

UNIVERSITÉ DE M'SILA
FACULTÉ DES MATHÉMATIQUES ET DE L'INFORMATIQUE
DEPARTEMENT DES MATHÉMATIQUES

Mémoire

Présenté pour l'obtention du diplôme de **Master**

Domaine: Mathématiques et Informatiques

Filière: Mathématiques

Option: Mathématiques Appliquées et discrètes

Par

Ouemlkheir DAIL

THÈME

SUR LES
Application de séparation et évaluation pour résoudre un problème d'optimisation
combinatoire

Soutenu le : 04/06./ 2016

Devant le jury composé de :

Mr. LamicheChaabane MCA. Université de M'sila Président

Mr. BelouadahHocine Prof. Université de M'sila Rapporteur

Dr. BounabNoura MCB. Université de M'sil Examinatrice

Dirigé par:

***Mr.* Hocine BELOUADAH**

Année: **2015/2016**

Remerciements

Je tiens à remercier, en premier lieu, **Allah** qui m'a donné la force de rédiger ce modeste travail.

Je tiens à remercier les membres du jury, qui ont accepté d'évaluer mon travail de mémoire.

Je tiens à remercier Dr.**Hocine BELOUADAH** directeur de mon mémoire, pour sa disponibilité et ses conseils judicieux tout au long de ce travail.

Je tiens à exprimer tout mes respects à mes parents, mes frères et Nasri Abderraouf ma soeur qui m'ont toujours encouragé .

Je remercie tous les professeurs du département de Mathématiques, sans oublier aussi mes collègues et amies, ainsi tous ceux qui ont participé de loin ou de près à l'élaboration de ce mémoire.

Dédicaces

Je dédie ce modeste travail :

-A mes parents ma mère et mon père.

- A mes soeurs

-A mes frères.

-A toute la famille.

-A toute mes amies.

- Je tiens à remercier l'ensemble de tous les étudiants et étudiantes de ma promotion,

En fin je dédie ce mémoire à mes collègues et tous ceux qui me sont chers.

NOTATIONS

- \mathcal{D} Domaine des solutionnes (La plupart \mathbb{N}).
- \mathcal{D}_{PL} Domaine des solutions de problème linéaire.
- \mathcal{D}_{RLP} Domaine des solutions de problème relaxation linéaire.
- N_A Liste des nœuds actifs.
- \mathcal{P} La classe \mathcal{P} Polynomial time.
- \mathcal{NP} Non-déterministe Polynomial time.
- POC Problème d'optimisation combinatoire.
- BB Branch and Bound.
- PLNE Programmes linéaires en nombres entiers.
- RLP Relaxation linéaire.
- P.S.À.D Le problème de sac à dos.
- P.V.C Le problème du voyageur de commerce.
- $|S|$ Cardinale d'ensemble des solutions.
- $N(t)$ Nombre des tournes

Table des figures

Figure 1	Classification des problèmes d'optimisation combinatoire.	Page 6
Figure 2	Exemple sac à dos.	Page 7
Figure 3	Arbre énumérant l'ensemble des permutations de 4 éléments.	Page 7
Figure 4	Classification des méthodes de résolution.	Page 10
Figure 5	Découpage d'ensemble dans une procédure des séparations et évaluations.	Page 11
Figure 6	La stratégie du séparation.	Page 15
Figure 7	La stratégie d'évaluation	Page 16
Figure 8	Organigramme de l'algorithme de Branch & Bound	Page 20
Figure 9	Organigramme de l'algorithme du simplexe	Page 23
Figure 10	Résolution graphique de (P)	Page 25
Figure 11	Résolution graphique et séparation (1) de (P)	Page 26
Figure 12	Résolution graphique et séparation (2) de (P)	Page 27
Figure 13	La séparation (1) sur simplexe.	Page 30
Figure 14	La séparation (2) sur simplexe.	Page 32
Figure 15	Arbre de recherche associé à la résolution par hxostive.	Page 37
Figure 16	Arbre de recherche associé à la résolution par branch and bound.	Page 37
Figure 17	Schéma récapitulatifs des séparations et évaluations.	Page 39
Figure 18	Le nœud initial de l'arbre de recherche.	Page 42
Figure 19	La séparation de nœud initial.	Page 43
Figure 20	Arbre de recherche et séparation N°(1).	Page 43
Figure 21	L'évaluation du séparation.	Page 45
Figure 22	Arbre de recherche et séparation N°(2).	Page 45
Figure 23	Arbre de recherche et séparation N°(3).	Page 46
Figure 24	Arbre de recherche et séparation N°(3).	Page 46
Figure 25	L'évaluation du séparation N°(3).	Page 47
Figure 26	Arbre de recherche et séparation N°(4).	Page 48
Figure 27	L'évaluation du séparation N°(4).	Page 49
Figure 28	Arbre de recherche finale.	Page 49

Table des matières

Introduction générale	1
1 Problème d'optimisation combinatoire et méthode de résolution exacte	2
1.1 Introduction	2
1.2 Le problème d'optimisation	2
1.2.1 Problème d'optimisation discrète	3
1.2.2 Problème d'optimisation combinatoire (POC)	3
1.3 Complexité théorique d'un problème	4
1.3.1 Problème facile et difficile.	5
1.4 Quelques problèmes d'optimisation combinatoire	6
1.4.1 Problème linéaire en nombres entiers (PLNE)	6
1.4.2 Le problème de sac à dos (P.S.À.D)	7
1.4.3 Le problème d'ordonnancement à une seule machine	8
1.4.4 Le problème du voyageur de commerce (P.V.C)	8
1.5 Résolution d'un problème d'optimisation combinatoire	9
1.5.1 Les méthodes exactes de résolution d'optimisation combinatoire . . .	10
1.6 Quelques notions de base	11
1.6.1 Relaxation et relaxation linéaire (RL)	11
1.6.2 Une solution réalisable	12
1.6.3 Une solution réalisable optimale	12
1.6.4 Un graphe	12
1.7 Conclusion	13

2	La méthode de séparation et évaluation et l'algorithme du simplexe	14
2.1	Introduction	14
2.2	La méthode de séparation et évaluation (B&B)	14
2.2.1	Trois principes fondamentaux	15
2.2.2	Organigramme de l'algorithme de Branch & Bound	20
2.2.3	L'algorithme	20
2.3	La méthode du simplexe	20
2.3.1	Principe de la méthode du simplexe :	21
2.3.2	Organigramme de l'algorithme du simplex	23
2.4	Conclusion	23
3	Application de la méthode de séparation et évaluation sur quelques problèmes	24
3.1	Introduction	24
3.2	Le problème linéaire en nombres entiers	24
3.2.1	Méthode algébrique (La méthode du simplexe)	28
3.3	Le problème du sac à dos	33
3.4	Le problème d'ordonnancement sur une seule machine	37
3.5	Le problème de voyageur de commerce (L'algorithme de little)	39
3.5.1	L'algorithme de Little	39
3.5.2	Le problème de voyageur de commerce	39
3.6	Conclusion	49
	Conclusion générale	50

Introduction générale

- Les méthodes arborescentes (La séparation et évaluation (nommée Branch and Bound en anglais) est notée (B&B)) sont des méthodes exactes d'optimisation qui pratiquent une énumération intelligente de l'espace des solutions. Il s'agit d'énumération complète améliorée. Elles partagent l'espace des solutions en sous ensembles de plus en plus petits, la plupart étant éliminés par des calculs de borne savant d'être construits explicitement. Appliquées à des problèmes NP-difficiles, ces méthodes restent bien sur exponentielles, mais leur complexité en moyenne est bien plus faible que pour une énumération complète. Elles peuvent pallier le manque d'algorithmes polynomiaux pour des problèmes de taille moyenne. Pour des problèmes difficiles de grande taille, leur durée d'exécution est très grande et il faut se retourner vers des heuristiques ou solutions approchées.

Pour un problèmes d'optimisation combinatoire, on peut inventer plusieurs méthodes arborescentes. Cependant elles auront trois composantes communes :

- * Une règle de séparation de l'ensemble des solutions.
- * Une fonction d'évaluation des ensembles de solutions.
- * Une stratégie d'exploration.

Le premier exemple développé de procédure par séparation est l'algorithme proposé en 1960 par LAND et DOIG pour les programmes mixtes.

En 1963, Little et al. Utilisent une PS pour résoudre le problème du voyageur de commerce et eut un grand retentissement. Il sera exposé dans la suite, dans le paragraphe des exemples.

On explore l'ensemble des solutions réalisables \mathcal{D} en de sous ensembles de plus en plus petits de façon à isoler une solution optimale dans l'un de ses sous ensembles.

Pour représenter cette exploration, on construit une arborescence dont le sommet de base correspond à l'ensemble \mathcal{D} et dont les autres sommets correspondent à des sous ensembles de \mathcal{D} . Pour définir l'exploration, nous devons définir une règle de séparation et une règle pour se déplacer dans l'arborescence. Il faut définir pour chaque sous ensemble, un minorant appelé "évaluation par défaut". Cela impose de savoir calculer rapidement une "bonne" évaluation par défaut. Une itération consiste à choisir une feuille de l'arborescence, nœuds ou sommets non encore séparés. Quand un sommet ne peut fournir de meilleures solutions, il est dit tué ou élagué. La recherche se termine quand tous les nœuds ont été tués. Les sommets sont numérotés dans l'ordre de leur création.

Toute technique de séparation est valable, à condition qu'aucune solution ne soit perdue. La fonction d'évaluation empêche en pratique que l'arborescence soit construite en entier.[1]

- C'est en fait de montrer la flexibilité de la méthode de branch and bound pour son application : C'est comment elle est appliqué au problème linéaire à valeurs entières, le problème de sac à dos, le problème d'ordonnancement sur une machine et enfin le problème de voyageur de commerce.
- On s'intéresse dans ce mémoire au méthode d'optimisation combinatoire et à l'une des méthodes les plus puissantes pour déterminer une solution optimale. En fait on considère quatre problèmes d'optimisation combinatoire à savoir : la programmation linéaire à valeur entières, le problème de sac à dos, le problème d'ordonnancement et le problème de voyageur de commerce. Le plan de ce mémoire est constitué de trois chapitres :

Le premier chapitre : Concerne la définition des problèmes combinatoires et leurs classifications selon leurs difficultés. Ainsi que la méthode de séparation et évaluation.

Le deuxième chapitre : est consacré à l'étude de la méthode de séparation et évaluation et une autre méthode de solution d'une programme linéaire à savoir la méthode de simplexe.

Le troisième chapitre : montre comment on adapte l'utilisation de la méthode de séparation et évaluation aux problèmes d'optimisation cités avant.

Finalement, On conclu le manuscrit par une conclusion général.

Chapitre 1

Problème d'optimisation combinatoire et méthode de résolution exacte

1.1 Introduction

Dans ce chapitre on considère le problème d'optimisation combinatoire et ses méthodes exactes de solution à savoir la programmation dynamique et la méthode séparation et évaluation. On va essayer de définir le problème d'optimisation combinatoire et d'introduire la théorie de complexité afin de pouvoir les classer selon leurs degrés de difficultés. Ensuite on va donner quelques exemples de fameux problèmes d'optimisation combinatoire. Après on a introduit les deux méthodes exactes de solution.

1.2 Le problème d'optimisation

L'optimisation est une branche des mathématiques et de l'informatique en tant que disciplines, cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à déterminer quelles sont les solutions satisfaisant un objectif quantitatif tout en respectant d'éventuelles contraintes.

L'optimisation joue un rôle important en recherche opérationnelle (domaine à la frontière entre l'informatique, les mathématiques et l'économie), dans les mathématiques appliquées (fondamentales pour l'industrie et l'ingénierie), en analyse et en analyse numérique, en statistique pour l'estimation du maximum de vraisemblance d'une distribution, pour la recherche de stratégies dans le cadre de la théorie des jeux,

ou encore en théorie du contrôle et de la commande.

Aujourd'hui, tous les systèmes susceptibles d'être décrits par un modèle mathématique sont optimisés. La qualité des résultats et des prédictions dépend de la pertinence du modèle, de l'efficacité de l'algorithme et des moyens pour le traitement numérique.

1.2.1 Problème d'optimisation discrète

- Les problèmes d'optimisation discrète, par opposition à l'optimisation continue, forment une classe de problèmes d'optimisation particulièrement étudiée. Toutes ou partie des variables de ce type de problèmes appartiennent à l'ensemble des entiers.
- Dans sa forme la plus générale, un problème d'optimisation combinatoire (on dit aussi d'optimisation discrète) consiste à trouver dans un ensemble discret un parmi les meilleurs sous-ensembles (ou solutions) réalisables, la notion de meilleure solution étant définie par une fonction objectif. Formellement, étant donnés :

- Un ensemble discret N .
- Une fonction **d'ensemble** $f : 2^N \rightarrow \mathbb{R}$, dite fonction objectif.
- Un ensemble fini R de sous-ensembles de N , dont les éléments sont appelés les solutions réalisables, un problème d'optimisation combinatoire consiste à déterminer

$$\max_{S \subseteq N} \{f(S) : S \in R\} \quad [2]$$

1.2.2 Problème d'optimisation combinatoire (POC)

Un modèle d'optimisation combinatoire consiste à affecter des valeurs aux variables de décision qui réalisent l'optimum (minimum ou maximum) de la fonction objectif parmi

l'ensemble de toutes les valeurs pour les variables de décision qui satisfont les contraintes données.

