

# TP Méthode approchées UMIN215

Bruno Y., Chloé T., Julien D. et Rémi F.

14 avril 2015

## Partie Théorique

### Exercice 1 - Sur le problème de la couverture sommet minimale : trois approches différentes

1. a) On a :

$$x_i = \begin{cases} 0 & \text{si } x_i \notin S \\ 1 & \text{si } x_i \in S \end{cases}$$

On peut donc déduire que le  $(PL)$  donne bien une solution optimale.  $z$  compte bien le nombre de sommets dans  $S$ . Si  $(v_r, v_s) \in E$ , on a  $x_r$  et  $x_s$  qui valent 1 : il faut qu'il y ait au moins un sommet couvre l'arête.

- b) On ne peut pas avoir  $x_s + x_r = 1$  car  $v_r$  et  $v_s$  peuvent tout deux appartenir à  $S$ .
- c) Soit une solution optimale, alors elle respecte les contraintes du  $(PL)$  et correspond à un  $z$  minimum : elle correspond donc à une solution du  $(PL)$ .
- d) Supposons qu'il existe  $(v_r, v_s) \in E$  tel que  $x_r < \frac{1}{2}$  et  $x_s < \frac{1}{2}$ , alors on aurait  $x_r + x_s < 1$ , ce qui viole une contrainte. Donc  $x_r \geq \frac{1}{2}$  ou  $x_s \geq \frac{1}{2}$ .
- e) Soit  $I$  une instance quelconque, alors on a  $\rho = \frac{\max(A(I))}{\text{Opt}(I)}$ . Par ailleurs, on sait que :

$$\text{Opt}(I) = S_{ILP}(I) \tag{1}$$

$$S_{LP}(I) \leq S_{ILP}(I) \tag{2}$$

On peut déduire de (2) que  $\frac{1}{S_{LP}(I)} \geq \frac{1}{S_{ILP}(I)}$ . Donc on arrive à l'inéquation suivante :  $\frac{A(I)}{S_{LP}(I)} \geq \frac{A(I)}{\text{Opt}(I)}$ . Notons que  $A(I)$  est la solution obtenue en arrondissant la solution de  $S_{LP}(I)$ .

On peut remarquer que puisqu'il existe  $i$  tel que  $x_i \geq \frac{1}{2}$ , donc  $2x_i \geq 1$ . On peut donc écrire :

$$A(I) = \sum_{i, x_i \geq \frac{1}{2}} 1 \leq \sum_{i \in S^*} 2x_i \leq 2 \sum_{i=1}^n x_i = 2\text{Opt}(I). \tag{3}$$

On déduit finalement que  $\frac{A(I)}{Opt(I)} = 2$  et  $\rho = \frac{\max(A(I))}{Opt(I)} \leq 2$  Montrons que  $\rho = 2$  en considérons le cas suivant :

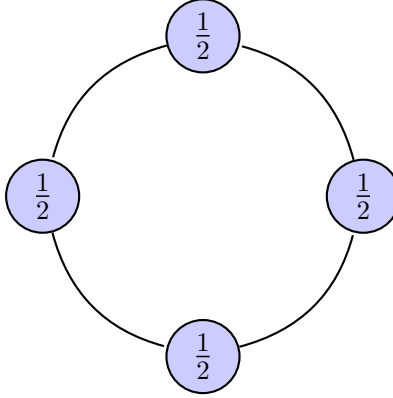


FIGURE 1 – Résultat donné par l'Algorithme 1 pour le problème Vertex cover.

On peut remarquer qu'ici,  $A(I) = 4$  alors que  $Opt(I) = 2$ .

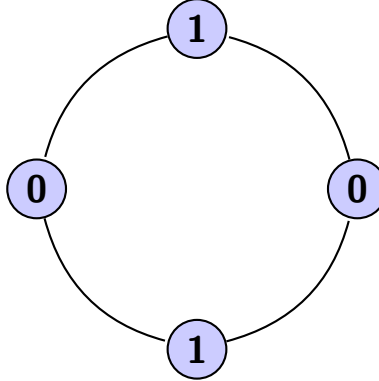


FIGURE 2 – Résultat optimal pour le problème Vertex cover.

