Procédure de séparation et évaluation (Branch & Bounds)

25 février 2020 – B. Colombel

Procédure de Séparation et d'Évaluation

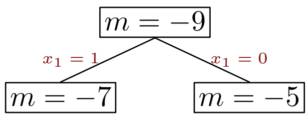
Exercice 1 : **Principes généraux**
On considère la fonction objectif :

$$\min z = -x_1 - x_2 - x_3 - x_4 + 3x_1x_2 - 3x_1x_3 + x_1x_4 + 2x_2x_3 - 2x_2x_4 + 2x_3x_4$$

avec $x_1,x_2,x_3,x_4 \in \{0;1\}$.

On fait les choix suivants pour la procédure arborescente :
- on sépare sur une variable x_i que l'on fixe à 1 ou à 0.
- on prend les variables dans l'ordre x_1, x_2, x_3 , x_4

On obtient le premier « branchement » :



Sur la figure 1 se trouve le demi-arbre des combinaisons possibles.

1. Compléter le demi-arbre en effectuant un parcours en *profondeur* d'abord (fils gauche d'abord).
2. Compléter le demi-arbre en effectuant un parcours en *meilleur* d'abord (fils gauche d'abord).

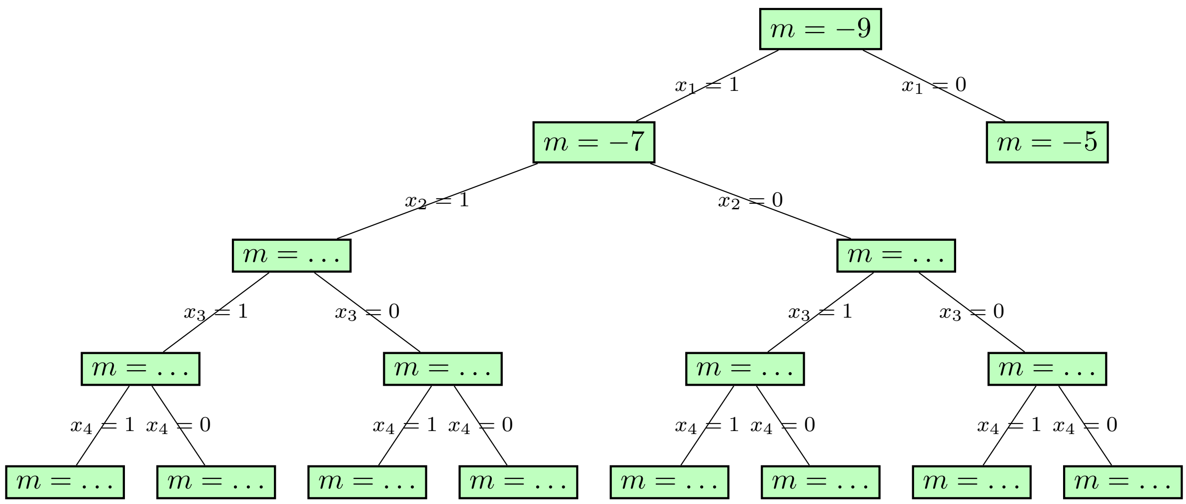


Figure 1 – Exercice 1 : Principe du Branch & Bound

Exercice 2 : Adapter la méthode précédente au problème :

$$\max z = x_1 + 2x_2 + x_3 - 2x_1x_2 + 3x_1x_3 - 3x_2x_3$$

avec $x_1,x_2,x_3 \in \{0;1\}$.

Exercice 3 : **Problème d'affectation**
On a n tâches à affecter à n ouvriers avec une unique tâche par ouvrier. Le problème est spécifié par la donnée d'une matrice D de n lignes et n colonnes. Les lignes représentent les ouvriers, les colonnes les tâches. Pour chaque coefficient D_{ij}, on a donc :

$$D_{ij} = \text{ coût d'affectation de l'ouvrier } i \text{ à la tâche } j$$

Le problème est donc d'affecter les tâches aux ouvriers de façon à minimiser les coûts engendrés.
On peut modéliser ce problème par un programme linéaire puis résoudre ce programme linéaire à l'aide d'un « solveur ». Ici, on va le résoudre à l'aide d'une procédure arborescente définie de la façon suivante :