1. Une (ou des) fonction (s) objectif.
2. Des variables de décision.
3. Des contraintes.

Un problème d'optimisation combinatoire (minimisation resp maximisation) est défini par la donnée :

- D'une fonction $f : S \rightarrow R$.
- D'un ensemble discret S , tq $|S|$ dénombrable ou bien fini.
- Et il s'agit de déterminer, s'il en existe un élément x^* de S tel que :

$$\left\{ f(x^*) \leq f(x), \left(\forall x \in S, f(x^*) = \min_{x \in S} f(x) \right) \right\} \\ \text{resp} \left\{ f(x^*) \geq f(x), \left(\forall x \in S, f(x^*) = \max_{x \in S} f(x) \right) \right\} \quad [3]$$

Proposition 1.2.1 *Un problème de maximisation d'une fonction f est équivalent au problème de minimisation de $(-f)$.*

$$\max_{x \in S} f(x) = - \min_{x \in S} (-f(x))$$

1.3 Complexité théorique d'un problème

La complexité d'un problème une estimation du nombre d'instructions à exécuter pour résoudre les instances de ce problème, cette estimation étant un ordre de grandeur par rapport à la taille de l'instance. la complexité de leur résolution [Papadimitriou, 1994].

Ordre de complexité : Une fonction $f(n)$ est $O(g(n))$, ($f(n)$ est de complexité $g(n)$), s'il existe un réel $c > 0$ et un entier positif n_0 tel que pour tout $n \geq n_0$ on a $|f(n)| \leq c.g(n)$ tq $f : \mathbb{N} \rightarrow \mathbb{N}$ et $g : \mathbb{N} \rightarrow \mathbb{N}$. [4]

1.3.1 Problème facile et difficile.

Tous les algorithmes étudiés étaient des algorithmes à temps polynomial : leurs temps d'exécution au pire des cas est $O(k^n)$ où n est la taille de l'entrée et k est une constante. Mais tous les problèmes ne peuvent être résolus en un temps polynomial. On peut donc distinguer deux types de problèmes :

1. **Problèmes faciles** : Tout problème possédant une solution de complexité polynomiale est considéré "facile".
2. **Problèmes difficiles** : Une solution exponentielle, ou pire qu'exponentielle, est associée à un problème "difficile".

► \mathcal{P} est la classe de tous les problèmes de décision qui peuvent être résolus par un algorithme polynomial.

► **\mathcal{P} -Complet** : Un problème décisionnel appartient à ce classement s'il fait partie de la classe \mathcal{P} et si tout problème \mathcal{P} dans \mathcal{P} peut s'y réduire en temps poly-logarithmique en utilisant un ordinateur avec un nombre polynomial de processeurs.

► Un problème est \mathcal{NP} est \mathcal{NP} -Complet si tout problème \mathcal{NP} s'y réduit en temps polynomial. Autrement, les problèmes \mathcal{NP} les plus difficiles sont \mathcal{NP} -Complets.

Proposition 1.3.1 *Si un seul problème \mathcal{NP} -Complet peut être résolu en un temps polynomial, alors tous les problèmes de \mathcal{NP} peuvent être résolus en un temps polynomial.*

Un problème \mathcal{NP} -Complet est un problème dans \mathcal{NP} au moins aussi difficile que tout autre problème de \mathcal{NP} .

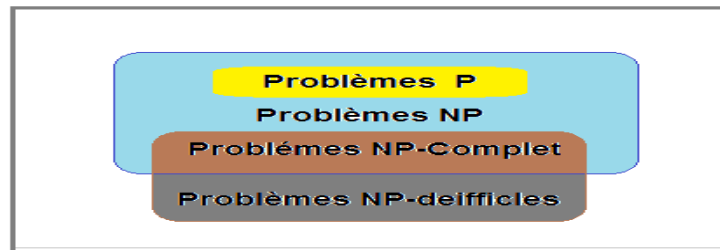
- *Il existe une réduction polynomiale qui permet de décrire n'importe quelle instance de n'importe quel problème de \mathcal{NP} comme une instance de \mathcal{NP} -Complet.*

- *Si on sait résoudre toutes les instances d'un problème \mathcal{NP} -Complet on sait résoudre toutes les instances de tous les problèmes \mathcal{NP} .*

- *Si un problème \mathcal{NP} -Complet est dans la classe \mathcal{P} (\mathcal{NP} -Complet $\cap \mathcal{P} \neq \emptyset$), alors $\mathcal{P} = \mathcal{NP}$. [5]*

Exemple 1.3.1 *Un problème est \mathcal{NP} -difficile si tout problème s'y réduit en temps polynomial.*

Remarque 1.3.1 On a en générale $\mathcal{P} \subsetneq \mathcal{NP}$ mais est ce que $\mathcal{P} = \mathcal{NP}$?



Classification des problèmes d'optimisation combinatoire.

1.4 Quelques problèmes d'optimisation combinatoire

1.4.1 Problème linéaire en nombres entiers (PLNE)

Les programmes linéaires ainsi obtenus sont appelés programmes linéaires en nombres entiers (PLNE). Ils sont souvent difficiles à résoudre.

* La programmation linéaire : La fonction objectif et les contraintes sont linéaires.

$$(PLNE) \begin{cases} \max z = c^T x \\ A.x \leq b \\ x_i \in \mathbb{N}, i = 1 \dots n \end{cases}$$

telle que $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, et $A(m \times n)$

Exemple 1.4.1

$$(PLNE) \begin{cases} \max Z = x_1 + x_2 \\ s.c. 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_i \geq 0, x_i \text{ sont des nombres entiers} \end{cases}$$

1.4.2 Le problème de sac à dos (P.S.À.D)

Le problème du sac à dos consiste à remplir un sac à dos de capacité C avec des produits P_1, P_2, \dots, P_n , qui prennent une place v_1, v_2, \dots, v_n et rapportent a_1, a_2, \dots, a_n . Par unité, de façon à maximiser le bénéfice. On suppose ici que l'on a autant d'unités de chaque produit que l'on veut.

Exemple 1.4.2 Données un sac à dos de poids 15 kg, 5 objets ayant chacun :

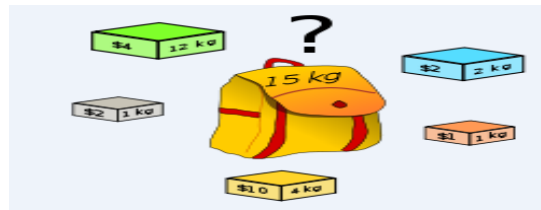


Figure (2) : Exemple sac à dos

Objectif : quelles objets choisir afin de maximiser la valeur emporté ne dépassant pas les 15 kg ?

Ce problème peut s'écrire comme un programme linéaire en nombres entiers. Si on note x_i le nombre d'unités du produit P_i à mettre dans le sac à dos, il s'agit de calculer le bénéfice maximal.

$$\max_{x_i \geq 0} \sum_{i=1}^n p_i x_i \text{ sous la contrainte } \sum_{i=1}^n v_i x_i \leq c. \quad [6]$$

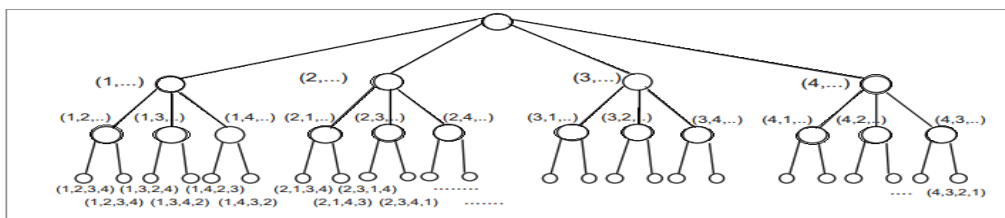


Figure (3) : Arbre énumérant l'ensemble des permutations de 4 éléments.

Exemple 1.4.3 Prenons l'exemple numérique suivant avec 4 produits : de poids respectifs : 2, 5, 10, et 5 et de valeurs respectives : 40, 30, 50, et 10).

Le poids maximal autorisé est 16.

$$(PLNE) \begin{cases} \text{Max } z = 40x_1 + 30x_2 + 50x_3 + 10x_4 \\ 2x_1 + 5x_2 + 10x_3 + 5x_4 \leq 16 \\ x_i \in \mathbb{N} \end{cases}$$

1.4.3 Le problème d'ordonnancement à une seule machine

L'étude du problème à une machine apparaît lorsque chaque tâche requiert une ressource dont la disponibilité est égale à l'unité. On doit ordonnancer n tâches, en une durée minimale, en respectant les conditions suivantes : une tâche i est disponible à la date r_i , a une durée P_i et un laps de temps d_i doit s'écouler entre la fin de la tâche i et l'achèvement de l'ordonnancement. Le problème est NP-difficile.

1.4.4 Le problème du voyageur de commerce (P.V.C)

Un voyageur de commerce doit visiter un ensemble de n villes, chacune une et seule fois, et retourner à sa ville de départ. Sachant que la distance entre la ville i et la ville j est c_{ij} il s'agit de déterminer une telle tournée de longueur minimal. Le problème du vc est NP-difficile

Définition des variables $x_{ij} = 1$ si la ville j est visitée immédiatement après la ville i , et $x_{ij} = 0$ sinon (les variables x_{ii} ne sont pas définies).

Définition des contraintes

- Le voyageur quitte la ville i une fois $\sum_{j=1}^n x_{ij} = 1, i = 1 \dots n$.
- Il arrive en chaque ville j une fois $\sum_{i=1}^n x_{ij} = 1, j = 1 \dots n$.
- Les variables sont en (0 ou 1), ie $x_{ij} \in \{0,1\}$.

• Un chemin t dans le graphe est représenté par une suite d'arêtes, et son coût est défini par :

$$f(t) = \min \sum_{(i,j) \in t} c_{ij} x_{ij}.$$

Remarque 1.4.1 Nombre de chemins t le cardinal de $N(t) = \frac{1}{2}(n-1)!$.

Exemple 1.4.4 On a un problème de voyageur de commerce de $n = 6$ villes.

N° du ville	1	2	3	4	5	6
Nom de ville	B	L	N	P	M	D

► La matrice de coûts est symétrique

$$(M) = \begin{pmatrix} & B & L & N & P & M & D \\ B & \infty & 780 & 320 & 580 & 480 & 660 \\ L & 780 & \infty & 700 & 460 & 300 & 200 \\ N & 320 & 700 & \infty & 380 & 820 & 630 \\ P & 580 & 460 & 380 & \infty & 750 & 310 \\ M & 480 & 300 & 820 & 750 & \infty & 500 \\ D & 660 & 200 & 630 & 310 & 500 & \infty \end{pmatrix}$$

1.5 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 S .
 - L'expression de l'objectif à optimiser $f(s)$.
 - 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.

Remarque 1.5.1 Dans la réalité, nous trouvons une solution à un modèle du problème. Tous les modèles sont des simplifications de la réalité

Problème → Modèle → Solution

1.5.1 Les méthodes exactes de résolution d'optimisation combinatoire

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, dites méthodes de la programmation dynamique et séparation et évaluation (**B&B**). [7]

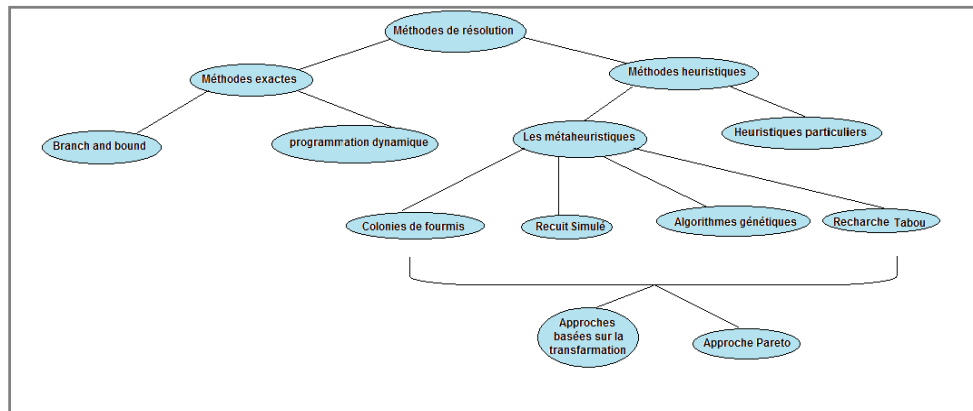


figure (4) : Classification des méthodes de résolution

La méthode de programmation dynamique.