On peut donc finalement conclure que  $\rho = \frac{\max(A(I))}{Opt(I)} = 2$ .

- f) i) Pour obtenir notre nouvel programme linéaire ( $PL'$ ), il suffit simplement d'introduire les poids  $w_i \in \mathbf{N}^*, i \in \{1, \dots, n\}$  et de poser  $z = \sum_{j=1}^n w_i x_i$ .

$$PL' \begin{cases} \min z = \sum_{j=1}^n w_i x_i \\ x_r + x_s \geq 1, \forall \{v_r, v_s\} \in E \\ x_j \in \{0, 1\}, j \in \{1, \dots, n\} \end{cases}$$

- ii) On peut remplacer le programme linéaire en nombre entiers ( $PL'$ ) par un autre utilisant une matrice  $A$ . Pour un graphe  $G = (V, E)$ , la matrice  $A$  utilisé dans le programme linéaire en nombres entiers possède les caractéristiques suivantes :
- $A$  à  $2 * |E|$  lignes et  $|V|$  colonnes.
  - Chaque ligne apparaît deux fois dans la matrice.
  - La matrice  $A$  est constituée uniquement de 1 et il n'y a que deux 1 par ligne.

$$PL' \left\{ \begin{array}{l} \min z = \sum_{j=1}^n w_j x_j \\ A * \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \geq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \\ x_j \in \{0, 1\}, j \in \{1, \dots, n\} \end{array} \right.$$

- iii) A terminer..
2. a) Dans les pires des cas, les deux sommets de chaque arêtes sont pris :  $A(I) \geq 2 * m$  avec  $m = |E|$ . De plus, il y a toujours, au moins un sommet par arête dans la solution optimale :  $Opt(I) \geq n$ . On peut donc écrire que  $\frac{1}{Opt(I)} \leq n$  et donc que  $\frac{A(I)}{Opt(I)} \leq 2$ .
- b) Considérons le graphe suivant :

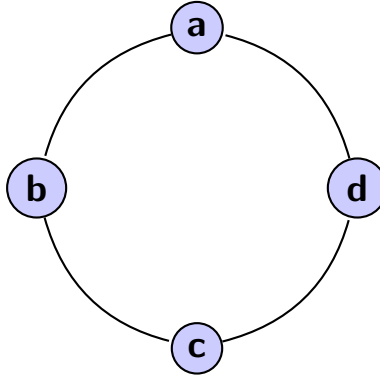


FIGURE 3 – Exemple d'instance pour lequel l'Algorithme 2 atteint la borne de deux.

On a donc  $E' = (\{a, b\}, \{a, d\}, \{b, c\}, \{c, d\})$ .

étape	$E'$	arête sélectionné	C
1	$\{ \{a,b\}, \{a,d\}, \{b,c\}, \{c,d\} \}$	$\{a,b\}$	$\{ \{a,b\} \}$
2	$\{ \{c,d\} \}$	$\{c,d\}$	$\{ \{a,b\}, \{c,d\} \}$
3	$\emptyset$	$\emptyset$	$\{ \{a,b\}, \{c,d\} \}$

On à ici  $A(I) = 4$  alors que  $Opt(I) = 2$  donc  $\rho = \frac{\max(A(I))}{Opt(I)} = 2$ .

- c) On commence par prendre les  $\frac{k!}{k}$  sommets de degrés  $k$  (les triangles), puis les  $\frac{k!}{k-1}$  sommets de degrés  $k-1$ , etc.. On va donc prendre en tout  $k! \sum_{i=0}^k \frac{1}{k-i} = k! \sum_{i=1}^k \frac{1}{i} = k! * H_k$ , avec  $H_k$  la k-ième somme partielle de la série harmonique. L'optimal est de prendre que les  $k!$  sommets de degrés  $k$  (les carrés). On a donc pour un  $k$  fixé,  $\frac{A(I)}{Opt(I)} = H_k$ . Or si  $k \rightarrow \infty$ ,  $H_k$  diverge et tend vers  $\infty$ .

