi-objectifs*. PhD thesis, Ecole Doctorale d'Angers, 2003.

[4] I.Othmani. *Optimisation multicritère : Fondements et Concepts*. PhD thesis, Université de Grenoble, 1998.

[5] Clarisse Dhaenens-Flipo. *Optimisation Combinatoire Multi-Objectif : Apport des Méthodes Coopératives et Contribution à l'Extraction de Connaissances*. PhD thesis, Université des Sciences et Technologies de Lille, 2005.

[6] D.CHAABANE. *Contribution à l'optimisation multicritère en variables discrètes*. PhD thesis, Faculté polytechnique de Mons, 2006.

[7] Matthias Ehrgott. Multicriteria optimization. *Lecture Notes in Economics and Mathematical Systems,Springer Verlag*, (491), 2000.

[8] M.Ehrgott. *Multicriteria Optimization*. 2005.

[9] A.M.Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of mathematical Analysis and applications*, 1968.

[10] Bowman. Multiple criteria decision making. *Springer*, 1976.

[11] E.L.Ulungu. *Optimisation combinatoire multi-critère :Détérmination de l'ensemble des solutions efficaces et méthodes intéractives*. PhD thesis, Université de Mons-Hainault, Faculté des sciences, 1993.

[12] E.L.Ulungu and J.Teghem. The two phases method : An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of computing and Decision Sciences*, 20 :149–165, 1995.

[13] J.Teghem D.Tuyttens. Performance of mosa method for the bi-criteria assignment problem. *Journal of Heuristics*, 6 :295–310, 2000.

[14] Matthias Ehrgott Anthony Przybylski, Xavier Gandibleux. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 2007.

[15] R.Malhotra R.Gupta. Multi-criteria integer linear programming problem. *Cahiers du CERO*, 34 :51–68, 1992.

[16] R.E.Gomory. Outline of algorithm for integer solutions to linear programs. *Bulletin of the AMS*, 64 :275–278.

[17] Michel Sakarovitch. *Programmation Discrète*. 1984.

[18] E.Hannan D.Klein. An algorithm for multiple objective integer linear programming problem. *European Journal of Operational Research*, 9 :378–385, 1982.

[19] Villareal B and Karwan M. An interactive dynamic programming approach to multicriteria discrete programming. *JMAA*, 81 :524–544, 1981.

[20] Villareal B and Karwan M. Multicriteria integer programming : A (hybrid) dynamic programming recursive approach. *Mathematical Programming*, 21 :204–223, 1981.

[21] Villareal B. and Karwan M. Parametric multicriteria integer programming. *Annals of Discrete Mathematics 11 ,Studies on Graphs and Discrete Programming, North Holland*, pages 371–379, 1981.

[22] G.Bitran. Linear multiple objective programs with zero-one variables. *Mathematical programming 13*, 13 :121–139, 1977.

[23] G.Bitran. Theory and algorithms for linear multiple objective programs with zero-one variables. *Mathematical Programming*, 17(362) :362–390, 1979.

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

[25] P.Serafini. Some considerations about computational complexity for multi-objective combinatorial problems. *Recent Advances and Historical Development of vector Optimisation*, 294, 1986.

[26] Green G.I. Lee S.M. and Kim C.S. Multiple criteria model for the location-allocation problem. *Computers and Operations Research*, 8 :1–8, 1981.

[27] Pelegrin P. and Fernandez F.R. Determination of efficient points in multiobjective location problems. *Naval Research and Logistics Quaterly,*, 35 :695–705, 1988.

[28] Ross T. and Soland R. A multicriteria approach to the location of public facilities,. *European Journal of Operational Research,*, 4 :307–321, 1980.

[29] Fischer R. and Richter K. Solving a multi-objective traveling salesman problem by dynamic programming,. *Math.Operationsforsch.Statist., Ser.Optimization,*, vol.13(2), 1982.

[30] Gupta A. and Warburton A. Approximation methods for multiple criteria traveling salesman problems,. *Lectures Notes in Economy and Mathematical Systems 285, Springer Verlag.*

[31] Keller C.P. and Goodchild M.F. The multiobjective vending problem : a generalization of the traveling salesman problem,. *Planning and Design*, 15, 1988.

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

[33] V. Barichard and J.K. Hao. Genetic tabu search for the multi-objective knapsack problem. *Tsinghua Science and Technology*, 8 :8–13, 2003.

[34] Harold W.Kuhn. The hungarian method for the assignment problem. *Naval Research logistics quarterly*, 2 :83–97, 1955.

[35] Harold w.Khun. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterty*, 3 :253–258, 1956.

[36] J.Munkres. Algorithms for the assignment and transportation problems. *Journal of the society of industrial and applied mathematics*, 5(1) :32–38, March 1957.

[37] Bhatia H.L. Malhotra R. and Puri M.C. Bicriteria assignment problems,. *OPSEARCH*, 19(2) :84–96, 1982.

[38] Rita Malhotra. Bi criteria assignement problem. *OPSEARCH*, 19(2), 1982.

[39] Lee S.M. and Schniederjans M.J. A multicriteria assignment problem : a goal programming approach. *Intefaces*, 13(4) :75–81, 1983.

[40] E.L. Ulungu and J. Teghem. Multi-objective combinatorial optimization. *A survey, journal of Multi-Criteria Decision Analysis*, 1994.

[41] M.Vecchi S.Kirkpatrick, C.D.Gelatt Jr. Optimization by simulated annealing. *Science 220*, 1983.

[42] X.Mouney P.Engrand. Une méthode originale d'optimisation multi-objectif. *Note interne EDF-DER*, mars 1998.

[43] Ph. Fortemps E.L. Ulungu, J. Teghem and D. Tuyttens. Mosa method : a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 1999.

[44] F.Glover. Future paths for integer programming and links to artificial intelligence. *Comput.and Ops.Res*, 13(5) :533–549, 1986.

[45] F.Glover. Artificial intelligence, heuristic frameworks and tabu search. *Managerial and decision economics*, 11 :365–375, 1990.

[46] T.L.Maganti R.K.Ahuja and J.B.Orlin. Network flows-theory,algorithms, and applications. *Prentice-Hall*, 1993.

[47] Marks D. Revelle C. and Liebman J.C. An analysis of public and private sector location models. *Management Science,*, 16 :692–707, 1970.

[48] Kwak N.K. Schnierdjans, M.J. and Helmer M.C. An application of goal programming to resolve a site location problem. *Interfaces,*, 12(3) :65–72, 1982.

[49] M.Ehrgot et X.Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Spanish journal of operations research*, 12 :1–90, 2004.

[50] P.Neumayer. Complexity of optimization on vector-weighted graphs. *Operations Research*, 93 :359–361, 1994.

[51] A.V.Zykina. A lexicographic optimization algorithm. *Automation and Remote Control, Russian journal*, 65(3) :363–368, 2003.