— **Séparation n-aire** : au niveau i, on fait toutes les affectations de tâches (non encore affectées) à l'ouvrier i. Cela crée, au plus, n sommets
— **Évaluation (borne inférieure)** : chaque tâche doit de toute façon être exécutée. Le coût minimum pour une tâche j est donc le coefficient D_{ij} minimum dans la colonne j de D. Ceci étant vrai pour toute tâche j, il suffit pour avoir une borne inférieure du coût total, d'ajouter les minimums de chaque colonne relative à chaque tâche. Cette borne inférieure se calcule donc facilement (algorithme polynomial) en chaque nœud de l'arborescence de recherche
— **Élagage** : pour un sous-problème donné (en un nœud de l'arborescence de recherche) lorsque les minimums de chaque colonne, sont sur des lignes distinctes, on a dans ce cas une solution optimale pour le sous-problème traité et donc une solution (une borne supérieure) du problème. Utiliser la borne supérieure et la borne inférieure pour élaguer l'arbre de recherche
— **Effectuer une recherche en meilleur d'abord** : explorer le sommet pendant de l'arborescence de recherche de meilleure borne inférieure c'est-à-dire la plus petite.

Traiter le cas suivant :

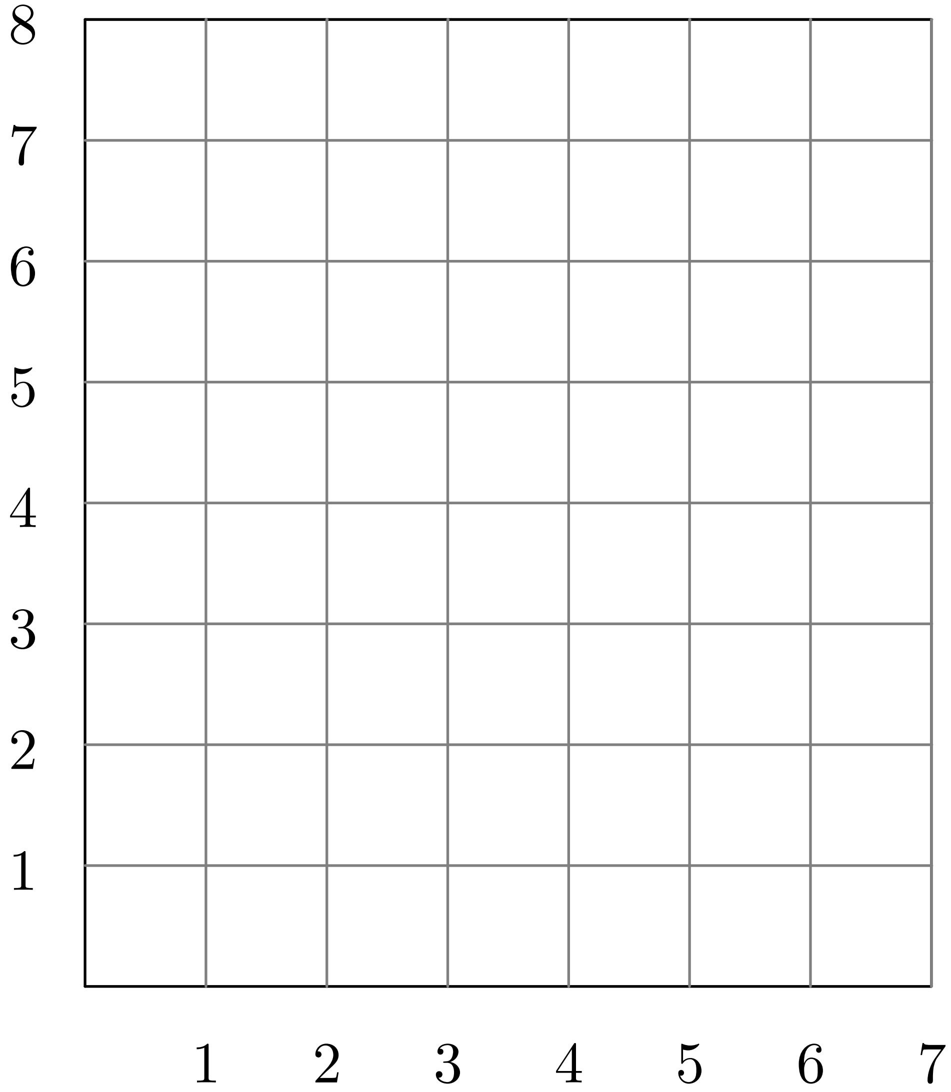
D (coûts)	Tâche 1	Tâche 2	Tâche 3	Tâche 4
Ouvrier 1	8	3	1	5
Ouvrier 2	11	7	1	6
Ouvrier 3	7	8	6	8
Ouvrier 4	11	6	4	9

