IFT-1575

Modèle de Recherche Opérationelle

Franz Girardin

Table des matières

1

Algorithme du Simplexe

1.1 Forme algébrique de l'algorithme

- **Forme standard** :

$$\text{Minimiser } z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\text{s.a.} \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leqslant b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leqslant b_m \\ x_1, x_2, \ldots, x_n \geqslant 0 \end{cases}$$

- **Introduction des variables d'écart** s_i :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + s_i = b_i, \quad s_i \geqslant 0$$

- **Fonction objectif ajustée** :

$$z + c_1x_1 + c_2x_2 + \cdots + c_nx_n = 0$$

- **Choix de la variable entrante** :
  - Sélectionner la variable avec le coefficient **le plus négatif** dans la fonction objectif.
- **Détermination de la variable sortante** :
  - Calculer le **ratio** pour chaque contrainte :

$$\boxed{\text{Ratio} = \frac{b_i}{a_{ij}}} \quad \text{avec } a_{ij} > 0$$

  - La variable avec le plus petit ratio positif est la **variable sortante**.
  - Si tous les a_{ij} sont négatifs, on conclue alors que le problème est **non borné** .
- **Pivot** :
  - Mettre à jour le tableau en pivotant sur le coefficient correspondant.
- **Critère d'optimalité** :
  - Si tous les coefficients $\overline{c}_j$de la fonction objectif sont tels que $\overline{c}_j \geqslant 0$, la solution est optimale.
- **Formules importantes** :
  - **Mise à jour des coefficients** :

Nouvelle ligne pivot :

$$\boxed{\text{Ligne pivot} \div \text{Coefficient pivot}}$$

Autres lignes :

$$\boxed{\text{Ligne} - (\text{Coefficient} \times \text{Nouvelle ligne pivot})}$$

- **Résumé des étapes** :
  - Formuler le problème en forme standard.
  - Identifier la variable entrante (coefficient le plus négatif).
  - Trouver la variable sortante (plus petit ratio positif).
  - Effectuer le pivot.
  - Répéter jusqu'à optimalité.

1.2 Forme Tabulée de l'algorithme

- **Construction du Tableau Simplexe** :
  - Organiser les équations du problème en **forme standard** dans un tableau.
  - La première colonne contient les **variables de base** (variables dépendantes).
  - Chaque ligne représente une équation, y compris la fonction objectif négative $(-z)$.

$$\begin{array}{ll} \text{Min} & -8x - 6y \\ \text{s.a.} & \\ & \begin{array}{rcl} 5x + 3y + \boldsymbol{u} & & \leqslant 30 \\ 2x + 3y \quad + \boldsymbol{p} & & \leqslant 24 \\ x + 3y \quad\quad + \boldsymbol{h} & & \leqslant 18 \\ \\ x, \quad y, \quad \boldsymbol{u}, \boldsymbol{p}, \boldsymbol{h} & & \geqslant 0 \end{array} \end{array}$$

v.d.	x	y	u	p	h	$-z$	t.d
$\boldsymbol{u}$	5	3	1				30
$\boldsymbol{p}$	2	3		1			24
$\boldsymbol{h}$	1	3			1		18
$-z$	−8	−6				1	0

Table 1.1 – État initial de la **forme standard**

- **Choix de la Variable Entrante** :
  - Sélectionner la variable avec le coefficient le plus négatif dans la ligne $(-z)$.
  - Cette variable améliorera le plus la valeur de l'objectif.

v.d.	x	y	u	p	h	$-z$	t.d
u	5	3	1				30
p	2	3		1			24
h	1	3			1		18
$-z$	−8	−6				1	0

TABLE 1.2 – Variable d'entrée choisie

- **Détermination de la Variable Sortante** :
  - Calculer les **ratios** pour chaque variable de base :

$$\boxed{\text{Ratio} = \frac{\text{Terme de droite}}{\text{Coefficient de la variable entrante}}}$$

  - Ignorer les coefficients négatifs ou nuls de la variable entrante.
  - La variable avec le plus petit ratio positif est la **variable sortante**.

