République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. Mira - Béjaia
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en Mathématiques Appliquées

Spécialité : Modélisation Mathématiques et Techniques de Décisions

Thème

La résolution des problèmes d'optimisation combinatoire bi-objectif, cas du problème d'affectation

Réalisé par :

Mr. BOUMAZA Houssam et **Mr. CHALLAL Amirouche**

Soutenu publiquement, le 1/07/2018 devant le jury composé de :

Mr.	**L. ASLI**	MAA	Université A-Mira de Béjaia.	Président
Mme	**A. ANZI**	MCB	Université A-Mira de Béjaia	Examinateur
Mme	**K. BOUIBED**	MCB	Université A-Mira de Béjaia.	Examinateur
Mme	**Z. AOUDIA**	MAA	Université A-Mira de Béjaia.	Promotrice

Remerciements

Nous remercions Dieu de nous avoir donné le courage et la patience afin de terminer ce travail.

Nous tenons à exprimer toute notre gratitude envers notre, promotrice pour son soutien et la confiance qu'elle nous a accordée en nous proposant ce sujet, et nous a aidés plus qu'elle ne le pense. En nous écoutant patiemment, et en discutant maintes fois de la nature et l'avancement de notre travail, elle nous a permis de synthétiser, comprendre et expliquer un grand nombre de questions. Ces conseils et sa gentillesse nous a apporté un précieux soutien, qu'elle soit chaleureusement remercie ici.

Nos remerciements sont aussi adressés aux membres du jury qui nous ont fait l'honneur d'accepter de juger notre travail. Nous remercions aussi tous les enseignants du département de Mathématiques Appliquée Appliquées qui nous ont permis d'améliorer notre formation.

Merci à nos parents et à nos frères et soeurs pour nous avoir inculqué le goût d'apprendre, de nous avoir enseigné à penser et de nous avoir encouragé sans cesse pour aller plus loin.

Enfin, nous n'oublions pas de remercier ceux qui nous ont aidé d'une manière oú d'une autre d'élaborer ce travail.

Dédicaces

Je dédie ce modeste travail à :

A mes très chers parents qui ont voulu voir en moi le fruit d'un long sacrifice, et je ne les remercierai jamais assez, pour tous les efforts qu'ils ont subis au long de cette durée d'études et je prie Dieu le tout puissant de les protéger et de leur accorder une longue vie.

A mes chères frère : Anis, Chahine et Abdelbasete qui ont été toujours à côté de moi.

Mon cher binôme et meilleur ami : Amirouche.

Toute ma famille et à tous les enseignants du déparetement de mathématiques Appliquées en particulier notre promotrice.

A toutes les personnes qui mon aidé, de près oú de loin, pour la réalisation de ce travail.

Houssam

Dédicaces

Je dédie ce modeste travail à :

Rien n'est aussi beau à offrir que le fruit d'un labeur qu'on dédie du fond du coeur à ceux qu'on aime jusqu'aux frontières de l'imagination.

Ma chère mère, mon cher père, sans eux, je n'aurais pas abouti à ce stade d'étude, que Dieu puisse m'aider à les honorer, les servir et les combler.

Je dédie aussi ce modeste travail à :

A mes chers frères.

A mes chères sœurs.

Mon cher binôme et meilleur ami : Houssam.

Toute ma famille et à tous les enseignants du département de mathématiques Appliquées, en particulier notre promotrice.

A toutes les personnes qui mon aidé, de près oú de loin, pour la réalisation de ce travail.

Amirouche

Table des matières

Liste des figures **vii**

Introduction Générale **1**

1 Optimisation combinatoire **3**

1.1 Définition de problème d'optimisation combinatoire 3
1.2 Résolution d'un problème d'optimisation combinatoire 4
1.3 Exemples de problèmes d'optimisation combinatoire 5
1.3.1 Le problème du voyageur de commerce 6
1.3.2 Problème de transport 6
1.3.3 Problème d'affectation 7
1.3.4 Le problème de sac à dos 9
1.3.5 Le problème du sac à dos quadratique 9
1.4 La résolution des problèmes d'optimisation combinatoire 10
1.4.1 Les approches exactes 10
1.4.2 Les méthodes approchées 13

2 Optimisation Multi-objectifs **14**

2.1 Optimisation mono-objectif 14
2.1.1 Programmation mathématique 14
2.1.2 Programmation linéaire 15
2.1.3 Programmation linéaire en variables entières et optimisation combinatoire 16

2.1.4 Programmation linéaire en variables mixtes 17
2.2 Optimisation multi-objectif 17
2.2.1 Définitions 17
2.3 Qu'est-ce que l'optimisation combinatoire multi-objectif ? 18
2.4 Problèmes d'optimisation combinatoire multi-objectif 19
2.4.1 Propriétés 20
2.5 Structure du front Pareto 22
2.5.1 Front minimal / front maximal complet 23
2.5.2 Solutions supportées / non supportées 23

3 Méthodes de résolution des problèmes d'optimisation combinatoire multi-objectifs 25
3.1 Méthodes exactes pour l'optimisation multi-objectifs 25
3.1.1 La recherche dichotomique 26
3.1.2 Méthode ϵ-contrainte 27
3.1.3 Méthode deux-phases 28
3.2 Les méthodes approchées 30
3.2.1 Métaheuristiques 31
3.3 Les méthodes hybrides 35
3.3.1 Coopération méta/méta 35
3.3.2 Coopération méta/exacte 35

4 Méthodes en deux phases pour les problèmes d'optimisation bi-objectifs 37
4.1 Algorithmes de ranking 37
4.1.1 Algorithme de Cheggiredy et Hamacher 37
4.1.2 Algorithme de Murty 38
4.1.3 Algorithme de Pascoal et al 39
4.2 Méthode en deux phases 40
4.2.1 Phase 1 : Détermination des solutions supportées 40
4.2.2 Phase 2 : Détermination des solutions non-supportées 42

4.3 Application de la méthode des deux phases aux problèmes d'affectation avec 1 contrainte de type sac à dos 61

Conclusion Générale **67**

Bibliographie **68**

Liste des figures

.

2.3.1 Exemple de front Pareto pour un problème à 3 objectifs 19
2.4.1 Illustration des points non dominants dans Y et de l'ordre naturel dans Y_N 22
2.5.1 Représentation des différents types de solutions en bi-objectif 24
2.5.2 Exemple de l'importance des solutions non supportées. 24
3.1.1 Problème (P_λ) défini par y^r et y^s. 27
3.1.2 Une nouveau point supporté est trouvé, la dichotomie continue 27
3.1.3 Illustration de la méthode ϵ-contrainte 28
3.1.4 Illustration des différentes étapes de la méthode deux phases 30
4.2.1 Phase 1, Cas a) 42
4.2.2 Phase 1, Cas b) 42
4.2.3 Exemple d'exécution de la méthode du ranking dans un triangle (y^1, y^2) dans le cadre de la seconde phase 45
4.2.4 La phase 1 appliquer au problème d'aectation bi-objectif 48
4.2.5 Exploration de triangle (y^1, y^4) 51
4.2.6 Exploration de triangle (y^4, y^3) 56
4.2.7 Exploration de triangle (y^3, y^2) 60
4.2.8 Résultats de l'application de la méthode en deux phase sur l'exemple 4.2.1 . 61
4.2.9 La phase 1 appliqué au problème d'aectation avec 1 contrainte de type sac à dos 63
4.2.10 La phase 2 appliquer au problème d'aectation avec 1 contrainte de

type sac à dos 64
4.2.11 L'application de la méthode en deux phases sur l'exemple 4.3.1. 66

Introduction Générale

Beaucoup de problème d'ordre pratique ou théorique nécessitant de prendre le meilleur choix, selon un critère donné parmi un ensemble de choix possibles, souvent très large.

Ces problèmes ont été étudiés depuis longtemps dans des axes de recherche indépendants et ce n'est qu'au milieu du vingtième siècle qu'ils ont été mis dans une seule structure en établissant des relations entre eux. Cette structure est nommée optimisation combinatoire [6].

Cependant, la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels.

Le domaine d'optimisation combinatoire multi-objectif possède ses sources dans les travaux de Edgeworth [18] et de Pareto [29] dans le cadre d'études d'économie au 19ème siècle. Cependant, l'optimisation multi-objectif connaît un intérêt croissant depuis le milieu des années 1980 [35] et le domaine connaît une expansion importante depuis le milieu des années 1990 avec l'apparition de méthodes évolutionnaires pour l'optimisation multi-objectif [13]. Actuellement, l'optimisation multi-objectif est appliquée dans de nombreux domaines académiques et industriels.

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

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

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

L'objectif assigné à ce mémoire, consiste dans un premier temps à la résolution de problème d'affectation bi-objectif par la méthode en deux phases. Par la suite, nous avons appliqué cette méthode sur un problème d'affectation avec une contrainte de type sac à dos (sur les ressources).

La méthode en deux phases est un cadre de résolution général qui a été popularisé par Ulungu en 1993 avec comme idée centrale d'exploiter la structure spécifique des problèmes d'optimisation combinatoire pour leur résolution dans un contexte multi-objectif. Elle a depuis été appliquée sur un grand nombre de problèmes, en se limitant toutefois au contexte bi-objectif [38]. Comme son nom l'indique, cette méthode est décomposée en deux étapes : la première consiste à trouver toutes les solutions supportées du front Pareto, puis la deuxième phase cherche parmi ces solutions les solutions Pareto non supportées. Cette méthode effectue le traitement donc essentiellement dans l'espace objectif [15].

Nous avons réparti ce mémoire en quatre chapitres, le premier chapitre comporte les rappels et concepts de base de l'optimisation combinatoire ainsi que certains problèmes fondamentaux.

Puis, le second chapitre sera consacré aux principales notions et propriétés de l'optimisation multi-objectif.

Dans le troisième chapitre, nous présenterons les défférentes méthodes (exactes, métaheuristique et hybrides) pour la résolution des problèmes d'optimisation combinatoire multi-objectif.

Et enfin, dans le quatrième et dernier chapitre, nous appliquons la méthode en deux phases sur un exemple numérique, et on suite adapter cette méthode pour un problème d'affectation avec 1 contrainte de type sac à dos..

Nous terminons ce travail avec une conclusion et perspectives.

CHAPITRE 1

Optimisation combinatoire

Dans ce chapitre nous définissons les problèmes d'optimisation combinatoire et on donne quelques rappels théoriques.

1.1 Définition de problème d'optimisation combinatoire

L'optimisation combinatoire regroupe une large classe de problèmes ayant des applications dans de nombreux domaines d'applications. Un problème d'optimisation combinatoire est défini par un ensemble fini de solutions discrètes D et une fonction objectif f associant à chaque solution une valeur (la plupart du temps, une valeur réelle). Ainsi, un problème d'optimisation combinatoire consiste en l'optimisation (minimisation ou maximisation) d'un certain critère sous différentes contraintes permettant de délimiter l'ensemble des solutions réalisables (ou solutions admissibles) [15].

Un problème d'optimisation combinatoire peut être défini comme suit : Étant donné un ensemble S de combinaisons, et une fonction $f : S \rightarrow \mathbb{R}$, il s'agit de trouver la combinaison de S minimisant f, i.e., $s^* \in S$, $f(s^*) \leq f(s_i), \forall s_i \in S$

Il est à noter que pour les problèmes de maximisation, il sufit de multiplier la fonction coût par -1.

Selon le contexte, f est appelée la fonction de coût, la fonction économique ou la fonction objectif,...

L'optimisation combinatoire trouve des applications dans des domaines aussi variés que la gestion, l'ingénierie, la conception, la production, les télécommunications, les transports, l'énergie, les sciences sociales et l'informatique ...

1.2 Résolution d'un problème d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points particuliers :

– la définition de l'ensemble des solutions réalisables,

– l'expression de l'objectif à optimiser,

– le choix de la méthode d'optimisation à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution.

Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application.

Le choix de l'objectif à optimiser requiert également une bonne connaissance du problème.

La définition de la fonction objectif mérite toute l'attention de l'analyste car rien ne sert de développer de bonnes méthodes d'optimisation si l'objectif à optimiser n'est pas bien défini.

Comme nous le verrons par la suite, il peut être très difficile de trouver un objectif unique d'optimisation. Nous pouvons donc être amené à proposer une modélisation multi-objectif.

Enfin, le choix de la méthode de résolution à mettre en œuvre dépendra souvent de la complexité du problème. En effet, suivant sa complexité, le problème pourra ou non être résolu de façon optimale. Dans le cas de problèmes classés dans la classe P, un algorithme polynomial a été mis en évidence. Il suffit donc de l'utiliser. Dans le cas de problèmes 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 & 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.

Nous voyons donc ici que la phase de modélisation du problème est très importante puisque c'est elle qui permettra par exemple de reconnaître un problème de la classe P d'un problème NP-difficile. En particulier, la définition de l'objectif est cruciale mais peut être difficile à réaliser, surtout lors de l'étude de problèmes réels [15].

1.3 Exemples de problèmes d'optimisation combinatoire

Le problème du plus court chemin

Ce problème est l'un des problèmes les plus anciens de la théorie des graphes qu'on peut rencontrer soit directement soit comme sous problèmes dans de nombreuses applications[6].

On peut citer enter autres :

– Les problèmes de tournées.

– Certains problèmes d'investissement et de gestion de stocks.

– Les problèmes de programmation dynamique à états discrets et temps discret.

– Les problèmes d'optimisation de réseaux (routiers, télécommunications).

– Certaines méthodes de traitement numérique du signal, de codage et de décodage de l'information.

1.3.1 Le problème du voyageur de commerce

Dans ce problème, il s'agit de trouver le chemin le plus court reliant n villes données, chaque ville ne devant être visitée qu'une et une seule fois, et revenir à la ville de départ. La difficulté du problème vient de l'explosion combinatoire du nombre de chemins à explorer lorsque l'on accroît le nombre de villes à visiter. Plus formellement, ce problème peut être modélisé comme suit : Etant donné un graphe non orienté complet $G = (S, A)$, et une fonction de distance $d : A \rightarrow \mathbb{R}^+$, on cherche à déterminer un cycle hamiltonien de distance totale minimale. Pour ce problème, trouver s'il existe un chemin hamiltonien est un problème NP-complet, la recherche du plus court chemin hamiltonien est un problème NP dificile [3].

Le modèle :

$$\begin{cases} minZ = \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{i,j} x_{i,j} \\ \sum_{j=1}^{n} x_{i,j} = 1 \qquad \forall i = 1, ..., n \\ \sum_{i=1}^{n} x_{i,j} = 1 \qquad \forall j = 1, ..., n \\ \sum_{i \in S, j \notin S} x_{i,j} \geq 1 \qquad \forall S \subset X, S \neq \varnothing \\ x_{i,j} \in \{0, 1\} \quad \forall i = 1, ..., n, \forall j = 1, ..., n \end{cases}$$

Les deux premières contraintes traduisent le fait que chaque ville doit être visitée exactement une fois, la troisième contrainte interdit les solutions composées de sous-tours disjoints, elle est généralement appelée contrainte d'élimination des sous-tours.

1.3.2 Problème de transport

Soit à acheminer une quantité de marchandises à partir de m origines vers n destination. Au niveau de chaque origine i, il y a une disponibilité de a_i articles. La demande de la

destination j est d_j. Le coût unitaire de l'expédition entre l'originr i et la destination j est de déterminer le plan de transport qui minimise le coût total, en tenant compte de l'offre et de la demande [6].

Les variables de décision :

X_{ij} La quantites à expéder de l'origine i, $i = 1, \ldots, m$, vers la destination j, $j = 1, \ldots, n$

Les contraintes :

(a) La disponibilité

$$\sum_{j=1}^{n} x_{ij} \leq a_i \qquad a_i > 0 \qquad i = 1, ..., m$$

(b) La demande

$$\sum_{i=1}^{m} x_{ij} \geq b_j \qquad d_j > 0 \qquad j = 1, ..., n$$

La fonction objectif :

$$minZ = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Le modèle :

$$\begin{cases} minZ = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} \leq a_i \qquad a_i > 0 \qquad i = 1, ..., m \\ \sum_{i=1}^{m} x_{ij} \geq b_j \qquad d_j > 0 \qquad j = 1, ..., n \\ x_{ij} \in \{0, 1\} \qquad i = 1, ..., m \qquad j = 1, ..., n \end{cases}$$

1.3.3 Problème d'affectation

Le problème d'affectation est un cas particulier du problème de transport dans lequel chaque source est affecter à une seul destination. Étant donnés n taches et n ouvriers.

Une affectation consiste à affecter la tâche i à l'ouvrier j de façon :

-chaque ouvrier j ait une et une seule tâche.

-Chaque tâche i est attribuée à un seul ouvrier.

L'affectation d'une tâche i à un ouvrier j coûte C_{ij}. Le problème d'affectation consiste à trouver une affectation de coût minimum [6].

Les variables de décision :

$$x_{ij} = \begin{cases} 1 & \text{si la tâche } i \text{ est affectée à l'ouvrier } j \\ 0 & \text{sinon} \end{cases}$$

Les contraintes :

(a) Le nombre d'ouvriers affectés à la tâche i est 1

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

(b) Le nombre de tâches aux quelles est affecté l'ouvrier j est 1

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

La fonction objectif :

$$minZ = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Le modèle :

$$\begin{cases} minZ = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} c_{ij} x_{ij} \\ \sum\limits_{j=1}^{n} x_{ij} = 1 \quad i = 1, ...n \\ \sum\limits_{i=1}^{n} x_{ij} = 1 \quad j = 1, ...n \\ x_{ij} \in \{0, 1\} \quad i = 1, ..., n \;\; j = 1, ..., n \end{cases}$$

1.3.4 Le problème de sac à dos

On dispose de n objet et d'un sac de capacité b, ou chaque objet j est muni d'une valeur p_j et d'un poids w_j. Le problème consiste à selectionner un sous ensemble d'objets qui maximise la valeur totale correspondante tout en n'exédant pas la capacité b du sac [6].

