Université Abdelmalek Essaadi
Faculté Polydisciplinaire de Larache A.U : 2017-2018

Cours de Programmation linéaire et Recherche Opérationnelle

Pr. Abdelghni LAKEHAL

SMI S5

Table des matières

Chapitre 1

Programmation linéaire

Sommaire

Problème

On veut acheter 4 types d'alimentations A, B, C et D qui ont chacun une teneur en calories et en vitamines comme suite :

Types 1	A	B	C	D
Calories	2	1	0	1
Vitamines	3	4	3	5
Coût	2	2	1	8

But :

Le consommateur cherche à minimiser le coût sachant qu'il a besoin d'au moins 12 calories et 7 vitamines.

Modélisation :

En notant x_1, x_2, x_3 et x_4 les quantités de types A, B, C et D respectivement achetés. On abouti au problème de minimisation suivant :

$$\begin{cases} \text{Minimiser} \quad 2x_1 + 2x_2 + x_3 + 8x_4 \\ \text{S.c} \\ 2x_1 + x_2 + x_3 \geq 12 \\ 3x_1 + 4x_2 + 3x_3 + 5x_4 \geq 7 \\ x_1, x_2, x_3, x_4 \geq 0. \end{cases}$$

Formulation :

Un programme linéaire écrit sous sa forme canonique, s'écrit sous la forme :

$$(PL)\begin{cases} \text{Max} \quad z = f.x \\ \text{S.c} \\ Ax \leq b \\ x \geq 0. \end{cases}$$

f : la fonction objective (fonction coût)

$A = (A_{ij})_{\substack{1\leq i\leq m \\ 1\leq j\leq n}} \in \mathbb{R}^{mn}$: matrice de contraintes
$b = (b_i)_{1\leq i\leq m}$: second membre des contraintes
$x = (x_j)_{1\leq j\leq n} \in \mathbb{R}^n$: variable

- On note $K = \{x \in \mathbb{R}^n,\ Ax = b, x \geq 0\}$ l'ensemble des solutions **réalisables** (vérifient les contraintes) de (PL).
- K est un polyèdre de $\mathbb{R}^n$, K peut être vide ou non, borné ou non.
- Soit $x^* \in \mathbb{R}^n$, x^* est solution optimale de (PL) si et seulement si $f(x^*) \geq f(x), \forall x \in K$.

Note :

- Si $x^* \in K$, x^* est dite solution réalisable optimale de (PL) et on note par $v(PL) = f(x^*)$ la valeur de (PL).
- Si $K = \emptyset$ on note $v(PL) = -\infty$
- Si K est non vide $v(PL) \in \mathbb{R} \cup +\infty$
- Si K est non borné $v(PL) = +\infty$

1.1 Forme standard d'un programme linéaire (PL)

La forme standard de (PL) s'écrit sous la forme :

$$(PL)\begin{cases} \text{Max} \quad z = f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0. \end{cases}$$

Contraintes égalités

$$\text{Soit } i \in \{1, 2, ..., m\} \quad A_i x = b_i \quad \Leftrightarrow \quad \begin{cases} A_i x \leq b_i \\ A_i x \geq b_i. \end{cases}$$

Contraintes inégalités

Soit $i \in \{1, 2, ..., m\}$
Si $A_i x \leq b_i$
Introduisons la variable x_{n+1} dite la variable d'écart on trouve :
$x_{n+1} = b_i A_i x$ i.e $A_i x + x_{n+1} = b_i$
Si $A_i x \geq b_i$
On prend $x_{n+1} = A_i x - b_i$ car x_{n+1} doit être toujours positif.
Si $\exists j \in \{1, 2, ..., n\}, x_j < 0$ on prend $y_j = -x_i > 0$.
S'il s'agit d'une variable x_j de signe quelconque, on introduit deux variables positives x_j^1 , x_j^2 tells que $x_j = x_j^1 - x_j^2$.

Exemple. 1 *Etant donné un programme linéaire suivant :*

$$\begin{cases} \text{Max} \quad x_1 - x_2 \\ \text{S.c} \\ -2x_1 + x_2 \geq 3 \\ x_1 + x_2 \leq 1 \\ x_1 \geq 0 \ \textit{et} \ x_2 \ \textit{qlq} \ . \end{cases}$$

La variable x_2 de signe qlq, donc on introduit deux variables positives x_2^1 , x_2^2 avec $x_2 = x_2^1 - x_2^2$.
le problème prend alors la forme suivante :

$$\begin{cases} \text{Max} \quad x_1 - x_2^1 + x_2^2 \\ \text{S.c} \\ -2x_1 + x_2^1 - x_2^2 - x_3 = 3 \\ x_1 + x_2^1 - x_2^2 + x_4 = 1 \\ x_1, x_2^1, x_2^2, x_3, x_4 \geq 0. \end{cases}$$

Théorème. 1 *L'ensemble $K = \{x \in \mathbb{R}^n, Ax = b, x \geq 0\}$ des solutions réalisables d'un programme linéaire est convexe.*

Démonstration : à titre d'exercice.

Théorème. 2 *On suppose l'ensemble des solutions réalisables K est non vide et borné.*

1. *La valeur de problème (PL) est atteint en au moins un point extrême de K.*
2. *Si la valeur de (PL) est atteint en au moins deux points extrêmes de K, alors le pb (PL) admet une infinité de solution optimales.*

Démonstration : à titre d'exercice.

Théorème. 3 *Etant donnée $I \subset \{1, \ldots, n\}$ tel que les colonnes $A^i (i \in I)$ de la matrice A soient linéairement indépendantes. S'il existe $|I|$ réels strictement positifs noté y_i $(i \in I)$ tq $\sum_{i \in I} A^i y_i = b$,*
Alors le point $x \in \mathbb{R}^n$ définie par $x_i = y_i$ $(i \in I)$ et $x_i = 0$ si $i \notin I$ est un point extrême de K.

Démonstration : à titre d'exercice

Théorème. 4 *Si x est un point extrême de K, alors les colonnes de A associées au composantes strictement positives sont linéairement indépendants.*

Démonstration :à titre d'exercice.

1.2 Algorithme primal de simplex

1.2.1 Principe de la méthode simplex

La résolution de (PL) se fait en se déplaçant de solution de base réalisable non optimale en solution de base réalisable et optimale.

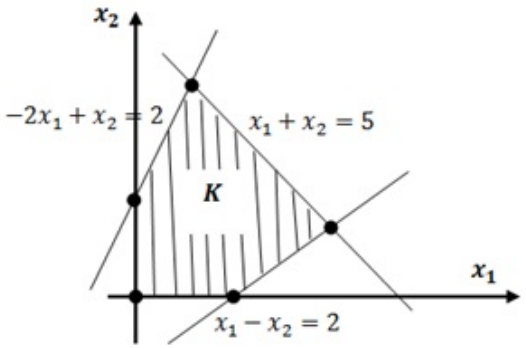
Exemple. 2

$$\begin{cases} \text{Max} \quad -x_1 + x_2 \\ \text{S.c} \\ -2x_1 + x_2 \geq 2 \\ x_1 - x_2 \leq 2 \\ x_1 + x_2 \leq 5 \\ x_1, x_2 \geq 0. \end{cases}$$

Le schéma ci-dessous représente le domaine réalisable de programme linéaire ci-dessus, limité par ses contraintes.
La méthode simplex consiste à trouver un cheminement d'un point extrême à un autre point extrêmes voisin c.-à-d. définir une suite de points x^1, x^2, ..., x^p $p \in \mathbb{N}^*$ vérifient $f(x^i) \geq f(x^{i-1})$ $\forall i \in \{2, 3, \ldots, p\}$ et que $f(x^p) = f(x^*) \geq f(x)$ $\forall x \in K$.

La forme standard de (PL) :

$$\begin{cases} \text{Max} \quad -x_1 + x_2 \\ \text{S.c} \\ -2x_1 + x_2 + x_3 = 2 \\ x_1 - x_2 + x_4 = 2 \\ x_1 + x_2 + x_5 = 5 \\ x_1, x_2, x_3, x_4, x_5 \geq 0. \end{cases}$$



Etant donné un programme linéaire écrit sous la forme

$$(PL)\begin{cases} \text{Max} \quad z = f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0. \end{cases}$$

Où b est le second membre supposé positif ou nul.
La matrice contrainte A est d'ordre $m \times n$.

Définition. 1 *On appelle* **base** *tout sous-matrice carrée régulière* $(m \times m)$ *extrait de* A *(il existe au moins une, puisque* $rg(A) = m$ *).*

Soit I *une base. Alors en permutant les colonnes de* A*, on peut toujours mettre* A *sous la forme* $A = [A^I, A^{\bar{I}}]$ *où* $A^{\bar{I}}$ *est la sous matrice formée par les colonnes de* A *qui ne sont pas dans la base.*
De même on peut partitionner la variable x *en* $[x_I, x_{\bar{I}}]$ *et la fonction coût* f *en* $[f^I, f^{\bar{\bar{I}}}]$ *comme il montre la Figure 1.1.*

Pour tout base I, on peut réécrire la forme standard en exprimant x_I (variable de base) en fonction de variables hors-base $x_{\bar{I}}$. Tout solution de (PL) vérifie $Ax = b$ et par suite :

$$\begin{aligned} [A^I, A^{\bar{I}}].\begin{pmatrix} x_I \\ x_{\bar{I}} \end{pmatrix} &= A^I.x_I + A^{\bar{I}}.x_{\bar{I}} \\ &= b \end{aligned} \tag{1.1}$$

On appel solution de base (associé à la base A^I), la solution particulière de 1.1 obtenue on faisant $x_{\bar{I}} = 0$. On peut alors déterminer x_I de façon unique par la résolution du système :

$$A^I.x_I = b \text{ soit : } x_I = (A^I)^{-1}.b$$

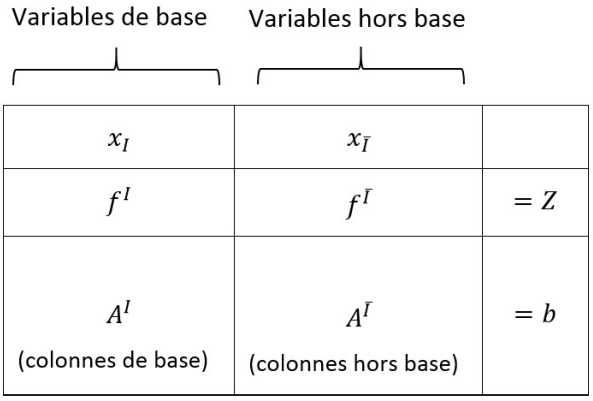


FIGURE 1.1 –

Une solution de base est dite ***réalisable*** si $x_I \geq 0$, autrement dit si : $(A^I)^{-1}.b \geq 0$.
Une base correspondant à une solution de base réalisable est appelée base réalisable.
Une solution de base est dite ***dégénéré*** si le vecteur x_I à des composantes nulles.
On a

$$x_I = (A^I)^{-1}.b - (A^I)^{-1}.A^{\bar{I}}.x_{\bar{I}}$$

Posons

$$t = (A^I)^{-1}.b \text{ et } T = (A^I)^{-1}.A$$

Donc

$$T^{\bar{I}} = (A^I)^{-1}.A^{\bar{I}}$$

Donc x_I s'écrit sous la forme :

$$x_I = t - T^{\bar{I}}.x_{\bar{I}}$$

L'expression nouvelle de la fonction objective est alors :

$$\begin{aligned} f.x &= [f^I, f^{\bar{I}}].\begin{pmatrix} x_I \\ x_{\bar{I}} \end{pmatrix} \\ &= f^I.x_I + f^{\bar{I}}.x_{\bar{I}} \end{aligned} \tag{1.2}$$

où $f = [f^I, f^{\bar{I}}]$

Alors $f.x = f^I(t - T^{\bar{I}}.x_{\bar{I}}) + f^{\bar{I}}.x_{\bar{I}}$

Donc $f.x = f^I.t + (f^{\bar{I}} - f^I T^{\bar{I}}).x_{\bar{I}}$

On note $d = f - f^I T$ donc $d^{\bar{I}} = f^{\bar{I}} - f^I T^{\bar{I}}$

Ainsi $f.x = f^I.t + d^{\overline{I}}x_{\overline{I}}$

Ainsi, la nouvelle formulation du problème (PL) s'écrit sous la forme :

$$(PL)\begin{cases} \text{Max} \quad z = f.x = f^I.t + d^{\overline{I}}x_{\overline{I}} \\ \text{S.c} \\ x_I + T^{\overline{I}}.x_{\overline{I}} = t \\ x_I, x_{\overline{I}} \geq 0. \end{cases}$$

Remarque. 1 *Pour toute base I*

- $d^I = f^I - f^I T^I = f^I - f^I((A^I)^{-1} - A^I) = 0$
- *On note par $x(I)$ la solution de base associée est telle que :* $x(I)_I = t$, $x(I)_{\overline{I}} = 0$, $f.x(I) = f^I.t$ *et* $t = (A^I)^{-1}.b$

Théorème. 5 *Soit un programme linéaire écrit sous forme*

$$(PL)\begin{cases} \text{Max} \quad z = f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0. \end{cases}$$

et I une base et $x(I)$ la solution de base correspondante.

1. Si les coûts des variables $d^{\overline{I}} \leq 0$ alors $x(I)$ est une solution optimale.
2. En cas de non dégénérescence, si $x(I)$ est une solution de base réalisable et s'il existe $s \in \overline{I}$ $d^s > 0$ alors :
   - **a-** Si $T^s \leq 0$ alors la valeur du programme linéaire (PL) est infinie,
   - **b-** Si $\exists i$ tq $T_i^s > 0$, il existe une autre solution de base réalisable de valeur supérieur à $f.x(I)$.

Démonstration :

1°/
Etant donné un programme linéaire (PL) et une base I,
Soit $x = [x^I, x^{\overline{I}}]$ une solution quelconque de (PL) (pas nécessairement de base)
$\forall x \in K$ $Ax = b$ et $f.x = f^I.t + d^{\overline{I}}x_{\overline{I}}$
Comme $d^{\overline{I}} \leq 0$, $x^{\overline{I}} \geq 0$, on a : $f.x \leq f^I.t = f(x(I))$

D'autre part, la valeur $f^I.t$ de z est atteinte par la solution de base réalisable :

$$x^0 = [x_I^0, x_{\overline{I}}^0] = [(A^I)^{-1}.b, 0]$$

Donc $f.x^0$ est la valeur optimale de problème (PL) et x^0 est une solution optimale.

$2°/$
On suppose $\exists s \in \overline{I}$, $d^s > 0$ on fait varier $x_s = \theta \in \mathbb{R}^+$ tout en lissant les autres variables hors-base fixées à 0 (i.e $x_{\overline{I}-s} = 0$) , la solution $x = [x^I, x^{\overline{I}}]$ associe cette base est définie par

$$\begin{cases} x_I = t - T^s\theta \\ x_{\overline{I}} = \theta e_s \end{cases}$$

(e_s est un vecteur de même dimension que $x_{\overline{I}}$ avec toutes ses composantes nulles sauf la composante s égale à 1).
Deux cas peuvent se présenter :

a- /
$T^s \leq 0$ (la colonne T^s a ses composantes toutes négatives)
Alors x_s peut prendre une valeur θ aussi grande que l'on veut, on aura toujours $x_I \geq 0$.
Comme $z = f.x = f.x(I) + d^s\theta$ avec $\theta > 0$
La valeur de z correspondante peut être aussi grand que l'on veut : le problème a un optimum non borné ($+\infty$).

b- /
$\exists i \in \{1, \ldots, m\}$ tq $T_i^s > 0$ (la colonne T^s n'a pas ses composantes toutes négatives)
Dans ce cas, la valeur de θ ne peut être augmentée indéfiniment : la plus grande valeur

$$\widehat{\theta} = \min_{i \in I^+} \frac{t_i}{T_i^s}$$

avec $I^+ = \{i | T_i^s > 0\}$
La nouvelle solution a pour composantes :

$$x_I = t - T^s\theta \geq 0, \quad x_s = \theta \geq 0, \quad x_{\overline{I}-s} = 0$$

et

$$f.\widehat{x} = f.x(I) + d^s\theta \geq f.x(I) \quad \text{avec } \theta \in [0, \widehat{\theta}]$$

1.2.2 Algorithme de simplex

But :
Le but de cet algorithme est de calculer la solution optimale d'un programme linéaire (PL) et la valeur correspondante v(PL).
On note par I une base, $t = (A^I)^{-1}b$, $T = (A^I)^{-1}A$ et $d = f - f^I.T$
Trouver une base réalisable I de (PL).
Calculer $d^{\overline{I}}$.
$v \leftarrow$ vrai (v vrai lorsque la valeur de (PL) est infinie).
Tan que $d^{\overline{I}} \not\leq 0$ et que non v faire.
Choix de la variable x_s entrant en base.
Choisir $s \in \overline{I}$, $d^s > 0$.
Par exemple $d^s = \max_{j \in \overline{I}} d^j$
Calculer T^s.
Si $T^s \leq 0$ alors la solution optimale est non bornée ($v(PL) = +\infty$).
$v \leftarrow vrai$.
Sionon Déterminer la variable x_r qui va sortir de la base.
Calculer t.
Déterminer $r \in I$ tq $\frac{t_r}{T_r^s} = \min\{\frac{t_i}{T_i^s} | i \in I, T_i^s > 0\}$.
Changement de base : $I \leftarrow I - r + s$
Finsi
Finsi
Si $nonv$ alors calculer t,
$v(PL) \leftarrow f^I$, $x_I \leftarrow t$, $x^{\overline{I}} \leftarrow 0$.
Sinon affecter valeur infinie.
Finsi.
Fin tan que.