Exemple

$$\lim(u) = \text{t.d.}(u) \div x(u) = 30 \div 5 = 6$$
$$\lim(p) = \text{t.d.}(p) \div x(p) = 24 \div 2 = 12$$
$$\lim(h) = \text{t.d.}(h) \div x(h) = 18 \div 1 = 12$$

- **Pivot** :
  - Identifier le **coefficient pivot** à l'intersection de la colonne de la variable entrante et de la ligne de la variable sortante.
  - Diviser la ligne pivot par le coefficient pivot pour obtenir un 1.
  - Utiliser des opérations sur les lignes pour annuler les autres coefficients dans la colonne de la variable entrante.

v.d.	x	y	u	p	h	$-z$	t.d
$u \to x$	5	3	1				30
p	2	3		1			24
h	1	3			1		18
$-z$	−8	−6				1	0

TABLE 1.3 – État durant le pivot

v.d.	x	y	u	p	h	$-z$	t.d
$u \to x$	5	3	1				30
p	2	3		1			24
h	1	3			1		18
$-z$	−8	−6				1	0

TABLE 1.4 – Coefficient du pivot identifié

v.d.	x	y	u	p	h	$-z$	t.d
x	1	$3/5$	$1/5$				$30/5$
p	2	3		1			24
h	1	3			1		18
$-z$	−8	−6				1	0

TABLE 1.5 – Division par le cofficient du pivot

v.d.	x	y	u	p	h	$-z$	t.d
x	1	$3/5$	$1/5$				6
p	0	$9/5$	$-2/5$	1			12
h	1	3			1		18
$-z$	−8	−6				1	0

TABLE 1.6 – $ligne(p) - 2 \times ligne(x)$

v.d.	x	y	u	p	h	$-z$	t.d
x	1	$3/5$	$1/5$				6
p	0	$9/5$	$-2/5$	1			12
h	0	$12/5$	$-1/5$		1		12
$-z$	−8	−6				1	0

TABLE 1.7 – $ligne(h) - 1 \times ligne(x)$

v.d.	x	y	u	p	h	$-z$	t.d
x	1	$3/5$	$1/5$				6
p	0	$9/5$	$-2/5$	1			12
h	0	$12/5$	$-1/5$		1		12
$-z$	0	$-6/5$	$8/5$			1	48

TABLE 1.8 – $ligne(-z) + 8 \times ligne(x)$

Nous avons complété **une itération** de l'algorithme. On obtien alors la solution suivante :

$$\boxed{[y, u = 0] \implies x = 6, p = 12, h = 12, z = -48}$$

- **Itération** :
  - Répéter le processus de sélection des variables entrante et sortante, suivi du pivot, jusqu'à ce que tous les coefficients de $(-z)$ soient positifs ou nuls.

1.2.1 Choix de la variable d'entrée

v.d.	x	y	u	p	h	$-z$	t.d
x	1	3/5	1/5				6
p	0	9/5	−2/5	1			12
h	0	12/5	−1/5		1		12
$-z$	0	−6/5	8/5			1	48

La variable d'entrée est donc $x_s = \boldsymbol{y}$

1.3 Choix de la variable de sortie

Démonstration

$$\lim(x) = \text{t.d.}(x) \div y(x) = 6 \div 3/5 = 10$$
$$\lim(p) = \text{t.d.}(p) \div y(p) = 12 \div 9/5 = 20/3$$
$$\lim(h) = \text{t.d.}(h) \div y(h) = 18 \div 12/5 = \mathbf{5}$$

La variable de sortie est donc $x_r = \boldsymbol{h}$, puisqu'après $y \leqslant 5$, h devient négatif, ce qui viole la contrainte de **non négativité**.

v.d.	x	y	u	p	h	$-z$	t.d
x	1	3/5	1/5				6
p	0	9/5	−2/5	1			12
$h \to y$	0	12/5	−1/5		1		12
$-z$	0	−6/5	8/5			1	48