- * La programmation dynamique a été appelée comme cela depuis 1940 par Richard Bellman et permet d'appréhender un problème de façon différente de celle que l'on pourrait imaginer au premier abord. Le concept de base est simple : une solution optimale est la somme de sous-problèmes résolus de façon optimale. Il faut donc diviser un problème donné en sous-problèmes.

La méthode par séparation et évaluation (B&B)

La séparation et évaluation est une technique qui effectue un parcours en profondeur de l'arbre de recherche et de fournir une ou plusieurs solutions optimales à partir d'un ensemble de solutions potentielles. Or chaque étape de la recherche, correspondant à un nœud 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 nœud. En début de résolution.

- ♣ Le principe d'une méthode arborescente est explorer l'ensemble des solutions réalisables en le scindant en des sous-ensembles de plus en plus petits de façon à isoler une solution optimale. Pour définir une telle procédure, nous devons élaborer une règle de séparation qui permettra de construire l'arborescence, choisir une règle de parcours de l'arborescence et préciser les évaluations que nous utilisons. Notre séparation consiste à remplacer l'intervalle d'exécution possible d'une tâche par deux intervalles recouvrant l'intervalle initial.
- ♣ On recommence la séparation en sous-ensembles avec chaque sous ensemble et ainsi de suite jusqu'à ce que tous les ensembles ne contiennent plus qu'un seul élément. Cette énumération peut se représenter par un arbre comme dans la figure (5) où la racine de l'arbre représente N_0 , ses fils représentent les sous-ensembles créés dans la première partition de N_1 , et ainsi de suite. [6]

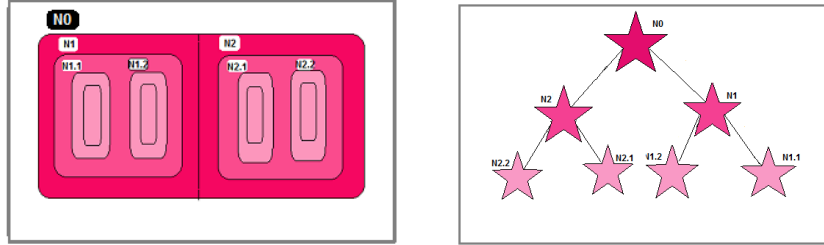


figure (5) : Découpage d'ensemble dans une procédure de séparation et évaluation

Remarque 1.5.2 *Le nombre de nœuds d'un arbre complet est de la forme $2^h = 2^{h+1} + 1$ tq h : Le nombre des niveaux de profondeur.*

1.6 Quelques notions de base

1.6.1 Relaxation et relaxation linéaire (RL)

Déterminer un problème relaxé peut s'avérer difficile en pratique. Cependant, dans le cas du problème en nombres entiers.

$$\begin{array}{ccc}
 (PLEN) \left\{ \begin{array}{l} \max z = c^T x \\ A.x \leq b \\ x_i \in \mathbb{N}, i = 1 \dots n \end{array} \right. & \begin{array}{c} relaxation \\ \Rightarrow \\ D_{PL} \subseteq D_{RPL} \end{array} & (RPL) \left\{ \begin{array}{l} \max z = c^T x \\ A.x \leq b \\ x_i \geq 0 \end{array} \right. \quad \text{et on a}
 \end{array}$$

Le problème linéaire associé où les contraintes d'intégralité ne sont pas incluses remplit cet objectif. Un tel problème est alors dit "relaxation linéaire" ou "relaxation continue", et on montre qu'il s'agit bien d'une relaxation, puisque le domaine réalisable du problème linéaire contient celui du problème en nombres entiers.

1.6.2 Une solution réalisable

Une **solution réalisable** est une affectation de valeurs aux variables telle que toutes les contraintes sont satisfaites. La valeur de la fonction objective d'une solution réalisable est obtenue en évaluant la fonction objective au point donné. Un problème est dit **problème faisable** est un problème qui possède au moins une solution réalisable. [3]

1.6.3 Une solution réalisable optimale

Une solution optimale minimisation (resp maximisation) est une solution réalisable dont la valeur de la fonction objective est inférieure (resp supérieure) ou égale à celle de toute autre solution réalisable.[3]

1.6.4 Un graphe

- * **Un graphe orienté** : Est un couple $G = (X, U)$, où X est un ensemble dont les éléments sont appelés sommets et U une partie de $X \times X$ dont les éléments sont appelés arcs.
- * **Un graphe value (réseau)** : Est un triplet $G = (X, U, V)$ où (X, U) est un graphe et V une application de U dans \mathbb{R} (ensemble des réels).
- * **Un arbre** : Un arbre est un graphe connexe sans cycle (avec $|X| \geq 2$) [8].

1.7 Conclusion

On a introduit ici une certaine classe de problèmes des mathématiques discrètes à savoir celle d'optimisation combinatoire on a choisit spécifiquement quatre problèmes NP difficiles. On a aussi introduit une méthode de solution exacte qui est l'algorithme par séparation et évaluation. Dans le chapitre suivant on essaye de voir comment cette méthode à été utilisé pour résoudre chacun de ces problèmes.

Chapitre 2

La méthode de séparation et évaluation et l'algorithme du simplexe

2.1 Introduction

Dans ce chapitre on considère la méthode de séparation et évaluation qui est une méthode très puissante pour résoudre le problème d'optimisation combinatoire avec taille modérée. Aussi une méthode très célèbre (Le simplexe) qui sert à résoudre un programme linéaire à valeurs réelles a été rigoureusement introduite vu son importance pour déterminer une évaluation par défaut (ou excès) par la valeur de la fonction objective d'un problème linéaire à valeurs entières.

2.2 La méthode de séparation et évaluation (B&B)

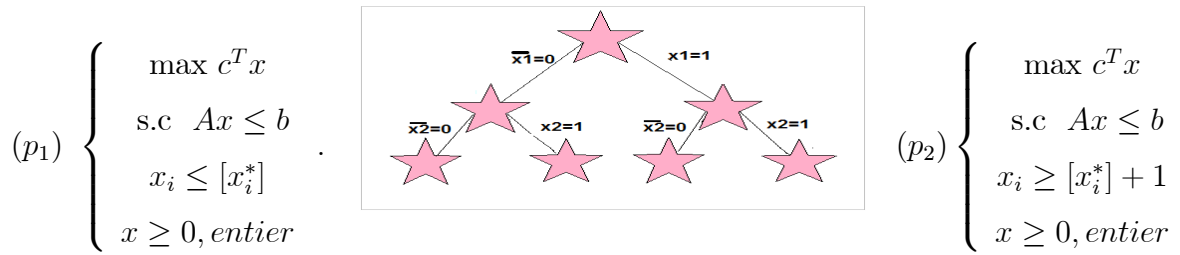
Les méthodes arborescentes (Méthode de séparation et évaluation (B&B)) sont des méthodes exactes d'optimisation qui pratiquent une énumération intelligente de l'espace des solutions. Il s'agit d'énumérations complètes améliorées. Elles partagent l'espace des solutions en sous-ensembles de plus en plus petits, la plupart étant éliminés par des calculs de bornes avant d'être construits explicitement. Appliquées à des problèmes NP-difficiles, ces méthodes restent bien sur exponentielles, mais leur complexité en moyenne est bien plus faible que pour un Branch P_1 si nous passons par (x, y) énumération complète. Elles peuvent pallier

le manque d'algorithmes polynomiaux pour des problèmes de taille moyenne. Pour des problèmes difficiles de grande taille, leur durée d'exécution est très grande et il faut se retourner vers des heuristiques ou solutions approchées.

2.2.1 Trois principes fondamentaux

La séparation(Branchement)

- La séparation (Branch) consiste à diviser le problème en sous-problèmes. Ainsi, en résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Cela revient à construire un arbre permettant d'énumérer toutes les solutions. L'ensemble de nœuds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des nœuds actifs.
- La séparation (Branch) consiste à séparer un ensemble de solutions en sous-ensembles
- Pour d'écrire l'opération de séparation, il suffit de dire comment on divise un ensemble de Solutions en sous-ensembles. Cela revient à d'écrire comment construire l'arbre permettant d'énumérer toutes Les solutions. L'ensemble de nœuds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de Contenir une solution optimale, c'est -à-dire encore à diviser, est appelé ensemble des nœuds actifs [6].



Figure(6) : La stratégie du séparation

L'évaluation

- L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale. L'objectif est d'essayer d'évaluer

l'intérêt de l'exploration d'un sous-ensemble de l'arborescence. La séparation et évaluation (B&B) utilise une élimination de branches dans l'arborescence de recherche de la manière suivante : la recherche d'une solution de coût minimal, consiste à mémoriser la solution de plus bas coût rencontré pendant l'exploration, et à comparer le coût de chaque nœud parcouru à celui de la meilleure solution. Si le coût du nœud considéré est supérieur au meilleur coût, on arrête l'exploration de la branche et toutes les solutions de cette branche seront nécessairement de coût plus élevé que la meilleure solution déjà trouvée. Consiste à évaluer les solutions d'un sous-ensemble de façon optimiste, c'est-à-dire en majorant la valeur de la meilleure solution de ce sous-ensemble.

- L'algorithme propose de parcourir l'arborescence des solutions possibles en évaluant chaque sous-ensemble de solutions de façon optimiste. Lors de ce parcours, il maintient la valeur M de la meilleure solution trouvée jusqu' à présent. Quand l'évaluation d'un sous-ensemble donne une valeur plus faible que M , il est inutile d'explorer plus loin ce sous-ensemble. Les paragraphes suivants d'entailent les différentes opérations de l'algorithme [6].



Figure (7) : La stratégies d'évaluation .

Évaluation optimiste

- Etant donné l'arbre énumérant toutes les solutions, chaque feuille contient une Solution dont on peut calculer la valeur exacte. Pour un nœud interne de l'arbre k on va évaluer ce nœud en calculant un majorant de la valeur de toutes les solutions contenues dans le sous-ensemble représenté par le sous arbre de racine k .

Si l'arbre entier était connu, on pourrait évaluer un nœud par la meilleure solution portée par ses feuilles. Mais ce n'est bien sûr pas le cas!. Il faut donc essayer d'estimer par majoration la meilleure solution qu'il est possible d'atteindre à partir du nœud. Comme un nœud interne représente une solution partielle (dont une partie des variables du problème est (partiellement) fixée), on calcule sa valeur en cherchant la meilleure valeur qu'on peut obtenir grâce aux degrés de liberté restants.

- Cette fonction d'évaluation, spécifique à chaque problème, est dite optimiste car elle calcule un majorant du meilleur résultat possible à partir d'une solution partielle. Une bonne fonction doit majorer au plus près la solution maximale, tout en restant le moins coûteuse possible d'un point de vue algorithmique. C'est un des aspects cruciaux quant aux performances de la résolution du problème.[6]

Elaguage : Une fois que la valeur d'un nœud interne est calculée, on peut utiliser cette valeur pour interrompre éventuellement l'exploration de cette partie de l'arbre. En particulier, il est inutile de diviser le nœud dans les cas suivants.

1. L'évaluation a permis de calculer une solution qui a exactement cette valeur. Cette solution est nécessairement optimale dans ce sous-ensemble de solutions. Si cette solution est la meilleure trouvée jusque-là, elle devient la meilleure solution courante. Ce cas est plutôt rare.
2. L'évaluation est inférieure ou égale à la valeur de la meilleure solution trouvée jusque-là. On a donc aucune chance de trouver mieux dans ce sous-ensemble. Ceci peut permettre des gains importants, car on élimine une partie de l'arbre de recherche.
3. Le sous-ensemble est réduit à un seul élément.

Dans les cas 1 et 2, on gagne dans l'exploration de l'arbre puisque la branche suivant le nœud considéré ne sera pas explorée. On dit que cette branche est élaguée (pruning en anglais). A noter que dans le cas 1, si la meilleure solution courante a changé, il convient de parcourir tous les nœuds actifs pour voir s'ils le restent [6].

Stratégies de parcours

En fonction de la structure de données utilisée pour la liste des nœuds actifs N_A , l'algorithme peut avoir des performances expérimentales très différentes. La façon dont on parcourt l'arbre des solutions et donc le choix du prochain nœud actif à diviser sont cruciaux. Plusieurs stratégies sont à envisager[6].

La largeur d'abord

- Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées.
- Stratégie en largeur : on divise les nœuds dans l'ordre de leur création [6].

La profondeur d'abord

- Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.
- Stratégie de recherche en profondeur : on choisit pour prochain nœud actif l'un des fils du nœud qui vient d'être divisé. Si aucun de ces nœuds n'est actif on revient en arrière (backtrack) dans l'arbre [6].

Le meilleur d'abord

- Cette stratégie consiste à explorer des sous problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.
- Stratégie de la meilleure évaluation : on divise le nœud de meilleure évaluation [6].

Stratégie du plus prioritaire

- la priorité d'un nœud peut être évaluée par pondération entre son évaluation et sa profondeur dans l'arbre (en fait sa distance à une feuille). En effet, les évaluations loin des feuilles sont souvent plus intéressantes mais moins fiables et il peut être avantageux de diviser un nœud potentiellement moins bon mais plus profond dans l'arbre.