## Exercice 2 - Sur le problème du couplage maximum de poids maximum : un début d'étude polyédrale sur le problème de couplage

Dans cet exercice, nous travaillerons sur un graphe  $G = (V, E)$  avec  $|V| = n$  et  $|E| = m$ . Chaque arête  $(i, j) \in E$  possède un poids que l'on not  $w_{i,j}$ .

1. On peut considérer le  $(PL)$  suivant afin de modéliser le problème :

$$PL \begin{cases} \max z = \sum_{(i,j) \in E} w_{i,j} x_{i,j} \\ \forall i \in \{1, \dots, n\}, \sum_{j \in V(i)} x_{i,j} \leq 1 \\ x_j \in \{0, 1\} \end{cases}$$

2. Considérons que le graphe de la figure 2 est le suivant :

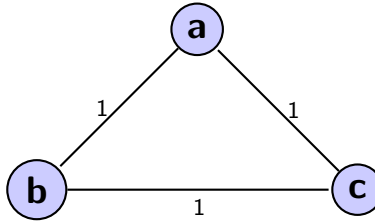


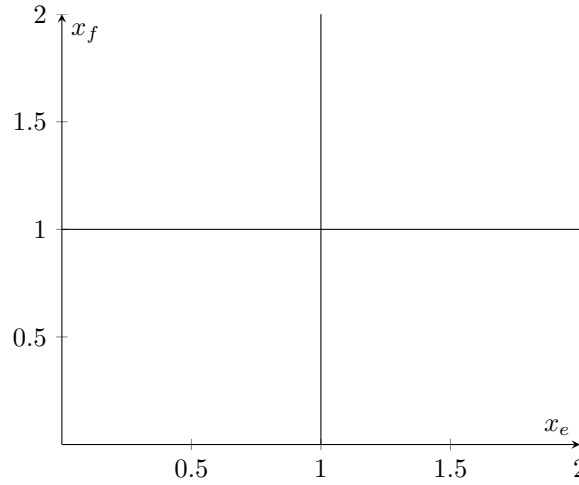
FIGURE 4 – Graphe représentant la figure 2.

Les équations des contraintes sont donc les suivantes :

$$\begin{cases} x_{a,b} + x_{a,c} \leq 1 \\ x_{a,c} + x_{b,c} \leq 1 \\ x_{a,b} + x_{b,c} \leq 1 \end{cases}$$

3. Une solution optimale du programme linéaire en nombre entiers est  $z = 1$  avec  $x_{a,b} = 1$ ,  $x_{b,c} = 0$  et  $x_{a,c} = 0$ .
4. Une solution pour le même programme relaxé est  $z = \frac{3}{2}$  avec  $x_{a,b} = \frac{1}{2}$ ,  $x_{b,c} = \frac{1}{2}$  et  $x_{a,c} = \frac{1}{2}$ .
5. La formulation est mauvaise puisqu'avec une solution de  $LP$ , on ne peut pas retrouver une solution de  $ILP$ .

6. Pour la figure 2,  $\frac{|S|-1}{2} = 1$ . On rajoute donc la contrainte suivante :  $\sum_{(i,j) \in E} x_{i,j} \leq 1$ . On remarque qu'avec cette contrainte,  $z(ILP) = z(LP) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$ . Cette solution ne peut être le résultat du PL car ce n'est pas un point extrême, puisqu'il est combinaison linéaire des solutions  $(1, 0, 0)$ ,  $(0, 1, 0)$  et  $(0, 0, 1)$ .
7. a) Pour la première figure, les solutions admissibles sont  $x_e = x_f = 1$  ;  $x_e = 1$  et  $x_f = 0$  ;  $x_e = 0$  et  $x_f = 1$  ;  $x_e = x_f = 0$