Changement de base

Etant donné une base I,
$t(I) = (A^I)^{-1}.b$, $T(I) = (A^I)^{-1}.A$ et $f.x(I) = f^I.t + d^{\overline{I}}.x_{\overline{I}}$
Si s est l'indice de la variable qui va entrer en base et r l'indice de variable qui va sortir de la base. On aura une nouvelle base $I' = I - r + s$.
D'après la démonstration du théorème définissant les critères d'arrêt de l'algorithme de simplex
$t(I')_s = \frac{t_r}{T_r^s}$

Exemple. 3 *Soit le problème de maximisation suivant :*

$$\begin{cases} \text{Max} \quad -x_1 + x_2 \\ \text{S.c} \\ -2x_1 + x_2 \leq 2 \\ x_1 - x_2 \leq 2 \\ x_1 + x_2 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

Ce dernier problème peut se présenter sous forme du tableau simplex suivant : Les

x_1	x_2	x_3	x_4	x_5	t	
-2	1	1	0	0	2	L_1
1	-2	0	1	0	2	L_2
1	1	0	0	1	5	L_3
-1	1	0	0	0	0	L_4

variables de base : x_3,x_4,x_5
Les variables hors base : x_1,x_2
La solution associée à cette base est $x = (0, 0, 2, 2, 5)$ qui est n'est pas optimale car les coûts de variables hors base ne sont pas tous négatifs.
La variable dont le coût est strictement positif x_2 entrera en base.
La variable x_3 quittera la base.
Nous effectuons des opérations sur les lignes on trouve le nouveau tableau :
Les coûts de variables hors base ne sont pas tous négatifs, alors la solution corres-

x_1	x_2	x_3	x_4	x_5	t	
-2	1	1	0	0	2	$L'_1 = L_1$
-3	0	2	1	0	6	$L'_2 = L_2 + 2L_1$
3	0	-1	0	1	3	$L'_3 = L_3 - L_1$
1	0	-1	0	0	-2	$L'_4 = L_4 - L_1$

pondant $x = (0, 2, 0, 6, 3)$ n'est pas optimale.
La variable x_1 de coût strictement négatif entrera en base.
Tous les coûts de variables hors-base sont négatifs, la solution correspond
$x^* = (1, 4, 0, 9, 0)$ est une solution de base réalisable optimale
$z(x^*) = v(PL) = -1 + 4 = 3$

Variables artificielles :

x_1	x_2	x_3	x_4	x_5	t	
0	1	1/3	0	2/3	4	$L''_1 = L'_1 + 2L'_3/3$
0	0	1	1	1	9	$L''_2 = L'_2 + L'_3$
1	0	-1/3	0	1/3	1	$L''_3 = L'_3/3$
0	0	-2/3	0	-1/3	-3	$L''_4 = L'_4 - L'_3/3$

On considère un problème linéaire sous sa forme la plus générale :

$$\begin{cases} \text{Max} \quad z = \sum_{j=1}^{p} f^j x_j \\ \text{S.c} \\ \sum_{j=1}^{p} A_i^j x_j \leq b_i \quad i = 1, \ldots, l \\ \sum_{j=1}^{p} A_i^j x_j \geq b_i \quad i = l+1, \ldots, u \\ \sum_{j=1}^{p} A_i^j x_j = b_i \quad i = u+1, \ldots, m \\ x_j \geq 0 \forall j = 1, \ldots, p \\ b_i \geq 0, i = 1, \ldots, m \end{cases}$$

La mise du programme (PL) sous forme standard nécessite l'introduction de u variables d'écart :

$$\begin{cases} \text{Max} \quad z = \sum_{j=1}^{p} f^j x_j \\ \text{S.c} \\ \sum_{j=1}^{p} A_i^j x_j + x_{p+i} = b_i \quad i = 1, \ldots, l \\ \sum_{j=1}^{p} A_i^j x_j - x_{p+i} = b_i \quad i = l+1, \ldots, u \\ \sum_{j=1}^{p} A_i^j x_j = b_i \quad i = u+1, \ldots, m \\ x_j \geq 0 \forall j = 1, \ldots, p+u \\ b_i \geq 0, i = 1, \ldots, m \end{cases}$$

La matrice des contraintes ne possède pas de sous matrice unité d'ordre m, on ajoute $m - l$ colonnes unité dans le seul but de pouvoir démarrer l'algorithme à partir de l'origine.

Cet ajout correspond donc à l'introduction de $m-l$ variables dites artificielles qui doivent être affectés d'un coefficient négatif dans l'objectif suffisamment grand en valeur absolue :

a- Pour pouvoir les éliminer en cours d'algorithme,

b- Aboutir à une solution de base réalisable pour le problème (PL),

c- Et enfin trouver une solution optimale réalisable de (PL).

On posant $n = p + u$ et $k = m - l$.

$$\begin{cases} \text{Max} \quad z = \sum_{j=1}^{p} f^j x_j - M \sum_{j=1}^{k} x_{n+j} \\ \text{S.c} \\ \sum_{j=1}^{p} A_i^j x_j + x_{p+i} = b_i \quad i = 1, \ldots, l \\ \sum_{j=1}^{p} A_i^j x_j - x_{p+i} + x_{n+i-l} = b_i \quad i = l+1, \ldots, u \\ \sum_{j=1}^{p} A_i^j x_j + x_{n+i-l} = b_i \quad i = u+1, \ldots, m \\ x_j \geq 0 \quad \forall j = 1, \ldots, n+k \\ b_i \geq 0, i = 1, \ldots, m \quad M \gg 0 \end{cases}$$

L'objectif peut s'écrire sous forme :

$$z = \sum_{j=1}^{p} f^j x_j - M \sum_{j=1}^{p} x_{n+i}$$

Et $x = (x_i)_{1 \leq i \leq n+k}$ par suite $z = (g + Mh)x$ où $x \in \mathbb{R}_+^{n+k}$
avec

$$g^j = \begin{cases} f^j & j = 1, \ldots, p \\ 0 & j = n+1, \ldots, n+k \end{cases}$$

et

$$h^j = \begin{cases} 0 & j = 1, \ldots, n \\ -1 & j = p+1, \ldots, n+k \end{cases}$$

ainsi le nouveau modèle avec variables artificielles s'écrit donc :

$$(PL) \begin{cases} \text{Max} \quad z = (g + Mh)x \\ \text{S.c} \\ Ax = b \\ x \geq 0 \end{cases}$$

Exemple. 4 *Etant donné le programme linéaire suivant :*

$$\begin{cases} \text{Max} \quad 2x_1 + 3x_2 \\ \text{S.c} \\ x_1 + x_2 \leq 4 \\ -x_1 + x_2 - x_3 \leq -2 \\ x_1 \geq 1 \\ x_1 + x_2 + x_3 = 2 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

La forme standard de (PL) avec variables artificielles s'écrit sous forme :

$$(FS) \begin{cases} \text{Max} \quad 2x_1 + 3x_2 - M(x_7 + x_8 + x_9) \\ \text{S.c} \\ x_1 + x_2 + x_4 = 4 \\ x_1 - x_2 + x_3 - x_5 + x_7 = 2 \\ x_1 - x_6 + x_8 = 1 \\ x_1 + x_2 + x_3 + x_9 = 2 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 \geq 0, M \gg 0 \end{cases}$$

Phase1 :
Détermination d'un point extrême de K c.-à-d. détermination d'une base $I = \{1, \ldots, n\}$, $A^I x_I = b$.
Ou
Prouver que le domaine réalisable K est vide c.-à-d. $\{x \in \mathbb{R}^n, \sum_{j=1}^{n} A_i^j x_j = b, b \geq 0\} = \emptyset$

Phase 2 :
Lorsqu'un point extrême de K été trouvé en phase 1, l'algorithme primal de simplex est appliquée au programme linéaire (PL) pour déterminer une solution de base réalisable optimale.

1.3 Dualité en programmation linéaire

Définition. 2 *On considère un problème linéaire (PL) mis sous forme standard :*

$$(P) \begin{cases} \text{Max} \quad f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0 \end{cases}$$

A est une matrice d'ordre $(m \times n)$.
On définit le dual de (PL) comme suit :

$$(D)\begin{cases} \text{Min} \quad u.b \\ \text{S.c} \\ u.A \geq f \\ (u) \end{cases}$$

avec

$$\begin{cases} m \quad \text{variable} \quad u_1, u_2, \ldots, u_m \\ \text{tq} \\ u.A^j \geq f^j \\ j = 1, \ldots, n \end{cases}$$

Le problème dual (D) peut s'écrire sous forme :

$$(D)\begin{cases} -\text{Max} \quad -b^t.u^t \\ \text{S.c} \\ A^t(-u^t) \leq -f^t \\ (u) \end{cases}$$

Modélisation de divers duaux :

La détermination du dual d'un programme linéaire présenté sous forme canonique

$$(P)\begin{cases} \text{Max} \quad f.x \\ \text{S.c} \\ Ax \leq b \\ x \geq 0 \end{cases}$$

Nécessite son écriture sous forme standard

$$(P)\begin{cases} \text{Max} \quad f.x \\ \text{S.c} \\ Ax + I_d.\begin{pmatrix} x_{n+1} \\ \vdots \\ x_{n+m} \end{pmatrix} = b \\ x \geq, x_{n+i} \geq 0 \quad i = 1, \ldots, m \end{cases}$$

le dual associé

$$(D)\begin{cases} \text{Min} \quad u.b \\ \text{s.c} \\ u.(A + I_d) \geq (f, 0) \\ (u) \end{cases}$$

Soit

$$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A \geq f\\ u.I_d = u \geq 0\end{cases}$$

C'est un programme linéaire sous forme canonique en minimisation.

Note :
En estimant que le nombre d'itération de l'algorithme du simplex est $\Theta(m)$ (m nombre de contrainte). Si A est d'ordre $m \times n$ avec $m \gg n$, il est clair que le résolution du problème dual apporte un gain en nombre d'opération $\Theta\frac{m}{n}$ par rapport à la résolution de (P).

Si on note $\overline{x}$ et $\overline{u}$ les solutions optimales respectives du problème (P) et de son dual (D).

Primal	Dual	Relation d'exclusion
$(P)\begin{cases}\text{Max} \quad f.x\\ \text{S.c}\\ Ax = b\\ x \geq 0\end{cases}$	$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A \geq f\\ (u)\end{cases}$	$(\overline{u}A - f)\overline{x} = 0$
$(P)\begin{cases}\text{Max} \quad f.x\\ \text{S.c}\\ Ax \leq b\\ x \geq 0\end{cases}$	$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A \geq f\\ u \geq 0\end{cases}$	$\overline{u}(A\overline{x} - b) = 0\,; (\overline{u}A - f)\overline{x} = 0$
$(P)\begin{cases}\text{Max} \quad f.x\\ \text{S.c}\\ Ax \geq b\\ x \geq 0\end{cases}$	$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A \geq f\\ u \leq 0\end{cases}$	$\overline{u}(A\overline{x} - b) = 0\,; (\overline{u}A - f)\overline{x} = 0$
$(P)\begin{cases}\text{Max} \quad f.x\\ \text{S.c}\\ Ax = b\\ (x)\end{cases}$	$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A = f\\ (u)\end{cases}$	
$(P)\begin{cases}\text{Max} \quad f.x\\ \text{S.c}\\ Ax \leq b\\ (x)\end{cases}$	$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A = f\\ u \geq 0\end{cases}$	$\overline{u}(A\overline{x} - b) = 0$
$(P)\begin{cases}\text{Max} \quad f.x\\ \text{S.c}\\ Ax \geq b\\ (x)\end{cases}$	$(D)\begin{cases}\text{Min} \quad u.b\\ \text{s.c}\\ u.A = f\\ u \leq 0\end{cases}$	$\overline{u}(A\overline{x} - b) = 0$

Exemple. 5

$$\begin{cases} \text{Max} \quad x_2 - 3x_3 + 2x_5 \\ \text{S.c} \\ x_1 + 3x_2 - x_3 + 2x_5 = 7 \\ -2x_2 + 4x_3 + x_4 = 12 \\ -4x_2 + 3x_3 + 8x_5 + x_6 = 10 \\ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{cases}$$

le dual est :

$$\begin{cases} \text{Min} \quad 7u_1 + 12u_2 + 10x_3 \\ \text{S.c} \\ (u_1, u_2, u_3) \begin{pmatrix} 1 & 3 & -1 & 0 & 2 & 0 \\ 0 & -2 & 4 & 1 & 0 & 0 \\ 0 & -4 & 3 & 0 & 8 & 1 \end{pmatrix} \geq (0, 1, -3, 0, 2, 0) \\ (u_1), (u_2), (u_3) \end{cases}$$

Soit

$$\begin{cases} \text{Min} \quad 7u_1 + 12u_2 + 10x_3 \\ \text{S.c} \\ 3u_1 - 2u_2 - 4u_3 \geq 1 \\ -u_1 + 4u_2 + 3u_3 \geq -3 \\ 2u_1 + 8u_3 \geq 2 \\ u_1, u_2, u_3 \geq 0 \end{cases}$$

Exemple. 6

$$\begin{cases} \text{Max} \quad x_1 - x_2 + x_3 \\ \text{S.c} \\ 3x_1 + 5x_2 - 4x_3 + x_4 \geq 6 \\ x_1 - 3x_2 + 2x_4 \leq 5 \\ 7x_2 + 5x_3 = 1 \\ x_1 \geq 0 \\ (x_2) \\ x_3 \geq 0 \\ x_4 \geq 0 \end{cases}$$

Le problème dual de (P) s'écrit sous forme :

$$\begin{cases} \text{Min} \quad 6u_1 + 5u_2 + u_3 \\ \text{S.c} \\ 3u_1 + u_2 \geq 1 \\ 5u_1 - 3u_2 + 7u_3 = -1 \\ -4u_1 - 5u_3 \geq 1 \\ u_1 + 2u_2 \geq 0 \\ u_1 \leq 0 \\ u_2 \geq 0 \\ (u_3) \end{cases}$$

Théorème. 6 *Etant donné un problème de programme linéaire (P) et (D) son problème dual alors : le dual du dual du programme linéaire (P) est identique à (P).*

Démonstration :

Soit (P) un programme linéaire mis sous forme standard

$$(P) \begin{cases} \text{Max} \quad f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0 \end{cases}$$

Le problème dual associé à (P) est

$$(D) \begin{cases} \text{Min} \quad u^t.b \\ \text{S.c} \\ u^t.A \geq f \\ (u) \end{cases}$$

Alors (D)peut s'ecrire sous forme :

$$(D) \begin{cases} -\text{Max} \quad -u^t.b \\ \text{S.c} \\ u^t.A \geq f \\ (u) \end{cases}$$

par transposition on a :

$$(D) \begin{cases} -\text{Max} \quad -b^t.u \\ \text{S.c} \\ A^t.u \geq f^t \\ (u) \end{cases}$$

Le dual de (D) noté (D') s'écrit alors sous la forme :

$$(D') \begin{cases} -\text{Min} \quad y.f^t \\ \text{S.c} \\ y.A^t = -b^t \\ y \leq 0 \end{cases}$$

si on transpose problème (D') :

$$(D') \begin{cases} \text{Max} \quad f.(-y^t) \\ \text{S.c} \\ A.(-y^t) = b \\ y \leq 0 \end{cases}$$

En posant $x = -y^t$ on aura :

$$(P) \begin{cases} \text{Max} \quad f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0 \end{cases}$$

Théorème. 7 *Etant donné un problème de programmation linéaire (P) et (D) son problème dual alors :*

1. *Si le problème primal (P) (resp (D)) a une solution optimale non borné, alors le dual (D) (resp primal (P)) a un domaine vide.*
2. *Si le problème primal (P) (resp (D)) a une solution optimale borné, alors le dual (D) (resp primal (P)) également, et leurs valeurs sont égales.*

Démonstration :

Soit (P) un programme linéaire écrit sous forme standard :

$$(P) \begin{cases} \text{Max} \quad f.x \\ \text{S.c} \\ Ax = b \\ x \geq 0 \end{cases}$$

On note par K et K' des domaines réalisables associés à (P) et (D).
$K = \{x \in \mathbb{R}^n \setminus Ax = b; x \geq 0\}$
$K' = \{u \in \mathbb{R}^m \setminus u^t.A \geq f\}$
$\forall x \in K, \forall u \in K'$

$$\begin{cases} uA \geq f \\ Ax = b \\ x \geq 0 \end{cases} \Rightarrow \begin{cases} u.A.x \geq f.x \\ Ax = b \\ x \geq 0 \end{cases} \quad \Rightarrow \quad u.b \geq f.x \quad \Rightarrow \quad v(D) \geq v(P)$$

1°/
si (P) admet une solution optimale non borné $\Rightarrow v(p) = +\infty$ donc forcément $v(D) \notin \mathbb{R} \bigcup \{-\infty\} \Rightarrow K' = \emptyset$.

2°/
Si $v(P) \in \mathbb{R}$, $v(P) = f.\overline{x}$.
$\overline{x}$: solution de base réalisable optimale.
Soit I la base réalisable optimale de (P). On note par $d = f - f^I(A^I)^{-1}A$ les coûts de variables.
Posons $\overline{u} = f^I(A^I)^{-1}$
Alors $d = f - \overline{u}A \leq 0 \Rightarrow \overline{u}A \geq f$. $\overline{u}$ est une solution réalisable de (D).
par suit $v(D) \leq \overline{u}b = f^I(A^I)^{-1}b = f^I t = v(P)$
de plus on a montré que $v(P) \leq v(D)$ donc (D) admet une solution optimale bornée $\overline{u}$.
Alors $v(P) = v(D)$

Théorème. 8 *Etant donné un problème de programmation linéaire (P) et (D) son problème dual. On note par $\overline{x}$ et $\overline{u}$ les solutions réalisables optimales associées*

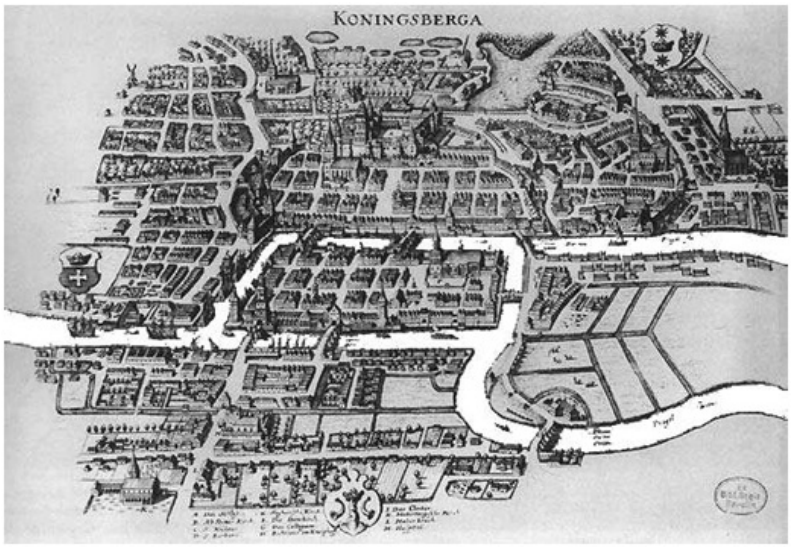
1. $x_j > 0 \Rightarrow \overline{u}A^j = f^j$ *(la contrainte j de (D) est vérifie en égalité)*
2. $u^i > 0$ $A_i\overline{x} = b_i$ *(la contrainte i de (P) est vérifie en égalité)*
3. *si la contrainte i de (P) est vérifié en inégalité, alors $\overline{u}^i = 0$*
4. *si la contrainte j de (D) est vérifié en inégalité, alors $\overline{x}_j = 0$*

Démonstration : à titre d'exercice.