TABLE 1.9 – Coefficient de pivot

Puisque le pivot est **12/5**, on divise $ligne(h)$ par cette même valeur pour que y soit variable dépendante à $ligne(h)$.

v.d.	x	y	u	p	h	$-z$	t.d
x	1	3/5	1/5				6
p	0	9/5	−2/5	1			12
y	0	1	−1/12		5/12		5
$-z$	0	−6/5	8/5			1	48

TABLE 1.10 – $ligne(h) \div 12/5$ devient $ligne(y)$

On soustrait maintenant $ligne(y)$ à $ligne(p)$ et $ligne(x)$ de façon à avoir un coefficient de 0 sous y.

v.d.	x	y	u	p	h	$-z$	t.d
x	1	3/5	1/5				6
p	0	0	−1/4	1	−3/4		3
y	0	1	−1/12		5/12		5
$-z$	0	−6/5	8/5			1	48

TABLE 1.11 – $ligne(p) \; - \; 9/5 \; ligne(y)$

v.d.	x	y	u	p	h	$-z$	t.d
x	1	0	1/4	0	−1/4		3
p	0	0	−1/4	1	−3/4		3
y	0	1	−1/12		5/12		5
$-z$	0	−6/5	8/5			1	48

TABLE 1.12 – $ligne(x) \; - \; 3/5 \; ligne(h)$

v.d.	x	y	u	p	h	$-z$	t.d
x	1	0	1/4	0	−1/4		3
p	0	0	−1/4	1	−3/4		3
y	0	1	−1/12		5/12		5
$-z$	0	0	3/2	0	1/2	1	54

TABLE 1.13 – $ligne(-z) \; - \; 6/5 \; ligne(h)$

Nous avons complété **une seconde itération** de l'algorithme. Le système est associé à la solution :

$$\boxed{[u, h = 0] \implies x = 3, p = 3, y = 5, z = -54}$$

Il n'est pas intéressant d'augmenter ni la valeur de u, ni la valeur de h car la valeur de z augmente. Ainsi, nous sommes à l'optimum.

- **Solution Optimale** :
  - Lorsque tous les coefficients de $(-z)$ sont positifs ou nuls, la solution optimale est atteinte.
  - Les valeurs des variables de base se trouvent dans la colonne des termes de droite.
- **Nombre de Solutions de Base** :
  - Pour un système avec n variables et m équations :

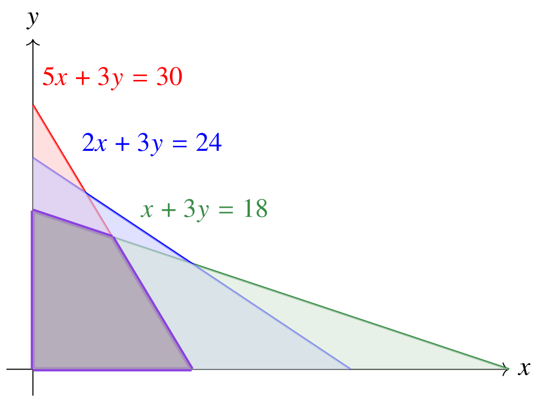
$$\boxed{\text{Nombre de solutions de base} = \binom{n}{m} = \frac{n!}{m!(n-m)!}}$$

- **Résumé des Étapes** :
  - **Organiser** les équations en tableau.

▷ **Choisir** la variable entrante (x_s) avec le coefficient le plus négatif dans ($-z$).
▷ **Déterminer** la variable sortante (x_r) avec le plus petit ratio positif.
▷ **Effectuer** le pivot sur le coefficient pivot.
▷ **Répéter** jusqu'à optimalité.