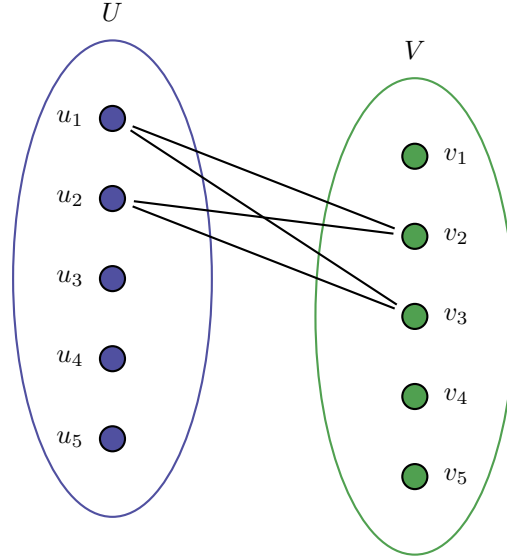


Le polytope associé est défini par les points  $(0, 0)$ ,  $(0, 1)$ ,  $(1, 0)$  et  $(1, 1)$ .

Pour la deuxième figure, les différentes combinaisons possibles sont :  $x_e = 0, x_f = 0, x_g = 0$  ;  $x_e = 0, x_f = 0, x_g = 1$  ;  $x_e = 0, x_f = 1, x_g = 0$  ;  $x_e = 1, x_f = 1, x_g = 0$  et  $x_e = 1, x_f = 0, x_g = 0$ . Le polytope associé est une pyramide définie par une base carré  $((0, 0, 0), (0, 1, 0), (1, 0, 0))$  et  $(1, 1, 0)$  et un sommet  $((0, 0, 1))$ .

Le cas du triangle les combinaisons possibles sont :  $x_e = 0, x_f = 0, x_g = 0$  ;  $x_e = 0, x_f = 0, x_g = 1$  ;  $x_e = 0, x_f = 1, x_g = 0$  et  $x_e = 1, x_f = 0, x_g = 0$ . Le polytope associé est une pyramide définie par une base triangulaire  $((0, 0, 0), (0, 1, 0) \text{ et } (1, 0, 0))$  et un sommet  $((0, 0, 1))$ .

- b) Si  $\lambda_1 M_1 + \lambda_2 M_2 + \lambda_3 M_3 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$   
 $\lambda_1(1, 0, 0) + \lambda_2(0, 1, 0) + \lambda_3(0, 0, 1) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$   
alors  $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{2}$  et donc  $\sum_{i=1}^3 \lambda_i = \frac{3}{2} > 1$ . Le point n'appartient donc pas au polytope.
- c) Soit un point  $x$  du polytope fractionnaire  $FM$ , alors  $\exists \lambda_1, \dots, \lambda_m, \exists M_1, \dots, M_m$  tel que  $\sum_{i=1}^m \lambda_i = 1$  et  $\sum_{i=1}^m \lambda_i M_i = x$ .  
Un graphe biparti n'est composé que de cycles pairs. Soit une solution optimale de LP.



Considérons le cycle  $(u_1, v_2, u_2, v_3, u_1)$ .

On a forcément  $x_{u_1, v_2} + x_{u_2, v_3} = x_{u_1, v_3} + x_{u_2, v_2}$  puisque l'on prend une arête sur deux le long du cycle. Donc on peut prendre  $x'_{u_1, v_2} = x_{u_1, v_2} + x_{u_1, v_3}$ ,  $x'_{u_2, v_3} = x_{u_2, v_3} + x_{u_2, v_2}$  et  $x'_{u_2, v_2} = x'_{u_1, v_3} = 0$ .

On répétant cette opération, on "casse" tous les cycles tout en gardant la même valeur pour la fonction objectif.

### Exercice 3 - Sur le problème de la coupe maximum

Dans cet exercice, nous travaillerons avec un graphe  $G = (V, E)$  avec  $|V| = n$  et  $|E| = m$ .

1. La boucle *while* ne peut s'effectuer plus de  $n$  fois puisqu'à chaque tour de boucle, nous déplaçons un sommet de l'autre côté de la coupe. Puis pour trouver un sommet qui augmente la valeur de la coupe, on parcourt potentiellement tout les sommet  $O(n)$  puis l'on doit calculer le nombre d'arcs dans la coupe et hors de la coupe  $O(n^2)$ . On obtient en tout du  $O(n^3)$ .
2. Soit  $(Y_1, Y_2)$  la coupe rendue par l'algorithme. Montrons par l'absurde que  $\forall v \in Y_1, |Voisins(v) \cap Y_1| \leq |Voisins(v) \cap Y_2|$ . Supposons qu'il existe un sommet  $v \in Y_1, |Voisins(v) \cap Y_1| > |Voisins(v) \cap Y_2|$ , alors on pourrait déplacer ce sommet de l'autre côté de la coupe et augmenter la valeur de celle-ci de  $|Voisins(v) \cap Y_1| - |Voisins(v) \cap Y_2| > 0$ . Ce qui contredit le fait que  $(Y_1, Y_2)$  soit le résultat de l'algorithme.