Chapitre 2

La théorie des graphes

Sommaire



Introduction

L'histoire de la théorie des graphes débute peut-être avec les travaux d'Euler au XVIIIe siècle et trouve son origine dans l'étude de certains problèmes, tels que celui des ponts de Königsberg (les habitants de Königsberg se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer deux fois par le même et de revenir à leur point de départ), la marche du cavalier sur l'échiquier ou le problème de coloriage de cartes.

La théorie des graphes s'est alors développée dans diverses disciplines telles que la chimie, la biologie, les sciences sociales. Depuis le début du XXe siècle, elle constitue une branche à part entière des mathématiques, grâce aux travaux de König, Menger, Cayley puis de Berge et d'Erdös.

De manière générale, un graphe permet de représenter la structure, les connexions d'un ensemble complexe en exprimant les relations entre ses éléments : réseau de communication, réseaux routiers, interaction de diverses espèces animales, circuits électriques,...

Les graphes constituent donc une méthode de pensée qui permet de modéliser une grande variété de problèmes en se ramenant à l'étude de sommets et d'arcs. Les derniers travaux en théorie des graphes sont souvent effectués par des informaticiens, du fait de l'importance qu'y revêt l'aspect algorithmique.

2.1 Définitions et premières exemples

Définition. 3 *:*

Un ***graphe non orienté*** *est un couple formé de deux ensembles finis : un ensemble* $X = \{x, x_2, \ldots, x_n\}$ *dont les éléments sont appelés sommets, et un ensemble fini* $A = \{a_1, a_2, \ldots, a_m\}$ *partie de l'ensemble des liens reliant deux éléments de* X*, appelé des arêtes. On notera* $a = \{x, y\}$ *lorsque* a *est l'arête reliant* x *à* y *(ou* y *à* x*). On dit aussi que l'arête* a *est d'extrémités* x *et* y*. les sommets* x *et* y *sont des adjacentes.*

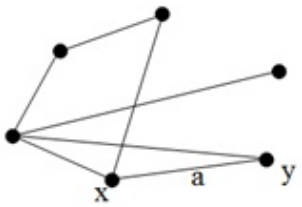


FIGURE 2.1 – Graphe non orienté

On parle de **graphe orienté** quand les arêtes ont un sens :

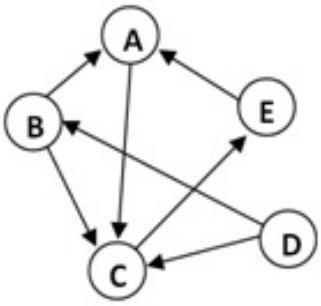
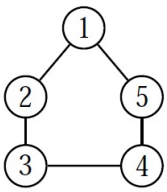


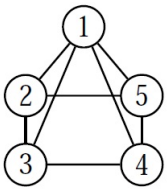
FIGURE 2.2 – Graphe orienté

Terminologie de la théorie des graphes

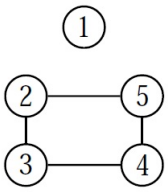
- **Ordre d'un graphe** : le nombre de sommets de ce graphe.
- **Degré d'un sommet** : le nombre d'arêtes reliées à ce sommet.
- **Chaîne** : suite finie de sommets reliés entre eux par des arêtes.
- **Chaîne simple** : chaîne qui n'utilise pas deux fois la même arête.
- **Chaîne eulérienne** : chaîne simple passant par toutes les arêtes d'un graphe.



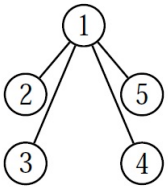
Hamiltonien et eulérien



Hamiltonien et non eulérien



Non hamiltonien et eulérien



Non hamiltonien et non eulérien

FIGURE 2.3 –

- **Chaîne hamiltonienne** : chaîne simple passant par tous les sommets d'un graphe une et une seule fois.
- **Cycle** : chaîne qui revient à son point de départ.
- **Cycle eulérien** : cycle simple passant par toutes les arêtes d'un graphe une et une seule fois.
- **Cycle hamiltonien** : cycle simple passant par tous les sommets d'un graphe une et une seule fois.
- **Chemin** : une suite de sommets reliés par des arcs dans un graphe orienté.
- **Circuit** : un chemin dont les sommets de départ et de fin sont les mêmes.
- **Arbre** : graphe connexe sans cycle simple et sans boucle.
- **Graphe eulérien** : graphe qui possède un cycle eulérien.
- **Graphe hamitonien** : graphe qui possède un cycle hamiltonien.
- **Graphe valué** : graphe où des réels sont associés aux arêtes (dans ce cours les réels sont positives).
- **Longueur d'une chaîne** : nombre des arêtes qui composent la chaîne.
- **Valeur d'une chaîne** : somme des valeurs des arêtes (arcs) d'une chaîne d'un graphe valué.
- **Distance entre deux sommets** : longueur de la plus courte chaîne joignant ces deux sommets.
- **Diamètre d'un graphe** : maximum des distances entre les sommets d'un graphe.

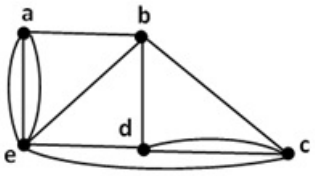
Théorème. 9 *Un graphe simple connexe $G = (V, E)$ est eulérien si et seulement si le nombre de sommet de degré impaire est 0 ou 2.*

Exemple. 7 *Soient les deux graphes suivants :*

G_1 n'est pas eulérien, le degré de ses sommets n'est pas tous pair.
G_2 est eulérien, un cycle eulérien est par exemple : a, b, c, d, c, e, d, b, e, a, e, a

Exemple. 8 ***Le problème des ponts de Königsberg***

(G_1)



(G_2)

La ville de Königsberg en Prusse (maintenant Kaliningrad) comprenait 4 quartiers, séparés par les bras du Prégel. Les habitants de Königsberg se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer deux fois par le même et de revenir à leur point de départ.

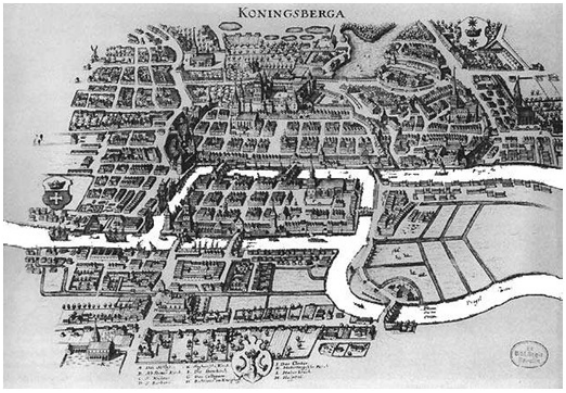
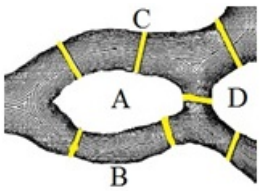
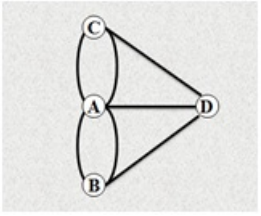


FIGURE 2.4 – Königsberg en 1652

Le plan de la ville peut se modéliser à l'aide du graphe ci-dessous, les quartiers sont représentés par les 4 sommets, les 7 ponts par des arêtes : La question posée





devient alors : ce graphe est-il eulérien ? Le théorème d'Euler répond immédiatement de façon négative aux habitants de Königsberg.

Exemple. 9 *Est-il possible de tracer une courbe, sans lever le crayon, qui coupe chacun des 16 segments de la figure suivante exactement une et une seule fois ?*

Considérons le multi-graphe dont les sommets sont les 6 régions de la figure, a,

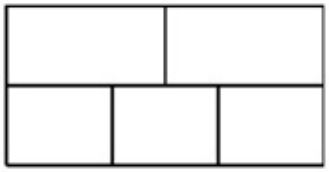


FIGURE 2.5 –

b, c, d, e, f, et dont les arêtes sont les 16 segments qui sont frontières entre les différentes régions.

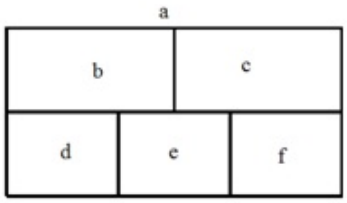
Le problème consiste à construire un cycle eulérien, ce qui est impossible, car le sommet a ; par exemple, est de degré 9.

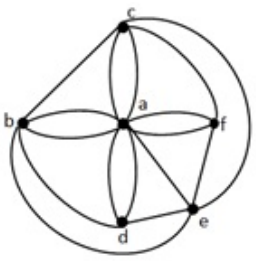
2.2 Représentation non graphique d'un graphe

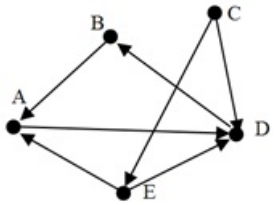
Les graphes peuvent être représentés de diverses manières non graphique dont en citant 3 type de représentations, de deux graphes ci-dessous :

2.2.1 Représentation par un tableau

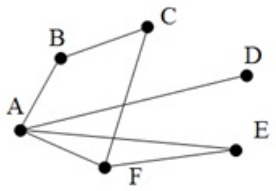
Les croix désignant les arcs (ou arêtes) entre sommets. Elles peuvent être remplacées par des valeurs désignant les longueurs des arcs (ou arêtes). Les tableaux suivants représentent les graphes précédents :







Graphe 1



Graphe 2

→	A	B	C	D	E
A				×	
B	×				
C				×	×
D		×			
E	×			×	

Graphe 1

–	A	B	C	D	E	F
A		×		×	×	×
B	×		×			
C		×				×
D	×					
E	×				×	
F	×		×		×	

Graphe 2

2.2.2 Dictionnaire des prédécesseurs (ou successeurs)

Il dresse pour chaque point les points qui le précédent dans le graphe (ou qui le succède), cette représentation n'a aucun sens pour les graphes non orientés. Pour le **Graphe 1** cette représentation est la suivante :

X	successeurs
A	D
B	A
C	D, E
D	B
E	A, D

Graphe 1

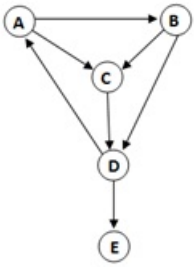
X	Prédécesseurs
A	B, E
B	D
C	-
D	A, C, E
E	C

Graphe 2

Les deux tableaux sont équivalents.

2.2.3 Représentation matricielle (ou matrice d'adjacences)

On peut représenter un graphe simple par une matrice d'adjacences, c'est une matrice carrée où les termes désignent la présence d'un arc (flèche) entre deux sommets donnés, un " 1 " à la position (i, j) signifie que le sommet i est adjacent au sommet j.



→	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	1	0
D	1	0	0	0	1
E	0	0	0	0	0

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrice d'adjacences

Dans l'exemple suivant un graphe orienté et la matrice associée :

Dans cet exemple, le sommet A est de degré 3. Le sommet D est de degré 4. Cette matrice a plusieurs caractéristiques :

1. Elle est carrée.
2. Il n'y a que des zéros sur la diagonale. Un "1" sur la diagonale indiquerait une boucle.
3. Une fois que l'on fixe l'ordre des sommets, il existe une matrice d'adjacences unique pour chaque graphe. Celle-ci n'est la matrice d'adjacences d'aucun autre graphe.

Exemple. 10 *Calculer les matrices M^2 et M^3 . M est la matrice d'adjacences du graphe de l'exemple ci-dessus. Pour chacune de ces matrices, à quoi correspondent les nombres obtenus ?*

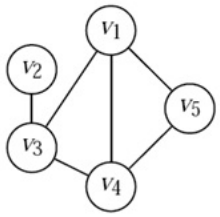
2.3 Coloration d'un graphe

2.3.1 Définitions

Soit $G = (V, E)$ un graphe. Un sous-ensemble S de V est un stable s'il ne comprend que des sommets non adjacents deux à deux. Dans le graphe ci-dessous, $\{v_1, v_2\}$ forment un stable ; $\{v_2, v_4\}$ aussi, ainsi que $\{v_2, v_5\}$ et $\{v_3, v_5\}$.

Le cardinal du plus grand stable est le nombre de stabilité de G on le note $\alpha(G)$. Dans le graphe ci-dessous, on a ? $(G) = 2$.

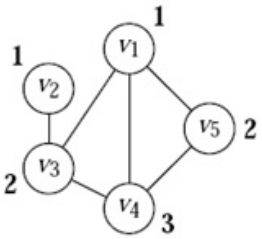
La coloration des sommets d'un graphe consiste à affecter à tous les sommets de ce graphe une couleur de telle sorte que deux sommets adjacents ne portent pas la même couleur. Une coloration avec k couleurs est donc une partition de l'ensemble



des sommets en k stables.

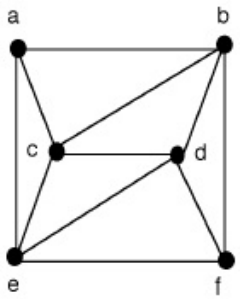
Le nombre chromatique du graphe G, noté $\gamma(G)$, est le plus petit entier k pour lequel il existe une partition de V en k stables.

Sur le graphe ci-dessous, on a eu besoin de trois couleurs (notées 1, 2 et 3) pour colorer les sommets de sorte que deux sommets adjacents aient des couleurs différentes. On a donc trois stables : $\{v_1, v_2\}$, $\{v_3, v_5\}$ et $\{v_4\}$. On ne peut pas utiliser moins de couleurs, à cause des cycles $\{v_1, v_4, v_5\}$ et $\{v_1, v_3, v_4\}$



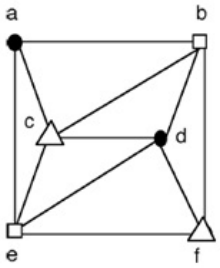
Remarquons enfin que le sommet v_2 aurait aussi pu être coloré "3". La coloration minimale n'est donc pas forcément unique.

Exemple. 11 *Déterminons le nombre chromatique du graphe G suivant :*



Considérons la partition de l'ensemble des sommets de G en sous-ensembles : $S_1 = \{a, d\}$; $S_2 = \{c, f\}$; $S_3 = \{b, e\}$;

On a donc $\gamma(G) \leq 3$. D'autre part, G contient un cycle d'ordre 3, donc $\gamma(G) \geq 3$. Finalement, le nombre chromatique de G est donc 3. La partition précédente en donne une 3-coloration :



Comment colorer un graphe ?

1. Repérer le degré de chaque sommet.
2. Ranger les sommets par ordre de degrés décroissants (dans certains cas plusieurs possibilités).
3. Attribuer au premier sommet (A) de la liste une couleur.
4. Suivre la liste en attribuant la même couleur au premier sommet (B) qui ne soit pas adjacent à (A).
5. Suivre (si possible) la liste jusqu'au prochain sommet (C) qui ne soit adjacent ni à A ni à B.
6. Continuer jusqu'à ce que la liste soit finie.
7. Prendre une deuxième couleur pour le premier sommet (D) non encore colorié de la liste.
8. Répéter les opérations 4 à 7.
9. Continuer jusqu'à avoir coloré tous les sommets.

2.3.2 Exemples d'applications

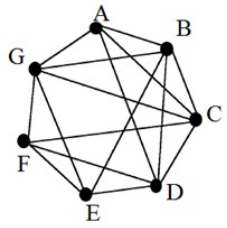
2.3.2.1 Problème d'emploi du temps

Une université doit organiser les horaires des examens. On suppose qu'il y a 7 épreuves à planifier, correspondant aux cours numérotés A, B, C, D, E, F, G et que les paires de cours suivantes ont des étudiants communs : A et B, A et C, A et D, A

et G, B et C, B et D, B et E, B et G, C et D, C et F, C et G, D et E, D et F, E et F, E et G et F et G. Comment organiser ces épreuves de façon qu'aucun étudiant n'ait à passer deux épreuves en même temps et cela sur une durée minimale ?

Solution :

Construisons le graphe G dont les sommets A, B, C, D, E, F, F, G, une arête relie deux de ses sommets lorsque les deux cours correspondant possèdent des étudiants communs :



1. On range les sommets du plus haut degré au plus petit (on ne tient pas compte de l'ordre pour les sommets de même degré).

Sommet	B	C	D	G	A	E	F
dergé	5	5	5	5	4	4	4

On choisit une couleur pour le premier sommet de la liste, Le rouge par exemple :

Sommet	B	C	D	G	A	E	F
dergé	5	5	5	5	4	4	4

On colorie en rouge les sommets non adjacents à B et non adjacents entre eux : F

Sommet	B	C	D	G	A	E	F
dergé	5	5	5	5	4	4	4

2- On réitère le procédé vu au 1 en prenant une autre couleur pour le premier sommet non colorié de la liste. On colorie C en bleu.

Sommet	B	C	D	G	A	E	F
dergé	5	5	5	5	4	4	4

On colorie ensuite E en bleu.

Sommet	B	C	D	G	A	E	F
dergé	5	5	5	5	4	4	4

3- On réitère le procédé. On colorie D en vert puis G.

Sommet	B	C	D	G	A	E	F
dergé	5	5	5	5	4	4	4

4- Enfin A en noire

Donc on a 4 stables $S_1 = \{B, F\}$, $S_2 = \{C, E\}$, $S_3 = \{D, G\}$ et $S_4 = \{A\}$.
Les examens peuvent être répartis en 4 périodes, de la manière suivante :
la 1^{re} période, épreuves des cours 2 et 6,
la 2^{me} période, épreuve du cours 3et 5,
la 3^{me} période, épreuves des cours 4 et 7,
la 4^{me} période, épreuves des cours 1.