Résolution de PLNE

Exercice 4 : **Résolution graphique**
On considère le PLNE suivant :

$$\max z = 4x_1 + x_2$$
$$\text{s.c.} \quad \left| \begin{array}{l} 7x_1 + \ \ x_2 \leq 36 \\ \ x_1 + 4x_2 \leq 22 \\ \ x_1 \qquad\quad \in \mathbb{N} \\ \qquad\quad x_2 \in \mathbb{N} \end{array} \right.$$

1. Dans le premier repère ci-dessous, représenter le domaine des solutions réalisables.
2. Déterminer graphiquement l'optimum de la relaxation linéaire et l'optimum entier.
3. On décide de séparer par rapport à la variable x_1. Tracer l'arbre de la résolution du PLNE en utilisant la méthode de *séparation et évaluation.*
4. Même question si l'on sépare suivant x_2.


8
7
6
5
4
3
2
1
1 2 3 4 5 6 7

Exercice 5 : **problème du sac à dos**

On considère le problème du sac-à-dos de capacité $W = 22$ avec les objets suivants :

Objets	a	b	c	d	e
poids	3	4	3	3	13
utilité	12	12	9	15	26

1. Trouver manuellement la solution optimale.
2. Modéliser ce problème comme un programme linéaire en nombres entiers.

Définition 1 (Efficacité)
On appelle efficacité d'un objet le rapport de sa valeur sur son poids.
Plus la valeur de l'objet est importante par rapport à ce qu'il consomme, plus l'objet est efficace.

Algorithme de résolution de la relaxation linéaire

Relaxation linéaire du cas à dos

Entrées : W : capacité du sac à dos ;
w : liste des poids des objets **triés par ordre décroissant d'efficacité**
Initialisation : $i \leftarrow 1$;
$w_{\text{dispo}} \leftarrow W$

début
    tant que $w_{dispo} \geqslant w[i]$ **faire**
        $x[i] \leftarrow 1$;
        $w_{\text{dispo}} \leftarrow w_{\text{dispo}} - w[i]$;
        $i \leftarrow i + 1$;
    fin
    $x[i] \leftarrow w_{\text{dispo}} \div w[i]$;
    tant que $i < n$ **faire**
        $i \leftarrow i + 1$;
        $x[i] \leftarrow 0$;
    fin
fin

3. Calculer manuellement la solution de la relaxation linéaire de ce programme (on remplace $x_i \in \{0; 1\}$ par $x_i \in [0; 1]$).
4. Indiquer pourquoi la solution trouvée à la question précédente est une borne supérieure de la solution optimale.
5. Décrire l'arbre de branchements pour ce problème en résolvant à chaque sommet de l'arbre la relaxation linéaire et en branchant, si elle existe, sur la variable non entière (à gauche en la mettant à 1 et à droite en la mettant à zéro).
6. Dans chaque sommet de l'arbre, indiquer une borne supérieure et une borne inférieure.
7. Avec un parcours en profondeur, fils gauche d'abord, indiquer l'ordre dans lequel les sommets sont visités et les sommets qui ne sont pas visités.
8. Même question avec un parcours en profondeur, fils droit d'abord.
9. Même question avec un parcours en profondeur, fils avec la meilleure borne supérieure d'abord.
10. Même question avec un parcours en profondeur, fils avec le plus petit écart entre la borne inférieure et la borne supérieure (gap) d'abord.