Il nous reste à montrer que l'Algorithme est 2-approché. Posons  $C$  la coupe optimale et  $(Y_1, Y_2)$  la coupe rendue par l'algorithme. On peut donc écrire que :

$$|C| = |\{\text{Arêtes de } Y_1 \text{ dans } Y_1 \in C\}| + |\{\text{Arêtes de } Y_1 \text{ dans } Y_2 \in C\}| + |\{\text{Arêtes de } Y_2 \text{ dans } Y_2 \in C\}| \\ \leq \dots$$

3. Considérons le graphe suivant :

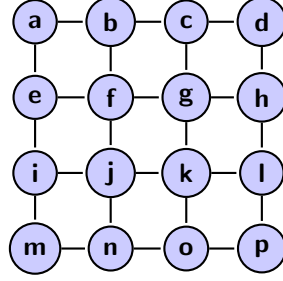


FIGURE 5 – Exemple pour lequel l'algorithme atteint sa borne.

Déroulement de l'algorithme :

étape	S	V - S	Taille de la coupe (S, V-S)
1	$\emptyset$	$\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$	0
2	$\{a\}$	$\{b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$	2
3	$\{a, b\}$	$\{c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$	3
4	$\{a, b, c\}$	$\{d, e, f, g, h, i, j, k, l, m, n, o, p\}$	4
5	$\{a, b, c, e\}$	$\{d, f, g, h, i, j, k, l, m, n, o, p\}$	5
6	$\{a, b, c, e, i\}$	$\{d, f, g, h, j, k, l, m, n, o, p\}$	6
7	$\{a, b, c, e, i, p\}$	$\{d, f, g, h, j, k, l, m, n, o\}$	8
8	$\{a, b, c, e, i, p, l\}$	$\{d, f, g, h, j, k, m, n, o\}$	9
9	$\{a, b, c, e, i, p, l, h\}$	$\{d, f, g, j, k, m, n, o\}$	10
10	$\{a, b, c, e, i, p, l, h, o\}$	$\{d, f, g, j, k, m, n\}$	11
11	$\{a, b, c, e, i, p, l, h, o, n\}$	$\{d, f, g, j, k, m\}$	12

L'algorithme s'arrête et renvoi donc cette coupe. Cependant on remarque que si l'on prend la coupe  $(S, V - S)$  avec  $S = \{a, c, f, h, i, k, n, p\}$ , on obtient une coupe de taille 24.

#### Exercice 4 - Sur le problème de Partition

#### Exercice 5 - Sur le problème du sac à dos simple

Soit  $w_1, \dots, w_n$  les poids de  $n$  objets,  $b$  la capacité du sac à dos. On cherche à trouver  $T \subseteq \{1, \dots, n\}$  tel que  $\sum_{i \in T} w_i$  soit maximum.

1. a) Trier les  $w_i, i \in \{1, \dots, n\}$  peut se faire en  $O(n \log(n))$ . On répète la boucle  $n$  fois et les éléments à l'intérieur s'effectuent en  $O(1)$ . On peut donc conclure que l'algorithme s'effectue en  $O(n \log(n))$ .

- b) On peut distinguer deux cas :
- Si  $\sum_{i=1}^n w_i \leq b$ , alors on peut mettre tous les objets dans le sac à dos. Donc  $T = \{1, \dots, n\}$  et le cas se résout trivialement.
  - Si par contre  $\sum_{i=1}^n w_i > b$ , on peut démontrer par l'absurde qu'il existe  $j$  tel que  $cost(T) + w_{j+1} > b$ . Supposons qu'un tel élément n'existe pas, alors à chaque étape de la boucle, l'élément  $i$  est ajouté à  $T$ . En particulier si  $i = n$ , on rajoute l'élément  $n$  à  $T$  puisque  $cost(T) + w_n = \sum_{i=1}^n w_i \leq b$ , ce qui est absurde.