Les variables de décision :

$$x_j = \begin{cases} 1 & \text{si l'objet } j \text{ est sélectionné} \\ 0 & \text{sinon} \end{cases}$$

Les contraintes : La contrainte de capacité

$$\sum_{j=1}^{n} w_j x_j \leq b$$

La fonction objectif :

$$maxZ = \sum_{j=1}^{n} p_i x_j$$

Le modèle :

$$\begin{cases} maxZ = \sum_{j=1}^{n} p_i x_j \\ \sum_{j=1}^{n} w_j x_j \leq b \\ x_{ij} \in \{0,1\} \quad j = 1, ..., n \end{cases}$$

1.3.5 Le problème du sac à dos quadratique

Le problème du sac à dos quadratique, Quadratic Knapsack Problème (QKP), consiste à sélectionner un sous-ensemble d'objets dont le poids total ne dépasse pas une capacité donnée c du sac à dos et de maximiser le profit total.

Plus formellement, le QKP est défini comme suit : Supposons $N = \{1..n\}$ un ensemble d'objets donné où chaque objet $j \in N$ a un poids positif w_j.

De plus nous disposons d'une matrice d'ordre n d'entiers non négatifs $P = \{p_{ij}\}$, où le p_{ij} est un profit accompli en sélectionnant deux objets difiérents i et $j \in N$. En outre, le profit p_{ij} est accompli si l'objet $i \in N$ est choisi. Des variables binaires x_j seront introduites et indiqueront si l'objet $j \in N$ est sélectionné. Ce problème peut être formulé comme suit :

$$\begin{cases} maxZ = \sum_{i \in N} \sum_{j \in N} p_{ij} x_i x_j \\ \sum_{j \in N} w_j x_j \leq c \\ x_j \in \{0,1\}, j \in N \end{cases}$$

Ce problème peut être considéré comme une variante du MKP, où on a une seule dimension, et où la fonction objectif est quadratique et non plus linéaire [3].

1.4 La résolution des problèmes d'optimisation combinatoire

La majorité des problèmes d'optimisation combinatoire, sont des problèmes NP-difficiles et donc ne possèdent pas à ce jour un algorithme efficace, i.e ; de complexité polynomiale, valable pour toutes les données. Ceci a motivé les chercheurs à développer de nombreuses méthodes de résolution en recherche opérationnelle. Ces méthodes appartiennent à deux grandes familles : les méthodes exactes et les méthodes approchées.

Nous citons dans ce qui suit les principales méthodes exactes et heuristiques connues dans la littérature [3].

1.4.1 Les approches exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des combinaisons de l'espace de recherche. Parmi les méthodes exactes, nous trouvons les méthodes de séparation et évaluation, dites méthodes de Branch and Bound, et la programmation dynamique [3].

Branch and bound

Le Branch and Bound est une technique qui effectue un parcours en profondeur de l'arbre de recherche afin de fournir une ou plusieurs solutions optimales à partir d'un ensemble de solutions potentielles. A chaque étape de la recherche, correspondant à un noeud de l'arbre de recherche, l'algorithme utilise une fonction Bound pour calculer une borne de l'ensemble des solutions du sous-arbre prenant sa racine à ce noeud. En début de résolution, cette borne est initialisée à une valeur maximale (en cas de minimisation). Si cette évaluation est moins bonne que la meilleure solution trouvée jusqu'à ce niveau de recherche, tout le sous-arbre peut être coupé.

Il importe de souligner que l'efficacité de l'algorithme Branch and Bound dépend étroitement du calcul de la borne utilisée [3].

Programmation dynamique

La programmation dynamique est une méthode ascendante : On commence d'habitude par les sous problèmes les plus petits et on remonte vers les sous problèmes de plus en plus difficiles. Elle est utilisée pour les problèmes qui satisfont au principe d'optimalité de Bellman : "Dans une séquence optimale (de décisions ou de choix), chaque sous-séquence doit aussi être optimale". Un exemple de ce type de problème est le plus court chemin entre deux sommets d'un graphe.

L'idée de base est d'éviter de calculer deux fois la même chose, généralement en utilisant une table de résultats déjà calculés, remplie au fur et à mesure qu'on résout les sous problèmes.

Il est à noter que la programmation dynamique est utilisée pour résoudre des problèmes polynomiaux (et non NP-difficiles) [3].

Algorithme hongrois

L'algorithme hongrois ou méthode hongroise, aussi appelé algorithme de Kuhn-Munkres [25], est un algorithme d'optimisation combinatoire, qui résout le problème d'affectation et qui déroule en quatre étapes.

ETAPE 0 : REDUCTION DU TABLEAU INITIAL

On soustrait à chaque ligne du tableau initial, le plus petit élément de la ligne

On fait de même avec les colonnes.

On obtient alors un problème équivalent avec une matrice ayant au moins un zéro par ligne et par colonne.

ETAPE 1 : ENCADRER ET BARRER DES ZEROS

On cherche la ligne comportant le moins de zéros non barrés (en cas d'égalité, choisir arbitrairement la plus haute)

On encadre un des zéros de cette ligne (arbitrairement le plus à gauche)

On barre tous les zéros se trouvant sur la même ligne ou sur la même colonne que le zéro encadré

On recommence l'opération jusqu'à ce qu'on ne puisse plus encadrer, ni barrer de zéros

Si l'on a encadré un zéro par ligne et par colonne, c'est terminé, on a la solution optimale

Sinon, on passe à l'étape 2.

ETAPE 2 : MARQUER ET BARRER DES LIGNES ET DES COLONNES

a) On marque d'une croix toutes les lignes ne contenant aucun zéro encadré

b) On marque toute colonne ayant un zéro barré sur une ligne marquée

c) On marque toute ligne ayant un zéro encadré dans une colonne marquée

On répète alternativement les opérations b) et c) jusqu'à ne plus pouvoir marquer de rangée

On trace alors un trait sur toute ligne non marquée et sur toute colonne marquée

ETAPE 3 : MODIFICATION DU TABLEAU

Les cases non traversées par un trait constituent un tableau partiel

On retranche à toutes les cases de ce tableau partiel le plus petit élément de celui-ci

On ajoute ce même élément à toutes les cases du tableau initial barrées deux fois

On obtient alors un nouveau tableau sur lequel on pourra répéter la succession des étapes 1 à 3

1.4.2 Les méthodes approchées

Nous distinguons parmi les méthodes approchées:

- Les méthode à solution unique, comme le recuit simulé et la recherche tabou.

- Les métaheuristiques, comme les algorithmes génétique et colonies de fourmis.

Conclusion

Il existe une multitude d'autres problèmes d'optimisation combinatoire, que nous ne pouvons citer en un chapitre d'un mémoire. Nous avons choisi les plus étudies et ceux qui se rapprochent du problème d'affectation.

CHAPITRE 2

Optimisation Multi-objectifs

Dans la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels. Dans les problèmes de conception, par exemple, il faut le plus souvent trouver un compromis entre des besoins technologiques et des objectifs de coût. L'optimisation multi-objectif consiste donc à optimiser simultanément plusieurs fonctions. La notion de solution optimale unique dans l'optimisation mono-objectif disparait pour les problèmes d'optimisation multi-objectif au profit de la notion d'ensemble de solutions Pareto optimales.

2.1 Optimisation mono-objectif

2.1.1 Programmation mathématique

La programmation mathématique vise à formaliser et résoudre les problèmes d'optimisation souvent rencontrés dans la vie réelle, tels que l'ordonnancement d'une ligne de production, la sélection des investissements, etc. Les décisions sont modélisées par des variables et des contraintes d'égalités ou d'inégalités. un probème de programmation mathématique peut être formulé comme suit:

$$
\begin{aligned}
&opt\ z(x) \\
&s.c.\ a_i(x)\Delta_i b_i \qquad i = 1, ..., m \\
&\qquad \Delta_i \in \{\leq, =, \geq\} \quad i = 1, ..., m \\
&\qquad x \in D \subseteq \mathbb{R}^n
\end{aligned}
$$

Dans ce manuscrit, la i-ième composante d'un vecteur v sera notée v_i et les vecteurs seront numérotés en utilisant des exposants.

n est le nombre de variables de décision du problème. D_j est le domaine de la variable x_j, c'est-à-dire l'ensemble des valeurs possibles pour x_j, $j \in \{1, ..., n\}$ Le problème est composé de m contraintes; chaque contrainte est définie par une fonction $a_i : \mathbb{R}^n \to \mathbb{R}$ sur les variables, une constante $b_i \in \mathbb{R}$ et un binaire opérateur $\Delta_i \in \{\leq, =, \geq\}$, $i = 1, ..., m$.

Une affectation d'une valeur pour tout $x_j \in \mathbb{R}$, $j = 1, ..., n$, est appelé une solution. Une solution x est réalisable si $x_j \in D_j$ pour tout $j \in \{1, ..., n\}$ et X satisfait à l'ensemble des contraintes. L'ensemble des solutions réalisables d'un problème est l'ensemble X.

$z : \mathbb{R}^n \to \mathbb{R}$ est la fonction objective à optimiser. Le terme «opt» signifie la maximisation ou la minimisation de la fonction objectif. Puisque les cas de maximisation et de minimisation sont symétriques, sans perte de généralité.

Parmi les solutions réalisables, nous nous intéressons particulièrement à celles optimisant la fonction objectif, c'est-à-dire les solutions $x^* \in X$ telles que pour tout $x' \in X$ nous avons $z(x^*) \leq z(x')$. Une telle solution est appelée une solution optimale. En général z n'est pas injectif, il peut donc exister plusieurs solutions optimales pour un même problème. Pour certains problèmes, il n'existe pas de solution optimale, en particulier si le problème est irréalisable ($X = \varnothing$) ou si z n'est pas borné sur X [10].

2.1.2 Programmation linéaire

La programmation linéaire (PL ou LP : Linear Programming) est un cas particulier de la programmation mathématique dans lequel la fonction objectif et les contraintes du problème sont linéaires et les variables sont non négatives. La fonction à minimiser s'écrit donc sous la forme d'une somme $z = \sum_{i=1}^{n} c_i x_i$ dont une forme équivalente est $z = c^T x$ où c^T est la transposée du vecteur de coûts $c = (c_{1,...,}c_n)^T$. Du fait de la linéarité des

contraintes, les coefficients des variables se représentent usuellement par une matrice des contraintes $A \in R^{m \times n}$, ce qui permet d'écrire un système de contraintes $Ax \geq b$, où $b \in R^m$ est le vecteur des membres de droite. Enfin, on suppose que les variables sont positives ou nulles et peuvent être bornées. On obtient donc la formulation suivante :

$$\begin{aligned} min \quad & c^T x \\ s.c \quad & Ax \geq b \\ & x_i \in R^+ \quad i = 1, ..., n \end{aligned}$$

L'ensemble admissible X d'un programme linéaire est un polytope, un polyèdre non borné ou l'ensemble vide et est en particulier convexe. Les programmes linéaires peuvent être résolus au moyen de la méthode du simplexe ou des méthodes de point intérieur [12].

2.1.3 Programmation linéaire en variables entières et optimisation combinatoire

En considérant un programme linéaire pour lequel on ajoute une contrainte d'intégralité qui restreint les valeurs des variables aux seuls nombres entiers, on obtient un problème de programmation linéaire en variables entières (PLNE ou ILP : Integer Linear Programming). Un tel problème se modélise donc comme suit :

$$\begin{aligned} min \quad & c^T x \\ s.c \quad & Ax \geq b \\ & x_i \in N \quad i = 1, ..., n \end{aligned}$$

À cause de ce changement de nature des variables, l'ensemble admissible X n'est plus convexe et les programmes linéaires en variables entières nécessitent donc d'autres méthodes de résolution que celles des programmes linéaires en variables continues. Pour ce type de problèmes, les méthodes de résolution les plus couramment utilisées sont la programmation dynamique, les méthodes polyédrales ainsi que les méthodes de séparation et évaluation (Branch & Bound) [40].

Optimisation combinatoire

Les problèmes d'optimisation combinatoire (CO : Combinatorial Optimization) peuvent être considérés comme un sous-ensemble de la programmation linéaire en variables binaires ($x_i \in \{0, 1\}$). Ces problèmes se distinguent par leurs contraintes donnant au problème une structure particulière [36].

2.1.4 Programmation linéaire en variables mixtes

Comme leur nom le laisse entendre, les problèmes de programmation linéaire en variables mixtes (PLNM ou MILP : Mixed Integer Linear Programming) sont une généralisation des deux classes de problèmes présentées précédemment. Les MILP possèdent en effet à la fois des variables continues et des variables entières, ce qui donne la formulation suivante:

$$\begin{aligned} min \quad & c^T x \\ s.c \quad & Ax \geq b \\ & x_j \in R_+ \qquad\qquad j = 1, ..., n_1 \\ & x_j \in N \qquad j = n_1 + 1, ..., n \end{aligned}$$

De même que pour les programmes linéaires en nombres entiers, la présence de variables discrètes rend l'ensemble réalisable non convexe en général. La difficulté de résolution provient donc des variables discrètes. D'un point de vue méthodologique, peu de choses différencient la résolution des ILP de celles des MILP. On peut toutefois noter que les variables continues peuvent être avantageusement exploitées, par exemple dans le calcul de coupes [20].

2.2 Optimisation multi-objectif

2.2.1 Définitions

Un problème d'optimisation multi-objectif ne considère plus une unique fonction à optimiser mais repose sur p fonctions objectifs, c'est-à-dire un vecteur de fonctions objectif. On a donc le modèle :

$$"min" \quad z(x) = (z_1(x), ..., z_p(x))$$
$$s.c \quad x \in X$$

où $x = (x_1, ..., x_n)$ et pour lequel le sens de «min» reste à définir.

Les pendants des classes de problèmes mono-objectifs déjà présentés sont la programmation linéaire multi-objectif (MOLP : Multiple Objective Linear Programming), la programmation linéaire multi-objectif en variables entières (MOILP : Multiple Objective Integer Linear Programming), l'optimisation combinatoire multi-objectif (MOCO : Multiple Objective Combinatorial Optimization) et la programmation linéaire multi-objectif en variables mixtes (MOMILP : Multiple Objective Mixed Integer Linear Programming) [39].

2.3 Qu'est-ce que l'optimisation combinatoire multi-objectif ?

L'optimisation combinatoire multi-objectif fait partie du domaine de l'optimisation combinatoire. Ainsi un certain nombre de définitions s'inspirent de l'optimisation combinatoire, mais différents concepts, spécifiques au multi-objectif, sont également introduits. En effet, la spécificité principale du multi-objectif étant l'existence de plusieurs fonctions à optimiser, il est en particulier nécessaire de revisiter la notion d'optimalité des solutions [15].

En plus du fait que l'optimisation soit multi-objectif, le nombre de solutions possibles est la plupart du temps gigantesque ce qui ne permet pas de créer des algorithmes qui recherchent les meilleurs solutions. Pour essayer tout de même de traiter au mieux ces problèmes on cherche donc à créer des heuristiques afin de s'approcher au mieux des optimums.

Ainsi, le but ne sera pas de trouver la meilleure solution mais de se rapprocher tant que possible des meilleures valeurs selon tous les critères afin de définir un front Pareto (meilleur front possible sans qu'aucune solution ne soit dominée par une autre) [17].

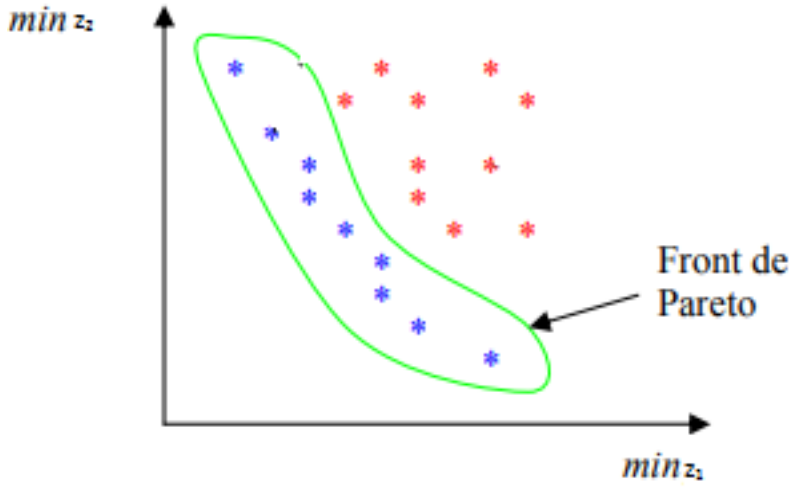


Fig 2.3.1 – Front de Pareto de min (z_1, z_2).

