Exercice 1 – Branch and bound et résolution graphique

La méthode arborescente dite "meilleur d'abord" est utilisée pour résoudre des programmes en nombres entiers. On résout la relaxation continue du problème. La séparation se fait sur la variable qui a la plus grande partie fractionnaire dans la solution en cours. Par exemple, si $x = a + \frac{b}{c}$ (avec $b < c$), on sépare en $x \leq a$ et $x \geq a+1$.

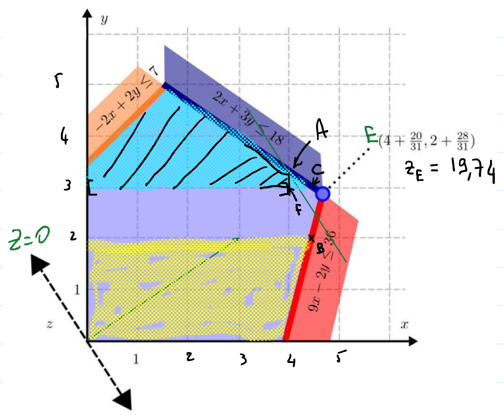
Développer l'arborescence pour l'exemple suivant :

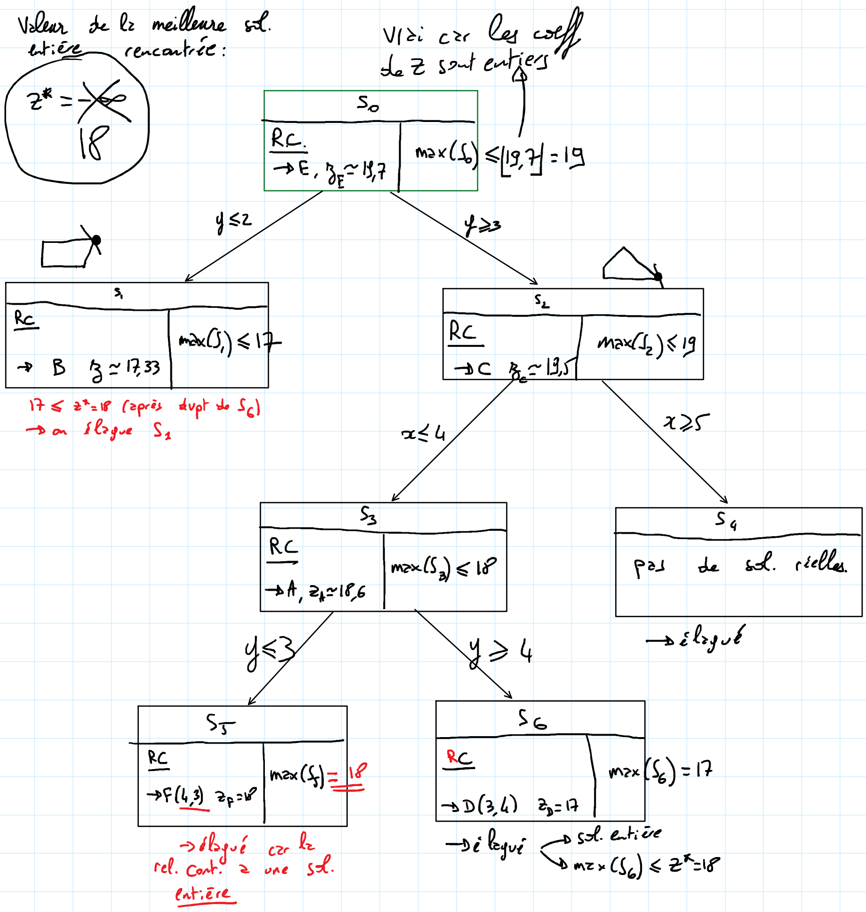
$$\begin{array}{lrrcr} \max z = & 3x & +2y & & \\ \text{s.c.} & -2x & +2y & \leq & 7 \\ & 2x & +3y & \leq & 18 \\ & 9x & -2y & \leq & 36 \\ & x, & y & \in & \mathbb{N} \end{array}$$