Montrons maintenant que dans le deuxième cas, on a forcément  $cost(T) \geq \frac{b}{2}$ . Etudions les trois cas possibles :

- On ne peut pas avoir  $j = 0$  puisque  $b \geq w_1$ .
- Si  $j = 1$ , alors on a  $w_1 + w_2 > b$  or on sait que  $w_1 \geq w_2$  et donc  $2w_1 \geq w_1 + w_2 > b$ , d'où  $cost(T) = w_1 \geq \frac{b}{2}$ .
- Si  $j \geq 2$ , on a alors  $T = \{1, \dots, j\}$  d'après la démonstration précédente.  $cost(T) = \sum_{i \in T} w_i \leq b$  et  $cost(T) + w_{j+1} > b$ . Supposons par l'absurde que  $cost(T) < \frac{b}{2}$  alors on aurait :

$$w_{j+1} + \frac{b}{2} > cost(T) + w_{j+1} > b$$

On conclut que  $w_{j+1} > \frac{b}{2}$ , et puisque  $j \geq 2$  alors  $cost(T) \geq w_1 + w_2 \geq 2w_{j+1} > b$ . C'est une contradiction.

- c) Montrons que l'algorithme admet une performance relative de deux. Soit  $I$  une instance quelconque du problème du sac à dos. Alors  $Opt(I) \leq b$ . Notre algorithme nous assure une solution  $A(I) \geq \frac{b}{2}$  d'après la question précédente. On a alors :

$$\frac{A(I)}{Opt(I)} \geq \frac{b}{2} * \frac{1}{b} = \frac{1}{2}$$

2.

**Exercice 6 - Programmation dynamique**

**Exercice 7 - Sur le produit matriciel**

**Exercice 8 - Résolution numérique**

**Exercice 9 - Seuil d'approximation pour le problème Bin Packing**

**Exercice 10 - Seuil d'approximation pour le problème de la coloration de sommets (reps. d'arêtes)**



### Exercice 11 - Comparaisons branch and bound and branch and cut

1. On peut tracer les droites correspondantes aux contraintes de  $PL_0$  :

$$\begin{aligned} &— y_1 = 5 - \frac{3}{2}x \\ &— y_2 = \frac{17}{5} - \frac{2}{5}x \end{aligned}$$

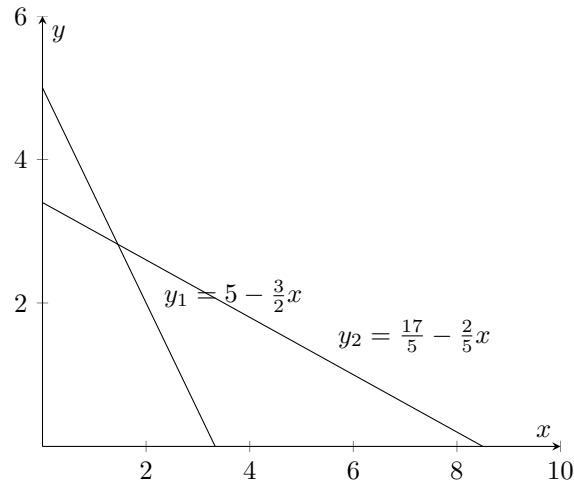
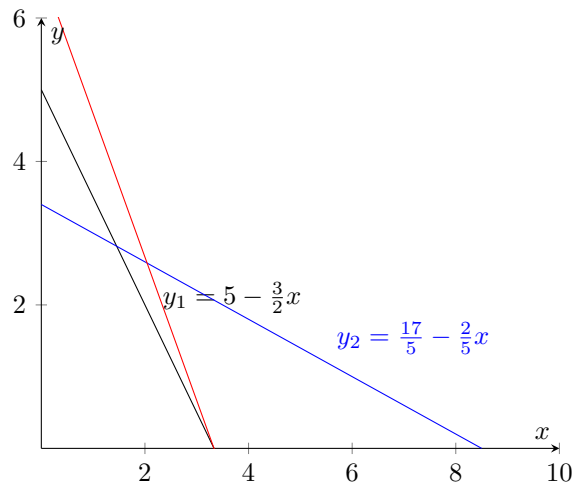


FIGURE 6 – Représentation des équations de  $PL_0$ .