2.3.2.2 Problème de wagons

On veut transporter des produits chimiques par le rail. A, B, C, D, E, F, G et H désignent huit produits chimiques. Dans le tableau ci-dessous, une croix signifie que les produits ne peuvent pas être entreposés dans le même wagon, car il y aurait risque d'explosion :

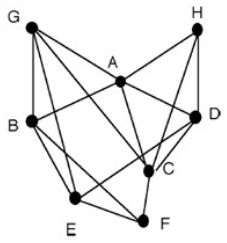
	A	B	C	D	E	F	G	H
A		×	×	×			×	×
B	×				×	×	×	
C	×			×		×	×	×
D	×		×		×			×
E		×		×		×	×	
F		×	×		×			
G	×	×	×		×			
H	×		×	×				

Quel nombre minimum de wagons faut-il ?

Solution :

Construisons le graphe G dont les sommets sont les huit produits chimiques tel que deux de ses sommets sont reliés lorsque les produits associés à ces sommets ne

peuvent pas être entreposés dans le même wagon. Le nombre minimum de wagons est égal au nombre chromatique de ce graphe.



La partition minimale des sommets est :

$$S_1 = \{A, E\}, S_2 = \{B, C\}, S_3 = \{D, F, G\}, S_4 = \{H\}$$

Il faut donc 4 wagons.

2.4 Problème du plus court chemin

2.4.1 Introduction

Beaucoup de problèmes peuvent être modélisés en utilisant des graphes valués. Les problèmes de cheminement dans les graphes, en particulier la recherche du plus court chemin, comptent parmi les problèmes les plus anciens de la théorie des graphes et les plus importants par leurs applications : coût de transport, temps de parcours, problème de trafic,...Les algorithmes de recherche de plus court chemin seront différents selon les caractéristiques du graphe.

Une méthode qui est largement utilisé pour trouver la plus court chemin dans un graphe est celle de présentée par R. Bellman. Cette méthode a été initialement consacrée aux problèmes économiques, mais elle peut être appliquée à d'autres domaines.

2.4.2 Description de la méthode

Pour déterminer le plus court chemin dans un graphe l'algorithme de Bellman consiste à suivre les étapes suivantes :

1. Les sommets x_0 sans précédents sont dit du 'premier niveau' ou niveau de depart : on leur affecte une fonction coût (ou distance), notée m, égale à 0 : $m(x_0) = 0$

2. Examiner les noeuds x_1 adjacents aux noeuds x_0(ceux de premier niveau). Pour chacun, la fonction m se calcul alors en ajoutant la distance à partir du noeud source par : $m(x_1) = d(x_1, x_0) + m(x_0) = d(x_1, x_0)$
3. Au niveau $i + 1$, on continue l'examination des noeuds adjacents à ceux visités dans le niveau i précedent. Quand un noeud $x_(i+1)$ a des liaisons avec plusieurs précédents, la valeur de $m(x_{i+1})$ se calcule alors en retenant le noeud x_i le plus proche, en d'autre terme : $m(x_{i+1}) = \min\{d(x_{i+1}, x_i) + m(x_i), \quad x_i$ noeud précédent à x_{i+1} $\}$
4. Répéter l'étape 3, niveau après niveau jusqu'à ce que le noeud de destination soit atteint.

2.4.3 Exemple d'application

La Figure 2.6 montre la distance entre 10 villes voisines avec l'unité de (1=100 kilomètre). Le problème est de trouver le plus court chemin entre la ville de départ (a) et la ville de destination (j).

Ce problème est souvent rencontré dans d'autres situations différentes, par exemple on peut associer à chaque arc de graphe le montant à payer au point de péage, et le but dans ce cas est de trouver le tragi dont le cout est minimal. **Niveau 1 :**

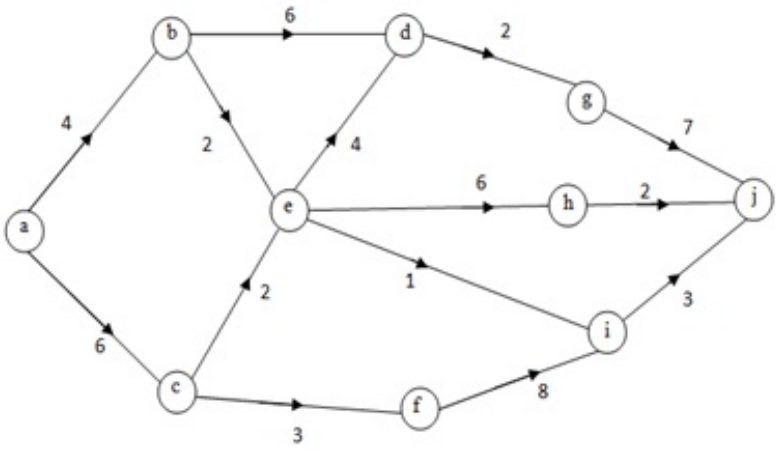


FIGURE 2.6 – Distances entre les villes

Point de départ est (a) donc m(a)=0

Niveau 2 : sommets (a) et (b)
Les sommets b et c sont adjacents à a. Pour ces noeuds on trouve la distance par l'ajout de m(a) avec la distance des noeuds à partir de a. On alors :

$m(b) = m(a) + d(a, b) = 0 + 4 = 4$
$m(c) = m(a) + d(a, c) = 0 + 6 = 6$

Niveau 3 : sommets (d), (e) et (f)

$m(d) = m(b) + d(b, d) = 4 + 6 = 10$
$m(e) = \min\{m(b) + d(b, e), m(c) + d(c, e)\} = \min\{4 + 2, 6 + 2\} = \min\{6, 8\} = 6$
$m(f) = m(c) + d(c, f) = 6 + 3 = 9$

Niveau 4 : sommets (g), (h) et (i)

$m(g) = m(d) + d(d, g) = 10 + 2 = 12$
$m(h) = m(e) + d(e, h) = 6 + 6 = 12$
$m(i) = \min\{m(e) + d(e, i), m(f) + d(f, i)\} = \min\{6 + 1, 9 + 8\} = \min\{7, 17\} = 7$

Niveau 5 : sommets (j)

$m(j) = \min\{m(g) + d(g, j), m(h) + d(h, j), m(i) + d(i, j)\} = \min\{12 + 7, 12 + 2, 7 + 3\} = \min\{19, 14, 10\} = 10$

Le plus court chemin à partir de a vers j est reconstitué de la fin vers le début comme suit : Pour le niveau 5 la ville plus la plus proche de (j) qui appartient au plus court chemin est (i) (la distance minimal pour les trois trajets).

Pour le niveau 4 la ville plus la plus proche de (i) dans le plus court chemin est (e).
Pour le niveau 3 la ville plus la plus proche de (e) dans le plus court chemin est (b).
Pour le niveau 1 la ville plus la plus proche de (b) dans le plus court chemin est (a).

Le plus court chemin est donc a, b, e, i, j
La distance minimal d'aller de (a) à (j) est de 10*100 = 1000km.

Exercice. 1 *Le graphe de la figure 2.7 correspond à des parcours possibles pour aller d'un point s à un point p.*

Quel est le plus court chemin de s à p en utilisant l'algorithme de Ford-Bellman ?

2.5 Problème de flot : Le flot maximal

Les flots permettent de modéliser une très large classe de problèmes. Leur interprétation correspond à la circulation de flux physique sur un réseau : distribution

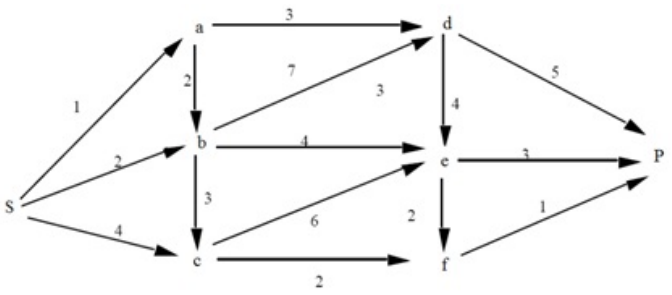


FIGURE 2.7 –

électrique, réseau d'évacuation, acheminement de paquets sur internet,...Il s'agit d'acheminement la plus grande quantité possible de matière entre une source s et une destination t. Les liens permettant d'acheminer les flux ont une capacité limitée, et il n'y a ni perte ni création de la matière lors de l'acheminement.

Définition. 4 *Soit $G(X, A)$ un graphe orienté avec une évaluation positive de ses arcs : on affecte à chaque arc (x, y) une valeur noté positive $c(x, y)$ souvent appelée coût.*
On distingue sur le graphe deux sommets particuliers : une source s et une destination t. Les autres sommets sont les noeuds intermédiaires du réseau.

Exemple. 12 *La figure 2.8 montre un graphe avec une capacité pour chaque arc :*

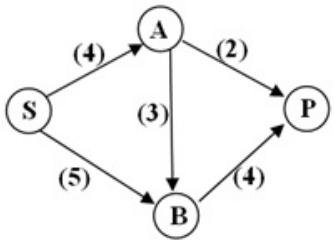


FIGURE 2.8 –

Données : Un graphe orienté $G(X, A)$, une évaluation $c : A \to \mathbb{R}^+$ ($c(a)$ est la capacité de l'arc a).

Un flot sur le réseau $G(X, A)$ est une application ϕ de A dans $\mathbb{R}^+$ telle que :

– le flot à chaque arc a est inférieur à la capacité de l'arc : $0 \leq \phi(a) \leq c(a)$ Figure 2.9

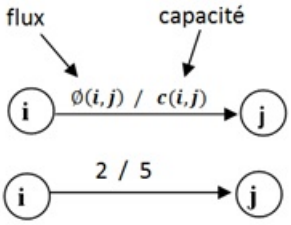


FIGURE 2.9 –

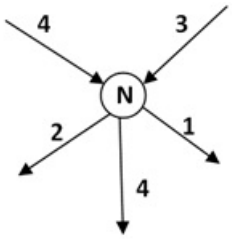


FIGURE 2.10 –

– pour tout sommet de $X \setminus \{s,t\}$, flot entrant = flot sortant Figure 2.10
– la valeur du flot ϕ est le flot sortant de s (égal au flot entrant en t).

Un arc a est dit **saturé** si $\phi(a) = c(a)$, sinon $c(a) - \phi(a)$ est la capacité résiduelle de l'arc a. Un flot ϕ est **saturé** si sur tout chemin de s à t il existe un arc saturé.

Définition. 5 *Soit $G(X,A)$ un graphe valué possédant un seul sommet source s et un seul sommet puits t. Une coupe dans ce graphe est une partition de ces sommets notée $(Y;\overline{Y})$ telle que :*
– $s \in Y et \quad t \in \overline{Y}$
– $X = Y \bigcup \overline{Y}$
– $Y \bigcap \overline{Y} = \emptyset$
*La **capacité de la coupe** $c(Y;\overline{Y})$ est définie par $c(Y,\overline{Y}) = \sum_{i \in Y, j \in \overline{Y}} c(i,j)$.*

Exemple. 13 *Soit le graphe figure 2.11, dont la valeur associe à chaque arc (i,j) est sa capacité $c(i,j)$.*

Dans cet exemple on a :
la coupe est $(Y,\overline{Y})$ avec $Y = \{S,A,B,C,E,F\}$ et $\overline{Y} = \{D,G,P\}$
les arcs sortant sont (A,D), (B,D), (E,P) et (F,G).
la valeur de cette coupe est $c(Y,\overline{Y}) = 4+5+4+3 = 16$.

Détermination du flot maximal : Méthode de Ford-Fulkerson

Le problème consiste à maximiser le flot d'une source (origine) s à une destination t dans un réseau de transport dont les arcs sont munis de capacité c_{ij}. La méthode la plus connue pour trouver le flot maximal dans un réseau de transport est celle de Ford-Fulkerson (1956) améliorée ensuite par plusieurs auteurs.

L'algorithme de Ford-Fulkerson (procédure de marquage)

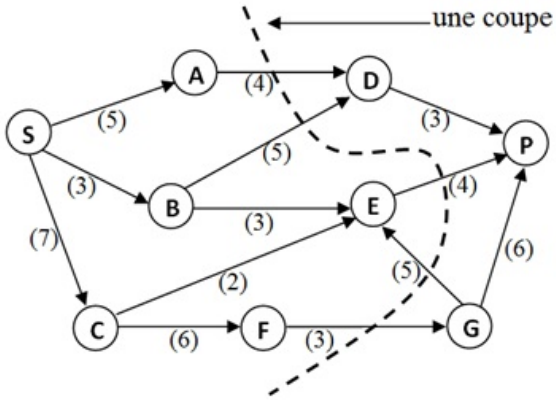


FIGURE 2.11 –

1. Initialisation du flot par un flot réalisable (ou trivial) c'est le flot nul,
2. Marquage des sommets :
   a) Marquer (+) le sommet source s
   b) On marque (+) un sommet qui extrémité d'un arc dont l'origine est déjà marqué (+/-) et sur lequel le flux peut s'augmenter.
   c) On marque (-) un sommet d'arc qui origine et dont l'extrémité est déjà marqué (+/-) et sur lequel peut démineur.
   d) Répéter (b) et (c) jusque ce que le puits soit marqué (le flux peut s'augmenter), ou on ne peut pas marquer le puits, et dans ce cas le flot est maximal.
3. Dans la chaine améliorante l'augmentation maximal de flot sur ce chemin est égal à ε

   $\varepsilon = \min\{\varepsilon^-, \varepsilon^+\}$
   $\varepsilon^+ = \min\{c(a) - \phi(a), a \in A^+\}$ A^+ les arcs de sens directs
   $\varepsilon^- = \min\{\phi(a), a \in A^-\}$ A^- les arcs de sens inverses.

Exemple. 14 *En utilisant la méthode de Ford-Fulkerson, trouver le flot maximal dans ce réseau.*

Au départ le flot est nul Puis, par la procédure de marquage on trouve la chaine améliorante (chemin d'augmentation) S, A, C, P dont la capacité est $\min\{5, 4, 3\} = 3$. On augmente le flot par 3 unités le long de ce chemin À la deuxième itération, on trouve le chemin S, B, D, P de capacité résiduelle 3.

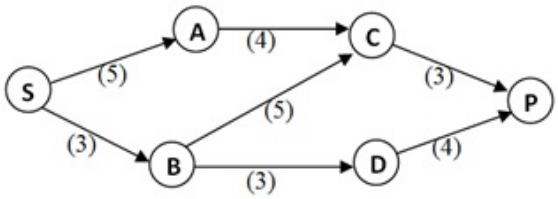


FIGURE 2.12 –

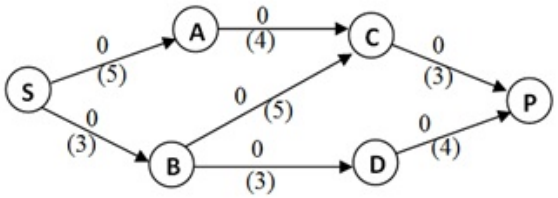


FIGURE 2.13 –

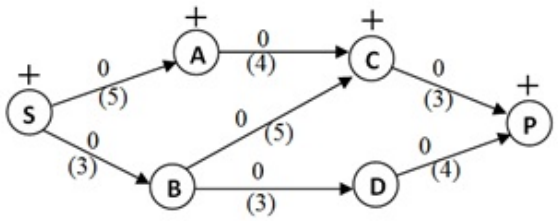


FIGURE 2.14 –

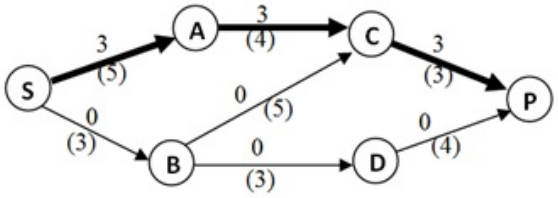


FIGURE 2.15 –

On augmente le flot par 3 unités le long de ce chemin, on trouve Le flot est saturé il n'existe aucune chaine améliorante, on ne peut marquer le puits à partir de la source. Mais il faut savoir est ce que le flot est maximal ou non.

Pour qu'un flot soit maximal il suffit qu'elle existe une coupe dont la valeur est égale le flot qui circule dans le réseau.

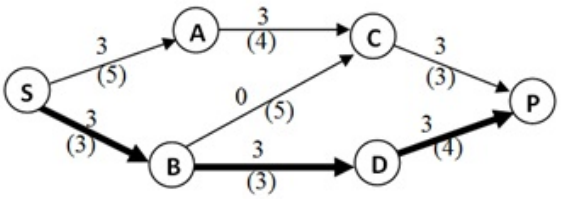


FIGURE 2.16 –

Pour l'exemple précédent

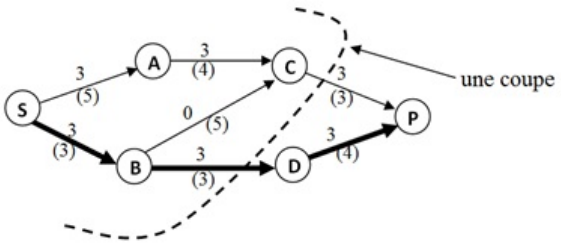


FIGURE 2.17 –

La coupe ayant comme arcs sortant (C,P) de capacité (3) et (B,D) de capacité (3), la valeur de cette coupe est 6 qui égale le flot qui circule dans le réseau. Par suit le flot est maximal.

Exercice. 2 *Un réseau de transport correspond au graphe Figure 2.18 :*

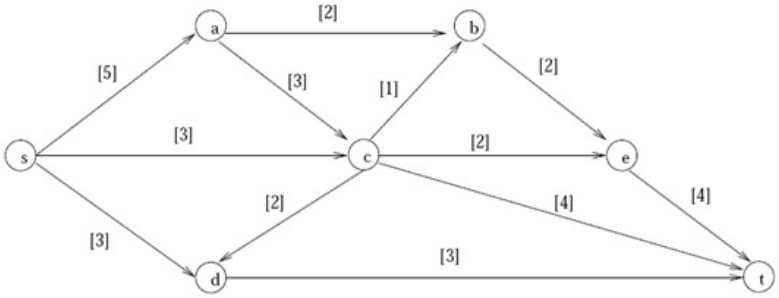


FIGURE 2.18 –

1. Quelle est la valeur de flot maximal ?
2. Donner une coupe minimal associe au flot maximal.

2.6 Problèmes d'ordonnancement de projet

2.6.1 Ordonnancement et planification

2.6.1.1 La WBS (Work Breakdown Structure)

Dès la conception préliminaire du projet. Il s'agit de décomposer de façon structurée et précise le projet en sous ensembles, de manière à visualiser l'ensemble du projet. Il se fait par niveau successifs jusqu'à un degré optimum de détail, afin d'éviter les oublis, et de permettre la consolidation des informations.