Stratégie mixte

- On va en profondeur tant qu'on le peut, mais quand on ne peut plus on saute au nœud de meilleure évaluation. On explorera les fils dans l'ordre d'évaluation : le meilleur d'abord [6].

2.2.2 Organigramme de l'algorithme de Branch & Bound

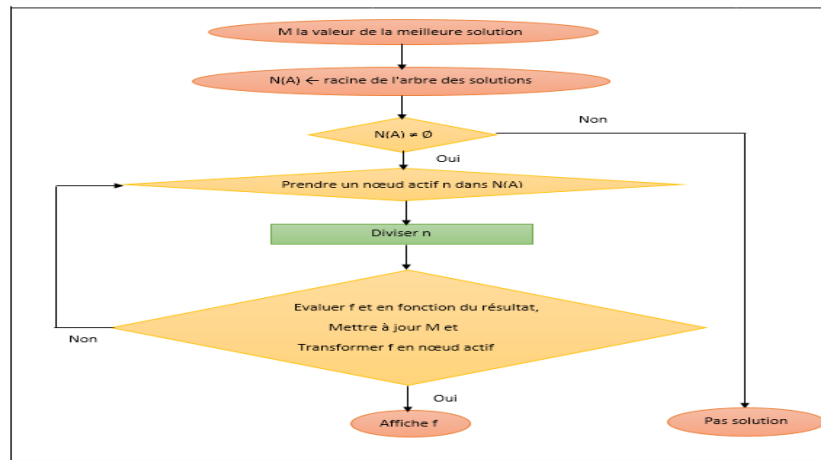


Figure (8) : Organigramme de l'algorithme de Branch & Bound

2.2.3 L'algorithme

L'algorithme maintient la valeur M de la meilleure solution trouvée

Jusqu' à présent et la liste N_A des nœuds actifs, susceptibles de contenir de meilleures solutions que M .

$N_A \leftarrow \{\text{racine de l'arbre des solutions}\}$

Tant que $N_A \neq \emptyset$ faire

Prendre un nœud actif n dans N_A ;

Diviser n ;

Pour chaque fils f de n

Évaluer f et en fonction du résultat, mettre à jour M et

transformer f en nœud actif (le mettre dans N_A) ou l'élaguer;

Fin pour

Fin tant que

2.3 La méthode du simplexe

Le programmes linéaire en nombres entiers :

$$(PLNE) \begin{cases} \max z = c^T x \\ A.x \leq b \\ x_i \in \mathbb{N}, i = 1 \dots n \end{cases}$$

tellque $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, et $A(m \times n)$.

► **PL** sous forme standard

$$\begin{cases} f(x) = \max_{x \in \mathbb{R}^n} c^T x. \\ Ax = b \\ x \geq 0 \end{cases}$$

On dispose d'une base B et d'une solution de base réalisable \underline{x} avec (à une permutation près des colonnes de A).

$$A = (A_B \mid A_H) \text{ et } \underline{x} = \begin{pmatrix} \underline{x}_B \\ \underline{x}_h \end{pmatrix}$$

Où A_B matrice $m \times m$, inversible (variables de base).

A_H matrice $m \times (n-m)$ (variables hors-base).

But : On veut trouver une autre base B^* et une solution de base réalisable \underline{x}^* telles que \underline{x}^* est meilleur que \underline{x} c-à-d $F(\underline{x}^*) > F(\underline{x})$.

2.3.1 Principe de la méthode du simplexe :

faire rentrer une variable hors-base dans la nouvelle base (variable entrante) et faire sortir à la place une variable de base (variable sortante).

Variable entrante-calcul des coûts réduits Fonction objectif f exprimée en fonction des variables hors-base. Ensemble des solutions réalisables $\mathcal{D}_R = \{x \in \mathbb{R}^n / Ax = b, x \geq 0\}$.

Proposition (coûts réduits) :

Pour tout $x \in \mathcal{D}_R$, on a $f(x) = f(\underline{x}) + L_H^T x_H$. où $L_H^T = c_H^T - c_B^T A_B^{-1} A_H$. Est le vecteur des coûts réduits.

· **Variable entrante** :

Si les coûts réduits sont tous négatifs i.e. $L_H^T \leq 0$, il n'est alors pas possible d'augmenter la fonction objectif f : l'algorithme se termine normalement c'est-à-dire qu'on a trouvé une solution de base réalisable \underline{x} optimale.

Dans le cas contraire (i.e. $\exists (L_H)_i > 0$), on a intérêt à faire entrer dans la base, la variable hors-base qui a le coûts réduit positif le plus grand possible.

On note $e \notin B$ l'indice de la variable entrante. On choisit e tel que $(L_H)_e = \max\{(L_H)_j, (L_H)_j > 0\}$ ce qu'on note par

$$e = \operatorname{argmax}\{(L_H)_j, (L_H)_j > 0\}.$$

• **Variable sortante :**

Une fois l'indice e choisi, il faut déterminer la variable qui doit quitter la base. En maintenant la relation $Ax = b$ avec $x \geq 0$, on augmente la variable entrante x_e jusqu'à annuler une des variables de base. Cette variable sera alors la variable sortante.

$$\text{On a } \boxed{z = A_B^{-1}A^e \in \mathbb{R}^m} \quad \text{on doit avoir } \boxed{x_B = \underline{x}_B - zx_e \geq 0}$$

Si $z \leq 0$, on peut augmenter x_e autant qu'on veut, on aura toujours la positivité de la variable de base x_B . La fonction objectif n'est pas majorée sur D_R ($\max f = +\infty$) arrêt de l'algorithme.

Sinon (i.e. il existe $z_i > 0$), pour avoir la positivité $(\underline{x}_B)_i - z_i x_e \geq 0$ pour tout i , on choisit la variable sortante x_s pour laquelle le rapport $(\underline{x}_B)_i / z_i$ pour $i = 1; \dots; m$ avec $z_i > 0$, est le plus petit possible :

Variable sortante (indice) :

$$\boxed{s = \operatorname{argmin}_i \left\{ \frac{(\underline{x}_B)_i}{z_i}, z_i > 0 \right\}}.$$

On a, dans ce cas, $x_s = 0$ et $x_B \geq 0$. La valeur de la variable entrante est donnée par :

$$\boxed{x_e = \min_i \left\{ \frac{(\underline{x}_B)_i}{z_i}, z_i > 0 \right\}} \quad [9]$$

2.3.2 Organigramme de l'algorithme du simplexe

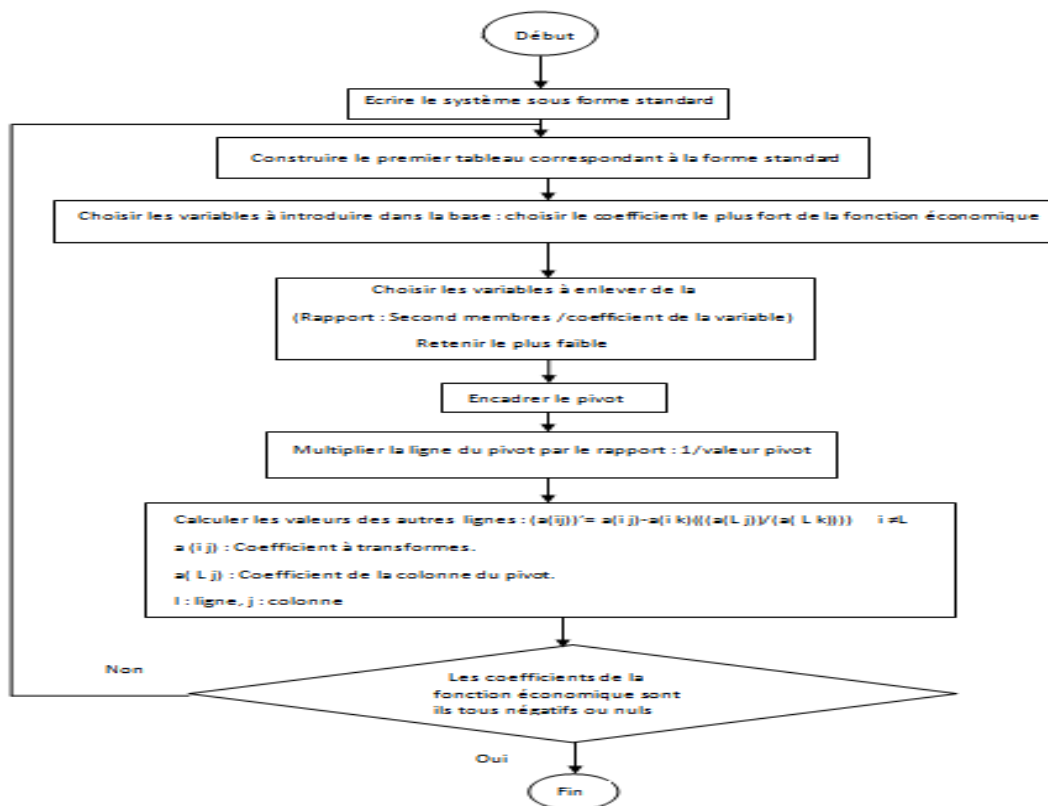


Figure (9) : Organigramme de l'algorithme du simplexe

2.4 Conclusion

On s'est penché dans ce chapitre à l'étude de deux méthodes formelles puissantes l'une pour résoudre un problème d'optimisations combinatoire. L'autre pour résoudre un programme linéaire à valeurs réelles. Dans le prochain chapitre on essaye d'utiliser ces deux outils pour résoudre quelques problèmes d'optimisation combinatoire à savoir: le problème linéaire en nombres entiers, le problème du sac à dos, le problème d'ordonnancement sur une seule machine et le problème de voyageur de commerce.

Chapitre 3

Application de la méthode de séparation et évaluation sur quelques problèmes

3.1 Introduction

Ici on s'intéresse spécifiquement à la méthode exacte de séparation et évaluation et son application à quatre problèmes d'optimisation combinatoire afin de montrer tout d'abord sa flexibilité et aussi de montrer comment que ses trois composantes principales, telles que l'évaluation, la séparation et la stratégie de recherche varient d'un problème à un autre.

Les quatre problèmes d'optimisation combinatoire considérés sont : le problème linéaire à valeurs entières, le problème de sac à dos, le problème d'ordonnancement sur une machine et enfin le problème de voyageur de commerce.

3.2 Le problème linéaire en nombres entiers

Considérons le programme linéaire en nombre entier :

$$(P) \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ \quad -12x_1 + 15x_2 \leq 7 \\ x_i \geq 0, x_i \text{ sont des nombres entiers} \end{array} \right.$$

L'évaluation :

Le problème relaxé (RLP)

$$(P') \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ \quad -12x_1 + 15x_2 \leq 7 \\ \quad x_i \geq 0, \end{array} \right.$$

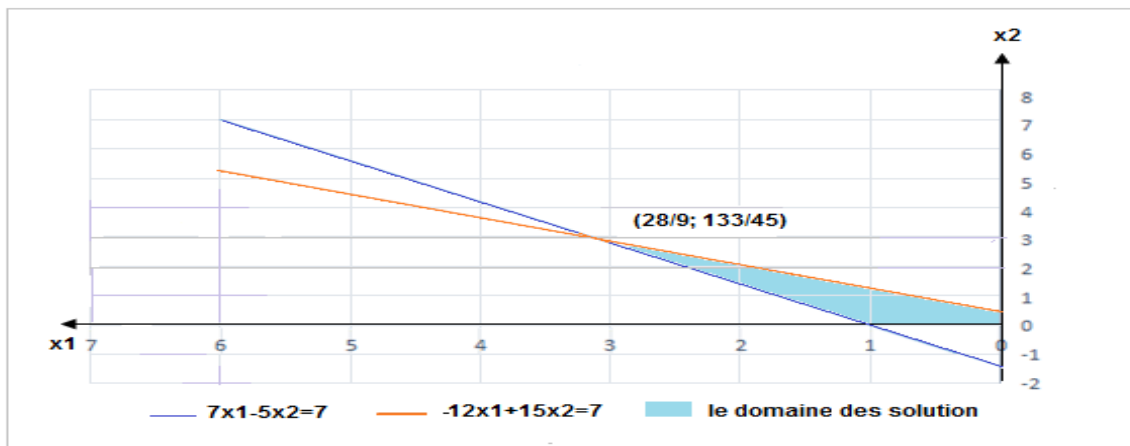


Figure (10) : Résolution graphique de (P)

- Une solution de la relaxation est $x = (28/9; 133/45)$ et $z = 273/45$.

La séparation :

- Ensuite, considérons deux cas $x_2 \leq 2$ et $x_2 \geq 3$.

L'évaluation :

- La relaxation linéaire est pour $x_2 \geq 3$:

$$(P_1)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_2 \geq 3 \\ x_i \geq 0, \end{array} \right.$$

- N'a pas de solution possible, pour $x_2 \geq 3$ car hors domaine des solutions admissibles.