2.4 Problèmes d'optimisation combinatoire multi-objectif

Un problème d'optimisation combinatoire multi-objectif peut s'écrire sous la forme du problème (2.1), où p représente le nombre d'objectifs et X un ensemble combinatoire.

$$\begin{aligned} & minz(x) = (z_1(x), ..., z_p(x)) \\ & x \in X \end{aligned} \qquad (2.1)$$

Afin de ne pas confondre les solutions réalisables et leurs images dans R^p, il est d'usage en optimisation multi-objectif de distinguer l'espace des décisions R^n, qui contient l'ensemble admissible X, et l'espace des objectifs $Y \subseteq R^p$ qui lui contient l'image de l'ensemble admissible X dans R^p. À toute solution réalisable x est associé un vecteur critère $z_1(x) = (z_1(x), ..., z_p(x))$ qui correspond à son image dans Y, telle que Y_k est une fonction objectif pour tout $k \in \{1, ..., p\}$. L'ensemble des points correspondant aux images des solutions admissibles dans Y est noté $Y = \{Z(x : x \in X\}$.

En raison de la nature conflictuelle des objectifs, il n'y a généralement pas une solution réalisable qui minimise simultanément tous les objectifs. L'optimalité dans un contexte multi-objectif est basée sur la notion de dominance et d'efficacité au sens de Pareto dont nous rappelons le formalisme dans les définitions suivantes [39].

Définition 2.4.1 *(Espace des décisions et espace des objectifs). On appelle espace des décisions* $R^n, X \subset R^n$ *et espace des objectifs* R^p, $Y = \{z(x) : x \in X\} \subset R^p$.$Y$, *l'image de* X *par le vecteur de fonctions objectif, est appelé ensemble réalisable. Les vecteurs de* R^p *et* Y *sont respectivement appelés points et points réalisables [39].*

2.4.1 Propriétés

Nous présentons ici les propriétés liées au contexte multi-objectif.

Efficacité et non dominance

Les fonctions objectif sont supposées conflictuelles. De ce fait, il n'existe pas de solution admissible optimisant tous les objectifs simultanément. La notion mono-objectif de solution optimale ne s'applique donc plus. Il est alors nécessaire de définir un contexte de résolution afin de donner un sens à «min» [39].

Définition 2.4.2 *(Dominance). Soient deux points* y^1, $y^2 \in R^p$. *On dit que :*

- y^1 *domine faiblement* y^2 *si* $y_i^1 \leq y_i^2$; $\forall i = 1, ..., p$. *On écrit alors* $y^1 \leqq y^2$.
- y^1 *domine strictement* y^2 *si* $y_i^1 < y_i^2$; $\forall i = 1, ..., p$. *On écrit alors* $y^1 < y^2$.
- y^1 *domine* y^2 *si* $y^1 \leqq y^2$ *et* $y^1 \neq y^2$. *On écrit alors* $y^1 \leq y^2$.

Deux points sont incomparables selon cet ordre élémentaire s'il n'existe pas de relation de dominance entre eux (c'est-à-dire si aucun d'entre eux ne domine faiblement l'autre) [39].

Exemple 2.4.1 *Considérons les trois points (3,4), (6,4) et (5,6). (3,4) domine et donc domine faiblement (6,4). (3,4) domine strictement (5, 6). (6,4) et (5,6) sont incomparables.*

Définition 2.4.3 *Solution efficace (Pareto optimale) [39]. Soit $x^* \in X$.*

– x^* est dite efficace s'il n'existe aucune autre solution admissible $x \in X$ telle que $z(x) \leq z(x^*)$. On dit alors que $y^* = z(x^*)$ est un point non dominé.

– x^* est dite faiblement efficace s'il n'existe aucune autre solution admissible $x \in X$ telle que $z(x) < z(x^*)$. On dit alors que $y^* = z(x^*)$ est un point faiblement non dominé.

Définition 2.4.4 *(Ensembles efficace et non dominé). L'ensemble des solutions efficaces est noté X_E et l'ensemble des points non dominés est Y_N.*

Y_N peut alternativement être défini par $Y_N := \left\{ y \in Y : (y - R^p_{\geqq}) \cap Y = \{y\} \right\}$.

Plus généralement, pour $S \subset R^p$, nous posons $S_N := \left\{ s \in S : (s - R^p_{\geqq}) \cap S = \{s\} \right\}$, l'ensemble des points de S qui ne sont pas dominés par d'autres points de S [39].

Remarque 2.4.1 *Dans le cas bi-objectif, il existe un ordre naturel des solutions efficaces. Il est en effet possible de trier ces solutions par valeur croissante d'un objectif et simultanément par valeur décroissante sur le second objectif. Comme nous le verrons par la suite, cette particularité rend les problèmes bi-objectifs bien plus simples à traiter et en fait une classe à part dans le domaine de l'optimisation multi-objectif [39].*

Définition 2.4.5 *(Équivalence, ensemble complet [24]). Deux solutions admissibles x et $x' \in X$ sont dites équivalentes si $z(x) = z(x')$.*

Un ensemble complet X_E est un ensemble de solutions efficaces tel que toute solution $x \in X \backslash X_E$ est soit équivalente, soit dominée par une solution $x' \in X_E$.

Un ensemble complet minimal X_{E_m} est un ensemble complet sans solution équivalente.

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

Le sens à donner à «min» dépend de l'ensemble calculé :

– l'ensemble non dominé complet.

– l'ensemble des solutions lexicographiquement optimales.

– l'ensemble de compromis, qui fait appel aux préférences du décideur.

Dans le cas des préférences, on parle usuellement de trois contextes :

– le contexte a priori, dans lequel on connaît à l'avance les préférences des décideurs.

– le contexte a posteriori, dans lequel aucune supposition n'est faite sur les préférences du décideur, ce qui impose de calculer la totalité de l'ensemble non dominé.

– le contexte interactif, dans lequel le décideur peut affiner ses préférences au cours de la résolution afin d'aboutir à une solution efficace le satisfaisant.

Exemple 2.4.2 *La figure 2.4.1 présente les points dominés et non-dominants d'un problème d'optimisation combinatoire bi-objectif. Les points l, avec $l \in \{1, ..., 6\}$ sont ordonnés selon l'ordre naturel. y^l et y^{l+1} sont adjacents dans l'ordre naturel de Y_N, pour $l \in \{1, ..., 6\}$ [10].*

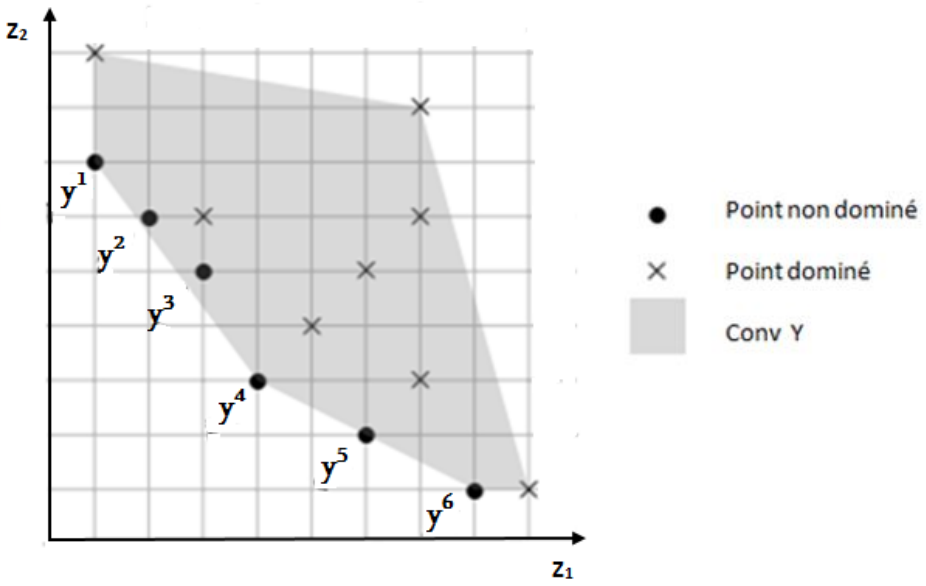


Fig 2.4.1 : Illustration des points non dominants dans Y et de l'ordre naturel dans Y_N.

2.5 Structure du front Pareto

L'objectif est donc de fournir aux décideurs un ensemble (le plus complet possible) de solutions Pareto, afin qu'ils puissent ensuite choisir les solutions qui les intéressent le plus.

Une question se pose donc sur la nature de ces solutions Pareto et la nécessité de les obtenir toutes. Une étude de la frontière Pareto doit donc être réalisée [15].

2.5.1 Front minimal / front maximal complet

La définition de front se réfère à l'espace des objectifs. Une solution appartient au front si elle n'est dominée par aucune autre solution réalisable.

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 Pareto, il est peut être préférable de privilégier une bonne approximation de l'ensemble de la frontière et donc favoriser la diversité (du côté objectif) des solutions retenues.

– Au contraire, lorsque la frontière Pareto 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.

Nous parlerons alors de recherche du front minimal, dans le premier cas, et du front maximal complet, dans le second [15].

2.5.2 Solutions supportées / non supportées

Sur le front Pareto, deux types de solutions peuvent être différenciées : les solutions supportées et les solutions non supportées. Les premières sont celles situées sur l'enveloppe convexe de l'ensemble des solutions (voir Fig. 2.5.1) et peuvent donc être trouvées à l'aide d'une agrégation linéaire des objectifs [19]. Elles sont donc plus simples à obtenir que les solutions non supportées. D'ailleurs, les premiers travaux en optimisation combinatoire multi-objectif se sont pour la plupart focalisés sur la recherche de ces solutions supportées en optimisant des combinaisons linéaires des objectifs utilisant différents vecteurs de poids.

Alors pourquoi ne pas se satisfaire des solutions supportées ? Tout d'abord parce que ces solutions peuvent ne représenter qu'un petit sous-ensemble des solutions efficaces. De plus, ces solutions supportées ne sont pas forcément bien réparties le long du front et ne

représentent pas toujours un bon compromis. La Figure 2.5.2 nous montre l'exemple d'un problème de flowshop bi-objectif où les deux seules solutions supportées sont des solutions extrèmes. Donc, si l'on veut obtenir des solutions de bon compromis entre les objectifs, il est nécessaire de considérer les solutions Pareto non supportées [15].

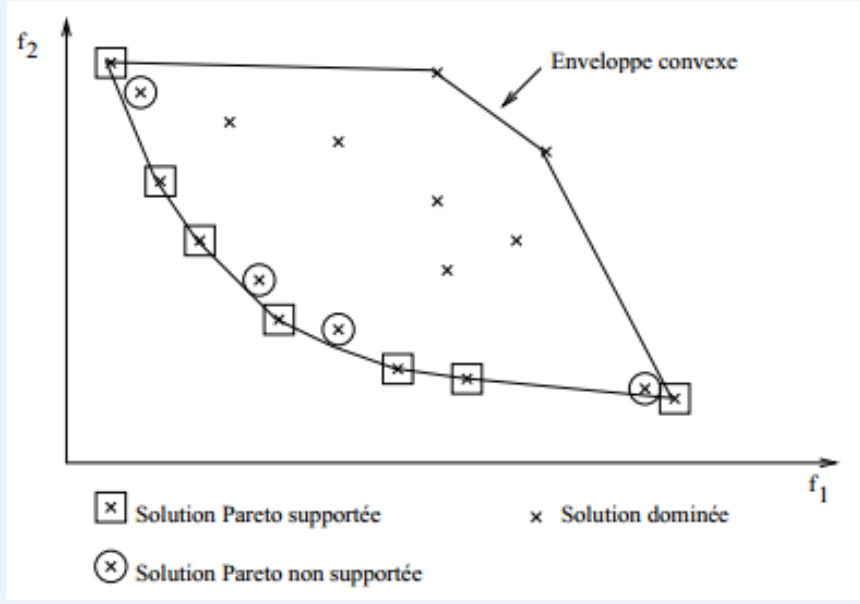


Fig. 2.5.1 –Représentation des différents types de solutions en bi-objectif.

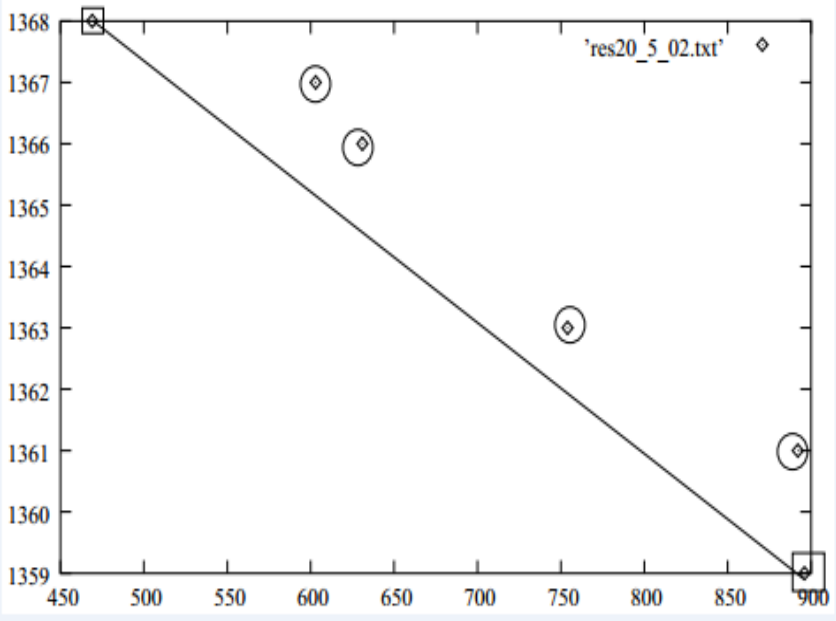


Fig. 2.5.2 –Exemple de l'importance des solutions non supportées.

CHAPITRE 3

Méthodes de résolution des problèmes d'optimisation combinatoire multi-objectifs

Ce chapitre est consacré pour la description des différentes méthodes pour la résolution des problèmes d'optimisation combinatoire.

3.1 Méthodes exactes pour l'optimisation multi-objectifs

Concernant les méthodes exactes, plusieurs approches basées sur des procédures de séparation et évaluation (branch and bound) [38], et la programmation dynamique [9] ont été proposées pour résoudre de petits problèmes à deux objectifs (problèmes bi-objectifs).

Une approche particulière pour l'optimisation multi-objectif est le "goal programming" (programmation par buts) [34]. Dans ce type d'approche, le décideur indique une valeur cible (but) et l'objectif est de minimiser l'écart avec cette cible. Souvent la program-

mation par buts est vue comme une discipline en elle-même, différente de l'optimisation multi-objectif. Une approche intéressante a été proposée par B. Ulungu et J. Teghem pour la recherche du front Pareto de problèmes bi-objectifs [38]. Leur méthode en deux phases consiste dans un premier temps à rechercher l'ensemble des solutions Pareto supportées, puis dans un deuxième temps à rechercher de façon indépendante les solutions non-supportées situées entre tous les couples de solutions supportées adjacentes. Cette approche a été utilisée efficacement sur des problèmes tels que l'affectation ou le sac à dos bi-objectifs. Cette méthode a ensuite été améliorée afin d'obtenir des fronts complets de façon plus efficace [33].

Pourtant, dès que le nombre d'objectifs ou la taille des problèmes augmentent, les méthodes exactes deviennent inefficaces étant donnée la nature NP-difficile des problèmes (déjà en mono-objectif) et l'aspect multi-objectif des problèmes.

Ainsi, il est nécessaire afin de résoudre des problèmes de grande taille et/ou des problèmes avec plus de deux objectifs, de faire appel à des méthodes heuristiques. Les méthodes exactes peuvent néanmoins être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet. Ceci représente une direction suivie par certains travaux présentés ci-après.

3.1.1 La recherche dichotomique

La recherche dichotomique offre un schéma d'application de l'agrégation linéaire permettant d'obtenir les solutions non supportées [14]. Cette méthode consiste à explorer de façon dichotomique des intervalles de front de plus en plus petits.

Tout d'abord les solutions extrèmes sont recherchées. Puis une recherche est menée entre ces solutions r et s suivant une direction perpendiculaire à la droite (r, s). En interdisant de ré-obtenir les solutions r et s et en éliminant les solutions dominées par ces solutions, cette recherche trouve la meilleure solution Pareto relativement à cette direction

de recherche, solution qui peut alors être non supportée. Cette nouvelle solution crée deux nouveaux intervalles qu'il faut explorer de la même façon.

Cette méthode, dédiée au bi-objectif, est intéressante mais nécessite de l'ordre de 2^n recherches, si n est le nombre de solutions du front Pareto [15].

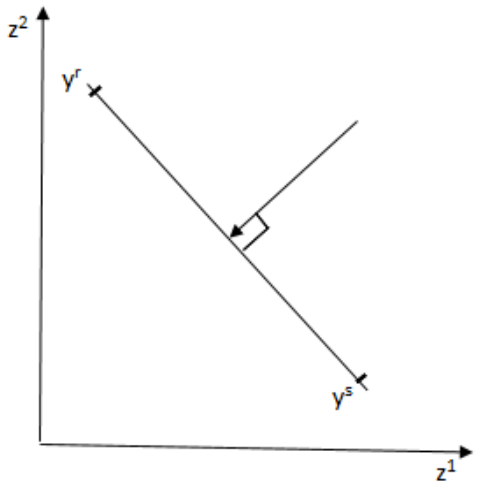


Fig 3.1.1 -Problème (P_λ) défini par y^r et y^s.