A $(4; 3+\frac{1}{3})$ $z = 18 + \frac{2}{3}$
B $(4+\frac{4}{9}; 2)$ $z = 17 + \frac{1}{3}$
C $(4+\frac{1}{2}; 3)$ $z = 19 + \frac{1}{2}$
D $(3; 4)$ $z = 17$

$F(4,3)$ $z_F = 18$

$z = 3x + 2y$





Conclusion : valeur optimale = 18, sol. optimale x=4 et y=3

Exercice 2 – Le problème du sac-à-dos

Pour embarquer dans l'avion, votre valise ne doit pas peser plus de B kg. Il ne va pas être possible d'emporter les n objets que vous vouliez y placer et vous devez faire des choix. Vous avez pesé chaque objet et p_i est le poids de l'objet i, $i = 1, \ldots, n$. Pour optimiser le contenu de la valise, vous attribuez à chaque objet une note d'"utilité" entre 1 et 20 : c_i est la note attribuée à l'objet i, $i = 1, \ldots, n$. Vous voulez maximiser l'utilité globale de la valise, égale à la somme des utilités des objets emportés.

objet i → utilité : c_i
objet i → poids : p_i

1) $x_i = \begin{cases} 1 & \text{si on prend l'objet } i \\ 0 & \text{sinon} \end{cases}$

"Sac-à-dos"
$c_i, p_i \geq 0$
$\sum^{n} p_i > B$

$\left(\max \sum_{i=1}^{n} c_i x_i \right.$

$$
(KP)\begin{cases}
\max & \sum_{i=1}^{m} c_i x_i \\
\text{s.c.} & \sum_{i=1}^{m} p_i x_i \leq B \\
& x_i \in \{0,1\} \quad \forall i \in \{1,\dots,m\}
\end{cases}
\qquad
\boxed{\begin{array}{l} c_i, p_i \geq 0 \\ \sum_{i=1}^{m} p_i > B \end{array}}
$$

1. Écrire le programme si le poids maximum autorisé est $B = 17$ kg et vous avez 4 objets de poids respectifs 3, 7, 9 et 6 kg et d'utilités respectives 8, 18, 20, 11.

$$
(KP_1)\begin{cases}
\max & 8x_1 + 18x_2 + 20x_3 + 11x_4 \\
\text{s.c.} & 3x_1 + 7x_2 + 9x_3 + 6x_4 \leqslant 17 \\
& x_i \in \{0,1\} \quad \forall i \in \{1,\dots,4\}
\end{cases}
$$

3) $\frac{8}{3} \simeq 2{,}66 \geqslant \frac{18}{7} \simeq 2{,}57 \geqslant \frac{20}{9} \simeq 2{,}22 \geqslant \frac{11}{6} \simeq 1{,}83$

Les variables sont déjà triées dans l'ordre des ratios décroissants

$\overline{KP_1}$: relaxat° continue de (KP_1)