L'évaluation :

- La relaxation linéaire est pour $x_2 \leq 2$:

$$(P_2)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_2 \leq 2 \\ x_i \geq 0, \end{array} \right.$$

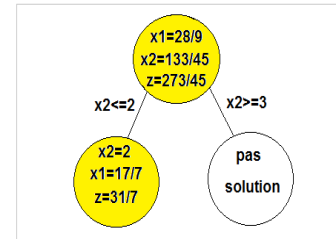
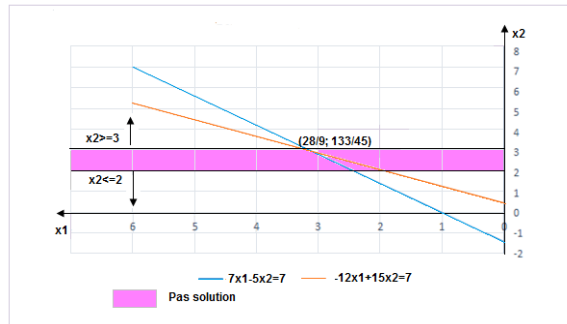


Figure (11) : Résolution graphique et separation (1) de (P)

- La solution de la relaxation est $x = (17/7; 2)$ avec $z = 31/7$.

La séparation :

- Ensuite, considérons deux cas $x_1 \leq 2$ et $x_1 \geq 3$.

L'évaluation :

- La relaxation linéaire est pour $x_1 \geq 3$:

$$(P_3)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_1 \geq 3 \\ x_i \geq 0, \end{array} \right.$$

- N'a pas de solution possible, pour $x_1 \geq 3$ car hors domaine des solutions admissibles.

L'évaluation :

- La relaxation linéaire est pour $x_1 \leq 2$:

$$(P_4)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 + 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_1 \leq 2 \\ x_1 \leq 2 \\ x_i \geq 0, \end{array} \right.$$

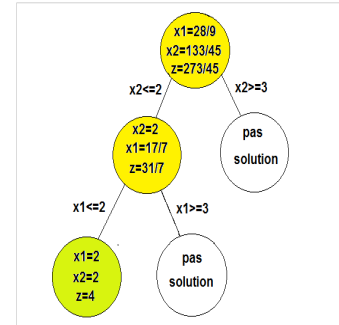
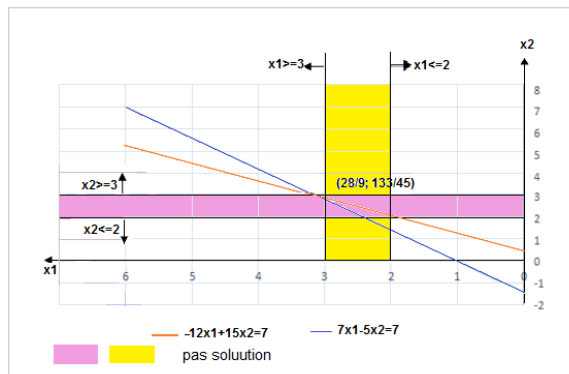


Figure (12) : Resolution graphique et separation (2) de (P).

- Donc $x^* = (2; 2)$ avec $z = 4$. Ceci est la solution optimale.

Remarque 3.2.1 Si le nombre de variables dépassent 3 on utilise l'algorithme algébrique de simplexe au lieu de la méthode graphique pour résoudre le problème relâché (réel).

3.2.1 Méthode algébrique (La méthode du simplexe)

Considérons le programme linéaire en nombres entiers :

$$(P) \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_i \geq 0, x_i \text{ sont des nombres entiers} \end{array} \right.$$

L'évaluation :

Le problème relaxé (P)

$$(P)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_i \geq 0. \end{array} \right.$$

► La méthode du simplexe:

$$(P)'' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 + x_3 = 7 \\ -12x_1 + 15x_2 + x_4 = 7 \\ x_i \geq 0. \end{array} \right.$$

Détermination de la solution de base optimale :

Les principaux étapes de l'algorithme de Simplexe sont :

1. Détermination de la variable entrante.
2. Détermination de la variable sortante.
3. Pivotage.

B = indices des variables en base, N = indices des variables hors base.

	z	x_1	x_2	x_3	x_4	b
z	-1	1	1	0	0	0
x_3	0	7	-5	1	0	7
x_4	0	-12	15	0	1	7

► Variable entrante : (Choisir la variable k hors base avec le profit marginal maximum

(max z))

$\boxed{Max\ z \rightarrow k = \arg \max_{i \in N} \bar{c}_i}$, si $\bar{c}_k \leq 0$ (max) pour tout $k \in N$, solution optimale,

STOP.

Donc x_1 variable entrante.

	z	$\downarrow x_1$	x_2	x_3	x_4	b
z	-1	1 Pivotage	1	0	0	0
x_3	0	7	-5	1	0	7
x_4	0	-12	15	0	1	7

► Variable sortante : (Choisir la variable L en base telle que)

$\boxed{L = \arg \min_{j \in B: \bar{a}_{jk} > 0} \frac{\bar{b}_j}{\bar{a}_{jk}}}$. si $\bar{a}_{Lk} < 0$ pour tout $L \in B$, problème non borné, **STOP.**

On a $\min(\frac{7}{7}, \frac{7}{-12}) = 1$ donc x_3 variable sortante.

Présentation du simplexe en tableaux :

Itération 1

	z	$\downarrow x_1$	x_2	x_3	x_4	b
z	-1	1	1	0	0	0
$\leftarrow x_3$	0	7	-5	1	0	7
x_4	0	-12	15	0	1	7

Règles de pivotage :

$$(\bar{a}_{ij})' = \begin{cases} \frac{\bar{a}_{Lj}}{\bar{a}_{Lk}} & i = L \\ \bar{a}_{ij} - \bar{a}_{ik}(\frac{\bar{a}_{Lj}}{\bar{a}_{Lk}}) & i \neq L \end{cases} \quad \text{et} \quad (\bar{b}_i)' = \begin{cases} \frac{\bar{b}_L}{\bar{a}_{Lk}} & i = L \\ \bar{b}_i - \bar{a}_{ik}(\frac{\bar{b}_L}{\bar{a}_{Lk}}) & i \neq L \end{cases}$$

	z	x_1	x_2	x_3	x_4	b
z	-1	0	$\frac{12}{7}$	$\frac{-1}{7}$	0	1
x_1	0	1	$\frac{-5}{7}$	$\frac{1}{7}$	0	1
x_4	0	0	$\frac{45}{7}$	$\frac{12}{7}$	1	19

calcul :

$$2.(-12, 15, 0, 1, | 7) - (-12)(1, \frac{-5}{7}, \frac{1}{7}, 0, | 1)$$

$$1.(7, -5, 1, 0, | 7) \frac{1}{7} \text{ tq } \bar{a}_{13} = 7$$

$$3.(1, 1, 0, 0, | 0) - (1)(1, \frac{-5}{7}, \frac{1}{7}, 0, | 1)$$

Itération 2

	z	x_1	$\downarrow x_2$	x_3	x_4	b
z	-1	0	$\frac{12}{7}$	$\frac{-1}{7}$	0	1
x_1	0	1	$\frac{-5}{7}$	$\frac{1}{7}$	0	1
$\leftarrow x_4$	0	0	$\frac{45}{7}$	$\frac{12}{7}$	1	19

	z	x_1	x_2	x_3	x_4	b	calcul :
z	-1	0	0	$\frac{-9}{15}$	$\frac{-4}{15}$	$\frac{273}{45}$	$3..(0, \frac{12}{7}, \frac{-1}{7}, 0, , 1) - (\frac{12}{7})(0, 1, \frac{4}{5}, \frac{7}{45} , \frac{133}{45})$
x_1	0	1	0	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{28}{9}$	$2.(1, \frac{-5}{7}, \frac{1}{7}, 0, , 1) - (\frac{-5}{7})(0, 1, \frac{4}{5}, \frac{7}{45} , \frac{133}{45})$
x_2	0	0	1	$\frac{4}{15}$	$\frac{7}{45}$	$\frac{133}{45}$	$1.(0, \frac{45}{7}, \frac{12}{7}, 1, 19) \frac{45}{7} \text{ tq } \overline{a_{24}} = \frac{45}{7}$

Si $\overline{c_k} \leq 0$ pour tout $k \in N$, solution optimale, **STOP**.

On a $\overline{c_k} = (0, 0, \frac{-9}{15}, \frac{-4}{15}) \leq 0$, donc **STOP**, la solution est $x = (\frac{28}{9}, \frac{133}{45})$, avec $z = \frac{273}{45}$.

Qui n'est pas entière. Donc le processus de la méthode de séparation et évaluation se poursuit :

La séparation :

On a

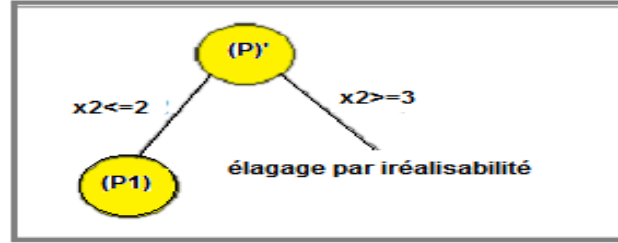


Figure (13) : La séparation (1) sur simplexe

Le problème linéaire à valeurs entières (P_1) :

$$(P_1) \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 < 7 \\ -12x_1 + 15x_2 < 7 \\ x_2 < 2 \\ x_i \in \mathbb{N}. \end{array} \right.$$

L'évaluation :

Le problème relaxé (P_1):

$$(P_1)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 \leq 7 \\ -12x_1 + 15x_2 \leq 7 \\ x_2 \leq 2 \\ x_i \geq 0. \end{array} \right.$$

► La méthode du simplexe:

$$(P_1)' \left\{ \begin{array}{l} \text{Max } Z = x_1 + x_2 \\ \text{s c } 7x_1 - 5x_2 + x_3 = 7 \\ -12x_1 + 15x_2 + x_4 = 7 \\ x_2 + x_5 = 2 \\ x_i \geq 0. \end{array} \right.$$

Présentation en tableau :

	z	$\downarrow x_1$	x_2	x_3	x_4	x_5	b
z	-1	1	1	0	0	0	0
$\leftarrow x_3$	0	7	-5	1	0	0	7
x_4	0	-12	15	0	1	0	7
x_5	0	1	0	0	0	1	2

Itération 1:

Variable entrante est x_1 , et variable sortante est x_3

	z	$\downarrow x_1$	x_2	x_3	x_4	x_5	b
z	-1	1	1	0	0	0	0
$\leftarrow x_3$	0	7	-5	1	0	0	7
x_4	0	-12	15	0	1	0	7
x_5	0	1	0	0	0	1	2

	z	x_1	x_2	x_3	x_4	x_5	b	calcul :
z	-1	0	$\frac{12}{5}$	$\frac{-1}{7}$	0	0	-1	$2.(1, 1, 0, 0, 0 \mid, 0) - (1)(1, \frac{-5}{7}, \frac{1}{7}, 0, 0 \mid, 1)$
x_1	0	1	$\frac{-5}{7}$	$\frac{1}{7}$	0	0	1	$1.(7, -5, 1, 0, 0, \mid, 7)^{\frac{1}{7}} \text{ tq } \overline{a_{13}} = 7$
x_4	0	0	$\frac{45}{7}$	$\frac{12}{7}$	1	0	19	$3.(-12, 15, 0, 1, 0, \mid, 7) - (-12)(1, \frac{-5}{7}, \frac{1}{7}, 0, 0 \mid, 1)$
x_5	0	0	1	0	0	1	2	$4.(0, 1, 0, 0, 1, \mid, 2) - 0(1, \frac{-5}{7}, \frac{1}{7}, 0, 0 \mid, 1)$

Itération 2 :

Variable entrante est x_2 , et variable sortante est x_5

	z	x_1	$\downarrow x_2$	x_3	x_4	x_5	b
z	-1	0	$\frac{12}{5}$	$-\frac{1}{7}$	0	0	-1
x_1	0	1	$-\frac{5}{7}$	$\frac{1}{7}$	0	0	1
x_4	0	0	$\frac{45}{7}$	$\frac{12}{7}$	1	0	19
$\leftarrow x_5$	0	0	1	0	0	1	2

	z	x_1	x_2	x_3	x_4	x_5	b	calcul :
z	-1	0	0	$-\frac{1}{7}$	0	$-\frac{12}{5}$	$-\frac{29}{9}$	$2.(0, \frac{12}{5}, -\frac{1}{7}, 0, 0 \mid -1) - (\frac{12}{5})(0, 1, 0, 0, 1 \mid 2)$
x_1	0	1	0	$\frac{1}{7}$	0	$\frac{5}{7}$	$\frac{17}{7}$	$3.(1, -\frac{5}{7}, \frac{1}{7}, 0, 0, \mid 1) - (-\frac{5}{7})(0, 1, 0, 0, 1 \mid 2)$
x_4	0	0	0	$\frac{12}{7}$	1	$-\frac{45}{7}$	$\frac{43}{7}$	$4.(0, \frac{45}{7}, \frac{12}{7}, 1, 0, \mid 19) - \frac{45}{7}(0, 1, 0, 0, 1 \mid 2)$
x_2	0	0	1	0	0	1	2	$1.(0, 1, 0, 0, 1, \mid 2) \frac{1}{1} \text{ tq } \overline{a_{25}} = 1.$

Si $\overline{c_k} \leq 0$ pour tout $k \in N$, solution optimale, STOP.