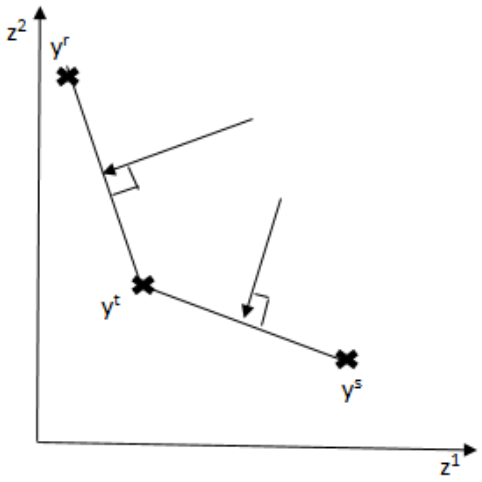


Fig 3.1.2 -Une nouveau point supporté est trouvé, la dichotomie continue.

3.1.2 Méthode ϵ-contrainte

Le principe de la méthode ϵ-contrainte qui consiste, dans le cas bi-objectif, à borner l'un des objectifs (en général le plus difficile à résoudre) et à optimiser l'autre objectif (optimisation mono-objectif) en tenant compte de cette borne [22], 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 de Pareto optimale.

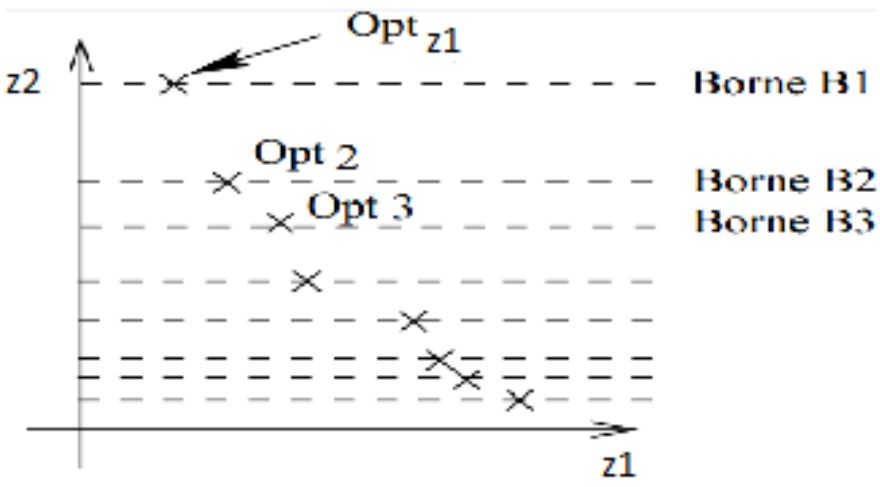


Fig. 3.1.3 –Illustration de la méthode ϵ-contrainte.

La figure 3.1.3 illustre un exemple pour lequel, la solution efficace optimale pour l'objectif z_1 est d'abord recherchée (solution Opt z_1). Cette solution détermine la borne B1 sur l'objectif z_2 en dessous de laquelle l'objectif z_2 va devoir être optimisé. Cela nous donne la solution Opt2 qui elle-même détermine la borne B2, etc...

L'inconvénient principal de cette méthode est qu'elle nécessite une résolution monoobjectif pour chacune des solutions du front. Lorsque ce nombre est élevé, cela peut être vu comme une limite, d'autant plus lorsque la méthode de résolution mono-objectif est coûteuse. De plus, lorsqu'il n'existe pas de méthode mono-objectif efficace, rechercher une solution particulière (respectant une borne, par exemple) est souvent synonyme d'énumération de nombreuses autres solutions dont certaines peuvent être Pareto optimales. Ainsi certaines solutions seront énumérées plusieurs fois sans que la méthode les repère [15].

3.1.3 Méthode deux-phases

La méthode deux-phases a initialement été proposée par Ulungu et Teghem pour la résolution d'un problème d'affectation bi-objectif [38]. Comme son nom l'indique, cette méthode est décomposée en deux étapes : la première consiste à trouver toutes les solutions supportées du front de Pareto, puis la deuxième phase cherche entre ces solutions

les solutions Pareto non supportées. Cette méthode travaille donc essentiellement dans l'espace objectif [15].

Première phase

L'objectif de la première phase est d'obtenir l'ensemble des solutions Pareto supportées. Comme nous l'avons vu précédemment, ces solutions ont l'avantage d'être relativement faciles à trouver puisqu'elles optimisent une certaine combinaison linéaire des objectifs.

Ainsi, durant la première phase de la méthode, les deux solutions extrêmes (solutions optimisant chacune un des deux objectifs) sont recherchées (voir figure 3.1.4.a). Puis, de façon récursive, dès que deux solutions supportées r et s sont trouvées, la méthode recherche d'éventuelles autres solutions supportées entre r et s, à l'aide de combinaisons linéaires bien choisies des objectifs (voi figure 3.1.4.b et 3.1.4.c). A la fin de la première phase l'ensemble des solutions supportées est donc trouvé (voir figure 3.1.4.d).

Cette première phase rappelle la méthode dichotomique, mais ici seules les solutions supportées sont recherchées. Pour cela, lors de l'exploration entre deux solutions, on s'autorise à retrouver l'une de ces deux solutions, lorsqu'il n'existe pas d'autre solutions supportées dans l'intervalle [15].

Deuxième phase

La deuxième phase consiste alors en la recherche des solutions non supportées appartenant au front Pareto. Ces solutions ne peuvent être obtenues par combinaisons d'objectifs. Ulungu et Teghem proposent alors d'utiliser les solutions supportées trouvées pour réduire l'espace de recherche en argumentant que les solutions Pareto non supportées restantes sont forcément dans les triangles rectangles basés sur deux solutions supportées consécutives (voir figure 3.1.4.e). Ainsi, une recherche de type deuxième phase est exécutée entre chaque couple de solutions supportées adjacentes (voir figure 3.1.4.f et 3.1.4.g). La méthode de recherche au sein de ces triangles dépend du problème étudié. A la fin de la deuxième phase, toutes les solutions Pareto sont trouvées. Notons, qu'il aura été

nécessaire au préalable de préciser si l'on recherche le front minimal ou maximal complet [15].

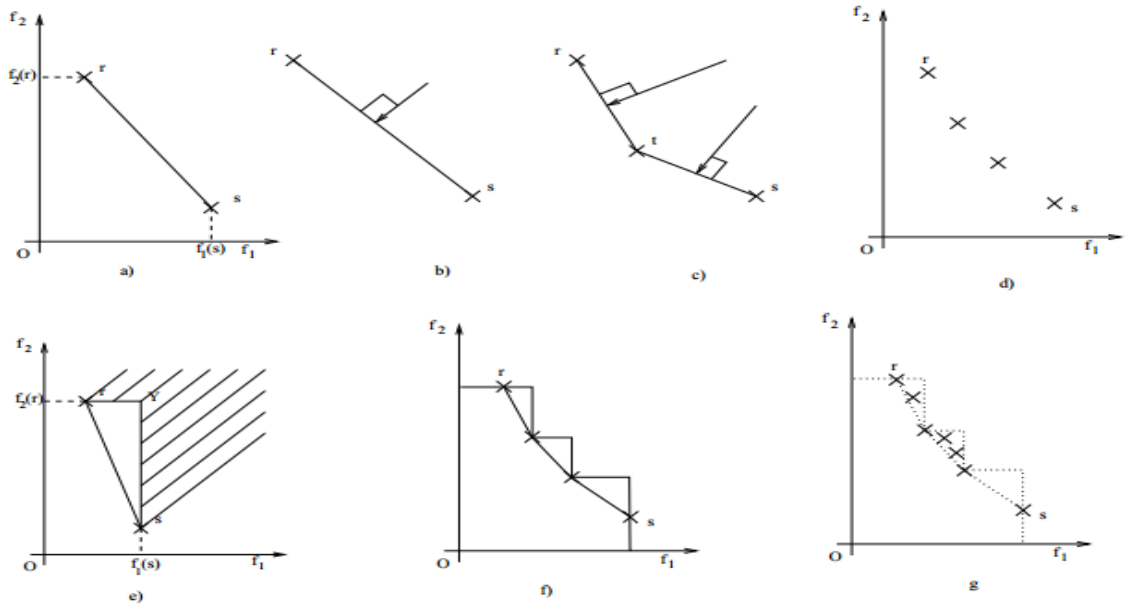


Fig. 3.1.4 –Illustration des différentes étapes de la méthode deux phases.

3.2 Les méthodes approchées

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

Plusieurs classifications des métaheuristiques ont été proposées, la plupart distinguent globalement deux catégories : celles se basant sur une solution unique et celles se basant sur une population de solution [26].

3.2.1 Métaheuristiques

Les métaheuristiques les plus utilisées

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

Leur particularité réside dans le fait qu'elles sont adaptables à un grand nombre de problèmes sans changements majeurs dans leurs algorithmes, d'où le qualificatif "méta". L'un des avantages de celles-ci est leur capacité à optimiser un problème à partir d'une quantité minimale d'information, cependant elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Les métaheuristiques sont des méthodes qui ont un comportement itératif, c'est-à-dire que le même schéma est reproduit un certain nombre de fois au cours de l'optimisation, et elles sont directes, dans le sens où elles ne font pas appel au calcul du gradient de la fonction. L'utilisateur est certes, demandeur de méthodes rapides et efficaces, mais il est aussi demandeur de méthodes simples d'utilisation. Un enjeu majeur des métaheuristiques est donc de faciliter le choix des méthodes et de simplifier leurs réglages, afin de les adapter au mieux aux problèmes posés. Les métaheuristiques sont en évolution permanente. De nombreuses méthodes sont proposées chaque année pour améliorer la résolution des problèmes les plus complexes. Du fait de cette activité permanente, un grand nombre de classes de métaheuristiques existe actuellement. Les méthodes les plus courantes sont le recuit simulé, la recherche tabou, les algorithmes de colonies de fourmis. La section suivante est dédiée à la présentation de ces méthodes [21].

Méthode de colonies de fourmis Le principe de la méthode provient analogiquement avec les comportements collectifs des insectes, les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis

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

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

L'algorithme de colonies de fourmis a été principalement utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. Récemment, son emploi s'est généralisé à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification, ou encore le traitement d'image [21].

Algorithme 3.2.1 *Algorithme de colonie de fourmis [21]*

1. Initialisation

Initialiser les pistes de phéromone

2. Construction de la solution

Pour chaque fourmi répéter

Construction de la solution en utilisant les pistes de phéromone

3. Mise à jour des pistes de phéromone

Jusqu'à atteindre la condition d'arrêt

Méthode de recuit simulé L'origine de la méthode du recuit simulé provient de la métallurgie, où, pour atteindre les états de basse énergie d'un solide on chauffe celui-ci jusqu'à des températures très élevées, après on le laisse refroidir lentement [21].

Lorsque le solide est à une température, chaque particule possède une très grande énergie et peut effectuer de grands déplacements aléatoires dans la matière. Au fur à mesure que la température est abaissée, chaque particule perd de l'énergie et sa capacité de déplacement se réduit. Les différents états transitoires de refroidissement permettent d'obtenir des matériaux très homogènes et de bonne qualité. Ce processus est appelé le recuit.

Cette méthode repose sur l'algorithme de Metropolis en 1953 [27]. Cet algorithme nous permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée, et de conserver les états les plus probables pour de très basses températures.

Algorithme 3.2.2 - *Recuit Simulé [8, 30]*

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

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

Sinon, elle sera acceptée avec une probabilité égale à $e^{\Delta E/T}$

5. Répéter 3. et 4. jusqu'à ce que la configuration optimale soit atteinte.
6. Décroitre la température et répéter jusqu'à ce que le système se solidifie.

Recherche Tabou La recherche tabou est une méthode de recherche locale introduite dans les années 80 par Fred Glover. Elle est basée sur des mécanismes inspirés de la mémoire humaine. Elle se distingue des méthodes de recherche locale simples par l'utilisation

d'une mémoire appelée liste tabou de taille k, qui enregistre les k dernières solutions visitées vers lesquelles il est interdit de se déplacer (d'où le nom attribué à la méthode par Glover). Cette liste est utilisée lors de déplacements dans le voisinage dans le but de favoriser une large exploration de l'espace des solutions et d'éviter d'y retourner trop rapidement à des solutions déjà visitées.

En effet, à partir d'une configuration courante s, on choisit une solution $s' \in N(s)$ en dehors des éléments de cette liste tabou T, même si elle dégrade la fonction objectif f, puis on ajoute s' à T. Quand le nombre k est atteint, chaque nouvelle solution sélectionnée remplace la plus ancienne dans la liste [2, 7, 16].

Algorithme 3.2.3 - *Recherche Tabou [16]*

1. Générer une solution s,
2. Choisir une nouvelle solution s' qui minimise $f(s')$ dans le voisinage de s et qui ne figure pas dans la liste tabou T,
3. Si $f(s') \leq f(s)$ alors :
   $s \leftarrow s'$,
4. Poser $s = s'$ et mettre à jour T,
5. Répéter 2. 3. et 4. jusqu'à ce que le critère d'arrêt soit atteint.

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

Algorithmes Génétiques Les algorithmes génétiques (AGs) sont une méthode d'optimisation numérique inspirée du processus naturel de l'évolution génétique (Darwin, 1859), Les AGs ont été largement utilisés dans la communauté multi-objectif. Ils sont trés appropriés pour résoudre des PMOs grâce à l'utilisation d'une population de solutions [3].

Nondominated Sorting Genetic Algorithm (NSGA) Proposée par (Srinivas et Deb 1994 [4]), dans cette méthode, avant que la sélection soit entamée, les solutions sont classifiées à base de non-dominance. Tous les individus non-dominés sont classés dans

une catégorie avec une valeur de fitness factice proportionnelle à la taille de la population afin de fournir une possibilité de reproduction égale pour tous les individus.

Nondominated Sorting Genetic Algorithm (NSGA II) Proposée par Deb et al. en 2002 [31], dans cette méthode, la population est classifiée selon la dominance de Pareto, elle attribue à chaque individu un rang de non-dominance et une distance de crowding qui permet d'évaluer de la densité de la région autour de cet individu. Cette méthode est bien moins complexe que sa devancière.

3.3 Les méthodes hybrides

L'idée de faire coopérer différents types de méthodes n'est pas nouvelle. Très vite, il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés et on a cherché à profiter des avantages des différentes méthodes [26].

3.3.1 Coopération méta/méta

Les coopérations étaient à l'origine essentiellement réalisées entre différentes métaheuristiques. A peu prés tous les types d'approches ont été proposés pour ce type de coopération, fait que les métaheuristiques hybrides sont devenues maintenant assez classiques dans le domaine de l'optimisation [5].

3.3.2 Coopération méta/exacte

Nous avons vu que les méthodes exactes permettaient de résoudre des petits problèmes tandis que les (méta)heuristiques sont capables d'appréhender de grands problèmes sans pouvoir donner la solution optimale ou prouver que la solution fournie est optimale. Cependant les méthodes exactes peuvent néanmoins être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique

du problème complet [15]. Cette constatation constitue le point de départ d'une approche hybride, assez originale dans le contexte des problèmes d'optimisation combinatoire visant à combiner résolutions exactes et métaheuristiques permettant de conserver aux mieux les avantages de chacune des approches. Nous nous intéressons à la coopération méta/exacte, afin d'obtenir toujours de meilleurs résultats [26].

CHAPITRE 4

Méthodes en deux phases pour les problèmes d'optimisation bi-objectifs

Dans ce chapitre, nous nous intéressons à la résolution du problème d'affectation bi-objectif par la méthode en deux phases proposée par Ulungu et Teghem [38].

4.1 Algorithmes de ranking

Un algorithme de ranking c'est un algorithme qui détermine les k-meilleures solutions d'un problème mono-objectif, ce qui signifie trouver k solutions $x^1, ..., x^k$ telles que $z(x^1) \leq z(x^2) \leq ... \leq z(x^k) \leq z(x)$ pour tout $x \in X$ avec $x \neq x^1, ..., x^k$ [32].

4.1.1 Algorithme de Cheggiredy et Hamacher

Cheggiredy et Hamacher [11] ont proposé plusieurs variantes d'algorithmes pour déterminer les k meilleurs couplage parfaits dans un graphe pondéré. En particulier, une des variantes considère le cas d'un graphe biparti et peut donc être utilisée pour énumérer

les k meilleures solutions d'un problème d'affectation. Cet algorithme est une application du principe d'exploration à l'aide d'un arbre binaire de recherche proposé par Hamacher et Queyranne [23]. Pour appliquer ce principe, il est nécessaire de pouvoir déterminer la deuxième meilleure solution d'un problème mono-objectif.

Dans un premier temps, la meilleure solution x^1 et la deuxième meilleure solution x^2 du problème son déterminer. Ensuite, l'ensemble admissible X du problème est partitionné en deux sous-ensembles X_1 et X_2 tel que x^1 et la meilleure solution dans X_1 et x^2 est la meilleure solution dans X_2. Dans chacun de ces nouveaux sous-ensembles X_1 et X_2, on détermine la deuxième meilleure solution x_1^2 et x_2^2 respectivement. La comparaison de ces deux nouvelles solutions permet d'obtenir la troisième meilleure solution x^3 du problème. Supposons par exemple que $x^3 = x_2^2$ alors X_2 est renommé $\tilde{X}_2$ et est partitionné en deux sous-ensemble X_2 et X_3 tel que x^2 et la meilleure solution dans X_2 et x^3 est la meilleure solution dans X_3. En comparant les deuxièmes meilleures solutions x_1^2, x_2^2, x_3^2 respectivement dans X_1, X_2 et X_3, on obtient la quatrième meilleure solution x^4.