1.4 Résolution Graphique

- **Méthode de Résolution Graphique** (pour problèmes à deux variables) :
  - ▷ **Tracer les contraintes** sur un plan cartésien :
    - ▷ Convertir chaque inégalité en équation pour tracer la droite correspondante.
    - ▷ Identifier la zone admissible (région des solutions réalisables) en tenant compte des inégalités.
  - ▷ **Déterminer les sommets** de la zone admissible :
    - ▷ Trouver les points d'intersection des droites (sommets du polygone formé).
  - ▷ **Calculer la valeur de la fonction objectif** en chaque sommet :
    - ▷ Évaluer $z = c_1x + c_2y$ pour chaque couple (x, y).
  - ▷ **Choisir la solution optimale** :
    - ▷ Pour une maximisation, sélectionner le sommet avec la valeur de z la plus élevée.
    - ▷ Pour une minimisation, choisir le sommet avec la valeur de z la plus faible.



1.4.1 Notes Importantes

- **Dans un problème à deux variables**, la zone admissible est un polygone convexe sur le plan xy, et la solution optimale se trouve toujours à l'un des sommets de ce polygone.
- **Vérification de la Faisabilité** :
  - ▷ S'assurer que les points trouvés respectent toutes les contraintes.
- **Cas Particuliers** :
  - ▷ **Solutions Multiples** : Si la fonction objectif est parallèle à une contrainte bordant la région admissible.
  - ▷ **Région Non Bornée** : Si la région admissible n'est pas limitée, la solution optimale peut être non bornée.

v.d.	x_1	x_2	$\cdots$	x_r	$\cdots$	x_m	x_{m+1}	$\cdots$	x_s	$\cdots$	x_n	$-z$		t.d.
x_1	1						$\overline{a}_{1,m+1}$	$\cdots$	$\overline{a}_{1s}$	$\cdots$	$\overline{a}_{1n}$			$\overline{b}_1$
x_2		1					$\overline{a}_{2,m+1}$	$\cdots$	$\overline{a}_{2s}$	$\cdots$	$\overline{a}_{2n}$			$\overline{b}_2$
$\vdots$			$\ddots$					$\ddots$		$\ddots$				$\vdots$
x_r				1			$\overline{a}_{r,m+1}$	$\cdots$	$\overline{a}_{rs}$	$\cdots$	$\overline{a}_{rn}$			$\overline{b}_r$
$\vdots$					$\ddots$		$\ddots$	$\ddots$		$\ddots$	$\overline{a}_{1n}$			$\vdots$
x_m						1	$\overline{a}_{m,m+1}$	$\cdots$	$\overline{a}_{ms}$	$\cdots$	$\overline{a}_{mn}$			$\overline{b}_m$
$-z$							$\overline{c}_{m+1}$	$\cdots$	$\overline{c}_s$	$\cdots$	$\overline{c}_n$	1		$\overline{z}$

1.5 Forme Générale de l'Algorithme

- **Notations** :
  - x_j : Variables de décision, $j = 1, 2, \ldots, n$.
  - c_j : Coefficients dans la fonction objectif.
  - a_{ij} : Coefficients des contraintes, $i = 1, 2, \ldots, m$.
  - b_i : Termes de droite des contraintes.
  - x_s : Variable d'entrée
  - x_r : Variable de sortie
- **Forme Standard** :

$$\text{Minimiser} \qquad z = \sum_{j=1}^{n} c_j x_j$$

$$\text{sous contraintes} \qquad \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, \ldots, m$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n$$

- **Tableau Simplexe Général** :
  - Les variables de base x_i forment une base identité.
  - Les variables non de base sont initialement fixées à zéro.
- **Choix de la Variable Entrante** :
  - Si tous les coefficients réduits $\overline{c}_j \geqslant 0$, la solution est optimale.
  - Sinon, choisir x_s tel que :

$$\boxed{\overline{c}_s = \min\{c_j \mid j = 1, 2, \ldots, n\}}$$

- **Détermination de la Variable Sortante** :
  - Calculer les ratios pour $\overline{a}_{is} > 0$:

$$\boxed{\text{Ratio} = \frac{\overline{b}_i}{\overline{a}_{is}}}$$

  - La variable sortante x_r correspond au plus petit ratio positif.
  - Si $\overline{a}_{is} \leqslant 0$ pour tout i, le problème est non borné.
- **Pivot et Mise à Jour du Tableau** :
  - **Ligne Pivot** (r) :

$$\boxed{\text{Nouvelle ligne } r = \frac{\text{Ancienne ligne } r}{\overline{a}_{rs}}}$$