$\hookrightarrow x_i \in [0;1]$ $\qquad\qquad$ $\hookrightarrow x_i \in \{0,1\}$

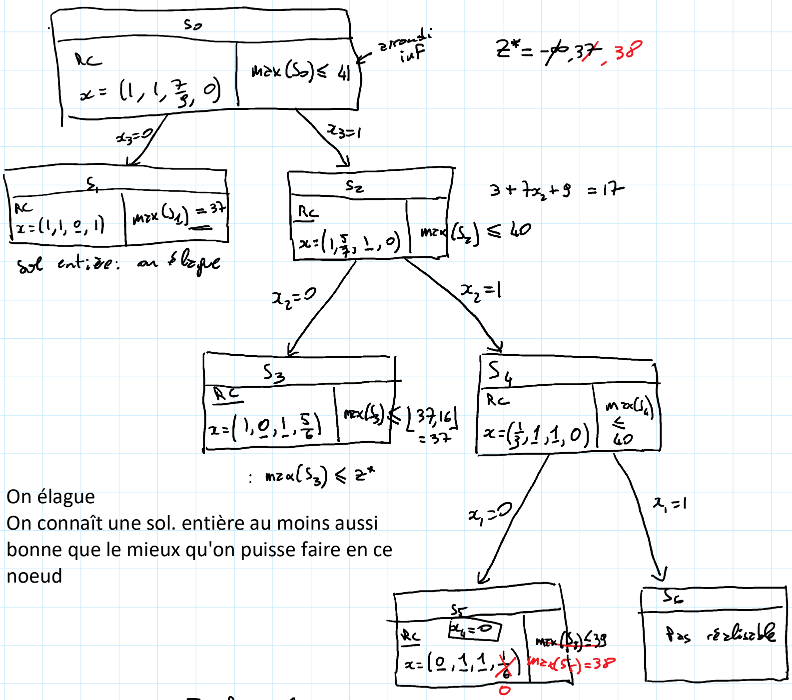
Sol. opt de $(\overline{KP_1})$

$$\boxed{x_1 = 1 \qquad x_2 = 1 \qquad x_3 = \frac{7}{9} \qquad x_4 = 0}$$

$9x_3 = 7$

$x_3 = \frac{7}{9}$

o)

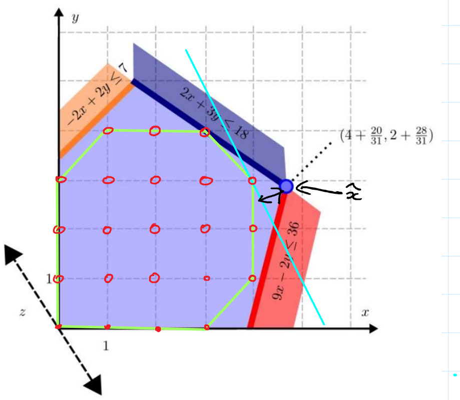


On élague
On connaît une sol. entière au moins aussi bonne que le mieux qu'on puisse faire en ce noeud

En fait, la contrainte devient $6x_4 < 1$
$\Rightarrow$ La variable entière x_4 peut être fixée à 0.

Conclusion : max(KP1) = 38, sol. opt. x=(0,1,1,0)

Coupes et inégalités valides



On appelle inégalité valide toute inégalité (linéaire) qui n'exclut aucun point entier.
Parmi les inégalités valides, on cherche des "coupes".
Une "coupe" est une inégalité valide qui exclut la solution de la relaxation continue du problème (sans toutefois exclure aucun point entier).

Exercice 3 – Problème de séparation par l'inégalité de couverture

On considère l'instance du problème du sac-à-dos (KP) en 0-1 définie à la question 1. de l'exercice précédent. Une **couverture** $\mathcal{C}$ est un ensemble d'objets qui ne tient pas dans le sac : $\mathcal{C} \subseteq \{1, \ldots, n\}$ est une couverture si $\sum_{i \in \mathcal{C}} p_i > B$. L'inégalité suivante est appelée **inégalité de couverture** :

$$\sum_{i \in \mathcal{C}} x_i \leq |\mathcal{C}| - 1$$

Ex de couvertures

Inégalité de couv. associée

$\mathcal{C}_1 = \{1, 2, 3, 4\}$

$x_1 + x_2 + x_3 + x_4 \leq 3$

→ n'exclut pas la sol. de la RC $(1, 1, \frac{7}{9}, 0)$
→ inégalité "redondante"

$\mathcal{C}_2 = \{1, 2, 3\}$

$x_1 + x_2 + x_3 \leq 2$

$1 + 1 + \frac{7}{9} > 2$

→ exclut $(1, 1, \frac{7}{9})$ → coupe.