A la K^{eme} itération de l'algorithme, l'ensemble admissible X est partitionné en K ensembles $X_1, \ldots, X_k$, et la meilleure et deuxième meilleure solution dans chacun de ces sous-ensembles est connue. Une comparaison de chacune de ces deuxièmes meilleures solutions $x_1^2, \ldots, x_K^2$ permet l'obtention de la $(K+1)^{eme}$ meilleure solution dans X [32].

4.1.2 Algorithme de Murty

L'algorithme de Murty [28] est basé sur un partitionnement différent de celui de Hamacher et Queyranne. On note x^1 la solution optimale du problème, ce n'est pas X qui est partitionné cette fois, mais $X \backslash \{x^1\}$.

En notant $x^1 = \{(1, j_1), ..., (n, j_n)\}$, les ensembles $X_i = \{x \in X : (1, j_1), ..., (i-1, j_{i-1}) \in x, (i, j_i) \notin x\}$ pour $i = 1, ..., n-1$ forment une partition de $X \backslash \{x^1\}$. La deuxième meilleure solution se trouve donc dans un de ces ensembles. Il est facile de trouver la solution optimale dans chacun des X_i, $i = 1, ..., n-1$, en adaptant la matrice des coûts puis en résolvant le problème d'affictation correspondant. Par comparaison des solutions optimales dans X_i, $i = 1, ...n$, on obtient la deuxième meilleure solution dans X.

Supposon que la deuxième meilleure solution dans X soit $x^2 = x_i^1$, la solution optimale dans X_i. Pour trouver la troisième meilleure solution, X_i est renommé $\tilde{X}$ et $\tilde{X}\backslash\{x^2\}$ est partitionné en $n-i$ sous-ensembles, il y a en effet déja $(i-1)$ affectations imposées et une interdite dans $\tilde{X}$. En notant $x^2 = \{(1, j_1), ..., (i-1, j_i-1), (i, j'), ..., (n, j'_n)\}$, les ensembles $\tilde{X}_k = \{x \in \tilde{X} : (i, j'_i), ..., (k-1, j'_{k-1}) \in x, (k, j'_k) \notin x\}$ pour $k = i, ..., n-1$, fortement une partition de $X\backslash\{x^2\}$. La solution optimale (si elle existe) est calculée dans chacun de sous-ensembles $\tilde{X}_k$ pour $k = i, ..., n-1$. En comparant les solutions des sous ensembles $\tilde{X}_k$ pour $k = 1, ..., n-1$ et celle des sous-ensembles X_j, $j = 1, ...i-1, i+1, n-1$, on obtient la troisième meilleure solution x^3.

A la k^{eme} itération de l'algorithme, l'ensemble $X\backslash\{x^1, ..., x^k\}$ est partitionné en un certain nombre de sou-ensemble dans lesquels on connait la solution optimale, la (k+1)^{eme} meilleure solution x^{k+1} en est déduit et l'ensemble $X'\backslash\{x^{k+1}\}$ afin de déterminer la $(k+2)^{eme}$ meilleure solution si cela est nécessaire. A chaque itération, on doit résoudre au plus $(n-1)$ problème d'affectation, chaque solution est donc générée en $O(n^4)$. La détermination des k meilleures solutions d'un problème d'affectaion avec cette méthode est donc en $O(kn^4)$ [32].

4.1.3 Algorithme de Pascoal et al

L'algorithme de Pascoal et al [30] est une amélioration de celui de Murty. Le partitionnement utilisé est le même, et l'avantage de cet algorithme est qu'à chaque itération il détermine au plus $n-1$ plus courts chemins dans des graphes avec $2n$ sommets alors que l'algorithme de Cheggiredy et Hamacher [11] détermine au plus $2n$ plus courts chemins (n pour chaque nouvel ensemble du partitionnement) dans un graphe avec au plus $2n$ sommets. Et pour dérerminer les k meilleures solutions d'un problème d'affictation, l'algorithme de Pascoal et al a une complexité en $O(kn^3)$ [15].

4.2 Méthode en deux phases

La méthode en deux phases [37, 38] est un schéma général de résolution pour les problèmes d'optimisation combinatoire bi-objectif. La phase 1 varie très peu en général et la phase 2 doit être spécifique au problème.

4.2.1 Phase 1 : Détermination des solutions supportées

La phase 1 est une variant de la méthode dichotomique de Aneja et Nair [4]. Elle détermine un ensemble X_{SE_M}. On note S l'ensemble des solutions efficaces obtenues par l'algorithme. $S \leftarrow \{x^1, x^2\}$ est initialisé avec les deux solutions lexicographiquement optimales correspondants respectivement à $lexmin_{x\in X}(z_1(x), z_2(x))$ et $lexmin_{x\in X}(z_2(x), z_1(x))$.

Pendant la recherche dichotomique, les solutions de S sont triées par valeur croissante 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 $z_2(x^r) > z_2(x^s)$, son considérées. Un problème d'optimisation $(2AP_\lambda)$ 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 son énumérées. La recherche dichotomique est initialisée avec $x^r = x^1$ et $x^s = x^2$ [32].

Remarque 4.2.1 *Dans les algorithmes, les symboles* $\downarrow, \uparrow$ *et* $\updownarrow$ *indiquent le mode de transmission des paramètres dans les procédures : ils correspondent respectivement au mode entrée, sortie et entrée/sortie. Le symbole –| indique le début d'une ligne de commentaires.*

Algorithme 4.2.1 *La phase 1.*

Paramètres $\updownarrow$: S

- -| Détermination de l'ensemble R des solutions optimales x de $(2AP_\lambda)$:
- -| $\min\{\lambda_1 z_1(x) + \lambda_2 z_2(x) : x \in X\}$
- -| où $\lambda_1 = z_2(x^r) - z_2(x^s)$, et $\lambda_2 = z_1(x^s) - z_1(x^r)$
- -| $C^\lambda_{ij} = \lambda_1 C^1_{ij} + \lambda_2 C^2_{ij}$, $i = 1, ..., n$; $j = 1, ..., n$

$SolveAP$ $(C^\lambda \downarrow, x \uparrow, \overline{C^\lambda} \uparrow)$; enumerate $(\overline{C}^\lambda \downarrow, x \downarrow, R \uparrow)$;

S$\leftarrow S \cup R$

Si $\{z^\lambda(x) : x \in R\} \cap [z(x^r)z(x^s)] = \varnothing$ **Alors**

- -| Cas a)

- -| Soient x^{t_1} et x^{t_2},les solutiont de R avec respectivement les valeurs minimales et maximales pour z_1

SolveRecursion ($x^r \downarrow, x^{t_1} \downarrow, S \updownarrow$); SolveRecursion ($x^{t_2} \downarrow, x^s \downarrow, S \updownarrow$)

Fin Si

- -| Si $\{z^\lambda(x) : x \in R\} \subset [z(x^r)z(x^s)]$ **alors**

- -| Cas b) : rien à faire

Soit $R = \{x^t : t \in T\}$ l'ensemble des solutions optimales de $(2AP_\lambda)$, où T est un ensemble d'indice tel que $|T|$ est le nombre de solution optimales de $(2AP_\lambda)$. Pour deux point 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 y a deux cas possibles :

a) $z(R) \cap [z(x^r)z(x^s)]$, voir figure 4.2.1. Alors toutes les solution x^t sont de nouvelles solutions supportées et sont ajoutées à S. ensuit 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 solution où le minimum et le maximum sont atteints. Pour poursuivre la recherche dichotomique, deux nouveaux problèmes définis par une somme pondérée sont considérés : un défini par les solutions x^r et x^{t_1} et un autre défini par les solutions x^{t_2}et x^s (en utilisant la procédure SolveRecursion dans l'algorithme 4.2.1). 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 conséquent aucune nouvelle solution ne peut être obtenue.

b) $z(R) \subset [z(x^r)z(x^s)]$, voir figure 4.2.2. 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.

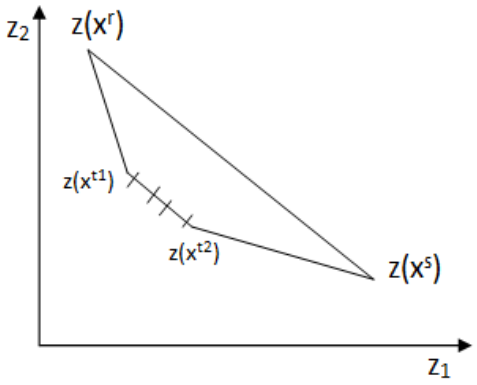


Fig 4.2.1- Phase 1, Cas a)

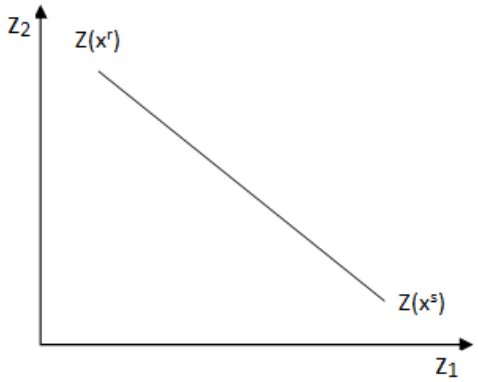


Fig 4.2.2- Phase 1, Cas b)

La phase 1 s'arrête s'il n'y a plus aucun problème pondéré ($2AP_\lambda$) à résoudre, on a alors $S = X_{SE_M}$. Remarquons que sans l'application de l'algorithme d'énumération, on retouve exactement la méthode de Aneja et Nair [4] qui permet seulement la détermination d'un ensemble $X_{SE_{1_m}}$ et éventuellement quelques autres solutions supportées. L'algorithme d'énumération est donc nécessaire pour garantir l'obtention d'un ensemble $X_{SE_{2_m}}$ et bien sûr d'un ensemble X_{SE_M} [32].

4.2.2 Phase 2 : Détermination des solutions non-supportées

Dans la phase 2, des solutions $x \in X$ telles que $z(x)$ se situe dans le triangle défini par deux points supportés consécutifs $z(x^r)$ et $z(x^s)$ dans l'espace des objectifs sont déterminées. Afin 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 [38] et Tuyttens et al [37].

Borne supérieure de Tuyttens et al

Dans la phase 2, chaque triangle défini par deux points supportés consécutifs et le point nadir doit être exploré. Soient x^r et x^s deux solutions supportées consécutive dans X_{SEm}, et λ le poids pour lequel x^r et x^s sont deux solutions optimales de ($2AP_\lambda$). Ulungu et Teghem [38] etTuyttens et al [37] ont proposé des bornes supérieures sur la valeur de la

fonction objectif z^λ définie par $\lambda_1 z_1(x) + \lambda_2 z_2(x)$ pour les solutions efficaces x avec $z(x)$ dans le triangle défini par $z(x^r)$ et $z(x^s)$. Soit $\Delta(z(x^r), z(x^s))$ l'intérieur de ce triangle.

Dans la phase 2, les points réalisables dans une bande du triangle $\Delta(z(x^r), z(x^s))$ 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 parallèle contient le point $(z_1(x^r)z_2(x^s))$, donc toute solution x telle que $z(x)$ se situe dans le triangle doit être énumérée. L'utilisation de bornes supérieures permet de rapprocher la droite parallèle de $(z(x^r)z(x^s))$ [32].

Exploration des triangles

Soient x^r et x^s deux solutions supportées consécutives dans X_{SE_m} et λ le poids tel que x^r et x^s sont des solutions optimales de $(2AP_\lambda)$. La phase 2 détermine des solutions admissibles x telles que :

$$z_1(x^r) < z_1(x) < z_1(x^s) et z_1(x^r) > z_1(x) > z_1(x^s)$$

Bien sûr, les points correspondants à toutes ces solutions se situent dans le triangle $\Delta(z(x^r), z(x^s))$. Pour l'exploration des triangles $\Delta(z(x^r), z(x^s))$ nous recherchons 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 phases. En effet, ce choix ne trahit pas l'esprit original de la méthode car il n'implique aucune modification de la structure du problème. Par rapport à une stratégie de fixation de variable, 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, les algorithmes de ranking ont été décrire dans la section 4.1.

Nous considérons le problème $(2AP)$ pour lequel les solutions supportées consécutives x^r et x^s sont optimales et nous appliquons un algorithme pour déterminer les k meilleures solutions. Et pour chaque solution obtenue pendant l'exploration, nous vérifions s'il est nécessaire 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 $z^\lambda > \beta_i$ pour $i = 1$ ou 2 [32].

Remarque 4.2.2 *Toute solution x non dominée au moment de son obtention le sera définitivement. En effet, s'il existe une autre solution x' telle que $z(x') \leq z(x)$, alors nécessairement $z^\lambda(x') < z^\lambda(x)$, ce qui implique que x' a déjà été obtenue [39].*

Algorithme 4.2.2 *La phase 2.*

Paramètres $\downarrow$: C^1, C^2, X_{SE_m}

Paramètres $\uparrow$: $S : X_{NE}$

$S \leftarrow \varnothing$

Pour tout x^r, x^s solutions consécutives dans X_{SE_m} **faire**

- -| $C^\lambda = [\lambda_1 c^1_{ij} + \lambda_2 c^2_{ij}]$ avec $\lambda_1 = z_2(x^r) - z_2(x^s)$, et $\lambda_2 = z_1(x^s) - z_1(x^r)$, $y^N = (y^s_1, y^r_2)$.

$SolveAP$ $(C^\lambda \downarrow, \overline{x} \uparrow, \overline{C^\lambda} \uparrow)$

- -|$\overline{C^\lambda}$ la matrice des coût réduit de z^λ

$L \leftarrow \{(i,j) : \overline{c}^\lambda_{ij} > 0\}$

$R \leftarrow \varnothing$ - -|R est la liste des solutions efficaces du triangle $\triangle(y^r, y^s)$

$BS \leftarrow \lambda_1 y^s_1 + \lambda_2 y^r_2$ - -| BS est la valeure initiale pour la borne supérieure

Si $BS > z^\lambda$ (x^r) **Alors**

Pour tout $(i,j) \in L$ **faire**

Teste **Si** la présence de l'affectation (i,j) donne une solution à l'extérieure du triangle

$c^\lambda_{ij} \leftarrow \infty$

Fin Si

Fin Pour

Si $L \neq \varnothing Alors$

–| debut du Ranking

$K \leftarrow 0$

Tant que $(z^{\lambda}\ (x^K) \leq BS)$ **faire**

$K \leftarrow K + 1$

Appliquer K-best $(K\downarrow, C^{\lambda}\downarrow, x^K\uparrow)$ - -|Calculer la prochaine meilleure solution

Si x^K est dans le triangle $\triangle(y^r, y^s)$ et x^K est non dominé (par tous les éléments de R)

$R \leftarrow R \cup \{x^K\}$

calculer la nouvelle borne supérieure

Fin Si

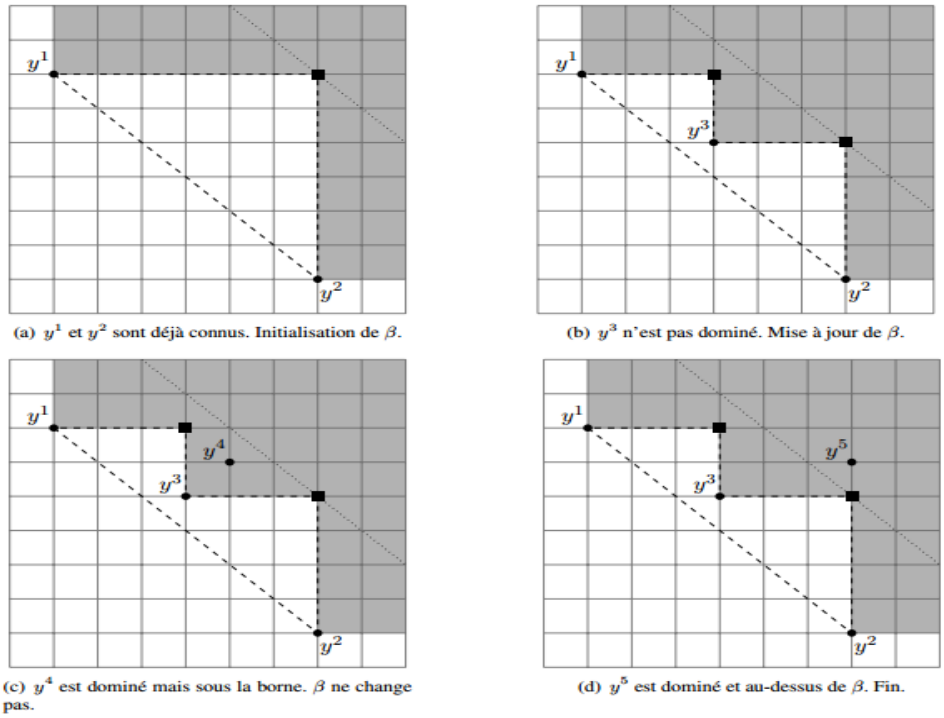
Fin Tant que

Fin Si

Fin Si

$S \leftarrow S \cup R$

Fin Pour tout



(a) y^1 et y^2 sont déjà connus. Initialisation de β.

(b) y^3 n'est pas dominé. Mise à jour de β.

(c) y^4 est dominé mais sous la borne. β ne change pas.

(d) y^5 est dominé et au-dessus de β. Fin.

Fig 4.2.3 – Exemple d'exécution de la méthode du ranking dans un triangle $\Delta(y^1, y^2)$ dans le cadre de la seconde phase.