  - **Autres Lignes** ($i \neq r$) :

$$\boxed{\text{Nouvelle ligne } i = \text{Ancienne ligne } i - \overline{a}_{is} \times \text{Nouvelle ligne } r}$$

  - **Fonction Objectif** :

$$\boxed{\overline{\overline{c}}_j = \overline{c}_j - \overline{c}_s \times \overline{a}_{rj}}$$

$$\boxed{\overline{\overline{z}} = \overline{z} + \overline{c}_s \times \overline{b}_r}$$

- **Résumé des Étapes de l'Algorithme** :
  - **Initialisation** : Formuler le problème en forme standard et construire le tableau initial.
  - **Itération** :
    - Choisir la variable entrante x_s avec le $\overline{c}_s$ le plus négatif.
    - Déterminer la variable sortante x_r en calculant les ratios.
    - Effectuer le pivot pour mettre à jour le tableau.
  - **Vérification d'Optimalité** :
    - Si tous les $\overline{c}_j \geqslant 0$, la solution optimale est atteinte.
    - Sinon, répéter l'itération.
- **Formules Clés** :
  - **Mise à Jour des Coefficients** :

$$\boxed{\overline{\overline{a}}_{ij} = \overline{a}_{ij} - \overline{a}_{is} \times \overline{a}_{rj}}$$

▷ **Mise à Jour des Termes de Droite** :

$$\boxed{\overline{\overline{b}}_i = \overline{b}_i - \overline{a}_{is} \times \overline{b}_r}$$

1.6 Forme Matricielle de l'Algorithme

1.6.1 Notations Matricielles

- **Variables de Décision** :

$$\mathbf{x}_{n\times 1} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- **Coefficients de l'Objectif** :

$$\mathbf{c}_{n\times 1} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

- **Matrice des Contraintes** :

$$\mathbf{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- **Termes de Droite** :

$$\mathbf{b}_{m\times 1} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

1.6.2 Formulation Matricielle du Problème

Le problème de programmation linéaire en forme standard peut être écrit en notation matricielle comme suit :

$$\begin{aligned} &\text{Minimiser} \quad z = \mathbf{c}^T\mathbf{x} \\ &\text{sous contraintes} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \\ &\qquad\qquad\quad \mathbf{x} \geqslant 0 \end{aligned}$$

1.6.3 Interprétation

▷ **Objectif** : $\mathbf{c}^T\mathbf{x} = \sum_{j=1}^{n} c_j x_j$

▷ **Contraintes** : Chaque équation $\sum_{j=1}^{n} a_{ij}x_j = b_i$ est représentée par le produit matriciel $\mathbf{A}\mathbf{x} = \mathbf{b}$

▷ **Non-négativité** : $\mathbf{x} \geqslant 0$ signifie que chaque $x_j \geqslant 0$

1.6.4 Exemple : Problème du Restaurateur

Considérons le problème du restaurateur avec les variables x et y, et les variables d'écart u, p, h.

$$\begin{aligned} &\text{Minimiser} \quad z = 8x + 6y \\ &\text{sous contraintes} \\ &5x + 3y + u = 30 \\ &2x + 3y + p = 24 \\ &x + 3y + h = 18 \\ &x, y, u, p, h \geqslant 0 \end{aligned}$$

En notation matricielle :

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ u \\ p \\ h \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} -8 \\ -6 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 5 & 3 & 1 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 \\ 1 & 3 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 30 \\ 24 \\ 18 \end{bmatrix}$$

1.6.5 Base et Variables de Base

- Une **base** B est une sous-matrice $m \times m$ non singulière de $\mathbf{A}$.
- Les variables correspondant aux colonnes de B sont les **variables de base**.
- Les autres variables sont les **variables hors-base**.