$\mathcal{C}_3 = \{2, 3, 4\}$

$x_2 + x_3 + x_4 \leq 2$

$1 + \frac{7}{9} + 0 \leq 2$

→ redondante.

1. Prouver que l'inégalité $\sum_{i \in \mathcal{C}} x_i \leq |\mathcal{C}| - 1$ est une inégalité valide pour (KP) ($\forall \mathcal{C}$ couverture).

Soit $\mathcal{C}$ une couverture. On a $\sum_{i \in \mathcal{C}} p_i > B$.

Soit x une solution 0-1 de (KP)

On veut montrer que x vérifie l'inégalité associée à $\mathcal{C}$:

$$\sum_{i \in \mathcal{C}} x_i \leq |\mathcal{C}| - 1. \qquad (1)$$

Comme x est réalisable pour (KP) : $\sum_{i=1}^{n} p_i x_i \leq B$

Par l'absurde, supposons que x ne vérifie pas (1) :

$$\sum_{i \in \mathcal{C}} x_i \geq |\mathcal{C}| \underset{x \in \{0,1\}^n}{\Longrightarrow} x_i = 1 \quad \forall i \in \mathcal{C}$$

On a donc $\sum_{i=1}^{n} p_i x_i \underset{(p_i \geq 0)}{\geq} \sum_{i \in \mathcal{C}} p_i x_i = \sum_{i \in \mathcal{C}} p_i > B$

ce qui contredit l'admissibilité de x.

En conclusion x vérifie l'inégalité (1).

2. Le **problème de séparation** consiste à chercher $\mathcal{C}$ telle que l'inégalité associée est l'inégalité de couverture la moins respectée par $\hat{x}$, c'est-à-dire $\mathcal{C}$ telle que $\sum_{i\in\mathcal{C}} \hat{x}_i$ est maximal (expliquez pourquoi).

Dire que $\hat{x}$ ne vérifie pas l'inégalité de couverture

$\iff$

$$\sum_{i\in\mathcal{C}} \hat{x}_i > |\mathcal{C}| - 1$$

Pour que l'écart entre gauche et droite soit maximal, comme $\hat{x}$ est une donnée, on doit trouver une couverture $\mathcal{C}$ t.q. $\sum_{i\in\mathcal{C}} \hat{x}_i - |\mathcal{C}| + 1$ soit maximal.

3. Modéliser le problème de séparation par un programme linéaire en variables 0-1 avec $z_i = 1$ si et seulement si $i \in \mathcal{C}$.
Note : dans cette question $\hat{x}$ est une donnée. Quelle condition doit vérifier la valeur optimale de ce programme pour que la solution soit bien une couverture ?

$$z_i = \begin{cases} 1 & \text{si } i \in \mathcal{C} \\ 0 & \text{sinon} \end{cases}$$

$$(\text{SEP}) \begin{cases} \max \ \text{ecart} = \sum_{i=1}^{m} \hat{x}_i z_i - \sum_{i=1}^{m} z_i + 1 \\ \qquad = 1 + \sum_{i=1}^{m} (\hat{x}_i - 1) z_i \\ \text{s.c.} \ \sum_{i=1}^{m} p_i z_i \geq B + 1 \\ z_i \in \{0, 1\} \quad \forall i \end{cases}$$

4) $$(\text{SEP}_1) \begin{cases} \max \ 1 - \frac{2}{9} z_3 - z_4 \\ \text{sc} \ 3z_1 + 7z_2 + 9z_3 + 6z_4 \geq 18 \\ z_i \in \{0,1\} \quad \forall i \end{cases}$$

$\hat{x} = (1, 1, \frac{7}{9}, 0)$

$\to$ Après B&B, on trouve $z^* = (1, 1, 1, 0)$

qui correspond à la couverture $\mathcal{C}^* = \{1, 2, 3\}$

Rq : L'écart pour (SEP_1) est de $1 - \frac{2}{9} = \frac{7}{9} > 0$ $\Rightarrow$ cette inégalité exclut bien $\hat{x}$ $\to$ c'est une coupe

Exercice 5 – Programmation quadratique 0-1

On veut montrer que tout programme quadratique en variables 0-1 peut se ramener à un programme linéaire en 0-1. Pour cela, on va "linéariser" le programme en remplaçant chaque terme quadratique $x_i x_j$ par une seule variable 0-1 : z_{ij}.