Exemple 4.2.1 *Nous considérons une instance du problème d'affictation bi-objectif* $(2AP)$ *définie par les matrices de coûts suivantes.*

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

PHASE 1

En applique la dichotomie sur ce problème. Avec l'initialisation, nous obtenons deux solutions :

$x^1 = \{x_{11} = x_{23} = x_{32} = x_{44} = 1\}$ dont le point correspondant est $y^1 = (20, 36)$.

$x^2 = \{x_{13} = x_{24} = x_{31} = x_{42} = 1\}$ dont le point correspondant est $y^2 = (28, 18)$.

La normale à la droite joignant les points y^1 et y^2 est $\lambda = (18, 8)$.

En résolvant le problème

$$(P_{(18,8)}) = \begin{bmatrix} 90 & 138 & 158 & 136 \\ 218 & 166 & 234 & 210 \\ 130 & 154 & 254 & 178 \\ 192 & 150 & 160 & 170 \end{bmatrix}$$

Nous obtenons une nouvelle solution supportée :

$x^3 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^3 = (24, 20)$.

$y^3 \cap [y^1 y^2] = \varnothing$ donc x^3 est une nouvelle solution supportée et $S = \{x^1, x^2, x^3\}$.

La normale à la droite joignant les points y^1 et y^3 est $\lambda = (16, 4)$.

Pour rechercher une nouvelle solution avec $\lambda = (16, 4)$ on résout le problème

$$(P_{(16,4)}) = \begin{bmatrix} 52 & 104 & 128 & 96 \\ 172 & 132 & 180 & 168 \\ 100 & 112 & 204 & 124 \\ 152 & 124 & 136 & 120 \end{bmatrix}$$

Nous obtenons une nouvelle solution supportée :

$x^4 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$ dont le point correspondant est $y^4 = (21, 27)$.

$y^4 \cap [y^1 y^3] = \varnothing$ donc x^4 est une nouvelle solution supportée et $S = \{x^1, x^2, x^3, x^4\}$.

La normale à la droite joignant les points y^1 et y^4 est $\lambda = (9, 1)$.

En résolvant le problème

$$(P_{(9,1)}) = \begin{bmatrix} 18 & 51 & 67 & 44 \\ 88 & 68 & 90 & 87 \\ 50 & 53 & 106 & 56 \\ 78 & 66 & 74 & 55 \end{bmatrix}$$

Nous obtenons la solution

$x^5 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$ dont le point correspondant est $y^5 = (21, 27)$.

$y^5 \cap [y^1 y^4] = y^4$ donc $x^5 = x^4$ et $S = \{x^1, x^2, x^3, x^4\}$.

La normale à la droite joignant les points y^4 et y^3 est $\lambda = (7, 3)$.

En résolvant le problème

$$(P_{(7,3)}) = \begin{bmatrix} 34 & 53 & 61 & 52 \\ 84 & 64 & 90 & 81 \\ 50 & 59 & 98 & 68 \\ 74 & 58 & 62 & 65 \end{bmatrix}$$

Nous obtenons la solution :

$x^7 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^7 = (24, 20)$.

$y^7 \cap [y^4 y^3] = y^3$donc $x^7 = x^3$ et $S = \{x^1, x^2, x^3, x^4\}$.

La normale à la droite joignant les points y^3 et y^2 est $\lambda = (2, 4)$.

En résolvant le problème

$$(P_{(2,4)}) = \begin{bmatrix} 38 & 34 & 30 & 40 \\ 46 & 34 & 54 & 42 \\ 30 & 42 & 50 & 54 \\ 40 & 26 & 24 & 50 \end{bmatrix}$$

Nous obtenons la solution :

$x^6 = \{x_{13} = x_{24} = x_{31} = x_{42} = 1\}$ dont le point correspondant est $y^6 = (28, 18)$.

$y^6 \cap [y^3 y^2] = y^2$donc $x^6 = x^2$ et $S = \{x^1, x^2, x^3, x^4\}$.

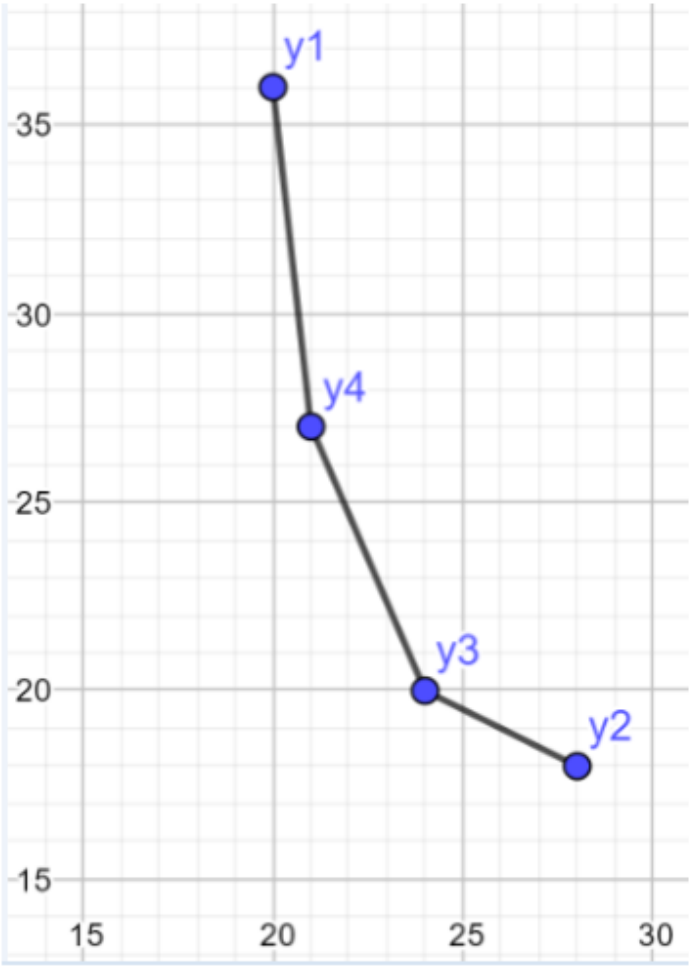


Fig 4.2.4 - La phase 1 appliquer au problème d'affectation bi-objectif

PHASE 2

Pour trouver les solutions non supportées on doit explorer les triangles $\Delta(y^1, y^4)$, $\Delta(y^4, y^3)$ et $\Delta(y^3, y^2)$ utilisant le ranking :

I) On recherche dans le triangle $\Delta(y^1, y^4)$ dont le point nadir local est $y^N = (21, 36)$.

La normale à la droite joignant les points y^1 et y^4 est $\lambda = (9, 1)$.

$$\text{et } (P_{(9,1)}) = \begin{bmatrix} 18 & 51 & 67 & 44 \\ 88 & 68 & 90 & 87 \\ 50 & 53 & 106 & 56 \\ 78 & 66 & 74 & 55 \end{bmatrix}$$

I.1) Pour applique l'algorithme de ranking on choisit de commencer par le point $y^1 = (20, 36)$ tel que la solution correspondante est $x^1 = \{x_{11} = x_{23} = x_{32} = x_{44} = 1\}$.

On interdit l'afféctation $(1, 1)$**, on aura :**

$$\begin{bmatrix} \infty & 51 & 67 & 44 \\ 88 & 68 & 90 & 87 \\ 50 & 53 & 106 & 56 \\ 78 & 66 & 74 & 55 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution:

$x^7 = x^3 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^7 = y^3 = (24, 20)$, alors $S = \{x^1, x^2, x^3, x^4\}$.

*** On force l'afféctation** $(1, 1)$ **et on interdit** $(\mathbf{2}, \mathbf{3})$**, on aura :**

$$\begin{bmatrix} 68 & \infty & 87 \\ 53 & 106 & 56 \\ 66 & 74 & 55 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^8 = x^4 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$ dont le point correspondant est $y^8 = y^4 = (21, 27)$, alors $S = \{x^1, x^2, x^3, x^4\}$.

*** On force l'afféctation** $(1,1)$ **et** $(2,3)$ **et on interdit** $(3,2)$**, on aura :**

$$\begin{bmatrix} \infty & 56 \\ 66 & 55 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution:

$x^9 = \{x_{11} = x_{23} = x_{34} = x_{42} = 1\}$ dont le point correspondant est $y^9 = (22, 32)$.

On a $y_1^9 > y_1^N$ donc y^9 est dominé. Alors $S = \{x^1, x^2, x^3, x^4\}$

I.2) Quand on impose l'affectation $(1,1)$ et on interdit $(2,3)$, nous obtenons la solution $x^4 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$ tel que le point correspondant est $y^4 = (21, 27)$ qui est un point non dominé supporté, alors pour continue on interdit l'affectation $(2,2)$ et on aura :

$$\begin{bmatrix} \infty & \infty & 87 \\ 53 & 106 & 56 \\ 66 & 74 & 55 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{10} = \{x_{11} = x_{24} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^{10} = (23, 22)$.

On a $y_1^{10} > y_1^N$ donc y^{10} est dominé. Alors $S = \{x^1, x^2, x^3, x^4\}$.

*** On force l'affectation** $(2,2)$ **et on interdit** $(3,4)$**, on aura :**

$$\begin{bmatrix} 106 & \infty \\ 74 & 55 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution:

$x^{11} = \{x_{11} = x_{22} = x_{33} = x_{44} = 1\}$ dont le point correspondant est $y^{11} = (24, 31)$.

On a $y_1^{11} > y_1^N$ donc y^{11} est dominé. Alors $S = \{x^1, x^2, x^3, x^4\}$.

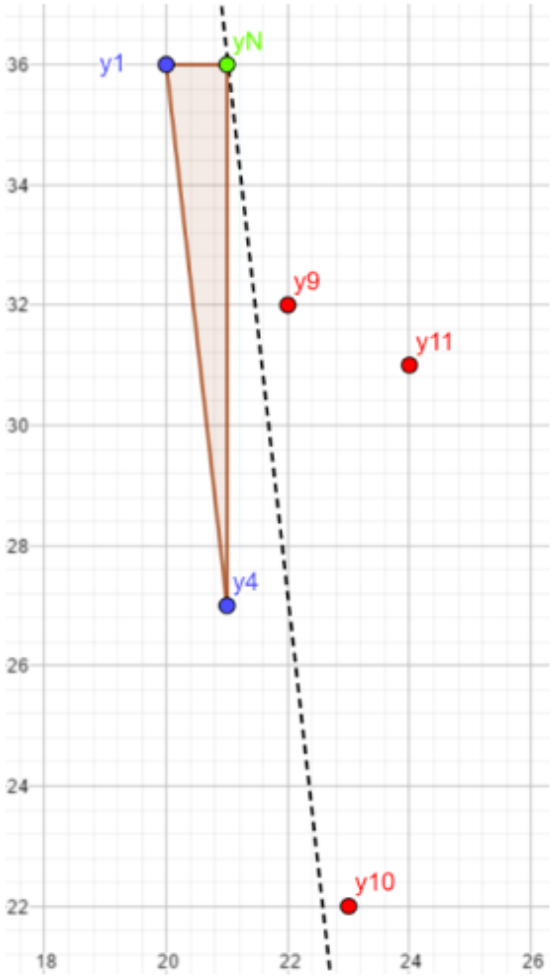


Fig 4.2.5 - Exploration de triangle $\Delta(y^1, y^4)$.

II) On recherche dans le triangle $\Delta(y^4, y^3)$ dont le point nadir local est $y^N = (24, 27)$. La normale à la droite joignant les points y^4 et y^3 est $\lambda = (7, 3)$.

$$\text{et } (P_{(7,3)}) = \begin{bmatrix} 34 & 53 & 61 & 52 \\ 84 & 64 & 90 & 81 \\ 50 & 59 & 98 & 68 \\ 74 & 58 & 62 & 55 \end{bmatrix}$$

II.1) On commence par le piont $y^4 = (21, 27)$ tel que la solution correspondante est $x^4 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$.

*** On interdit l'affectation** $(1, 1)$**, on aura :**

$$\begin{bmatrix} \infty & 53 & 61 & 52 \\ 84 & 64 & 90 & 81 \\ 50 & 59 & 98 & 68 \\ 74 & 58 & 62 & 55 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution:

$x^{12} = x^3 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^{12} = y^3 = (24, 20)$, alors $S = \{x^1, x^2, x^3, x^4\}$.

*** On force l'affectation** $(1, 1)$ **et on interdit** $(2, 2)$**, on aura :**

$$\begin{bmatrix} \infty & 90 & 81 \\ 59 & 98 & 68 \\ 58 & 62 & 65 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{13} = \{x_{11} = x_{24} = x_{32} = x_{43} = 1\}$ dont le point correspondant est $y^{13} = (23, 25)$.

On a $y_1^{13} < y_1^N$ et $y_2^{13} < y_2^N$ donc x^{13} est une solution non-supportée, et $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation** $(1, 1), (2, 2)$ **et on interdit** $(3, 4)$**, on aura :**

$$\begin{bmatrix} 98 & \infty \\ 62 & 65 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{14} = \{x_{11} = x_{22} = x_{33} = x_{44} = 1\}$ dont le point correspondant est $y^{14} = (24, 31)$.

On a $y_2^{14} > y_2^N$ donc y^{14} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

II.2) Quand on interdit l'affectation $(1,1)$, nous obtenons la solution $x^3 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$ tel que le point correspondant est $y^3 = (24, 20)$ qui est un point non dominer supporté, alors pour continue on interdit l'affectation $(1,4)$ et on aura :

$$\begin{bmatrix} \infty & 53 & 61 & \infty \\ 84 & 64 & 90 & 81 \\ 50 & 59 & 98 & 68 \\ 74 & 58 & 62 & 65 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution : $x^{15} = \{x_{13} = x_{22} = x_{31} = x_{44} = 1\}$ dont le point correspondant est $y^{15} = (24, 24)$.
On a $y_1^{15} = y_1^N$ donc y^{15} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation**$(1,4)$ **et on interdit** $(2,2)$**, on aura :**

$$\begin{bmatrix} 84 & \infty & 90 \\ 50 & 59 & 98 \\ 74 & 58 & 62 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution : $x^{16} = \{x_{14} = x_{23} = x_{31} = x_{42} = 1\}$ dont le point correspondant est $y^{16} = (25, 25)$.

On a $y_1^{16} > y_1^N$ donc y^{16} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation** $(1,4), (2,2)$ **et on interdit** $(3,1)$**, on aura :**

$$\begin{bmatrix} \infty & 98 \\ 74 & 62 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution : $x^{17} = \{x_{11} = x_{22} = x_{33} = x_{41} = 1\}$ dont le point correspondant est $y^{17} = (30, 26)$.

On a $y_1^{17} > y_1^N$ donc y^{17} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

II.3) Quand on impose l'affectation $(1,1)$ et on interdit $(2,2)$ nous obtenons la solution $x^{13} = \{x_{11} = x_{24} = x_{32} = x_{43} = 1\}$ tel que le point correspondant est $y^{13} = (23, 25)$ qui est un point non dominé non-supporté, alors pour continue on interdit l'affectation $(2,4)$ et on aura :

$$\begin{bmatrix} \infty & 90 & \infty \\ 59 & 98 & 68 \\ 58 & 62 & 65 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{18} = x^1 = \{x_{11} = x_{23} = x_{32} = x_{44} = 1\}$ dont le point correspondant est $y^{18} = y^1 = (20, 36)$, alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation (2,4) et on interdit** $(3,2)$**, on aura :**

$$\begin{bmatrix} \infty & 98 \\ 58 & 62 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{19} = \{x_{11} = x_{24} = x_{33} = x_{42} = 1\}$ dont le point correspondant est $y^{19} = (28, 25)$.

On a $y_1^{19} > y_1^N$ donc y^{19} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

II.4) Quand on interdit l'affectation $(1,1)$, $(1,4)$ nous obtenons la solution $x^{15} = \{x_{13} = x_{22} = x_{31} = x_{44} = 1\}$ tel que le point correspondant est $y^{15} = (24, 24)$ qui est un point dominée, alors pour continue on interdit l'affectation $(1,3)$ et on aura :

$$\begin{bmatrix} \infty & 53 & \infty & \infty \\ 84 & 64 & 90 & 81 \\ 50 & 59 & 98 & 68 \\ 74 & 58 & 62 & 65 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution : $x^{20} = \{x_{12} = x_{24} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^{20} = (27, 19)$. On a $y_1^{20} = y_1^N$ donc y^{20} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation** $(1, 3)$ **et on interdit** $(2, 2)$**, on aura :**

$$\begin{bmatrix} 84 & \infty & 81 \\ 50 & 59 & 68 \\ 74 & 58 & 65 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution : $x^{21} = x^2 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^{21} = y^2 = (28, 18)$, alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation** $(1, 3), (2, 2)$ **et on interdit** $(3, 1)$**, on aura :**

$$\begin{bmatrix} \infty & 68 \\ 74 & 65 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution : $x^{22} = \{x_{13} = x_{22} = x_{34} = x_{41} = 1\}$ dont le point correspondant est $y^{22} = (27, 26)$.