► On a $\overline{c_k} = (0, 0, -\frac{1}{7}, 0, -\frac{12}{5}) \leq 0$, donc STOP, la solution est $x = (\frac{17}{7}; 2)$ avec $z = -\frac{29}{9}$, $x_1 = \frac{17}{7}$. Qui n'est pas entière. Donc le processus de la méthode de séparation et évaluation se poursuit :

La séparation :

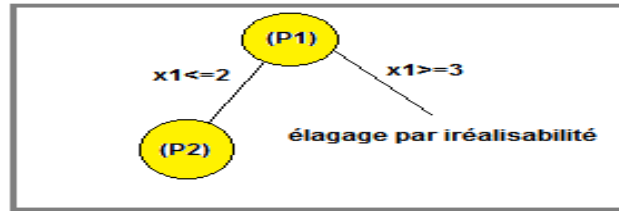


Figure (14) : La séparation (2) sur simplexe

Le problème linéaire à valeurs entières (P_2) :

$$(P_2) \left\{ \begin{array}{l} \text{Max } Z = x_1 + 2 \\ \text{s c } 7x_1 \leq 17 \\ -12x_1 \leq -23 \\ x_1 \leq 2 \\ x_i \in \mathbb{N}. \end{array} \right.$$

Le programme de (P_2) est:

$$(P_2) \left\{ \begin{array}{l} \text{Max } Z = x_1 + 2 \\ \text{s c } 7x_1 \leq 17 \\ -12x_1 \leq -23 \\ x_1 \leq 2 \\ x_1 \in \mathbb{N}. \end{array} \right. \Rightarrow (P_2) \left\{ \begin{array}{l} \text{Max } Z = x_1 + 2 \\ \text{s c } x_1 \leq \frac{17}{7} \\ x_1 \leq \frac{-23}{-12} \\ x_1 \leq 2 \\ x_1 \in \mathbb{N}. \end{array} \right. \Rightarrow (P_2) \left\{ \begin{array}{l} \text{Max } Z = x_1 + 2 \\ \text{s c } x_1 \leq \frac{23}{12} \\ x_1 \in \mathbb{N}. \end{array} \right. \Rightarrow$$

$x_1 = 1$

Donc $x^* = (1; 2)$ avec $z = 3$ Ceci est la solution optimale.

La comparaison entre méthode de graphique et la méthode du simplexe

* Il est à noter que l'algorithme du simplexe est une méthode itérative qui passe d'une solution dite réalisable de base à une solution de base réalisable meilleur jusqu'à l'aboutissement à la solution optimale.

Aussi il est à noter que chaque solution de base réalisable correspond à un point corner du polyèdre de la région des solutions admissibles dans le cas où on résout le problème par la méthode graphique. C'est pour cela qu'une solution optimale correspond à un sommet du polyèdre de l'ensemble des solutions admissibles.

3.3 Le problème du sac à dos

- Le Problème peut s'écrire comme un programme linéaire en nombres entiers. Si on note x_i le nombre d'unités du produit P_i à mettre dans le sac à dos, il s'agit de calculer le bénéfice maximal.

On a 4 produits suivants avec : de poids respectifs : (2, 5, 10, et 5 et de valeurs respectives : 40, 30, 50, et 10). Le poids maximal autorisé est 16.

$$(P) \left\{ \begin{array}{l} \max z = 40x_1 + 30x_2 + 50x_3 + 10x_4 \\ 2x_1 + 5x_2 + 10x_3 + 5x_4 \leq 16 \\ x_i = \{0, 1\} \end{array} \right.$$

<i>objets</i>	1	2	3	4
p_i	\$40	\$30	\$50	\$10
w_i	2	5	10	5
p_i/w_i	\$20	\$6	\$5	\$2

La séparation :

► trier tous les objets par ordre décroissant de cette valeur p_i/w_i .

$$\text{On a } \left\{ \begin{array}{l} \text{poids total} = \text{poids} + \sum_{j=i+1}^{k-1} w_j \\ \text{borne} = (\text{bénéfice} + \sum_{j=i+1}^{k-1} p_j) + [(W - \text{poids total}) \times \frac{p_k}{W_k}] \end{array} \right.$$

Notation 3.3.1 La couple (x, y) :

$$\left\{ \begin{array}{l} x : \text{Le numéro de l'article} \\ y : \text{Le numéro de nœud} \end{array} \right.$$

L'évaluation :

$$\begin{array}{l} (0.1) \left\{ \begin{array}{l} p = \$0 \\ w = 0 \\ UB = \$115 = (0 + 40 + 30 + (16 - 7) \times \frac{50}{10}) \end{array} \right. \\ (1.2) \left\{ \begin{array}{l} p = \$40 \\ w = 2 \\ UB = \$115(0 + 40 + 30 + (16 - 7) \times \frac{50}{10}) \end{array} \right. \\ (1.3) \left\{ \begin{array}{l} p = \$0 \\ w = 0 \\ UB = \$82 = (0 + 30 + 50 + (16 - 15) \times \frac{10}{5}) \end{array} \right. \\ (2.4) \left\{ \begin{array}{l} p = \$70 \\ w = 7 \\ UB = \$115(0 + 40 + 30 + (16 - 7) \times \frac{50}{10}) \end{array} \right. \end{array}$$

$$\begin{aligned}
 (2.5) \quad & \left\{ \begin{array}{l} p = \$40 \\ w = 2 \\ UB = \$98 = (0 + 40 + 50 + (16 - 12) \times \frac{10}{5}) \end{array} \right. \\
 (3.6) \quad & \left\{ \begin{array}{l} p = \$120 \\ w = 17. \text{ Impossible } (17 > 16) \\ UB = \$0 \end{array} \right.
 \end{aligned}$$

$$\begin{aligned}
 (3.7) \quad & \left\{ \begin{array}{l} p = \$70 \\ w = 7 \\ UB = \$80 = (0 + 40 + 30 + 10) \end{array} \right. \\
 (3.8) \quad & \left\{ \begin{array}{l} p = \$90 \\ w = 12 \\ UB = \$98(0 + 40 + 50 + (16 - 12) \times \frac{10}{5}) \end{array} \right. \\
 (3.9) \quad & \left\{ \begin{array}{l} p = \$40 \\ w = 2 \\ UB = \$50(0 + 40 + 0 + 0 + 10) \end{array} \right.
 \end{aligned}$$

$$\begin{aligned}
 (4.10) \quad & \left\{ \begin{array}{l} p = \$100 \\ w = 17. \text{ Impossible } (17 > 16) \\ UB = \$0 \end{array} \right. \\
 (4.11)^* \quad & \left\{ \begin{array}{l} p = \$90 \\ w = 12 \\ UB = \$90 = (0 + 40 + 50) \end{array} \right.
 \end{aligned}$$

donc $x^* = (1, 0, 1, 0)$ **et** $w = 12$ \$, $p = \$90$

Nous montrons comment on utilise la stratégie de dessin de séparation et évaluation dans le problème de sac à dos.

D'abord nous discutons la version simple qui appelé "la recherche de largeur d'abord".

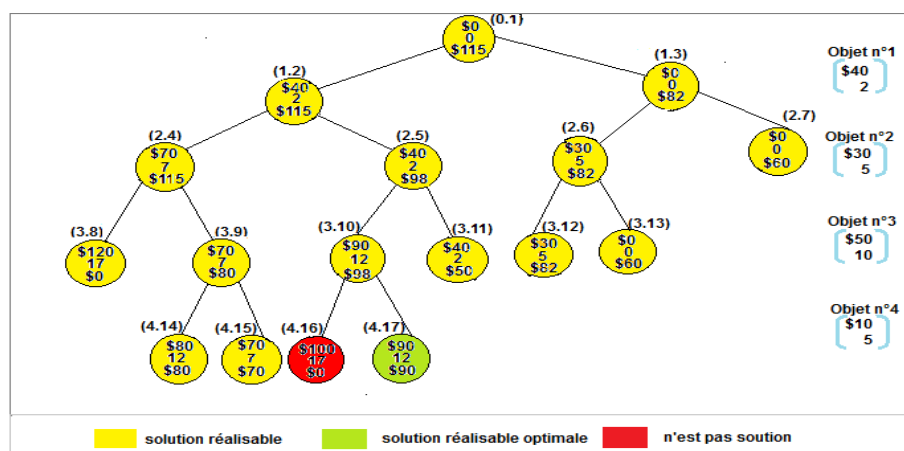


Figure (15) : Arbre de recherche associé à la résolution.

Après ce la, Nous montrons l'amélioration sur la version simple dite la recherche de " Le meilleur d'abord " avec le pruneau de séparation et évaluation.

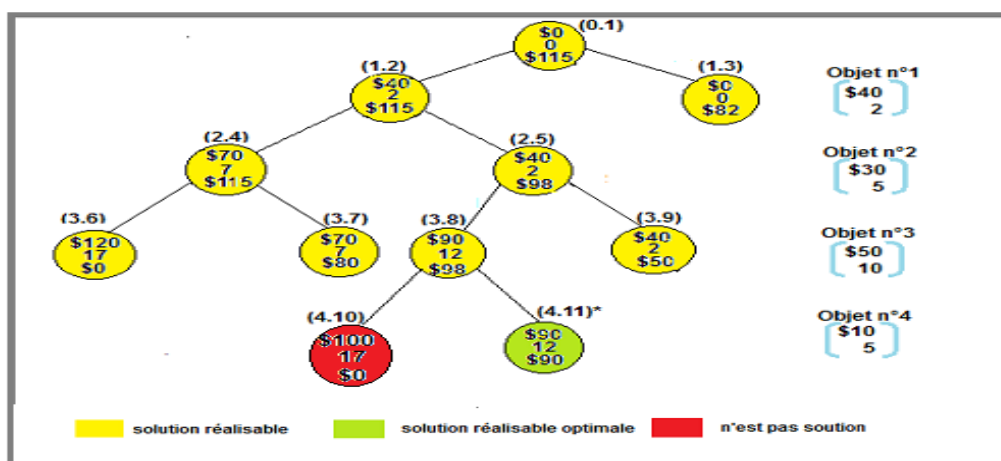


Figure (16) : Arbre de recherche associé à la résolution par branch and bound

3.4 Le problème d'ordonnancement sur une seule machine

De nombreux problèmes d'ordonnancement classés NP-difficiles peuvent être résolus par des procédures par séparation et évaluation. Nous présentons des problèmes d'ordonnancement à une seule machine et un job shop.

Un atelier comporte une seule machine toujours disponible. “ n ” tâches doivent y être exécutées les unes à la suite des autres. Elles sont toutes disponibles dès l'instant Zéro. La tâche i dure p_i unités de temps et doit être terminée avant l'instant d_i sinon il faut payer une pénalité de retard w_i par unité de temps de retard. Il s'agit de trouver le meilleur ordre de passage des tâches sur la machine de manière à minimiser la somme des pénalités de retard à payer.

Ensemble des solutions réalisable N_0 est formé des $n!$ Permutation possibles des n tâches.

La séparation : se fera sur la tâche que l'on fixe en dernier, puis en avant dernier, ... pour le calcul des bornes ou minorant, on prend en considérations la somme des pénalités des tâches placées en queue. L'élimination des sous ensemble se fera sur le sommet ayant le plus petite mineront. Le problème suivant est à 1 machine et 3 tâches. Telle que les paramètres de problèmes :

$$\left(1/d_i / \sum_{i=0}^3 w_i T_i\right)$$

	tâche 1	tâche 2	tâche 3
p_i	2	4	1
d_i	3	5	5
w_i	4	2	3

L'évaluation :

On relâche la contrainte des dates d'échéance d_i par la séparation $d_i = \infty$

Il est clair que

$$L_i = c_i - d_i \leq \max(c_i - d_i, 0) = T_i \Rightarrow w_i L_i \leq w_i T_i \Leftrightarrow \sum w_i L_i \leq \sum w_i T_i$$

Or le problème est facile il peut être résolu par la règle EDD (Earliest due date). Donc la valeur de la solution optimale de ce dernier peut être utilisée comme évaluation par

défaut pour le problème

$$1/d_i / \sum_{i=0}^3 w_i T_i.$$

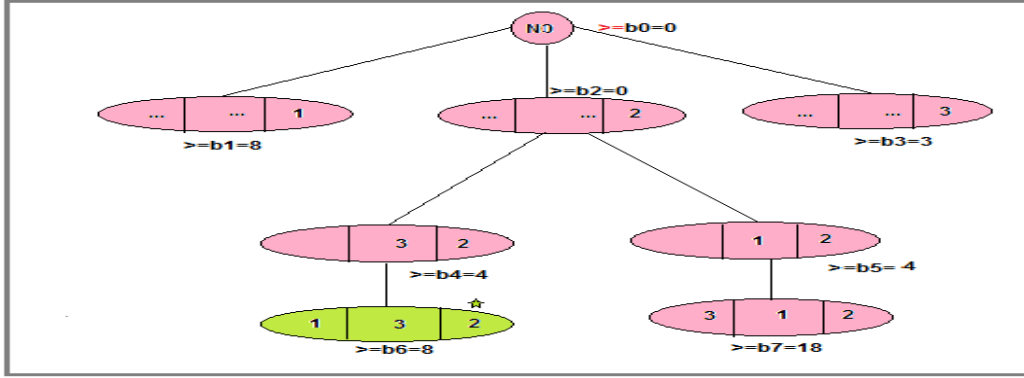


Figure (17) : Schéma récapitulatif des séparations et évaluation.