1. Donner les contraintes à ajouter pour que le programme linéaire obtenu soit équivalent au programme quadratique initial, c'est-à-dire qu'on a bien $z_{ij} = x_i x_j$ pour tout couple (i, j).
2. Une telle transformation est-elle possible si les variables sont continues (dans $\mathbb{R}$) ?

$$\begin{cases} \max \ z = 2x_1 + 3x_2 - 5x_3 + 2x_1x_2 + 3x_1x_3 - 2x_2x_3 \\ \text{sc} \quad x_1 + x_2 - 2x_3 \leq 12 \\ x_1, x_2, x_3 \in \{0, 1\} \end{cases}$$

$$x_i \in \{0,1\} \iff x_i^2 = x_i$$

1) On remplace chaque pdt $x_i x_j$ par une var. z_{ij} en s'assurant que $z_{ij} = x_i x_j$, en ajoutant des contraintes <u>linéaires</u> qui assurent $z_{ij} = x_i x_j$

x_i	x_j	$x_i x_j$
0	0	0
0	1	0
1	0	0
1	1	1

On doit avoir :

x_i	x_j	z_{ij}
0	0	0
0	1	0
1	0	0
1	1	1

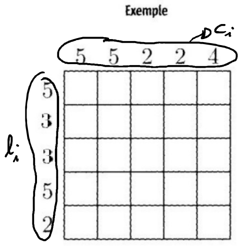
$$\begin{cases} z_{ij} = x_i x_j \\ x_i, x_j \in \{0,1\} \end{cases} \iff \begin{cases} z_{ij} \le x_i \\ z_{ij} \le x_j \\ z_{ij} \ge x_i + x_j - 1 \\ x_i, x_j \in \{0,1\},\ \cancel{z_{ij} \in \{0,1\}}\ \ z_{ij} \ge 0 \end{cases}$$

$$\left(z_{ij} \le \frac{x_i + x_j}{2} \right)$$

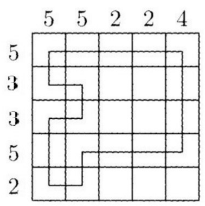
Tour de ville

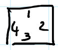
RÈGLE DU JEU :
Retrouvez un circuit d'un seul tenant passant par le centre de certaines cases du plan. Le circuit est composé de segments horizontaux ou verticaux joignant les centres de deux cases voisines. Les nombres extérieurs à la grille indiquent le nombre de cases visitées (lors du tour) dans la ligne ou la colonne de l'indice.

Exemple



Solution





Modélisation

$$x_{ij} = \begin{cases} 1 & \text{si on utilise } (i,j) \\ 0 & \text{sinon} \end{cases} \qquad y_{ijk} = \begin{cases} 1 & \text{si on traverse la porte } k \text{ de la case } (i,j) \\ 0 & \text{sinon} \end{cases}$$

Contraintes Projections

$\forall i \quad \sum_j x_{ij} = l_i$

$\forall j \quad \sum_i x_{ij} = c_i$

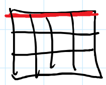
Contrainte liante

$$2x_{ij} = \sum_{k=1}^{4} y_{ijk}$$

Continuité du trait

$$y_{ij1} = y_{i-1,j,3} \quad \forall i \in [\![2, I]\!] \quad \forall j \in [\![1, J]\!]$$

bord



$$y_{1,j,1} = 0 \qquad \forall j$$

+ autres bords

Pb à régler : éliminat° des sous-tours qui peuvent apparaître