Le découpage de produit, ou " Product Breakdown Structure " (P.B.S) établit l'arborescence des différents composants du projet. La décomposition des produit est effectuée par niveaux, selon un principe de " filiation " : pour tout produit de niveau 'n', doivent apparaître les produits le constituant à niveau 'n+1'.

Pour planifier un projet, il est donc nécessaire de le décomposer en sous-ensembles, tâches ou activités. Cette opération est appelée décomposition structurée des tâches (WBS). La WBS doit inclure tous les composants nécessaires à la réalisation d'un projet et qui influent sur sa durée et son cout : tâches de conception, fabrication, matières consommables, équipements, rapports,...

On peut associer directement à une WBS les couts de chaque tâche, les ressources prévues ainsi que les ressources alternatives. L'élaboration d'une WBS simplifie considérablement :

- La planification
- L'identification précoce des activités critiques
- Le contrôle des coûts
- L'élaboration des budgets

Si le nombre de tâches auquel on parvient est trop élevé et par conséquent si cela nuit à la clarté de la lecture du planning, on peut, une fois effectue ce découpage minutieux, rassembler différents tâches sous un même terme et constituer ainsi des " work packages " ou lots de travaux. En découpant le projet en éléments successifs depuis le haut de l'arborescence, il est possible d'identifier des éléments de plus en plus simples, dont les coûts, délais et ressources deviennent plus faciles à estimer.

Le degré optimal de décomposition est atteint lorsque les trois critères sont remplis :

- La possibilité de maîtriser la durée d'une activité
- La connaissance des ressources requises
- La possibilité de connaitre le coût de l'activité

Plus la décomposition sera fine et plus la planification et le contrôle seront précis mais plus la coordination entre tâches sera ardue.

On distingue alors :

- Projet
  - Un seul début et une seule fin
  - Début et fin identifiés en tant qu'événements (décision, revue...)
- Sous-projet
  - Projet contenu dans le projet principal
  - Lié à un objet ou un dérivable partiel du projet
- Phase (étape)
  - Ensemble d'actions qui marque un avancement significatif
  - Lié à un type de compétence et à un degré dans la progression du projet
- Tâche
  - Maille la plus fine de la planification
  - Action exécutable par une seule ressource (ou un seul ensemble de ressource)

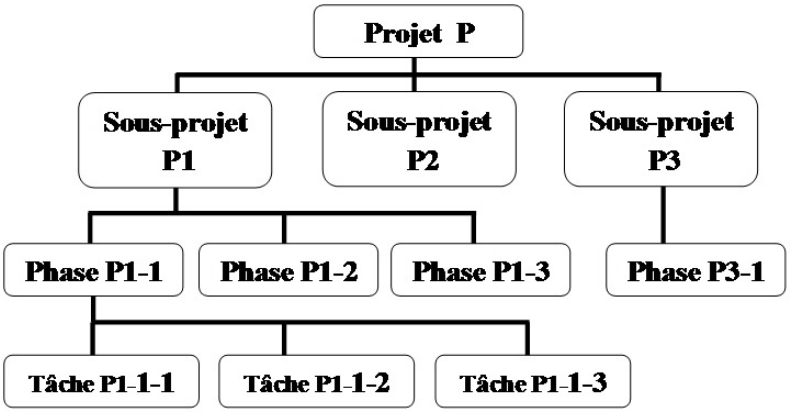


FIGURE 2.19 – Schéma général d'une décomposition d'un projet en WBS

2.6.1.2 Contraintes d'ordonnancement

Les problèmes d'ordonnancement sont apparus au départ dans la planification de grands projets. Le but était de gagner du temps sur leur réalisation. De tels projets sont constitués étapes, également appelées tâches. Des relations temporelles

existent entre dernières. Par exemple :

- Une étape doit commencer à une date précise ;
- Un certain nombre de tâches doivent être terminées pour pouvoir en démarrer une autre ;
- Deux tâches ne peuvent être réalisées en même temps (elles utilisent une même machine par exemple) ;
- Chaque tâche nécessite une quantité de main d'oeuvre. Il faut donc éviter, à chaque instant, de dépasser la capacité totale de main d'oeuvre disponible.

Toutes ces contraintes ne sont pas simples à prendre en compte dans la résolution de problème. Ici, nous allons nous intéresser uniquement aux deux premiers types de contraintes.

On cherche à déterminer une planification, un ordonnancement des étapes qui minimise le temps total de réalisation de projet. A partir de cette planification, nous verrons que le temps de certaines étapes peut éventuellement être modifié sans entraîner un retard du projet, alors que d'autre, les tâches dites " critiques ", retardent entièrement le projet au moindre retard local.

La réalisation d'un projet nécessite souvent une succession des tâches auxquelles s'attachent certaines contraintes :

- De temps : délais à respecter pour l'exécution des tâches ;
- D'antériorité : certaines tâches doivent s'exécuter avant d'autres ;
- De production : temps d'occupation du matériel ou des hommes qui l'utilisent...

Les techniques d'ordonnancement dans le cadre de la gestion d'un projet ont pour objectif de répondre au mieux aux besoins exprimés par un client, au meilleur coût et dans les meilleurs délais, en tenant compte des différentes contraintes.

L'ordonnancement se déroule en trois étapes :

- La planification : qui vise à déterminer les différentes opérations à réaliser, les dates correspondantes, et les moyens matériels et humains à y attacher.
- L'exécution : qui consiste à la mise en oeuvre des différentes opérations définies dans la phase de planification.
- Le contrôle : qui consiste à effectue une comparaison entre planification et exécution, soit au niveau des coûts, soit au niveau des dates de réalisation.

Il existe trois méthodes d'ordonnancement : la méthode PERT (Program Evaluation and Research Task), la méthode MPM (Méthode des Potentiels Métra) et le diagramme de Gantt.

2.6.2 Technique d'ordonnancement

L'objectif d'une technique d'ordonnancement est de visualiser sous forme d'un graphe (ou réseau d'ordonnancement) l'enchainement des tâches de projet, d'estimer la durée global du projet, de déterminer les tâches ne tolèrent pas le retard, les retards tolérées pour certaines tâches, etc. Voici une technique la plus utilisée en pratique de la gestion des projets :

La méthode PERT (Program Evaluation and Research Task)

Les objectifs de la méthode PERT sont : Ordonnancer le projet, calculer la durée du projet, déterminer les tâches critiques, etc.

a - Le réseau PERT

La méthode PERT (Program Evaluation and Research Task) : c'est la méthode la plus connue. Elle a été développée par la marine américaine dans les années 1950 pour coordonner l'action de près de 6000 constructeurs pour la réalisation de missiles à ogives nucléaires POLARIS. C'est aussi la plus "compliquée" à mettre en oeuvre.

Le réseau PERT est un graphe permettant de visualiser et coordonner les l'enchainement des tâches. Il permet également de focaliser l'attention sur des points (ou étapes) où peuvent être prises des mesures pour respecter les délais et optimiser les coûts. Il est utilisé chaque fois quenous nous trouvons en présence d'activités simultanées avec ou sans partage de ressources. Sa conception s'appuie bien entendu sur la WBS.

Le réseau PERT est constitue d'étapes et de tâches élémentaires ou opérations :

Une étape est le commencement ou la fin d'une ou plusieurs tâches. Elle est représentée graphiquement par une cellule dans laquelle sont indiqués le " départ au plus tôt " et le " départ au plus tard " des tâches reliées à ce sommet.

La tâche élémentaire est une action bien déterminée s'inscrivant dans la réalisation du projet. Elle a une durée et consomme de la main d'oeuvre, des matières,

des équipements ou d'autres ressources. Elle est représentée graphiquement par une flèche. Une tâche fait évoluer l'ouvrage vers son état final.

Dans un graphe PERT :

- Chaque tâche est représentée par une flèche (ou arc). Dans la figure ci-dessous, la tâche est mentionnée par la lettre T et sa durée par le nombre d souvent en unité de temps (seconde, minute, heure, jour...).
- Entre les flèches figurent des cercles appelées " sommets " ou " étapes " qui marquent l'aboutissement d'une ou plusieurs tâches. Ces cercles sont numérotés afin de suivre de succession des diverses étapes du projet (la figure ci-dessous montre deux étapes i et j).

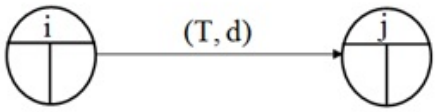


FIGURE 2.20 –

Pour construire un graphe **PERT**, on établit d'abord un tableau donnant les tâches et, pour chacune d'entre elles, les opérations pré-requises. Puis, on utilise la méthode des niveaux qui consiste à :

- Déterminer les tâches sans antécédents, qui constituent le niveau 1.
- Identifier les tâches dont les antécédents sont exclusivement du niveau 1. Ces tâches constituent le niveau 2, et ainsi de suite...

Exemple. 15

Soit l'exemple suivant :

Tâches	Durée	Taches antérierures
E	3	néant
F	5	néant
G	6	E
H	7	E, F
I	5	G, H

Le niveau 1 : E, F
Le niveau 2 : G, H
Le niveau 3 : I

Sur le graphique, les tâches d'un même niveau se retrouvent sur une même verticale.

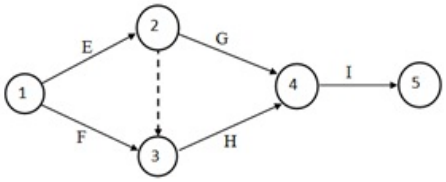


FIGURE 2.21 –

Remarque. 2 *Il a été nécessaire d'introduire une tâche fictive de durée égale à 0, pour représenter la relation antérieur d'antériorité entre E, H : en effet, 2 tâches ne peuvent être identifiées par 2 flèches ayant la même origine et la même extrémité.*

Ainsi si 2 tâches sont simultanées, elles seront représentées par 2 flèches différentes en partant de la même origine :

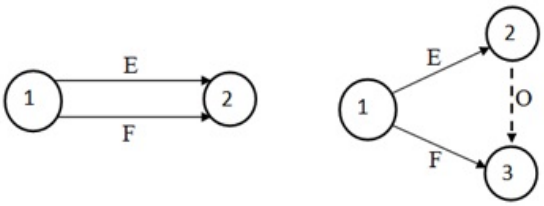


FIGURE 2.22 –

Notons qu'il est très utile, dans le cas général, de déterminer les tâches précédentes directes de chaque tâche. Ceci permettra de savoir quand il est nécessaire d'introduire des tâches fictives : en effet, chaque fois qu'une tâche admet deux (ou plus) précédentes directes qui ont même sommet de début, il est nécessaire de faire dévier une des deux par une flèche fictive. (c'est la cas de la tâche H qui a deux précédentes directes E et F qui ont le même départ figure 2.21) .

Enfin les boucles sont interdites ; elles sont contraires à la logique du système. C'est un point qu'il faut toujours vérifier car, à la suite de modification dans les antériorités, on peut créer des boucles de manière insidieuse.

b - Calcul des dates des tâches

Ayant estimé les durées de tous les tâches constitutives du réseau, on calcule les dates de début au plus tôt et les dates de fin au plus tard respectivement, pour chaque tâche du projet. Comme il montre la figure ci-dessus, on a :

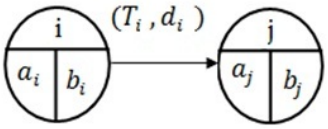


FIGURE 2.23 –

T_{ij} une tâche commence à l'étape i (événement i) et se termine à l'étape j (événement j) de durée d_i (en unité de temps).

a_i : La date de **début au plus tôt** de la tâche T_i.
Elle est égale au temps cumuléle plus long à partir de l'origine en considérant tous les chemins qui aboutissent à la dite tâche. (c'est la date à laquelle toutes les tâches précédentes de cette tâche sont terminées).Pour la calculer, il faut ajouter à la date au plus tôt de chacune des étapes immédiatement précédentes de l'étape considérée la durée de cette tâche. Parmi les valeurs obtenues, on choisit la plus élevée. On initialise le somme "Début" avec une date au plus tôt = 0.

b_i : La date de **début au plus tard** de la tâche T_i.
C'est la date limite à laquelle cette tache doit débutée au plus tard afin de ne pas causer de retard dans la réalisation totale du projet. Pour la calculer, on part de la fin du projet en remontant à son début en suivant les liaisons. Elle est égale au temps le plus faible obtenu en remontant de la dernière tâche vers la première. On initialise à l'étape terminale, le dernier sommet "Fin" par la date au plus tard = date au plus tôt.

Calcul des dates au plus tôt des taches

On cherche à quelles dates, au plus tôt, peuvent être exécutées les différentes tâches du projet. Dans une première phase, on établit les dates au plus tôt de chaque étape, début et fin de tâche. La technique est la suivante :

- On initialise à 0 l'étape de début, numéroté souvent par 1.
- On parcourt le réseau à partir de l'origine dans le sens des flèches. Pour chaque

tâche T_i on a :

$$a_j = a_i + d_i$$

Date de fin au plus tôt de tâche = Date de début au plus tôt de tâche + durée de cette tâche

Si plusieurs chemins aboutissent à une même étape ou noeud, on suit chacun des chemins possibles et on prend pour date au plus tôt de l'étape la plus élevée de toutes les dates obtenues en suivant les différents chemins.

$$a_j = \max\{a_i + d_{ij}\,;\text{ l'étape } i \text{ prédécesseur de } j\ \}$$

On arrive ainsi à dater au plus tôt l'étape finale.

Calcul des dates au plus tard des taches

Partant de cette date finale, on procède au datage au plus tard, qui correspond à un calage de toutes les tâches par l'aval. On remonte le réseau dans le sens inverse des flèches. Pour chaque étape i début d'une tâche T_i on a :

$$b_i = b_j - d_{ij}$$

Date de début au plus tard de tâche = Date de fin au plus tard de tâche - durée de cette tâche

Si plusieurs chemins partent de cette étape, on suit chacun des chemins possibles dans le sens inverse des flèches et on prend pour date au plus tard de cette étape la moins élevée de toutes les dates obtenues en parcourant les différents chemins.

$$b_i = \min\{b_j - d_{ij}\,;\text{ l'étape } j \text{ sucesseur de } i\ \}$$

On inscrit date au plus tôt et date au plus tard de chaque étape. Si date au plus tôt et date au plus tard sont identiques, la tâche est dite critique. Si les dates sont différentes, il y a un battement ou marge. Il convient d'analyser cette marge, qui peut être une marge libre pour une tâche ou une marge totale pour un chemin.

On appelle chemin critique une succession de tâches critiques du début de projet à la fin du projet. Il représente le chemin le plus long du projet au sens des durées. Pour toutes les tâches du chemin critique, les dates au plus tôt et au plus tard coïncident.

c - Marge libre et marge totale

La marge libre : C'est le retard que l'on peut prendre dans la réalisation d'une tâche sans retarder la date de début au plus tôt de tout autre tâche qui suit.

$$\mathrm{ML(t)} = a_j - a_i - d$$

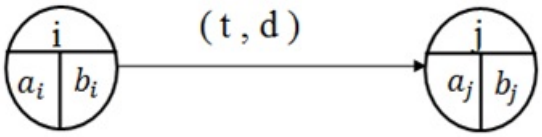


FIGURE 2.24 –

La marge totale d'une tâche : C'est le retard que l'on peut prendre dans la réalisation de cette tâche sans retarder l'ensemble du projet, elle est obtenue par la différence entre la date au plus tard d'une tâche et la date au plus tôt.

$$MT = b_i - a_i$$

Pour toute tâche du chemin critique, la marge totale et la marge libre est nulle.

Exercice corrigé

Nous devons déterminer la durée maximale des travaux nécessaires à la construction d'un entrepôt.

Tâches	Tâches antérieures	durée
A. Etude, réalisation et acceptation des plans	-	4
B. Préparation du terrain	-	2
C. Commande matériaux (bois, briques,...)	A	1
D. Creusage des fondations	A,B	1
E. Commandes portes, fenêtres	A	2
F. Livraison des matériaux	C	2
G. Coulage des fondations	D,F	2
H. Livraison portes, fenêtres	E	10
I. Construction des murs, du toit	G	4
J. Mise en place portes et fenêtres	H,I	1

Tracé du réseau PERT

La figure 2.25 représente le graphe PERT du projet, les dates au plus tôt et au plus tard sont calculées pour chaque tâche, on prend par exemple : La date de début au plus tôt de la tâche D est le max de {4+0,0+2} est égal à 4.

La date de début au plus tard de tâche C est le min de {6-2,8-1} est égal à 4.

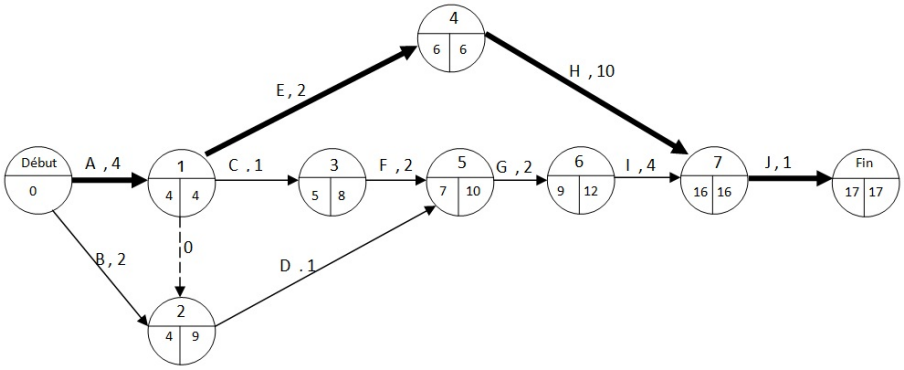


FIGURE 2.25 –

Le ***chemin critique*** du graphe est construit par les tâches **A-E-H-J** a pour durée 17 unité du temps.