- On a séparé par rapport à une tâche placée en dernière position.
- Pour calculer b_1 , quel que soit la séquence (2.3.1) ou (3.2.1), la tâche 1 ne peut commencer son exécution qu'à la date $P_2 + P_3 = 5$. Elle sera en retard de 2 unités. Le coût de retard sera d'au moins $2 \times 4 = 8$ d'où $b_1 = 8$.
- Ainsi de suite $b_2 = 0$ car la tâche 2 ne sera pas en retard de 1 unité, $b_3 = 3 \times 1 = 3$.
- On sépare par rapport au sommet de b_2 . On trouve que pour les séquences (1.3.2) ou (3.1.2), les pénalités sont respectivement de 4 unités.
- Il faut encore séparer le sommet où la tâche 3 est placée la dernière. On trouve les 2 sous-ensembles de coût 8 et 18. D'où (1.3.2) et (3.1.2) sont optimaux. [1]

3.5 Le problème de voyageur de commerce (L'algorithme de little)

3.5.1 L'algorithme de Little

L'algorithme de Little est une simplification du Branch And Bound. Il se décompose de la manière suivante :

- Initialisation.

1. Réduction de la matrice (RM)

- Réduction en ligne : soustraire le plus petit élément de chaque ligne, ce qui nous permet d'avoir pour chaque ligne au moins un 0.
- Réduction en colonne : soustraire le plus petit élément de chaque colonne, ce qui nous permet d'avoir pour chaque colonne au moins un 0.

2. Calcul de la borne inférieure (BI).

- Branch and Bound

1. Calcul des coûts d'évitement (CE)

$$CE(x, y) = \min \{P_{xi} + P_{jy} | (i \neq y) \wedge (j \neq x)\}$$

2. Sélection de l'arc (x,y) avec CE(x,y) maximal

3. Branchement (création de l'arborescence)

- Branche P_2 si nous évitons de passer par (x, y) . [15]

3.5.2 Le problème de voyageur de commerce

Problème du voyageur de commerce (Traveling Salesman Problem-TSP) : Calculer une tournée de longueur minimale passant une et une seule fois par chacune des n villes.

- Le problème du voyageur de commerce consiste à calculer un cycle hamiltonien de coût minimal.

- L'algorithme de Little est un algorithme de résolution du TSP par séparation et évaluation :

- La séparation consiste à considérer l'inclusion ou l'exclusion d'un trajet (i, j) dans une tournée. Chaque séparation produisant deux branches.

L'évaluation fournit une borne inférieure du coût de la tournée en effectuant des opérations sur la matrice de coûts.

On a un problème de voyageur de commerce de $n = 6$ villes.

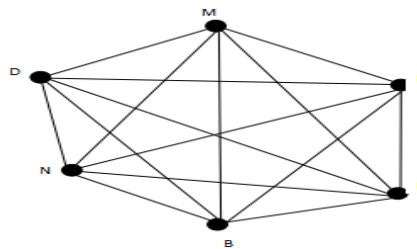
N° du ville	1	2	3	4	5	6
Nom de ville	<i>B</i>	<i>L</i>	<i>N</i>	<i>P</i>	<i>M</i>	<i>D</i>

On a $N(t) = \frac{1}{2}(6 - 1)! = 60$ tournées possibles.

Remarque 3.5.1 Lower Bound (borne inférieure) :

• On suppose que pour chaque sous-problème on peut calculer efficacement une borne inférieure (LB) sur le coût optimal, i.e. $(LB) \leq f(t)$.

$$f(t) = \min \sum_{(i,j) \in t} c_{ij} x_{ij}.$$



Graphe complet

L'évaluation :

Matrice de coûts (symétrique et la distance par km)

► Réduction de la matrice lignes

3.5. Le problème de voyageur de commerce (L'algorithme de little)

	<i>B</i>	<i>L</i>	<i>N</i>	<i>P</i>	<i>M</i>	<i>D</i>	Min ligne :
<i>B</i>	∞	780	320	580	480	660	320
<i>L</i>	780	∞	700	460	300	200	200
<i>N</i>	320	700	∞	380	820	630	320
<i>P</i>	580	460	380	∞	750	310	310
<i>M</i>	480	300	820	750	∞	500	300
<i>D</i>	660	200	630	310	500	∞	<u>200</u>

1650 total supprimé.

► Réduction de la matrice colonnes

	<i>B</i>	<i>L</i>	<i>N</i>	<i>P</i>	<i>M</i>	<i>D</i>
<i>B</i>	∞	460	0	260	160	340
<i>L</i>	580	∞	500	260	100	0
<i>N</i>	0	380	∞	60	500	310
<i>P</i>	270	150	70	∞	440	0
<i>M</i>	180	0	520	450	∞	200
<i>D</i>	460	0	430	110	300	∞

Min colonne 0 0 0 60 100 0 | 160 total supprimé

Le nœud initial de l'arbre de recherche

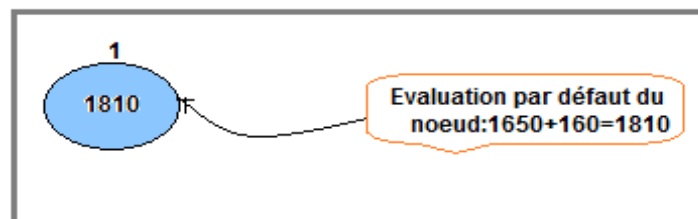


Figure (18) : Le nœud initial de l'arbre de recherche

Regret = min ligne + min colonne, calcul des **regrets** dans toutes les cases de valeur 0.

	<i>B</i>	<i>L</i>	<i>N</i>	<i>P</i>	<i>M</i>	<i>D</i>	Min ligne
<i>B</i>	∞	460	0(130)	200	60	340	60
<i>L</i>	580	∞	500	200	0(60)	0(0)	
<i>N</i>	0(180)	380	∞	0(50)	400	310	
<i>P</i>	270	150	70	∞	340	0(70)	
<i>M</i>	180	0(180)	520	390	∞	200	
<i>D</i>	460	0(50)	430	50	200	∞	
	Min colonne		70		$60 + 70 = 130$		

On privilégie l'étude du trajet de regret maximal: N-B.

La séparation :

On crée deux fils (2 et 3) correspondant à l'inclusion ou à l'exclusion du trajet N-B dans la tournée

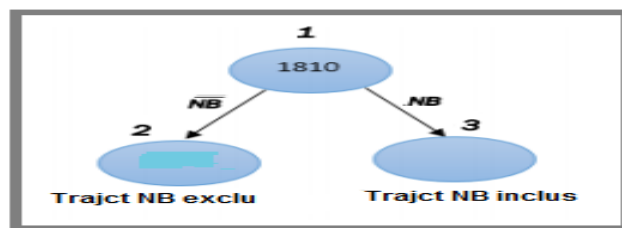


Figure (19) : La séparation de nœud initial

L'évaluation d'un nœud de type "trajet exclu" est déjà connue, sans autre calcul : il s'agit de la **valeur du père + la valeur de regret**

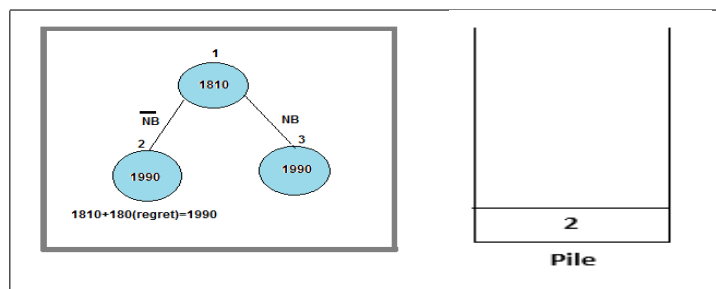


Figure (20) : Arbre de recherche et séparation N° (1).

L'évaluation :

On manipule une nouvelle matrice : la ligne de la ville départ (N) et la colonne de la ville d'arrivée (B) du trajet a été supprimée

	L	N	P	M	D
B	460	0	200	60	340
L	∞	500	200	0	0
P	150	70	∞	340	0
M	0	520	390	∞	200
D	0	430	50	200	∞

On supprime tous les trajets aboutissant à des sous-tournées incomplètes (n'incluant pas toutes les villes)

→ Ici BN , aboutissant à la sous-tournée $NB - BN$

	L	N	P	M	D
B	460	∞	200	60	340
L	∞	500	200	0	0
P	150	70	∞	340	0
M	0	520	390	∞	200
D	0	430	50	200	∞

Réduction de la matrice lignes et colonnes

	L	N	P	M	D	min ligne:
B	460	∞	200	60	340	60
L	∞	500	200	0	0	0
P	150	70	∞	340	0	0
M	0	520	390	∞	200	0
D	0	430	50	200	∞	0

min colonne 0 70 50 0 0 | $\overline{180}$ total supprimé

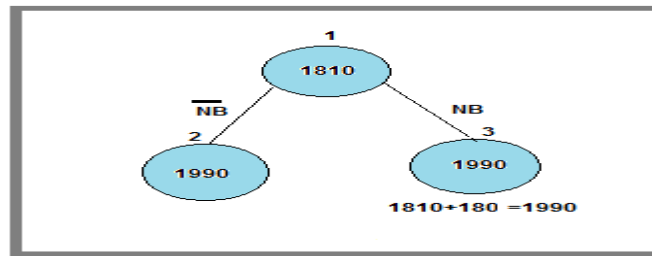


Figure (21) : L'évaluation du séparation

Regret maximal : P-N (360)

	L	N	P	M	D
B	400	∞	90	0	280
L	∞	430	150	0	0
P	150	0	∞	340	0
M	0	450	340	∞	200
D	0	360	0	200	∞

\Rightarrow

	L	N	P	M	D
B	400	∞	90	0(90)	280
L	∞	430	150	0(0)	0(0)
P	150	0(360)	∞	340	0(0)
M	0(200)	450	340	∞	200
D	0(0)	360	0(90)	200	∞

► Création des fils 4 et 5.

► Affectation de la valeur de 4.

La séparation :

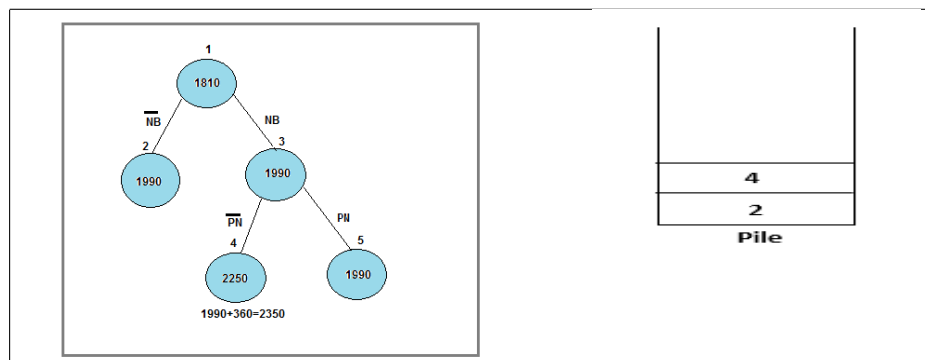


Figure (22) : Arbre de recherche et séparation N°(2)

► Élimination du trajet BP (aboutissant à la sous-tournée PN – NB – BP)

	L	P	M	D	
B	400	∞	0	280	0
L	∞	150	0	0	0
M	0	340	∞	200	0
D	0	0	200	∞	0
	0	0	0	0	$\bar{0}$ total supprimé

\Rightarrow

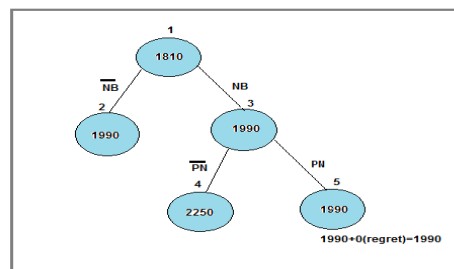


Figure (23) : L'évaluation du séparation N°2

L'évaluation :

	L	P	M	D
B	400	∞	0	280
L	∞	150	0	0
M	0	340	∞	200
D	0	0	200	∞

\Rightarrow

	L	P	M	D
B	400	∞	$0(280)$	280
L	∞	150	$0(0)$	$0(200)$
M	$0(200)$	340	∞	200
D	$0(0)$	$0(150)$	200	∞

Regret maximal: B-M (280)

- Création des fils 6 et 7.
- Affectation de la valeur de 6.
- Empilage de 6.