Le polytope associé aux équations de  $PL_0$  est donc celui formé par les points :  $P_0(0, 0)$ ,  $P_1(0, \frac{17}{5})$ ,  $P_2(\frac{16}{11}, \frac{31}{11})$  et  $P_3(\frac{10}{3}, 0)$

2. On peut tracer la fonction objective :

$$— y = -2x + k, k \in \mathbf{R} \text{ (rouge)}$$



La solution optimale pour  $PL_0$  est donc  $x_1 = \frac{10}{3}$  et  $x_2 = 0$  avec  $z = \frac{20}{3}$ .

3. On commence par reprendre le programme linéaire donné :

$$PL_0 \begin{cases} \max z(x_1, x_2) = 2x_1 + x_2 \\ 2x_1 + 5x_2 \leq 17 \\ 3x_1 + 2x_2 \leq 10 \\ x_1, x_2 \geq 0 \end{cases}$$

On ajoute les variables d'écarts  $x_3$  et  $x_4$  pour obtenir  $PL_1$  :

$$PL_1 \begin{cases} \max z(x_1, x_2) = 2x_1 + x_2 \\ 2x_1 + 5x_2 + x_3 = 17 \\ 3x_1 + 2x_2 + x_4 = 10 \\ x_1, x_2 \geq 0 \end{cases}$$

On obtient donc le tableau initial :

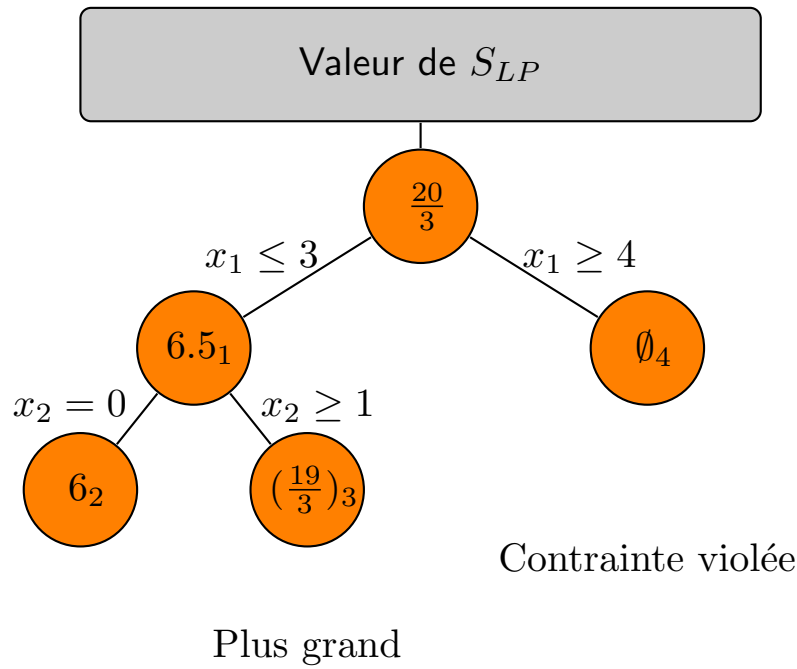
			2	1	0	0
			$x_1$	$x_2$	$x_3$	$x_4$
0	$x_3$	17	2	5	1	0
0	$x_4$	10	3	2	0	1
	$z$	0	-2	-1	0	0

La variable entrante est  $x_1$  et la variable sortante est  $x_4$ . Le tableau final est donc :

			2	1	0	0
			$x_1$	$x_2$	$x_3$	$x_4$
0	$x_3$	$\frac{31}{3}$	0	$\frac{11}{3}$	1	$-\frac{2}{3}$
2	$x_1$	$\frac{10}{3}$	1	$\frac{2}{3}$	0	$\frac{1}{3}$
	$z$	$\frac{20}{3}$	0	$\frac{1}{3}$	0	$\frac{2}{3}$

La solution optimale est donc  $x_1 = \frac{10}{3}$  et  $x_2 = 0$  avec  $z = \frac{20}{3}$ .