Calcul des marges libres et des marges totales

Pour la tâche G par exemple, on a :
La marge libre est ML(B) = 4-0-2 = 0
La marge total est MT(B) = 9-0-2 = 7

le tableau suivant montre la marge libre et total de chaque tâche du projet :

Tâche	A	B	C	D	E	F	G	H	I	J
Marge libre	0	2	0	2	0	0	0	0	3	0
Marge total	0	7	3	5	0	3	3	0	3	0

Chapitre 3

Programmation linéaire en nombres entiers

Sommaire

3.1 Introduction

Ce chapitre concerne la résolution des programmes en nombre entiers où les variables sont astreintes à prendre des valeurs entières. Il s'agit là d'un des domaines

les plus riches et les plus actifs de la programmation mathématique. Le but de ce chapitre est de présenter une synthèse de deux principales méthodes pour résoudre un programme linéaire en nombres entiers.

Considérons le programme linéaire :

$$(PL)\begin{cases} Minimiser \quad z = c.x \\ Sous \quad les \quad contraintes : \\ A.x = b \\ x \geq 0 \end{cases}$$

Tous les coefficients c_j $(j = 1, \ldots, n)$, $a_{ij}(i = 1, \ldots, m; j = 1 \ldots, n)$ et b_i $(i = 1, \ldots, m)$ sont supposés entiers.

Supposons maintenant les variables x_i représentent physiquement de nombres d'objets indivisibles (des voitures, des avions, des appareilles, ect). Une société de transport, par exemple, cherchera à réaliser un certain programme annuel de voyages, tout en minimisant le coût total de ses bus. Les variables représenteront alors le nombre d'appareilles de chaque type à acheter ou à louer. Il n'est alors pas admissible d'avoir une solution optimale fractionnaire.

On devra dans ce cas imposer aux variables des contraintes supplémentaires (dites contraintes d'intégrité) du type :

x_i entier $\forall j = 1, \ldots, n$.

Le problème deviendra donc :

$$(PNE)\begin{cases} \text{Minimiser} \quad z = c.x \\ \text{Sous les contraintes :} \\ A.x = b \\ x \geq 0 \\ x_i \quad \text{entier} \qquad (\forall i = 1, \ldots, n) \end{cases}$$

Un tel problème est appelé *programme linéaire en nombres entiers.*

Par opposition, le problème (PL) obtenu à partir de (PNE) en relaxant (en "oubliant") les contraintes d'intégrité sera appelé *programme linéaire continu.*

3.2 Différentes classes en programmation en nombres entières

La modélisation de plusieurs problèmes se ramené au programme linéaire en nombres entiers, dans ce qui suit pour cette section nous présentons les modèles les plus utilisés.

3.2.1 Le problème du sac à dos : Knapsack-Problem

Un alpiniste se préparant à partir en expédition est confronté au douloureux problème de savoir ce qu'il doit mettre dans son sac. Chaque objet présente pour lui une utilité plus ou moins importante et le poids total du sac est limité. La modélisation de ce problème est a priori très simple.
On connaît pour chaque objet son poids p_j, ainsi que l'utilité c_j que l'on peut en retirer. On connaît aussi le poids maximum P du sac.

Les **decisions** concernent le nombre de chacun des objets à mettre dans le sac. Elles sont representées par des variables $x_j : x_j \geq 0$

La seule contrainte porte sur le poids des objets.

$$\sum_{j=1}^{n} p_j c_j \leq P$$

L'**objectif** concerne la maximisation de l'utilité totale. On fait l'hypothèse que pour chaque type d'objets, l'utilité est proportionnelle au nombre d'objets.

$$\text{Max} \quad \sum_{j=1}^{n} c_j x_j$$

Pour le moment, on a, à priori, un problème très simple de programmation linéaire :

$$(PNE) \begin{cases} Max \quad c.x = \sum_{j=1}^{n} c_j x_j \\ S.c \\ \sum_{j=1}^{n} p_j x_j \leq P \\ x \geq 0 \\ x_j \quad \text{entier} \quad (\forall j = 1, \ldots, n) \end{cases}$$

Ce problème est l'exemple le plus simple de problème de programmation linéaire en nombres entiers. Le challenge est que ce type de problème fait partie des problèmes

difficiles, ceux pour lesquels le temps de calcul croit de manière exponentielle avec la taille du problème.

3.2.2 Problème d'affectation

Le modèle d'affectation sert à représenter des problèmes dans lesquels on dispose de deux ensembles de n éléments, tout élément de l'ensemble 1 devant être affecté à un et seul élément de l'ensemble 2 et réciproquement, l'affectation d'un élément à un autre induisant un coût. L'objectif est de réaliser cette affectation en minimisant le coût total.

Les 2 ensembles peuvent, par exemple, correspondrent à des tâches et des personnes susceptibles de les réaliser ou, à des jetons et des récipients !

Une grande compagnie aérienne a centralisé ses correspondances dans un hub.

Elle souhaite coupler les vols arrivant et partant de ce hub, afin qu'un maximum de passagers puisse continuer leur voyage sans changer d'avion.

On considère n vols arrivant au hub et n vols quittant le hub.

n_{ij} est le nombre de passagers arrivant par le vol numéro i $(i = 1, \ldots, n)$ et poursuivant sur le vol numéro j $(j = 1, ..., n)$.

Dans ce problème, le but est de déterminer, pour chaque vol arrivant, le vol partant avec lequel il sera couplé.

Etant donné le vol arrivant numéro i, il s'agit de savoir si le vol partant numéro j lui est ou non affecté, d'où l'introduction des variables de **décision** x_{ij}. $j = 1 \ldots n$.
On peut donc modéliser le problème d'affectation sous la forme :

$$x_{ij} = \begin{cases} 1 & si \quad le \quad vol \quad arrivant \quad i \quad est \quad couple \quad au \quad vol \quad partant \quad j \\ 0 & sinon \end{cases}$$

Il faut que chaque vol arrivant soit couplé avec un seul vol partant.
Pour le vol arrivant numéro i, il suffit de compter le nombre de variables x_{ij} qui valent 1.
Cette **contrainte** peut donc s'écrire :

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

Ceci ne suffit pas car plusieurs vols arrivant pourraient être couplés au même vol quittant le hub ; on ajoute donc les contraintes qui expriment que chaque vol quittant est couplé avec un seul vol arrivant :

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

L'**objectif** est de maximiser le nombre de passagers qui ne changeront pas d'avion.

$$\sum_{i=1}^{n} \sum_{i=1}^{n} n_{ij} x_{ij}$$

Ce problème est un problème d'affectation dont la forme générale est :

$$(PNE) \begin{cases} \text{Min(Max)} & \sum_{i=1}^{n} \sum_{i=1}^{n} n_{ij} x_{ij} \\ \text{S.c :} & \\ \sum_{j=1}^{n} x_{ij} = 1 & \forall i = 1, \ldots, n \\ \sum_{i=1}^{n} x_{ij} = 1 & \forall j = 1, \ldots, n \\ x_{i,j} \in \{0, 1\} & \forall i = 1, \ldots, n \quad j = 1, \ldots, n \end{cases}$$

Le problème d'affectation est un des rares problèmes de (PNE) qui soit facile à résoudre. Il existe de très bons algorithmes pour cela. On peut aussi le résoudre comme un problème de programmation linéaire standard, sans tenir compte des contraintes d'intégrité, on constatera a posteriori que les variables à l'optimum valent 0 ou 1.

3.2.3 Le problème de voyageur de commerce : (Traveling Salesman Problem ou TSP)

Le "problème du voyageur de commerce", ou TSP (Traveling Salesman Problem), est le suivant :

Un représentant de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer exactement une fois par chaque ville et de revenir à son point de départ pour un moindre coût, c'est-à-dire en parcourant la plus petite distance possible.

Soit $G = (X, U)$, un graphe dans lequel l'ensemble X des sommets représente les villes à visiter, ainsi que la ville de départ de la tournée, et U, l'ensemble des arcs de

G, représente les parcours possibles entre les villes. À tout arc $(i,j) \in U$, on associe la distance de parcours d_{ij} de la ville i à la ville j. La longueur d'un chemin dans G est la somme des distances associées aux arcs de ce chemin. Le TSP se ramène alors à la recherche d'un circuit hamiltonien (i.e., un chemin fermé passant exactement une fois par chacun des sommets du graphe) de longueur minimale dans G.

On peut formuler le TSP de manière équivalente en associant :

- à chaque couple (i,j) de villes à visiter ($i = 1 \dots n$, $j = 1 \dots n$ et $i \neq j$) une distance δ_{ij} égale à d_{ij} s'il existe un moyen d'aller directement de i à j (i.e., $(i,j) \in U$ dans G), fixée à ∞ sinon ;
- une variable de succession, x_{ij} , binaire, qui prend la valeur 1 si la ville j est visitée immédiatement après la ville i dans la tournée et qui prend la valeur 0 sinon.

et Le TSP est alors modélisé par :

$$(PNE) \begin{cases} \text{Minimiser} \quad \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{ij} x_{ij} \\ \text{S.c :} \\ \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1 \dots n \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1 \dots n \\ \displaystyle\sum_{i \in S j \notin S} x_{ji} \geq 2 \quad \forall S \subset X, S \neq \emptyset \\ x_{ij} \in \{0,1\} \quad \forall i,j = 1, \dots, n \end{cases}$$

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

3.2.4 Problème de minimisation du coût du flot circulant sur un graphe

Soit $G = (X,U)$ un réseau orienté définie par un ensemble de noeuds **X** et **U** définit un ensemble de **m** arcs. Pour chaque arc $(i,j) \in U$ un coût c_{ij} est associé pour définir le coût par unité de flot sur cet arc. On assume que le coût du flot varie linéairement avec la quantité du flot. On associe avec chaque arc $(i,j) \in U$ une capacité u_{ij} qui définie la quantité maximale de flot qui peut circuler sur cet arc. On associe avec chaque noeud $i \in X$ un nombre entier $e(i)$ représentant la production ou la consommation (en terme de flot) de ce noeud. Si $e(i) > 0$, le noeud "i" est un noeud producteur ; si $e(i) < 0$, le noeud "i" est un noeud consommateur ; et si

$e(i) = 0$, le noeud "i" est un noeud intermédiaire servant de relais. Les variables de decision dans ce probleme sont les flots sur les arcs et on represente le flot sur l'arc $(i, j) \in U$ par x_{ij}. Le problème de minimisation de coût du flot est un probleme d'optimisation formulé comme suit :

$$(PNE)\begin{cases} \text{Minimiser} \quad \sum_{(i,j)\in U} c_{ij}x_{ij} \\ \text{S.c :} \\ \sum_{j:(i,j)\in U} x_{ij} - \sum_{j:(j,i)\in U} x_{ij} = e(i) \qquad \forall i \in U \qquad (1) \\ \\ x_{ij} \leq u_{ij} \qquad \forall (i,j) \in U \qquad (2) \\ x_{ij} \geq 0 \end{cases}$$

Au niveau de la contrainte 1, le premier terme désigne le flot sortant du noeud "i" et le second terme désigne le flot entrant. Cette contrainte est connu sous le nom de la contrainte de conservation de flot. Tandis que la contrainte 2 est connue sous le nom de contrainte de capacité, et elle assure que le flot circulant sur un arc ne dépasse pas la capacité disponible sur ce dernier.

Le problème présenté peut servir à résoudre le problème de plus court chemin entre une source $s \in X$ et une destination $t \in U$. En effet, il suffit de mettre :

- $e(s) = 1$
- $e(t) = -1$
- $e(i) = 0 \qquad \forall i \in X \setminus \{s, t\}$

3.2.5 Problème de localisation

Une entreprise livre ses marchandises à m clients et dispose pour cela d'un ensemble de n dépôts potentiels. L'utilisation d'un dépôt j entraine un coût fixe f_j et le transport de la commande d'un client i à partir d'un dépôt j entraine un coût c_{ij}.

$$x_{ij} = \begin{cases} 1 & \text{si le client } j \text{ est affécté au dépôt } i \\ 0 & \text{sinon} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{si le dépôt } i \text{ est ouvert} \\ 0 & \text{sinon} \end{cases}$$

Quels dépôts ouvrir et quel plan d'acheminent établir pour livrer tous les clients en minimisant la somme des coûts fixes et variables ?

Le problème est alors modélisé par :

$$(PNE)\begin{cases} \text{Minimiser} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j y_j \\ \text{S.c :} \\ \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1 \dots m \\ x_{ij} \leq y_i \quad i = 1 \dots m, j = 1 \dots n \\ x_{ij} \in \{0,1\} \quad i = 1 \dots m, j = 1 \dots n \\ y_i \in \{0,1\} \quad i = 1 \dots m \end{cases}$$

3.3 Les méthodes de recherche arborescente par séparation et évaluation

Considérons un programme linéaire en nombres entiers du type :

$$(PNE)\begin{cases} \text{Minimiser} \quad z = c.x \\ \text{S.c} \\ A.x = b \\ x \geq 0 \\ x_i \quad \text{entier} \dots (\forall j = 1, \dots, n) \end{cases}$$

où : A ets une matrice $m \times n$, $b \in \mathbb{Z}^m$, $x \in \mathbb{Z}^n$.

Le polytope :

$$\mathcal{P} = \{x \in \mathbb{R} / Ax = b, x \geq 0\}$$

étant supposé borné (c'est donc un polydre convexe), il est toujours possible d'associer à chaque variable x_i des bornes de variation : $\alpha_i \leq x_i \leq \beta_i$

pour déterminer β_i, par exemple, on pourra résoudre le programme linéaire (continu) :

$$\begin{cases} \text{Maximiser} \quad x_i \\ \text{S.c} \\ A.x = b \\ x \geq 0 \end{cases}$$

Ainsi on peut toujours se ramener au cas où chaque variable x_i ne peut prendre qu'un nombre fini de valeurs.

3.3.1 Réduction à un problème en variables bivalentes

Toute variable x ne pouvant prendre que $k+1$ valeurs entiers $0, 1, 2, ..., k$ on peut substituer une combinaison linéaire :

$$x = y_0 + 2y_1 + 4y_2 + \cdots + 2^p y_p$$

de $p+1$ variables : $y_0, \ldots, y_p$, chacune d'elles astreinte à ne prendre que deux valeurs 0, ou 1 (variables bivalentes).
p est le plus petit entier tel que $k \leq 2^{p+1} - 1$.
On peut toujours pour un (PNE) se ramener à un programme linéaire en variables bivalentes.

Exemple. 16 *Soit le programme linéaire en nombres entiers à variables bornées :*

$$\begin{cases} \text{Minimiser} \quad 2x_1 - 3x_2 \\ \text{S.c} \\ x_1 - x_2 \leq 4 \\ -2 \leq x_1 \leq 5 \\ 1 \leq x_2 \leq 3 \\ x_1 \quad \text{et} \quad x_2 \quad \text{entiers} \end{cases}$$

En utilisant 5 variables bivalentes $y_1, y_2, \ldots, y_5$, effectuons le changement de variables :

$x_1 = -2 + y_0 + 2y_1 + 4y_2$
$x_2 = 1 + y_3 + 2y_4$.

La substitution donne :

$$\begin{cases} \text{Minimiser} \quad -7 + 2y_0 + 4y_1 + 8y_2 - 3y_3 - 6y_4 \\ \text{S.c} \\ y_0 + 2y_1 + 4y_2 - y_3 - 2y_4 \leq 7 \\ y_0 + 2y_1 + 4y_2 \leq 7 \\ y_3 + 2y_4 \leq 2 \\ y_i \in \{0, 1\} \quad (i = 0, \ldots, 4) \end{cases}$$

Pour cette raison, nous nous restreindrons dans ce paragraphe à l'étude des programmes linéaires en variables bivalentes de la forme :

$$(PB) \begin{cases} \text{Minimiser} \quad z = c.x \\ \text{S.c} \\ A.x = b \\ x_j = 0 ou 1 \quad (\forall j = 1, \ldots, n) \end{cases}$$

ou encore, en notant S l'ensemble des n-vecteurs à composantes 0 ou 1 :

$$(PB)\begin{cases} \text{Minimiser} \quad z = c.x \\ \text{S.c} \\ A.x = b \\ x_j \in S \end{cases}$$

Comme $|S| = 2^n$, l'ensemble des solutions de (PB) comporte un nombre fini d'éléments.

3.3.2 Définition de l'arborescence : Le concept

Il est clair qu'un tel algorithme serait fini. Mais dès que le nombre n de variable deviendrait grand ($>$50) il faudrait des siècles à l'ordinateur le plus puissant pour énumérer les 2^n éléments de S.
Pour résoudre effectivement sur un calculateur des problèmes de taille importante (100 variables bivalentes ou plus) il faut donc rechercher un principe algorithmique permettant de déterminer une solution optimale sans avoir à énumérer effectivement l'ensemble des éléments de S.
C'est cette idée d'une énumération de solutions qui est à la base des méthodes de recherche arborescente par *séparation* et *évaluation*.

Les sommets de l'arborescence correspondent à des sous-ensembles de S sont répartis de la façon suivante. Il existe un seul sommet de niveau 0 (appelé racine de l'arborescence) lequel correspond à l'ensemble S tout entier. Pour construire le niveau 1 de l'arborescence il faut d'abord choisir (le choix est arbitraire) une variable, par exemple x_1. Le niveau 1 comporte alors deux sommets notés S_1 et $S_{\overline{1}}$ qui correspondent : le premier au sous ensemble de vecteurs pour lesquels la variable x_1 a la valeur 0 ; le second au sous ensemble de vecteurs pour lesquels la variable x_1 a la valeur 1. Évidemment on a : $S_1 \subset S$ et $S_{\overline{1}} \subset S$, de plus S_1 et $S_{\overline{1}}$ forment une *partition* de S.
Les sommets S et S_1 (resp.S et $S_{\overline{1}}$)sont reliés par un arc qui représente la relation d'inclusion. On dit que l'on a "séparé" S relativement à la variable x_1. De la même manière, pour construire le niveau 2 on choisira (arbitrairement) une seconde variable, par exemple x_2. On trouve alors quatre (2^2) sommets de niveau 2 :
$S_{\overline{1},\overline{2}}$ ensemble des vecteurs pour lesquels $x_1 = 0$, $x_2 = 0$
$S_{\overline{1},2}$ ensemble des vecteurs pour lesquels $x_1 = 0$, $x_2 = 1$
$S_{1,\overline{2}}$ ensemble des vecteurs pour lesquels $x_1 = 1$, $x_2 = 0$
$S_{1,2}$ ensemble des vecteurs pour lesquels $x_1 = 1$, $x_2 = 1$

Là encore, les relations d'inclusion sont représentées par arcs joignant le sommet $S_{\overline{1}}$ aux sommets $S_{\overline{1},\overline{2}}$ et $S_{\overline{1},2}$ d'une part ; le sommet S_1 aux sommets $S_{1,\overline{2}}$ et $S_{1,2}$ d'autre

part. On dit que l'on a séparé $S_{\overline{1}}$ (resp. S_1) relativement à la variable x_2.
La figure 3.1 représente l'arborescence associée à un problème de type (PB) comportant 3 variables bivalentes x_1, x_2, x_3.