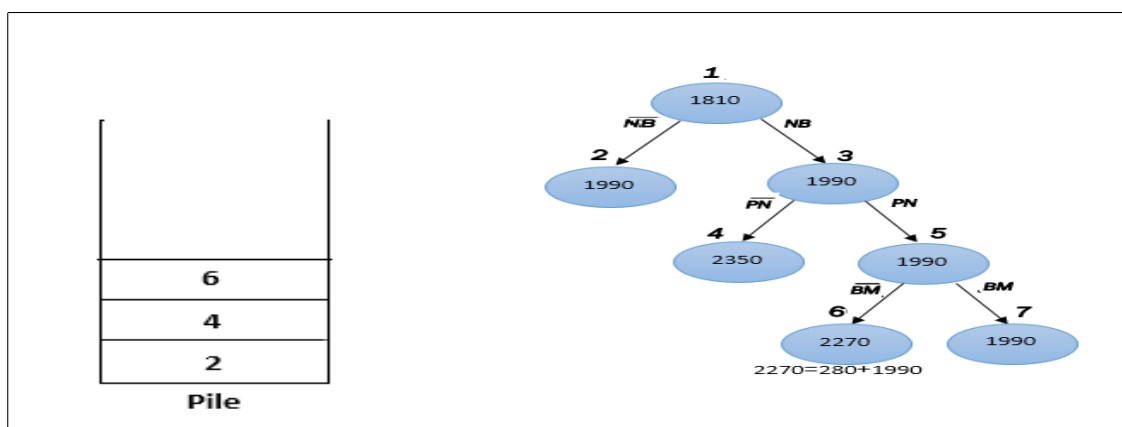


Figure (24) : Arbre de recherche et séparation N °(3)

Élimination du trajet MP (aboutissant à la sous-tournée $PN - NB - BM - MP$)

	L	P	D	
L	∞	150	0	0
M	0	∞	200	0
D	0	0	∞	<u>0</u>
	0	0	0	$\bar{0}$ total supprimé

La séparation :

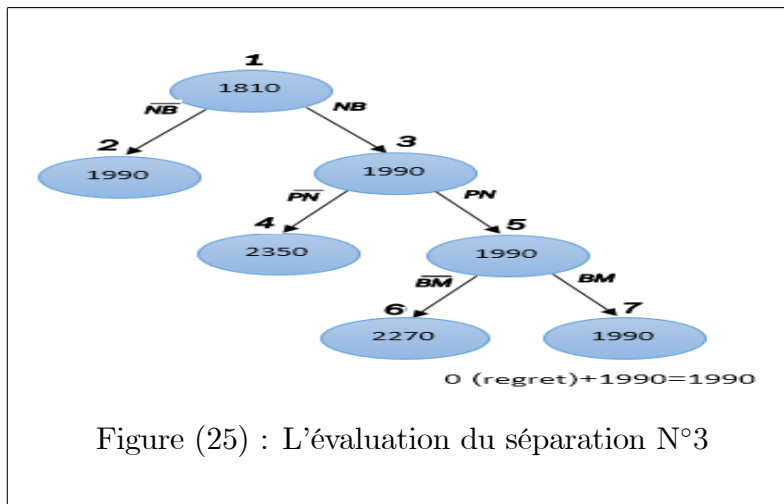


Figure (25) : L'évaluation du séparation N°3

L'évaluation :

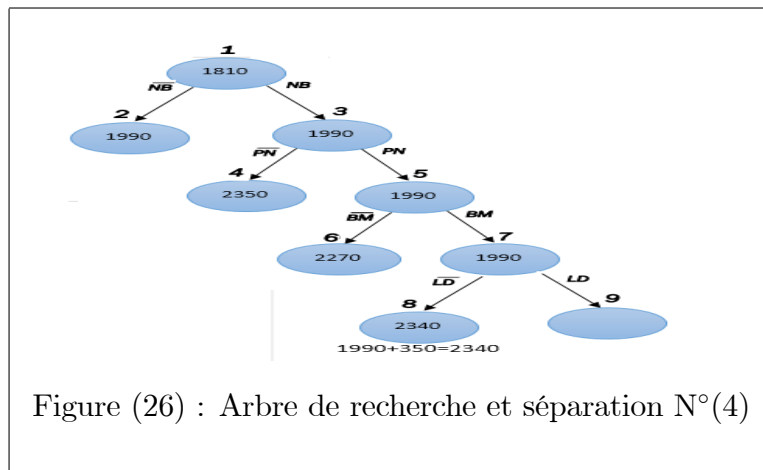
► Calcul des regrets :

	L	P	D
L	∞	150	0(350)
M	0(200)	∞	200
D	0(0)	0(150)	∞

► Regret maximal: L-D (350) :

- Création des fils 8 et 9.
- Affectation de la valeur de 8.
- Empilage de 8.

La séparation :



L'évaluation :

► Matrice de coûts

	L	P
M	0	∞
D	0	0

► Élimination du trajet DL (aboutissant à la sous-tournée $LD - DL$) :

	L	P
M	0	∞
D	∞	0

► Réduction de la matrice

	L	P
M	0	∞
D	∞	0

0 0 | $\bar{0}$ total supprimé

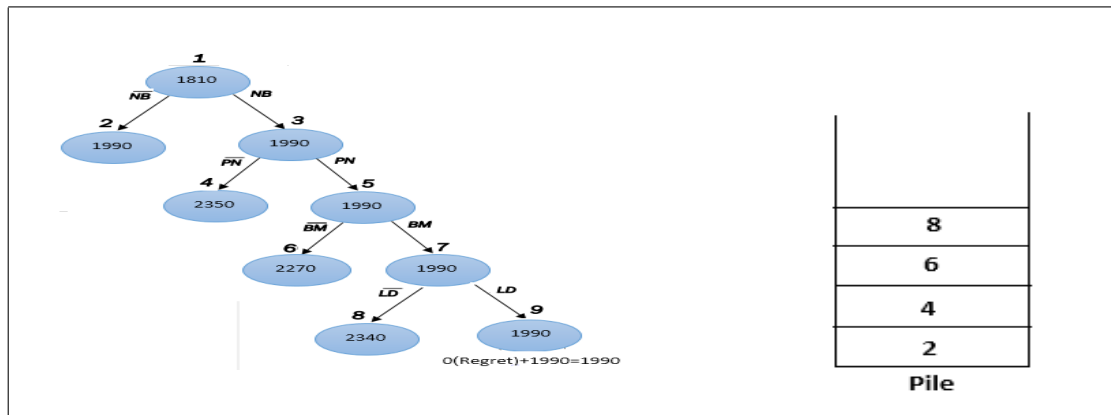


Figure (27) : L'évaluation du séparation N°4

► Branche terminée. Une première solution trouvée, contenant les trajets :

NB, PN, BM, LD, ML et DP

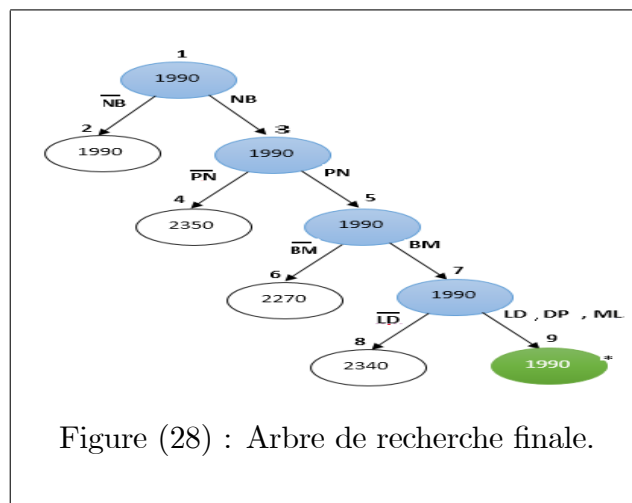


Figure (28) : Arbre de recherche finale.

► On peut donc construire une tournée au départ de n'importe quelle ville, par exemple B :

$BM - ML - LD - DP - PN - NB$

► Valeur de référence (*): 1990km [13]

3.6 Conclusion

Dans ce chapitre on a montré comment on adapte l'utilisation de la méthode par séparation et évaluation pour résoudre quelques problèmes d'optimisation combinatoire :

L'évaluation : Qui se calcule généralement par relaxation des contraintes, l'intégralité des variables, une tâche peut ne pas avoir une date d'échéance.

La séparation : la division de l'espace de recherche en deux sous régions, l'article peut ou ne peut pas être pris dans le sac , le choix de la tâche dans une position donnée, une arête peut ou ne peut pas être choisie dans la tournée cherchée.

NP : A la fin nous avons signalé que notre recherche n'est pas encore prête puisque le temps est limité. Nous espérons que la programmation de la méthode de B&B sera affichée le jour de l'exposé.

Conclusion générale

Dans ce mémoire, on s'est concentré à l'étude de quelques problèmes d'optimisation combinatoire dans la classe des NP-difficiles où un algorithme polynomial de résolution est introuvable. Ces problèmes sont les plus célèbres : le problème linéaire en nombres entiers (PLNE), le problème de sac à dos, et le problème d'ordonnancement sur une seule machine et le problème de voyageur de commerce (PVC) . C'est via des exemples numériques qu'on va montrer comment chacun pourrait être résolu par ce qu'on appelle la méthode de séparation et évaluation. C'est une méthode très performante dans le cas où le problème sous considération est de taille modérée. Cette méthode qui est aussi connu sous le nom de "branch and bound" (en Anglais)est basée sur trois aspects : l'évaluation par défaut (ou excès), la séparation et la stratégie de recherche.

Pour la stratégie du parcours dans l'arbre de recherche correspondant à la méthode de séparation évaluation on considère dans les trois cas stratégie "meilleur d'abord" vu son efficacité et son supériorité sur les autres stratégies à savoir : largeur d'abord, profondeur d'abord.

Bibliographie

- [1] A.DERBALA : *Programmation dynamique et métaheuristiques*, universite saad dahlab, Blida.
- [2] Bellman, Richard (1957) : *Dynamic Programming*, Princeton University Press. Dover paperback edition (2003), ISBN 0-486-42809-5.
- [3] M. AÏDER : *Optimisation combinatoire*, El Alia, Bab Ezzouar 16111 Algiers, Algeria,1 October 2014.
- [4] S.M. DOUIRI, S. ELBERNOUSSI, H.LAKHBAB : *Cours des Méthodes de Résolution Exactes Heuristiques et Métaheuristiques*.Laboratoire de recherche mathématiques.
- [5] Cours sur internet. NP - Complétude. <http://pedrov.kwain.net/utbm/AG51/Cours>.
- [6] Cours sur internet. Méthodes de résolution par séparation et évaluation. <http://sebastien-viardot.imag.fr/Enseignements/DocumentsAlgoWiki/bb.pdf>
- [7] Imen Harbaoui Dridi. Optimisation heuristique pour la résolution du m-PDPTW statique et dynamique. Sciences de l'ingénieur [physics]. Ecole Centrale de Lille; Ecole Nationale d'Ingénieurs de Tunis (Tunisie), 2010. Français.<NNT : 2010ECLI0031>. <tel-00590443>.
- [8] Jacques :Optimisation Combinatoire, CARLIER RECHERCHE OPERATIONNELLE .
- [9] J.-F. Scheid. Méthode du Simplexe, Graphes et RO-TELECOM Nancy 2A.
- [10] K. GHEDIRA et C. SOLNON : *Optimisation multi-objectif par colonies de fourmis : cas des problèmes de sac à dos*, Inès Alaya.

- [11] Du cours de mastère de Christine Solnon disponible à l'url [http ://www710.univ-lyon1.fr/ csolnon](http://www710.univ-lyon1.fr/~csolnon).
- [12] Lyes Belhoul. R esolution de probl emes d'optimisation combinatoire mono et multi-objectifs par enum eration ordonn ee. Autre [cs.OH]. Universit e Paris Dauphine - Paris IX, 2014. Fran cais.<NNT : 2014PA090060>. <tel-01127114>.
- [13] Little, J. D; Murty, K. G; Sweeney, D. W; & Karel, C : *An algorithm for the traveling salesman problem*. Operations research, ISEN école d'ingénieurs, 1993.
- [14] Ana.busic@inria.fr, Conception d'algorithmes et applications (LI325), Cours 7 et 8: Algorithmes Gloutons.
- [15] INFOB321-Théorie des graphes : JP Leclercq, Cours de Théorie des Graphes et réseaux de Petri (Septembre 2008).