Exemple de Bases

$$B_1 = \begin{bmatrix} u & p & h \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} x & u & h \\ 5 & 1 & 0 \\ 2 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad B_3 = \begin{bmatrix} x & p & y \\ 5 & 0 & 3 \\ 2 & 1 & 3 \\ 1 & 0 & 3 \end{bmatrix}$$

1.6.6 Partitionnement des Matrices

- **Variables** :

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_R \end{bmatrix}$$

où $\mathbf{x}_B$ sont les variables de base et $\mathbf{x}_R$ les variables hors-base.

- **Coefficients de l'Objectif** :

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_B \\ \mathbf{c}_R \end{bmatrix}$$

- **Matrice des Contraintes** :

$$\mathbf{A} = [B \ : \ R]$$

où B est la base et R le reste de la matrice.

1.6.7 Formulation Réécrite

Le problème peut être réécrit comme :

$$\begin{aligned}
&\text{Minimiser} \quad z = \mathbf{c}_B^T \mathbf{x}_B + \mathbf{c}_R^T \mathbf{x}_R \\
&\text{sous contraintes} \quad B\mathbf{x}_B + R\mathbf{x}_R = \mathbf{b} \\
&\mathbf{x}_B \geqslant 0, \quad \mathbf{x}_R \geqslant 0
\end{aligned}$$

1.6.8 Résolution par le Simplexe en Notation Matricielle

1. **Expression de $\mathbf{x}_B$** :
$$\mathbf{x}_B = B^{-1}\mathbf{b} - B^{-1}R\mathbf{x}_R$$
2. **Substitution dans l'Objectif** :
$$z = \mathbf{c}_B^T(B^{-1}\mathbf{b} - B^{-1}R\mathbf{x}_R) + \mathbf{c}_R^T\mathbf{x}_R$$
3. **Définition des Multiplicateurs du Simplexe** :
$$\pi^T = \mathbf{c}_B^T B^{-1}$$
4. **Calcul des Coûts Réduits** :
$$\tilde{\mathbf{c}}_R^T = \mathbf{c}_R^T - \pi^T R$$
5. **Nouvelle Forme de l'Objectif** :
$$z = \pi^T\mathbf{b} + \tilde{\mathbf{c}}_R^T\mathbf{x}_R$$

1.6.9 Exemple Numérique

- **Matrice de Base** B et son inverse B^{-1}
$$B = \begin{bmatrix} 5 & 0 & 3 \\ 2 & 1 & 3 \\ 1 & 0 & 3 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 1/4 & 0 & -1/4 \\ -1/4 & 1 & -3/4 \\ -1/12 & 0 & 5/12 \end{bmatrix}$$
- **Calcul de** π^T :
$$\begin{aligned}
\pi^T = \mathbf{c}_B^T B^{-1} &= \begin{bmatrix} -8 & 0 & -6 \end{bmatrix} \begin{bmatrix} 1/4 & 0 & -1/4 \\ -1/4 & 1 & -3/4 \\ -1/12 & 0 & 5/12 \end{bmatrix} \\
&= \begin{bmatrix} -\frac{3}{2} & 0 & -\frac{1}{2} \end{bmatrix}
\end{aligned}$$
- **Calcul des Coûts Réduits** :
$$\begin{aligned}
\tilde{\mathbf{c}}_R^T = \mathbf{c}_R^T - \pi^T R &= \begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} -\frac{3}{2} & 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{3}{2} & \frac{1}{2} \end{bmatrix}
\end{aligned}$$
- **Valeur de l'Objectif** :
$$z = \pi^T\mathbf{b} = \begin{bmatrix} -\frac{3}{2} & 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 30 \\ 24 \\ 18 \end{bmatrix} = -54$$