On a $y_1^{22} > y_1^N$ donc y^{22} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

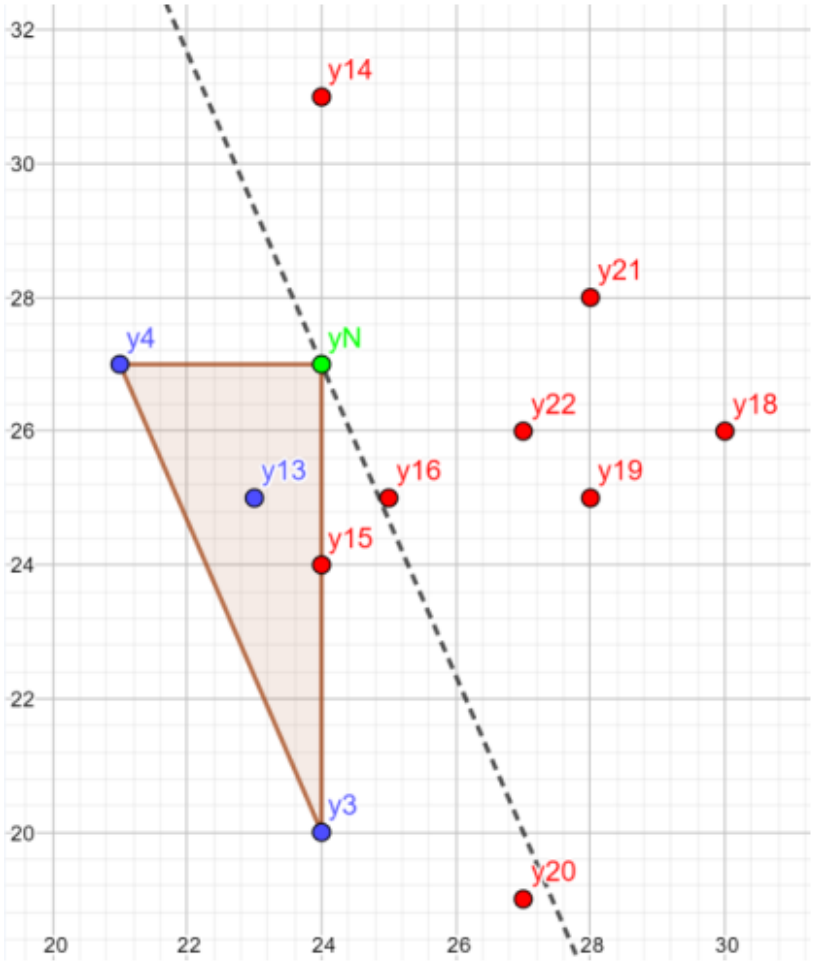


Fig 4.2.6 - Exploration de triangle $\Delta(y^4, y^3)$.

III) On recherche dans le triangle $\Delta(y^3, y^2)$ dont le point nadir locale est $y^N = (28, 20)$. La normale à la droite joignant les points y^3 et y^2 est $\lambda = (2, 4)$.

$$\text{et } (P_{(2,4)}) = \begin{bmatrix} 38 & 34 & 30 & 40 \\ 46 & 34 & 54 & 42 \\ 30 & 42 & 50 & 54 \\ 40 & 26 & 24 & 50 \end{bmatrix}$$

III.1) On commence par le point $y^3 = (24, 20)$ tel que la solution correspondant est $x^3 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$.

*** On interdit l'affectation** $(1, 4)$**, on aura :**

$$\begin{bmatrix} 38 & 34 & 30 & \infty \\ 46 & 34 & 54 & 42 \\ 30 & 42 & 50 & 54 \\ 40 & 26 & 24 & 50 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution:

$x^{23} = x^2 = \{x_{13} = x_{24} = x_{31} = x_{42} = 1\}$ dont le point correspondant est $y^{23} = y^2 = (28, 18)$, alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation** $(1, 4)$ **et on interdit** $(2, 2)$**, on aura :**

$$\begin{bmatrix} 46 & \infty & 54 \\ 30 & 42 & 50 \\ 40 & 26 & 24 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{24} = \{x_{14} = x_{23} = x_{31} = x_{42} = 1\}$ dont le point correspondant est $y^{24} = (25, 25)$.

On a $y_2^{24} > y_2^N$ donc $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

*** On force l'affectation** $(1, 4), (2, 2)$ **et on interdit** $(3, 1)$**, on aura :**

$$\begin{bmatrix} \infty & 50 \\ 40 & 24 \end{bmatrix}$$

Après la résolution du problème avec l'algorithme hongrois, nous obtenons une solution:

$x^{25} = \{x_{14} = x_{22} = x_{33} = x_{41} = 1\}$ dont le point correspondant est $y^{25} = (30, 26)$.

On a $y_1^{25} > y_1^N$ et $y_2^{25} > y_2^N$ donc y^{25} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}\}$.

III.2) Quand on interdit l'affectation $(1,4)$, nous obtenons la solution $x^2 = \{x_{13} = x_{24} = x_{31} = x_{42} = 1\}$ tel que le point correspondant est $y^2 = (28, 18)$ qui est un point non dominer supporté, alors pour continue on interdit l'affectation $(1,3)$ et on aura :

$$\begin{bmatrix} 38 & 34 & \infty & \infty \\ 46 & 34 & 54 & 42 \\ 30 & 42 & 50 & 54 \\ 40 & 26 & 24 & 50 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{26} = \{x_{12} = x_{24} = x_{31} = x_{44} = 1\}$ dont le point correspondant est $y^{26} = (27, 19)$.

On a $y_1^{26} < y_1^N$ et $y_2^{26} < y_2^N$ donc y^{26} est point non-dominé non-supporté. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}, x^{26}\}$.

*** On force l'affectation**$(1,3)$ **et on interdit** $(2,4)$**, on aura :**

$$\begin{bmatrix} 46 & 34 & \infty \\ 30 & 42 & 50 \\ 40 & 26 & 50 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{27} = \{x_{13} = x_{22} = x_{31} = x_{44} = 1\}$ dont le point correspondant est $y^{27} = (24, 24)$.

On a $y_2^{27} > y_2^N$ donc y^{27} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}, x^{26}\}$.

*** On force l'affectation** $(1,3),(2,4)$ **et on interdit** $(3,1)$**, on aura :**

$$\begin{bmatrix} \infty & 42 \\ 40 & 26 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{28} = \{x_{13} = x_{24} = x_{32} = x_{41} = 1\}$ dont le point correspondant est $y^{28} = (29, 24)$.

On a $y_1^{28} > y_1^N$ et $y_2^{28} > y_2^N$ donc y^{28} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}, x^{26}\}$.

III.3) Quand on interdit l'affectation $(1, 4), (1, 3)$ nous obtenons la solution $x^{26} = \{x_{12} = x_{24} = x_{31} = x_{44} = 1\}$ tel que le point correspondant est $y^{26} = (27, 19)$ qui est un point non dominer non-supporté, alors pour continue on interdit l'affectation $(1, 2)$ et on aura :

$$\begin{bmatrix} 38 & \infty & \infty & \infty \\ 46 & 34 & 54 & 42 \\ 30 & 42 & 50 & 54 \\ 40 & 26 & 24 & 50 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{29} = x^{13} = \{x_{11} = x_{24} = x_{32} = x_{43} = 1\}$ dont le point correspondant est $y^{29} = y^{13} = (23, 25)$, alors $S = \{x^1, x^2, x^3, x^4, x^{13}, x^{26}\}$.

*** On force l'affectation $(1, 2)$ et on interdit $(2, 4)$, on aura :**

$$\begin{bmatrix} 46 & 54 & \infty \\ 30 & 50 & 54 \\ 40 & 24 & 50 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{30} = \{x_{12} = x_{21} = x_{34} = x_{43} = 1\}$ dont le point correspondant est $y^{30} = (27, 26)$.

On a $y_2^{30} > y_2^N$ donc y^{30} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}, x^{26}\}$.

*** On force l'affectation $(1, 2)$ et $(2, 4)$ on interdit $(3, 1)$, on aura :**

$$\begin{bmatrix} \infty & 50 \\ 40 & 24 \end{bmatrix}$$

En résolvant le problème avec l'algorithme hongrois, nous obtenons une solution :

$x^{31} = \{x_{12} = x_{24} = x_{33} = x_{41} = 1\}$ dont le point correspondant est $y^{31} = (33, 25)$.

On a $y_1^{31} > y_1^N$ et $y_2^{31} > y_2^N$ donc y^{31} est dominé. Alors $S = \{x^1, x^2, x^3, x^4, x^{13}, x^{26}\}$.

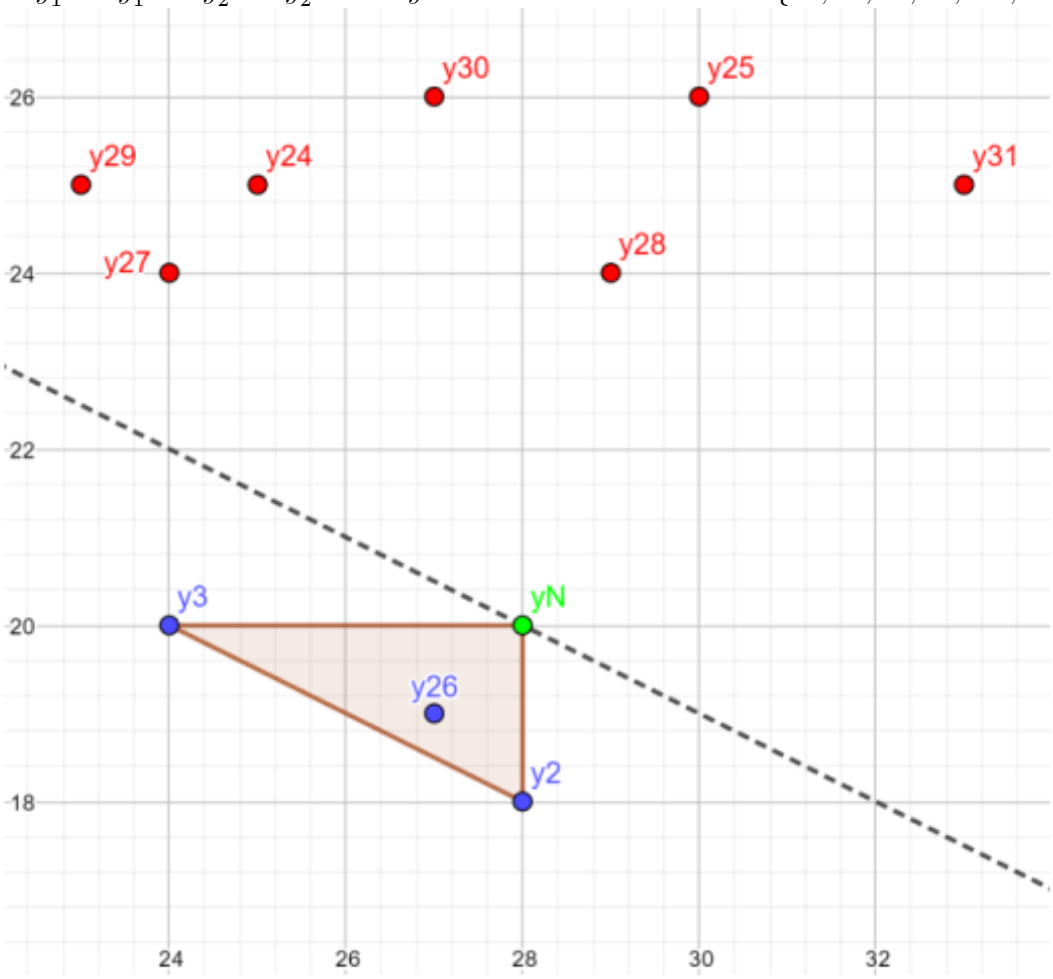


Fig 4.2.7 - Exploration de triangle $\Delta(y^3, y^2)$.

Les solutions efficaces de ce problème sont:

- $x^1 = \{x_{11} = x_{23} = x_{32} = x_{44} = 1\}$ dont le point correspondant est $y^1 = (20, 36)$.
- $x^2 = \{x_{13} = x_{24} = x_{31} = x_{42} = 1\}$ dont le point correspondant est $y^2 = (28, 18)$.
- $x^3 = \{x_{14} = x_{22} = x_{31} = x_{43} = 1\}$ dont le point correspondant est $y^3 = (24, 20)$.
- $x^4 = \{x_{11} = x_{22} = x_{34} = x_{43} = 1\}$ dont le point correspondant est $y^4 = (21, 27)$.
- $x^{13} = \{x_{11} = x_{24} = x_{32} = x_{43} = 1\}$ dont le point correspondant est $y^{13} = (23, 25)$.
- $x^{26} = \{x_{12} = x_{24} = x_{31} = x_{44} = 1\}$ dont le point correspondant est $y^{26} = (27, 19)$.

Les points y^1, y^2, y^3et y^4 sont supportés, et y^{13}, y^{26} sont non-supporté.

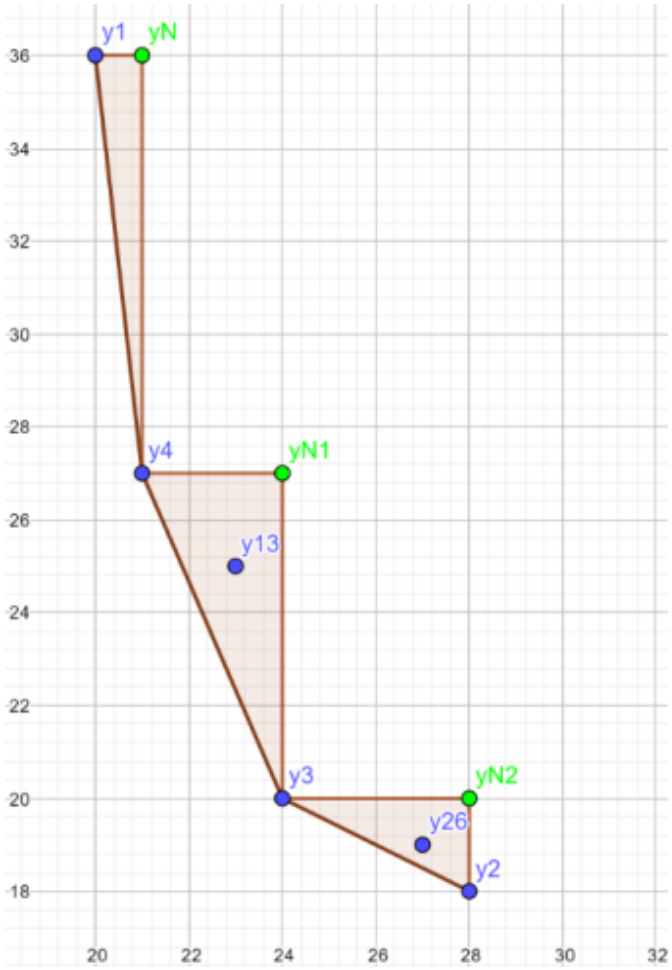


Fig 4.2.8 - Résultats de l'application de la méthode en deux phase sur l'exemple 4.2.1.

4.3 Application de la méthode des deux phases aux problèmes d'affectation avec 1 contrainte de type sac à dos

On considère(P) le problème d'affectation avec 1 contrainte de type sac à dos :

$$\begin{cases} minZ = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \quad \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \leq b \\ \quad \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, ...n \\ \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, ...n \\ \quad x_{ij} \geq 0 \ , \ \textit{entier pour tout } (i,j) \end{cases}$$

(P) est un problème mono-objectif, et qui a été déjà résolu avec la méthode Branch and Bound. Mais nous, nous intéressons à sa résolution avec la méthode en deux phases, et pour cela en le transforment en un problème bi-objectif c'est-à-dire en considérant :

$min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$ comme le premier objectif et $min \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$ comme le second .

Soit x^1 et x^2 les deux solutions initiales (lexicographiques). Nous appliquons la phase 1 mais en considération de la valeur de b, on aura trois cas possible :

1. $z_2(x^1) > b$ et $z_2(x^2) > b$, alors il n'ya pas de solution, $S = \varnothing$.

2. $z_2(x^1) < b$ et $z_2(x^2) < b$, alors x^1 est une solution optimale et la contrainte $\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$ est redondante.

3. $z_2(x^1) > b$ et $z_2(x^2) \leq b$, alors on résout le problème (P_λ), λ déduit à partir de x^1 et x^2 et on trouve une solution x^3.

Si $z_2(x^3) \leq b$ en continue la recherche dichotomique, alors un nouveau problème (P_λ) est définis avec λ est déduit à partir de x^1 et x^3. Sinon, si $z_2(x^3) > b$ alors λ est déduit à partir de x^3 et x^2.

On répète la même procédure jusqu'à trouver une solution déjà existante, afin d'explorer le triangle $\triangle(y^k, y^{k'})$ (triangle critique) définis par les deux point y^k, $y^{k'}$et le point nadir $y^N = (y_1^{k'}, y_2^k)$.