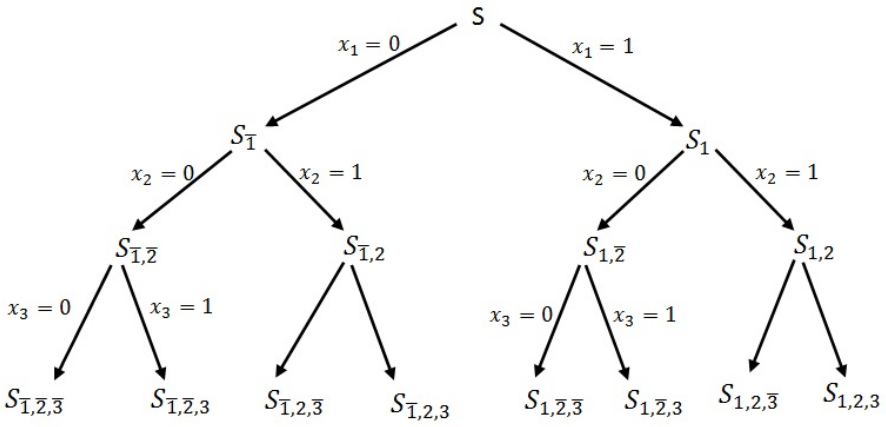


FIGURE 3.1 – Arborescence correspondant à un problème en nombres entiers à 3 variables bivalentes x_1, x_2, x_3

3.3.3 Évaluation

Après avoir introduit le concept de séparation passons à celui d'évaluation. Supposons que, pour chaque sommet S_i de l'arborescence précédente, on puisse déterminer, par calcul, une évaluation par défaut, c'est-à-dire un minorant $f(S_i)$ du coût $c.x$ de la meilleur solution $x \in S_i$, autrement dit :

$$f(S_i) \leq \min_{x \in S_i} c.x$$

La fonction f sera appelée fonction d'évaluation.
Construisons progressivement à partir du niveau 0 (sommet S), l'arborescence des sous-ensembles de S, la séparation étant faite en examinant les variables dans un certain ordre (a priori arbitraire).

Soit $\overline{x}$ une solution du problème (PB), de coût $\overline{z} = c.\overline{x}$, déterminée soit au cours des étapes précédentes de la construction de l'arborescence, soit par une quelconque méthode approchée (heuristique) (éventuellement, si on n'en a pas trouvé, on conviendra de poser $\overline{z} = +\infty$

Supposons alors qu'un sommet S_i de l'arborescence soit tel que : $f(S_i) > \overline{z}$
Par définition de la fonction f, aucune solution contenue dans S_i ne peut être

meilleur que $\overline{z}$, et par conséquent on est sûr que S_i ne contient aucune solution optimale.
Cette constatation permet d'éviter l'exploration (l'énumération) de tous solution de successeurs directes ou indirects du sommet S_i dans l'arborescence.
En limitant à chaque étape, l'énumération aux successeurs des seuls sommets S_i tels que $f(S_i) \leq \overline{z}$ on peut arriver à une réduction considérable du nombre de sommets effectivement examinés, suffisante pour traiter des problèmes de taille importante

3.3.4 Mise en pratique

Ceci étant, il subsiste beaucoup de degrés de liberté dans ce qui précède, et l'on peut dire qu'il existe autant d'algorithmes que de façons de choisir :

(a) La nature de la fonction d'évaluation ;

(b) Le sommet à séparer à chaque étape ;

(c) La variable suivant laquelle s'effectuera la séparation du sommet choisi.

Examinons successivement ces trois points :

Le choix de (a) n'est pas toujours évident. On peut toujours utiliser comme évaluation par défaut la solution optimale continue du programme linéaire restreint aux variables libres. Cependant, une telle évaluation est généralement coûteuse en temps de calcul. Aussi on pourra préférer une fonction d'évaluation moins bon (c'est-à-dire s'écartant davantage de la valeur de l'optimum entier) mais beaucoup plus rapide à obtenir.

Pour le choix de (b) de nombreuses stratégies ont pu être proposées, dont aucune à notre connaissance ne s'avère systématiquement meilleure. Dans la méthode dite " profondeur d'abord ", on convient de choisir le sommet de niveau le plus élevé parmi les sommets non encore séparés. S'il en existe plusieurs, on pourra choisir celui qui correspond à l'évaluation la plus faible. Cette a pour but de mettre en évidence le plus tôt possible une solution du problème. Dans la méthode dite " largeur d'abord " encore appelée méthode SEP (séparation et évaluation progressive) on choisit systématiquement le sommet ayant l'évaluation la plus faible, en remarquant que c'est celui qui intuitivement a la plus de chances de contenir une solution optimale parmi ses successeurs. Le risque avec cette méthode est d'avoir à explorer une fraction importante de l'arborescence avant de découvrir une solution. En revanche, la première solution trouvé est généralement de meilleure qualité (i.e de coût moins élevé) que celle déterminée par une procédure " profondeur d'abord ".
Le choix de (c), donne lieu lui aussi à de nombreuses variantes. Très souvent, l'ordre

des variables est fixé une fois pour toutes, soit de façon totalement arbitraire, soit suivant des critères heuristiques justifiés de façon intuitive. Mais il faut savoir que ce choix peut être déterminant (suivant l'ordre adopté le temps de résolution peut varier de 1 à 10 ou de 1 à 100 !). Ceci a amené à concevoir des stratégies beaucoup plus souples (dynamiques) dans lesquelles on choisit à chaque étape la variable qui semble la meilleur suivant un certain critère. Par exemple, dans la méthode dite des " pénalités " (cf. Hansen 1971) on associe à chaque variable x_i pouvant servir à la séparation un nombre p_i (pénalité), valeur exacte ou approchée de la différence entre les nouvelles évaluations obtenues en faisant $x_i = 0$ d'une part, $x_i = 1$ d'autre part. Si l'on effectue alors la séparation suivant la variable x_i associé à la pénalité p_i maximale, on obtiendra deux nouveaux sous-ensembles dont l'un aura beaucoup plus de chances que l'autre de contenir une solution optimale. Il s'agit en quelque sorte, du choix " le plus informant ". Il conduit à minimiser les risques d'explorer en vain une branche de l'arborescence alors que la solution optimale est contenue dans l'autre.

3.3.5 Exemple

Pour illustrer le fonctionnement des méthodes arborescentes, considérons le problème suivant :

$$\begin{cases} \text{Minimiser} \quad z = -20x_1 - 16x_2 - 11x_3 - 9x_4 - 7x_5 - x_6 \\ \text{S.c} \\ 9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12 \\ x_i \in \{0,1\} \quad (i = 1, \ldots, 6) \end{cases}$$

qui est de la forme :

$$\begin{cases} \text{Minimiser} \quad z = \sum c_j x_j \\ \text{S.c} \\ \sum a_j x_j \leq b \\ x_i \in \{0,1\} \quad (i = 1, \ldots, n) \end{cases}$$

C'est un problème de "sac à dos".

On remarque que les variables sont ordonnées suivant les rapports croissants et que la solution continue (obtenue en remplacant la contrainte d'intergrité $x_i = 0$ ou 1 par : $0 \leq x_i \leq 1$).
Est : $x_1 = 1, \quad x_2 = \frac{3}{8}$ de valeur $z = -20 - 6 = -26$.
Cette valeur constitue donc, sur l'ensemble de toutes les solutions entières possibles, une évaluation par défaut.

Remarquons dès maintenant que l'on peut obtenir aisément une solution entière, approche pour ce problème en utilisant une règle (heuristique) du type :

(a) Au début toutes les variables sont à 0.

(b) Compte tenu des choix précédents, rechercher la variable ayant le plus petit rapport à laquelle on puisse attribuer la valeur 1, tout en respectant la contrainte, et fixer cette variable à 1. Continuer (b) jusqu'à ce qu'on ne trouve plus aucune variable satisfaisant ces conditions.

Ici cette règle donnerait : $x_1 = 1$
puis compte tenu de ce choix : $x_6 = 1$.

La solution entière ainsi obtenue a une valeur $z = -21$.
Compte tenu de ce qui précède, nous étudions maintenant les étapes successives de la recherche arborescente.

Au premier niveau séparons l'ensemble S en deux sous-ensembles : l'ensemble S_1 des vecteurs pour lesquels $x_1 = 1$ et l'ensemble $S_{\overline{1}}$ des vecteurs pour lesquels $x_1 = 0$.

Le choix de x_1 est arbitraire. Mais dans la suite de l'exemple, les séparations successives se feront suivant l'ordre des $\frac{c_j}{a_j}$ croissants, qui se trouve être l'ordre naturel des variables. Ce choix peut être justifié par des considérations intuitives ; on pourra vérifier, en outre qu'il conduit à la solution beaucoup plus rapidement qu'en adoptent l'ordre inverse par exemple.

A ce niveau, on peut obtenir facilement une évaluation par défaut de la meilleur solution entière continue dans chacun des sous-ensembles, S_1 et $S_{\overline{1}}$ respectivement.

Considérons par exemple le sous-ensemble S_1.
Puisque l'on sait que x_1, le problème restreint aux variables $x_2, \ldots, x_6$ (qui seront appelées variables " libres ") est :

$$(P_1) \begin{cases} \text{Min} \quad -20 - 16x_2 - 11x_3 - 9x_4 - 7x_5 - x_6 \\ \text{S.c} \\ 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 3 \\ x_i \in \{0, 1\} \quad (i = 2, \ldots, 6) \end{cases}$$

qui admet la solution continue : $x_2 = \frac{3}{8}$ de valeur $z = -20 - 6 = -26$

-26 est donc une évaluation par défaut de la meilleur solution entière contenue dans S_1. Nous la noterons $\widehat{z}_1$.

Prenons maintenant le sous-ensemble $S_{\overline{1}}$. Puisque l'on sait que $x_1 = 0$, le problème

restreint aux variables libres est :

$$(P_2)\begin{cases} \text{Min} \quad -16x_2 - 11x_3 - 9x_4 - 7x_5 - x_6 \\ \text{S.c} \\ 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12 \\ x_i \in \{0,1\} \quad (i = 2, \ldots, 6) \end{cases}$$

lequel admet la solution continue : $x_1 = 1$, $x_3 = \frac{2}{3}$ de valeur : $\widehat{z}_1 = -16 - \frac{22}{3} = \frac{-70}{3} (\approx -23,3)$.

Comme la meilleur solution entière trouvée jusqu'ici ($x_1 = 1$,$x_6 = 1$) a pour valeur -21, il est clair que chacun des sous-ensembles S_1, ou S_1 peut contenir une solution meilleur, en particulier une solution optimale. Il va donc falloir séparer chacun de ces sous-ensembles en deux (ou plusieurs) sous-ensembles qui, à leur tour donneront lieur à une évaluation et ainsi de suite jusqu'à la découverte d'une solution optimale.

Cependant au point ou nous en sommes, il y a un problème important : lequel de S_1, ou de $S_{\overline{1}}$ doit-on séparer le premier ?

Comme nous l'avons dit, bien des règles peuvent être adoptées pour cela. Choisissons ici, la méthode d'exploration " largeur d'abord " ou SEP qui se fonde sur la remarque suivante :

Si l'évaluation par défaut d'un sous-ensemble est une bonne approximation (c'est-à-dire est peu différente) de la meilleure solution entière contenue dans ce sous-ensemble, il est raisonnable de rechercher d'abord la solution optimale dans le sous-ensemble donnant lieu à la plus petite évaluation. Dans l'exemple qui nous intéresse c'est le sous ensemble S_1 qui a la plus petite évaluation (-26).
Séparons donc S_1 en deux sous-ensembles : Le sous-ensemble $S_{1,\overline{2}}$ des vecteurs pour lesquels $x_1 = 1$ et $x_2 = 0$; Le sous-ensemble $S_{1,2}$ des vecteurs pour lesquels $x_1 = 1$ et $x_2 = 1$;

On remarque immédiatement que l'ensemble $S_{1,2}$ est l'ensemble vide $\emptyset$ car $x_1 = 1$ et $x_2 = 1$ ne permet en aucun cas de satisfaire la contrainte. Il ne sera donc jamais pris en considération par la suite, et un bon moyen pour s'en assurer est de lui associer une évaluation $+\infty$.

Pour l'ensemble on obtient l'évaluation par défaut :
$-20 - 11/2 = -25,5$ avec la solution $x_3 = \frac{1}{2}$

L'ensemble des opérations effectuées jusqu'ici peut être résumé par l'arborescence de la figure 3.2.

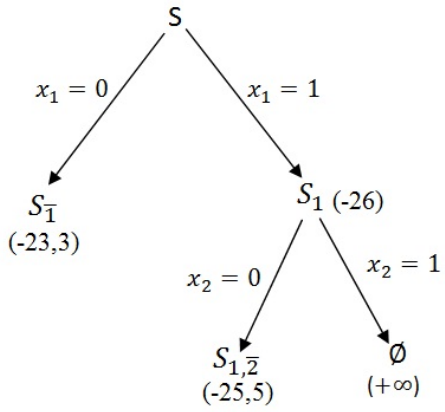


FIGURE 3.2 – Début de la construction de l'arborescence

A ce point, c'est le sous-ensemble ensemble $S_{1,\overline{2}}$ qui a l'évaluation la plus petite -25,5. Si on le sépare (relativement à la variable x_3) on obtient les deux sous-ensembles :
$S_{1,\overline{2},\overline{3}}$ d'évaluation -20-27/5=-25,4, $S_{1,\overline{2},3}$ d'évaluation $+\infty$ ($S_{1,\overline{2},3} = \emptyset$)
En continuant ainsi à construire cette branche de l'arborescence, on obtient successivement $S_{1,\overline{2},\overline{3},\overline{4}}$ (d'évaluation -25,2) et $S_{1,\overline{2},\overline{3},\overline{4},\overline{5}}$ (d'évaluation -23).

Comme l'évaluation de $S_{\overline{1}}$ est inférieure (-23,3) on repart de $S_{\overline{1}}$ que l'on sépare en $S_{\overline{1},\overline{2}}$ (-21,7) et $S_{\overline{1},2}$ (-23,3). On repart alors de $S_{\overline{1},2}$ que l'on sépare en $S_{\overline{1},2,\overline{3}}$ (-23,2) et $S_{\overline{1},2,3}$ ($+\infty$). Puis on repart de $S_{\overline{1},2,\overline{3}}$ que l'on sépare en $S_{\overline{1},2,\overline{3},\overline{4}}$ (-23 : la solution continue de $S_{\overline{1},2,\overline{3},\overline{4}}$ est entière : $x_2 = 1$, $x_5 = 1$) et $S_{\overline{1},2,\overline{3},4}$ ($+\infty$).

A ce stade, la solution continue du problème $S_{\overline{1},2,\overline{3},\overline{4}}$ restreint aux variables libres fournit une solution entière : $x_2 = 1$, $x_5 = 1$ de valeur -23. Comme aucun des sous-ensembles restant à examiner (sommets pendants) n'a une évaluation inférieure à -23, on peut en conclure que cette solution est une solution optimale du problème. Cependant, ce n'est peut être pas la seule, car il y a un autre sous-ensemble d'évaluation -23, c'est $S_{1,\overline{2},\overline{3},\overline{4},\overline{5}}$. Si l'on veut rechercher toutes les solutions optimales, il faut donc continuer, et séparer ce sous-ensemble relativement à la variable x_6.

Ceci donne deux sous-ensembles réduits à une seule solution :

- $S_{1,\overline{2},\overline{3},\overline{4},\overline{5},\overline{6}}$ qui correspond à la solution $x_1 = 1$ (toutes les autres variables nulles) de valeur -20.

- $S_{1,\overline{2},\overline{3},\overline{4},\overline{5},6}$ qui correspond à la solution $x_1 = 1$, $x_6 = 1$ de valeur -23 est l'unique solution optimale, et l'algorithme se termine (figure 3.3).