4. a) On effectue notre Branch and Bound dans l'ordre :  $x_1$  puis  $x_2$ . Puisque dans la solution optimale du (PL),  $x_1 = \frac{20}{3}$ , on a  $x_1 \leq 3$  ou que  $x_1 \geq 4$ .
- Si l'on considère que  $x_1 \leq 3$ , la solution optimale devient  $z = \frac{13}{2}$  avec  $x_1 = 3$  et  $x_2 = \frac{1}{2}$ . On a ensuite,  $x_2 = 0$  ou  $x_2 > 0$ .
    - Si l'on considère que  $x_2 = 0$ , alors la solution optimale devient  $z = 6$  avec  $x_1 = 3$  et  $x_2 = 0$ .
    - Si l'on considère que  $x_2 \geq 1$ , alors la solution optimale devient  $z = \frac{19}{3} \approx 6.33$  avec  $x_1 = \frac{8}{3}$  et  $x_2 = 1$ . On ne développe pas plus cette branche puisque l'on a déjà trouvé une solution avec  $z = 6$ .
    - Si l'on considère que  $x_1 \geq 4$ , la deuxième contrainte est violée.
- L'arbre du Branch and bound récapitulatif est donné plus bas.



b) Puisque que l'on a  $\frac{31}{3} = \frac{11}{3}x_2 + x_3 - \frac{2}{3}x_4$ , on peut déduire que :  $\frac{11}{3}x_2 - \frac{2}{3}x_4 \geq \frac{1}{3}$ .

Puisque l'on a  $\frac{10}{3} = x_1 + \frac{2}{3}x_2 + \frac{1}{3}x_4$ , on peut donc déduire que  $x_4 = 10 - 3x_1 - 2x_2$ .

On obtient finalement la contrainte :  $2x_1 + 5x_2 \geq 7$ .

Le tableau final donne :

			2	1	0	0	0
			$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
0	$x_3$	10	0	0	1	0	1
2	$x_1$	$\frac{36}{11}$	1	0	0	$\frac{5}{11}$	$\frac{2}{11}$
1	$x_2$	$\frac{1}{11}$	0	1	0	$\frac{-2}{11}$	$\frac{-3}{11}$
	z	$\frac{73}{11}$	0	0	0	$\frac{8}{11}$	$\frac{1}{11}$

On choisi l'équation  $\frac{36}{11} = x_1 + \frac{5}{11}x_4 + \frac{2}{11}x_5$ , on déduit donc  $\frac{5}{11}x_4 + \frac{2}{11}x_5 \geq \frac{3}{11}$ . Ce qui revient à écrire par l'équation précédente que  $\frac{36}{11} - x_1 \geq \frac{3}{11}$  et donc que  $x_1 \leq 3$ .

			2	1	0	0	0	0
			$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
0	$x_3$	$\frac{17}{2}$	0	0	1	$\frac{-5}{2}$	0	$\frac{11}{2}$
0	$x_4$	$\frac{3}{2}$	0	0	0	$\frac{5}{2}$	1	$\frac{-11}{2}$
1	$x_2$	$\frac{1}{2}$	0	1	0	$\frac{1}{2}$	0	$\frac{-3}{2}$
2	$x_1$	3	1	0	0	0	0	1
	$z$	$\frac{13}{2}$	0	0	0	$\frac{1}{2}$	0	$\frac{1}{2}$

On sélectionne l'équation  $\frac{1}{2} = x_2 + \frac{1}{2}x_4 - \frac{3}{2}x_6$ . Donc on déduit que  $\frac{1}{2} \leq \frac{-3}{2}x_6$

## Partie Pratique

### 2.1 Programmation dynamique

### 2.2 Branch and Bound

### 2.3 Comparaisons entre un algorithme de complexité exponentielle et un FP-TAS