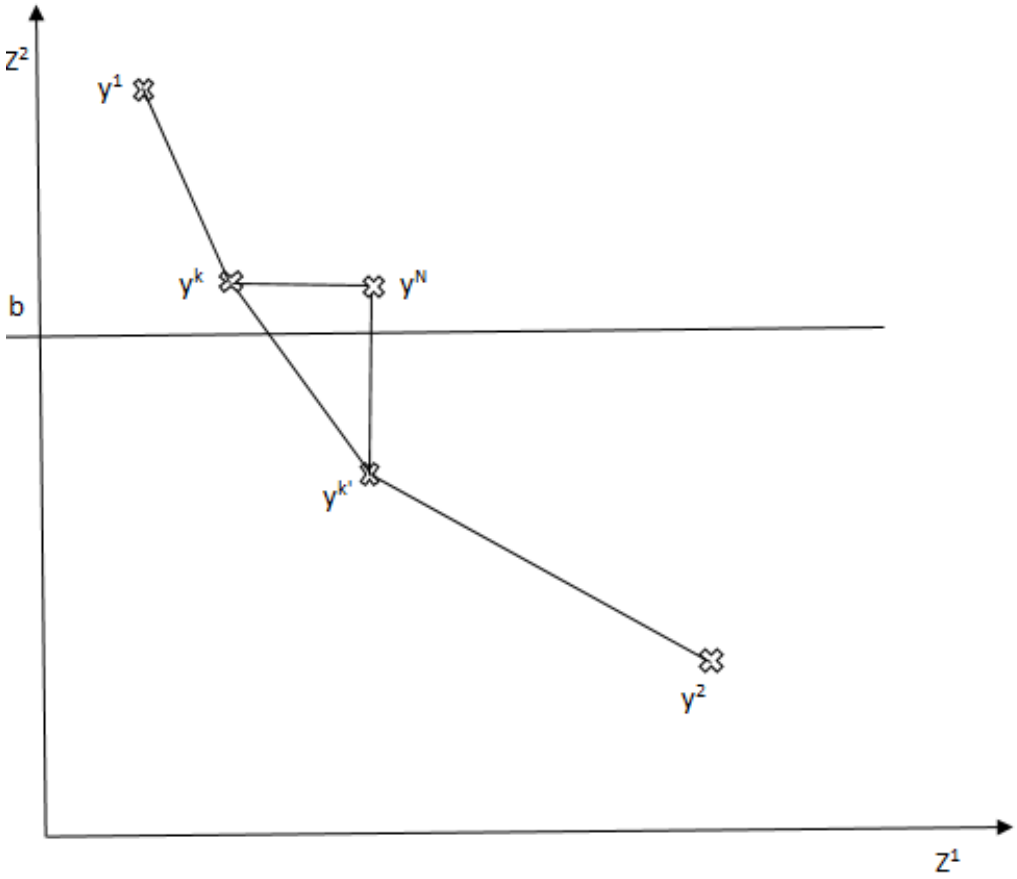


Fig 4.2.9 - La phase 1 appliqué au problème d'affectation avec 1 contrainte de type sac à dos.

Dans la deuxième phase, on détermine le triangle d'exploration. On commence par rechercher la deuxième meilleure solution x^m tel que le point correspondant est $y^m = (y_1^m, y_2^m)$.

Soit $y^s = (y_1^s, b)$ l'intersection de la droit (B) : $(z^2 = b)$ est le segment $[y^k y^{k'}]$ et $y^c = (y_1^{k'}, b)$ l'intersection de la droit (B) et le segment $[y^{k'} y^N]$. Pour déterminer y_1^s, on utilise ce qui suit.

D'après le théorème de Thalès on a :

$$\frac{x}{\lambda_2} = \frac{(b - y_2^{k'})}{\lambda_1}, avec\ x = y_1^{k'} - y_1^s ...(1)$$

donc :

$$x = \frac{\lambda_2}{\lambda_1}(b - y_2^{k'}) ...(2)$$

D'après (1) et (2) on déduit que $y_1^s = y_1^{k'} - \frac{\lambda_2}{\lambda_1}(b - y_2^{k'})$. Alors le triangle d'exploration est $\triangle(y^m, y^s)$ qui est définir par les deux point y^m, y^s et le point nadir correspondant $y^{N1} = (y_1^m, b)$.

Si la meilleure prochaine solution x^{m1} est supportée, on explore $\triangle(y^{m1}y^s)$. Sinon, si x^{m1} est non-supportée alors le triangle d'exploration est $\triangle(y^{m1}, y^{s1})$ avec le point nadir est $y^{N2} = (y_1^{m1}, b)$ et y^{s1}c'est l'intersection de la droit $(B) : (z^2 = b)$ avec la droite qui passe par y^{m1}, et qui est aussi parallèle à la droite $(y^k y^{k\prime})$. On répète cette procédure jusqu'à qu'on trouve une solution x^d dominé, qui vérifie $(\lambda_1 C^1 + \lambda_2 C^2)x^d \geq (\lambda_1 z_1^d + \lambda_2 b)$.

Remarque 4.3.1 *Cette exploration ne dépend pas du point y_1^s. Elle dépend uniquement du point nadir.*

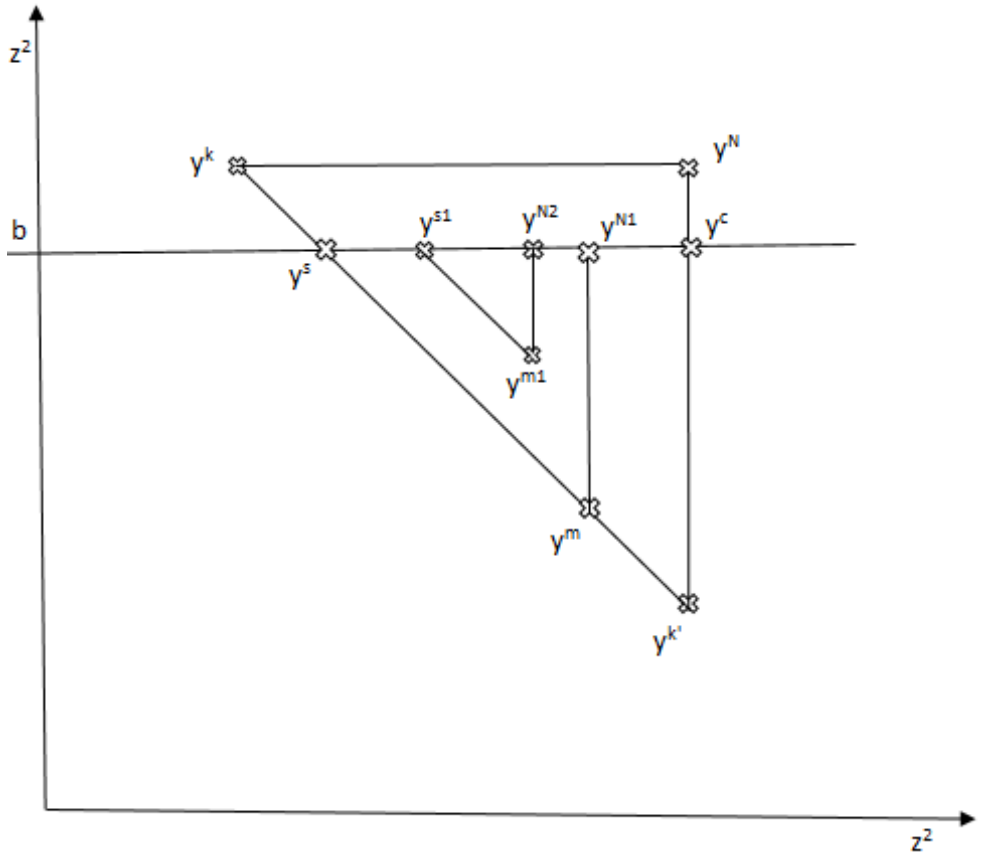


Fig 4.2.10 - La phase 2 appliquer au problème d'affectation avec 1 contrainte de type sac à dos.

Remarque 4.3.2 *La contrainte de type sac à dos est également appelée contrainte sur les ressources.*

Remarque 4.3.3 *Le schéma de résolution du problème d'affectation avec 1 contrainte de type sac à dos est semblable à celui de branch and bound, car la phase 2 (Ranking) se fait avec une arboresence de résolution.*

Exemple 4.3.1 *On prend le même exemple que les deux phases [1].*

$$C = \begin{bmatrix} 1 & 5 & 7 & 4 \\ 9 & 7 & 9 & 9 \\ 5 & 5 & 11 & 5 \\ 8 & 7 & 8 & 5 \end{bmatrix} \qquad D = \begin{bmatrix} 9 & 6 & 4 & 8 \\ 7 & 5 & 9 & 6 \\ 5 & 8 & 7 & 11 \\ 6 & 3 & 2 & 10 \end{bmatrix}$$

avec $b = 26$.

PHASE 1

on appliquons la phase 1 on détermine le triangle critique $\triangle(y^4y^3)$ avec $y^4 = (21, 27)$ et $y^3 = (24, 20)$.

PHASE 2

En applique la phase 2 on trouve que la prochaine meilleure solution est X^{13} tel que le point correspondant est $y^{13} = (23, 25)$.

On explore le nouveau triangle, tel que le point nadir correspondant est $Y^N = (23, 26)$ en trouve que tous les point sont dominés, ce qui veut dire que la meilleure solution est x^{13}.

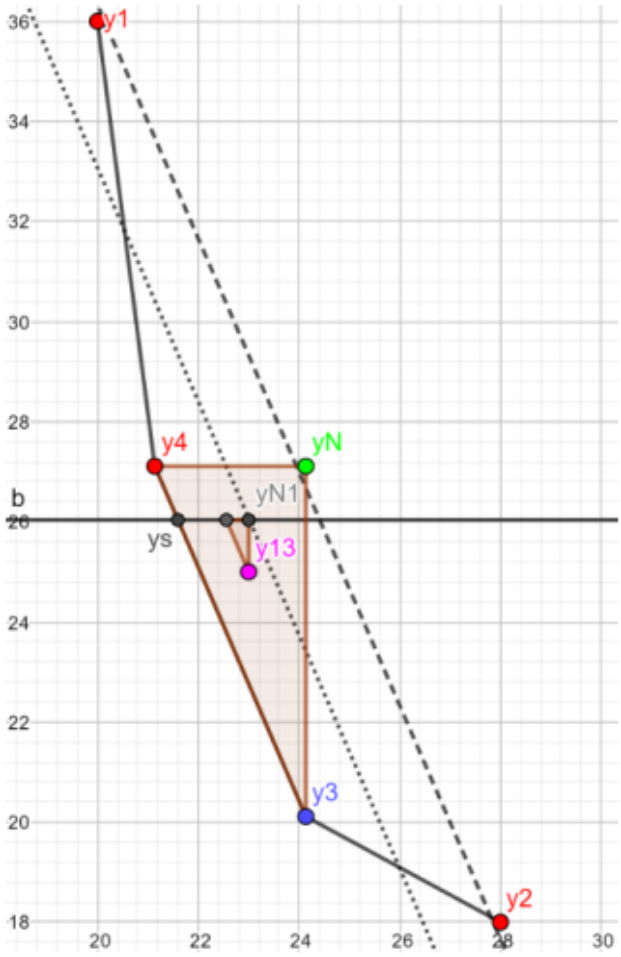


Fig 4.2.11 - L'application de la méthode en deux phases sur l'exemple 4.3.1.

Discussion

La méthode deux-phases présente un schéma de résolution exacte très intéressant car très général et qui ne dépend pas du problème. Son intérêt réside dans une décomposition de l'espace de recherche et l'utilisation de méthodes mono-objectifs pour les différentes résolutions successives (recherche des extrêmes, résolution des agrégations...). Appliquer la méthode deux-phases pour la résolution d'un problème bi-objectif nécessite donc d'avoir une méthode mono-objectif efficace (si possible polynomiale). C'est le cas pour les problèmes traités par Ulungu et Teghem, et c'est ce qui rend leur méthode performante pour ces problèmes là [15].

Conclusion Générale

Dans ce mémoire, nous nous sommes intéressés à la résolution des problèmes d'optimisation combinatoire multi-objectif, pour cela nous traitons le problème d'affectation comme cadre d'application.

Avant de présenter la méthode, nous avons donné quelques rappels sur les notions fondamentales de l'optimisation combinatoire et nous avons présenté les principales notions, propriétés et méthodes de résolution pour l'optimisation combinatoire multi-objectif.

Ensuite nous avons appliqué la méthode sur un exemple numérique, après l'adaptation de la méthode en deux phases à un problème d'affectation avec une contrainte sur les ressources, nous avons repris le même exemple que l'affectation.

Finalement, après l'application de la méthode sur un exemple, nous avons remarqué qu'elle peut être plus rapide que Branch and bound, car cette dernière explore tout l'ensemble de solutions, alors que avec la méthode en deux phases c'est possible de trouver la solution optimale après un nombre réduit d'itérations, car la recherche se fait dans un triangle réduit.

Comme suite au présent travail, nous pouvons proposer un très grand nombre de problèmes à étudier en perspective, pour ne citer que les suivants :

- Problème d'affectation bi-objectif avec contrainte de ressource.

- Problème d'affectation bi-objectif pour des problèmes autre que l'affectation (voyageur de commerce, problème de sac à dos, problème de transport. . .).

- Améliorer les bornes (supérieures).

- Etude comparative avec les résultats donnés par le solveur Polyscip.

- Etc...

Bibliographie

[1] V. Aggarwal. A lagrangean-relaxation method for the constrained assignment problem. *Computers and Operations Research*, 12(1):97 – 106, 1985.

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

[3] I. Alaya. *Optimisation multi-objectif par colonies de fourmis Cas des problèmes de sac à dos*. PhD thesis, Université Claude Bernard Lyon1, 2009.

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

[5] M. Basseur. *Conception D'algorithmes Coopératifs Pour L'optimisation Multiobjectif : Application Aux Problèmes D'ordonnancement De Type Flow-Shop*. PhD thesis, Université de Lille U.F.R. D.I.E.E.A, 2005.

[6] C. Boughani. Cours d'optimisation combinatoire. Technical report, 2015.

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

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

[9] R. Carraway, T. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44:95–104, 1990.

[10] A. Cerqueus. *Bi-objective branch-and-cut algorithms applied to the binary knapsack problem*. PhD thesis, Université de Nantes, 2015.

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

[12] V. Chvatal. *Linear programming*. W. H. Freeman Company, 1983.

[13] C. A. Coello Coello and D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[14] F. Degoutin and X. Gandibleux. Un retour d'expérience sur la résolution de problèmes combinatoires bi-objectifs. In *Programmation Mathématique MultiObjectif (PM2O)*.

[15] C. Dhaenens. *Optimisation Combinatoire Multi-Objectif : Apport des méthodes coopératives et contribution à l'extraction de connaissances*. PhD thesis, Université de Lille, 2005.

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

[17] Y. Dufresne. Pji - algorithmes de recherche locale pour l'optimisation combinatoire multiobjectif. Technical report, 2012.

[18] F. Y. Edgeworth. *Mathematical physics*. P. Keagan, 1881.

[19] A. Geoffrion. proper efficiency and the theory of vector minimization. *Journal of Mathematical Analysis and Applications*, 22:618–630, 1968.

[20] R. Gomory. An algorithm for the mixed integer problem. Technical report, DTIC Document, 1960.

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

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

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

[24] P. Hansen. *Bicriterion path problems*, chapter Multiple criteria decision making theory and application, pages 109–127. Springer, 1980.

[25] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 52:7–21, 2005.

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

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

[28] K. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[29] V. Pareto. Cours d'économie politique. Technical report, 1896.

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

[31] K. Deb A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[32] A. Przybylski. *Méthode en deux phases pour la résolution exacte de problèmes d'optimisation combinatoire comportant plusieurs objectifs : nouveaux développemants et application au problème d'affectation linéaire.* PhD thesis, Université de Nantes, 2006.

[33] A. Przybylski, X. Gandibleux, and M. Ehrgott. Seek and cut algorithm computing minimal and maximal complete efficient solution sets for the biobjective assignment problem. In *In 6th Int. Multi-Objective Programming and Goal Programming conf (MOPGP-04)*, 2004.

[34] E. Sandgren. *Advances in design optimization*, chapter Multicriteria design optimization by goal programming. 1994.

[35] R. E. Steuer. *Multiple criteria optimization : Theory, computation and application.* Wiley, 1986.

[36] J. Teghem. *La recherche opérationnelle Tome 1 : Les méthodes d'optimisation.* Editions Ellipses et Editions de l'ULB, 2012.

[37] D. Tuyttens, J. Teghem, Ph. Fortemps, and K. Van Nieuwenhuyse. Proper efficiency and the theory of vector minimization. *Performance of the mosa method for the bicriteria assignment problem*, 6:295–310, 2000.

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

[39] T. Vincent. *Caracterisation des solutions efficaces et algorithmes d'énumération exacts pour l'optimisation multiobjectif en variables mixtes binaires.* PhD thesis, Lina, 2013.

[40] L. Wolsey. *Integer Programming.* Wiley, 1998.

Résumé

Dans ce travail, nous nous sommes intéressés à la résolution des problèmes d'optimisation combinatoire multi-objectif. Pour cela, nous traitons le cas du problème d'affectation bi-objectif.

La méthode en deux phases est un cadre de résolution générale qui a été proposée par Ulungu et Teghem pour la résolution d'un problème d'affectation bi-objectif.

L'application de cette méthode sur un problème d'affectation avec une contrainte sur les ressources nous a permis de déduire qu'elle peut être plus rapide que la méthode Branch and bound.

Mots-clés: optimisations combinatoires multi-objectifs, méthode en deux phases, problème d'affectation bi-objectif.

Abstract

In this work, we are interested in solving multi-objective combinatorial optimization problems. For this, we treat the case of bi-objective assignment problem.

The two-phase method is a general resolution framework that has been proposed by Ulungu and Teghem for the resolution of a bi-objective assignment problem.

Applying this method to an allocation problem with a resource constraint allowed us to deduce that it can be faster than the Branch and bound method.

Keywords: multi-objective combinatorial optimizations, two-phase method, bi-objective assignment problem.