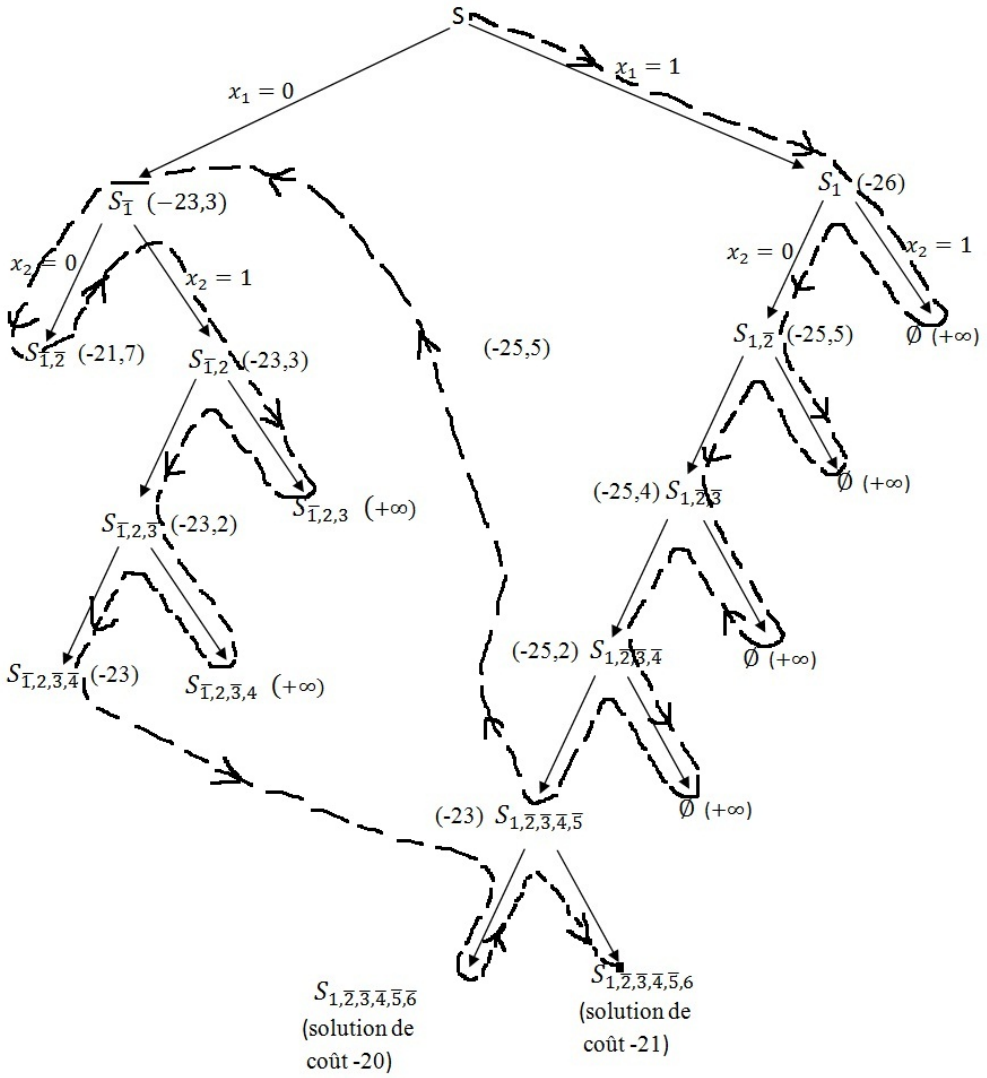


FIGURE 3.3 – Exemple de résolution d'un problème à 6 variables par la méthode SEP. On a indiqué en pointillés l'ordre dans lequel les sommets sont examinés

Cet exemple permet également d'apprécier l'efficacité de la méthode. En effet, sur les $2^7 - 1 = 127$ sommets que comporte théoriquement l'arborescence complète, on constate que 19 seulement ont été examinés, ce qui constitue une réduction significative.

3.4 Les Méthodes de coupes

3.4.1 Principe des méthodes de coupes

L'idée de base sur laquelle se fondent ces méthodes est la suivante.
On commence par résoudre le programme linéaire en continu (PL). Si la solution optimale obtenue est un point extrême à coordonnées entières c'est terminé.

Dans le cas contraire, il est facile de voir que l'on peut toujours tronquer le domaine des solutions (en rajoutant une contrainte supplémentaire au problème) de façon à éliminer ce point extrême sans exclure aucune solution entière. Un telle contrainte est appelée une coupe (on dit encore une troncature).

Soit le problème de minimisation en nombres entiers de type "sac à dos" :

$$(PNE)\begin{cases} \text{Min} \quad -8x_1 - 9x_2 \\ \text{S.c} \\ 8x_1 + 10x_2 \leq 39 \\ x_1, x_2 \quad \text{entiers} \quad \geq 0. \end{cases}$$

L'optimum du problème relaxé (continu) du problème (PNE) est le point extrême : $(\frac{39}{8}, 0)$. La contrainte supplémentaire :

$$x_1 \leq 4$$

élimine ce point sans exclure aucune solution entière. C'est donc bien une coupe. De même toute contrainte supplémentaire de type

$$x_1 + x_2 \leq \alpha$$

avec $4 \leq \alpha \leq \frac{39}{8}$ est une coupe.

Beaucoup de coupes sont possibles, en principe, il en existe toujours une infinité.

Après avoir rajouté une coupe (ou éventuellement plusieurs) le programme augmenté des contraintes correspondantes est à nouveau résolu (en continu) par la méthode du simplexe.

Si la solution optimale de ce nouveau problème est entière, c'est terminé : on a obtenu une solution optimale du problème en nombres entiers. Sinon le raisonnement précédent peut être répété : on recherche une nouvelle coupe (éventuellement plusieurs) que l'on rajoutera à l'ensemble des contraintes ; puis le programme linéaire ainsi augmenté sera optimisé à nouveau, etc.

Si les coupes sont correctement choisies à chaque étape, le polyèdre initial sera ainsi progressivement réduit jusqu'à coïncider avec l'enveloppe convexe des solutions entières, au mois au voisinage de la solution optimale. La solution continue du problème augmenté deviendra alors entière et le problème sera résolu.

Il claire maintenant que le choix des coupes est déterminant pour la convergence de la méthode. Malheureusement, et c'est la difficulté essentielle, on ne connaît pas de méthode systématique pour engendrer tous les équations ou inéquations définissant l'enveloppe convexe des points entières contenus dans un polyèdre convexe donné. C'est pourquoi, un des résultats les plus importantes en programmation en nombres entiers a été mis en évidence (*Gomory* 1958) de coupes d'un type particulier permettant, moyennant certaines précautions, d'obtenir la convergence finie de la méthode.

3.4.2 Les coupes de Gomory

Considérons le programme linéaire continu (obtenu à partir de (PNE) en relaxant les contraintes d'intégrité) :

$$(PL)\begin{cases} \text{Min} \quad z = c.x \\ \text{S.c} \\ A.x = b \\ x \geq 0. \end{cases}$$

et soit B la matrice de base optimale (sous-matrice carrée régulière extraite de A).

Puisque les coefficients de A sont entières, $D = |\det(B)|$ est entier. D'autre part, en multipliant le système $A.x = b$ par B^{-1}, toute variable de base x_i ($i \in I =$ ensemble des indices des variables de base) peut s'exprimer en fonction des variables hors base x_j ($j \in J =$ ensemble des indices des variables hors base) par :

$$x_i = \frac{\beta_i}{D} - \sum_{j \in J} \frac{\alpha_{ij}}{D}.x_j \tag{3.1}$$

où les variables α_{ij} et β_i sont entiers.

Puisque l'on se place dans le cas où la solution optimale n'est pas entière, une des variables de base, x_i par exemple, est fractionnaire.

Exprimons le fait que l'on recherche une solution optimale dans laquelle la variable x_i est entier. Cette condition fournit avec l'équation 3.1 l'équation de congruence :

$$\sum_{j \in J} \alpha_{ij} x_j \equiv \beta_i (\mod D) \tag{3.2}$$

On remarque que l'on obtient une équation de congruence équivalent en multipliant les deux membres de 3.2 par un même nombre λ, premier avec D :

$$\sum_{j \in J} (\lambda \alpha_{ij}) x_j \equiv (\lambda \beta_i) (\mod D) \tag{3.3}$$

Pour un entier relatif quelconque y notons $|y|_D$ le représentant de y dans $[0, D-1]$ modulo D.

En posant alors : $f_j = |\lambda \alpha_{ij}|_D$ (pour $j \in J$) et : $f_0 = |\lambda \beta_i|_D$ on déduit de 3.3 qu'il existe un entier s tel que :

$$\sum_{j \in J} f_j x_j = f_0 + sD$$

Si s est négatif, $f_0 + sD$ est nécessairement négatif, ce qui contredit le fait que $f_j \geq 0$ et $x_j \geq 0$. Donc s est un entier positif ou nul, et l'inéquation :

$$\sum_{j \in J} f_j x_j \geq f_0 \tag{3.4}$$

Est nécessairement vérifiée par toute solution dans laquelle la variable x_i est entière (donc par toute solution entière de (PNE)).
Par ailleurs, on observe que cette inéquation n'est pas vérifiée par la solution de base courante puisque celle-ci est définie par :

$$x_j = 0 \qquad \forall j \in J$$

L'inéquation 3.4 constitue donc une *coupe*.

On remarque qu'une certaine latitude est laissée dans le choix de λ pour définir une coupe à partir de l'équation 3.1. On peut en profiter pour rechercher, parmi toutes les coupes de ce type, celle qui a des chances d'être la plus efficace (qui amputera le plus le polyèdre courant). Une règle intéressante en pratique, est de choisir λ de facon à ce que le second membre $f_0 = |\lambda \beta_i|_D$ soit le plus grand possible. Une facon évidente d'obtenir cette valeur de λ est d'énumérer tous les nombres λ

premier avec D. Mais en peut aussi remarquer que, si $\delta = pgcd(\beta_i, D)$, l'algorithme d'Euclide permet de définir deux entiers relatifs λ et μ tels que :

$$pgcd(\lambda, D) = 1 \text{ et } -\lambda\beta_i + \mu D = \delta.$$

C'est ce λ qui permet d'obtenir la plus grande valeur f_0, soit $D - \delta$.

3.4.3 L'algorithme dual de Gomory

On va ici expliquer les grandes lignes seulement de cet algorithme de Gomory, utilisant les coupes de même non.

L'algorithme de Gomory est une version " gomorisée " de l'algorithme dual du simplexe.

Supposons que l'on puisse toujours trouver une coupe de Gomory $\sum_{j\in J} f_j x_j \geq f_0$ garantissant la décroissance stricte de z. Dès lors :

- Tant que z n'est pas entier, on fabrique une coupe de Gomory déduit de $z = z_B + \sum_{i\in J} \overline{c}_j x_j$. Le pivot x_j de l'agorithme correspond alors à un $\overline{c}_j$ non nul et on a une décroissance stricte de z.
- Si z est entier, on rajoute une coupe de Gomory sur l'intégralité des variables de base. z décroit alors strictement mais peut devenir fractionnaire. On se retrouve dans le cas précédent.
- Et ainsi de suite, jusqu'à obtenir un minimum entier de z.

On obtient alors un algorithme qui converge. En fait il y a juste un petit problème : par rapport à la supposition initiale, on n'arrive justement pas toujours à trouver une coupe qui engendre une décroissance stricte de z.

Que-fait-il alors ?

Dans le cas où z est entier, on est dans un cas de dégénérescence où le pivot x_j a un $\overline{c}_j$ nul. Il faut alors considérer alors le polytope

$$\mathcal{P} = \{x' \in \mathbb{R}^{n'} / A'.x' = b, x' \geq 0\}$$

(où A' est le sous-matrice de A dont les colonnes correspondent aux $n' = |J'| + |J|$ variables de coût réduit nul) a, ou non, un point extrême entier ; en effet, dans l'affirmative, un tel sommet entier correspondrait à une solution optimale du problème.

Dans le cas contraire, on est certain qu'aucune combinaison des seules variables $x_j, j \in J'$ ne suffit à satisfaire la condition d'intégrité pour les variables de base et l'on peut alors rajouter la contrainte (coupe) :

$$\sum_{j \in J - J'} x_j \geq 1$$

Le pivot sur cette contrainte amène alors nécessairement une décroissance stricte de la fonction z puisque ses coefficients non nuls correspondent aux variables de coût réduit $\bar{c}_j$ non nul. On est alors ramené au problème précédent.

3.4.4 Exemple

Pour terminer, il est intéressant d'illustrer le fonctionnement de la méthode de coupe sur un petit exemple.

Prenons l'exemple à deux variables et une seule contrainte précédent :

$$(PNE) \begin{cases} \text{Min} \quad -8x_1 - 9x_2 \\ \text{S.c} \\ 8x_1 + 10x_2 \leq 39 \\ x_1, x_2 \quad \text{entiers} \quad \geq 0. \end{cases}$$

Comme $\frac{c_1}{a_{12}} = \frac{-8}{8} < \frac{c_2}{a_{12}} = \frac{-9}{10}$, la solution continue de ce problème est obtenue pour $x_1 = \frac{39}{8}$ et $x_2 = 0$ et la valeur correspondante de z est -39.

Le tableau simplexe, sous forme canonique relativement à la base optimale $\{x_1\}$ est :

	x_1	x_2	x_3	
fonction à minimiser $\rightarrow$	0	1	1	39
contrainte $\rightarrow$	1	$\frac{10}{8}$	$\frac{1}{8}$	$\frac{39}{8}$

x_3 est une variable d'écart ($x_3 \geq 0$) permettant de mettre le tableau sous forme standard (tous les contraintes sous forme d'égalités).

Cette solution n'est pas entière, on va appliquer la méthode des congruences décroissantes pour obtenir une solution entière. La condition d'intégrité sur la variable x_1 se déduit de l'équation :

$$x_1 = \frac{39}{8} - \frac{10}{8}x_2 - \frac{1}{8}x_3$$

et s'écrit :

$$10x_2 + x_3 \equiv 39 (\mod 8)$$

soit : $2x_2 + x_3 \equiv 7(\mod 8)$

d'où la coupe :

$$2x_2 + x_3 \geq 7$$

(il s'agit bien de la coupe ayant le second membre f_0 le plus élevé ; donc ici $\lambda = 1$).
En rajoutant une variable d'écart $x_4 \geq 0$ cette coupe se met sous la forme :

$$-2x_2 - x_3 + x_4 = -7$$

En rajoutant cette contrainte au tableau précédent on obtient la forme canonique du problème augmenté relativement à la base : $\{x_1, x_4\}$:

	x_1	x_2	x_3	x_4	
	0	1	1	0	39
	1	$\frac{10}{8}$	$\frac{1}{8}$	0	$\frac{39}{8}$
ligne pivot $\rightarrow$	0	-2	-1	1	-7

Cette base est dual réalisable (coûts réduits tous positifs ou nuls) mais pas primale réalisable ($x_4 = -7$ ne vérifié pas les conditions de positivité).
Effectuons une étape de l'algorithme dual su simplexe.
Le pivot sur la 2^e contrainte montre que c'est x_2 qui rentre en base car

$$Min\{-\frac{1}{-2}, -\frac{1}{-1}\} = \frac{1}{2}$$

Après pivotage on obtient le nouveau tableau (forme canonique relative à la nouvelle base $\{x_1, x_2\}$) :

x_1	x_2	x_3	x_4	
0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{71}{2}$
1	0	$-\frac{1}{2}$	$\frac{5}{8}$	$\frac{1}{2}$
0	1	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{7}{2}$

qui donne la solution : $x_1 = \frac{1}{2}$, $x_2 = \frac{7}{2}$.

Cette solution n'est toujours pas entière.
Continuant à appliquer la méthode des congruences décroissantes, exprimons l'intégrité d'une variable en base, par exemple x_2. On a : $x_2 = \frac{7}{2} - \frac{1}{2}x_3 + \frac{1}{2}x_4$

ce qui donne l'équation de congruence :

$$x_3 + x_4 \equiv 1(\mod 2)$$

d'où on déduit la coupe :

$$x_3 + x_4 \geq 1$$

En rajoutant cette coupe au tableau précédent après avoir introduit la variable d'écart $x_5 \geq 0$ on obtient :

x_1	x_2	x_3	x_4	x_5	
0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{71}{2}$
1	0	$-\frac{1}{2}$	$\frac{5}{8}$	0	$\frac{1}{2}$
0	1	$\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{7}{2}$
0	0	-1	-1	1	-1

Le pivot sur la dernière ligne conduit à choisir indifféremment x_3 ou x_4 comme variable candidate pour rentrer en base. En choisissant x_3 par exemple,

x_1	x_2	x_3	x_4	x_5	
0	0	0	0	$\frac{1}{2}$	35
1	0	0	$\frac{9}{8}$	$-\frac{1}{2}$	1
0	1	0	-1	$\frac{1}{2}$	3
0	0	1	1	-1	1

Cette fois la solution obtenue : $x_1 = 1$, $x_2 = 3$, $z = -35$
est entière.

De plus il se trouve qu'elle est aussi primale réalisable, car positive ; donc c'est une solution optimale du problème en nombre entiers.

Pour l'illustration graphique nous allons introduire les coupes engendrées précédemment :

La première coupe est :

$$2x_2 + x_3 \geq 7$$

En élimminant x_3 définie par :

$$x_3 = -8x_1 - 10x_2 + 39$$

On exprime la contrainte en fonction de x_1 et x_2 seulement, ce qui donne :

$$x_1 + x_2 \leq 4$$

La seconde coupe est :

$$x_3 + x_4 \geq 1$$

En effectuant les substitutions :

$$x_3 = -8x_1 - 10x_2 + 39$$

$$x_4 = -8x_1 - 8x_2 + 32$$

on peut exprimer $x_3 + x_4$ en fonction de x_1 et x_2 seulement :

$$x_3 + x_4 = -16x_1 - 18x_2 + 71$$

d'où la contrainte :

$$8x_1 + 9x_2 \leq 35$$

ces deux contraintes se coupent au point (1,3) comme il apparît sur la figure 2.5.
Une seule de ces deux contraintes (la première) est une face de l'enveloppe convexe

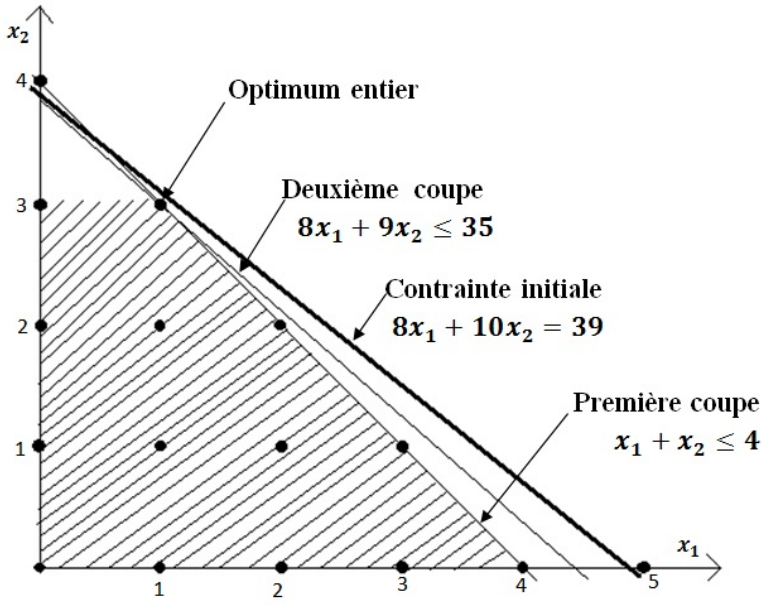


FIGURE 3.4 – les deux coupes engendrées par l'algorithme, et le polyèdre réduit (hachuré)

des solutions entières.