1.6.10 Résumé des Étapes

1. **Initialisation** : Choisir une base initiale B (par exemple, les variables d'écart).
2. **Calculer** B^{-1} et $\pi^T = \mathbf{c}_B^T B^{-1}$.
3. **Calculer** les coûts réduits $\tilde{\mathbf{c}}_R^T = \mathbf{c}_R^T - \pi^T R$.
4. **Vérifier** la condition d'optimalité : si tous les $\tilde{c}_j \geqslant 0$, la solution est optimale.
5. **Sinon**, choisir la variable entrante correspondante au $\tilde{c}_j < 0$ le plus négatif.
6. **Déterminer** la variable sortante en utilisant la règle du rapport minimal.
7. **Mettre à jour** la base B et répéter les étapes jusqu'à l'optimalité.

1.7 Analyse Post-Optimale

- **Modification des Termes de Droite** (b) :
  - Si les termes de droite b sont modifiés par Δb, la nouvelle solution est :
$$\boxed{\bar{\bar{b}} = \bar{b} + B^{-1}\Delta b}$$
  - La nouvelle valeur de la fonction objectif est :
$$\boxed{\tilde{z} = \bar{z} + \pi^T\Delta b}$$
  - **Interprétation** :
    - Si $\bar{\bar{b}} \geqslant 0$, la base reste optimale.
    - Si $\bar{\bar{b}}$ contient des valeurs négatives, une réoptimisation est nécessaire.
- **Modification d'un Coût dans la Fonction Objectif** (c_j) :
  - Si un coût c_j est modifié par Δc_j, le nouveau coefficient réduit est :
$$\boxed{\bar{\bar{c}}_j = \bar{c}_j + \Delta c_j}$$
  - **Interprétation** :
    - Si $\bar{\bar{c}}_j \geqslant 0$ pour toutes les variables hors-base, la solution reste optimale.
    - Sinon, il faut réoptimiser en introduisant la variable correspondante dans la base.
- **Modification d'une Variable Hors-Base (Colonne** a_j**)** :

- Si une colonne a_j est modifiée par Δa_j, le nouveau coefficient réduit devient :

$$\boxed{\bar{\tilde{c}}_j = \bar{c}_j - \pi^T \Delta a_j}$$

- **Interprétation** :
  - Si $\bar{\tilde{c}}_j \geqslant 0$, la base reste optimale.
  - Si $\bar{\tilde{c}}_j < 0$, une réoptimisation est nécessaire.

- **Ajout d'une Nouvelle Variable (Nouvelle Action)** :
  - Pour une nouvelle variable x_r avec coût c_r et colonne a_r :

$$\boxed{\tilde{a}_r = B^{-1} a_r}$$

$$\boxed{\tilde{c}_r = c_r - \pi^T a_r}$$

  - **Interprétation** :
    - Si $\tilde{c}_r \geqslant 0$, la nouvelle variable n'améliore pas la solution actuelle.
    - Si $\tilde{c}_r < 0$, il est avantageux d'introduire x_r dans la base et de réoptimiser.

- **Généralités sur l'Analyse Post-Optimale** :
  - L'analyse post-optimale permet d'évaluer l'impact de modifications mineures sans refaire tout le calcul.
  - Elle utilise les informations du dernier tableau optimal, notamment B^{-1} et les multiplicateurs de Lagrange π.
  - Elle est utile pour les **analyses de sensibilité** et la prise de décision informée.

2

Rappel d'algèbre linéaire

2.1 Définitions

Soit $x, y \in \mathbb{R}^n$, nous avons les quantités suivantes :

$$\text{Produit scalaire} \quad : \quad x^T y = y^T x = \sum_{j=1}^{n} x_j y_j$$

$$\text{Vecteur :} \quad \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{Tranposée :} \quad [x_1, \ldots x_n]$$

2.1.1 Exemple de produit scalaire

Prenons $n = 3, \boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$, et $x^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
Calculons le produit scalaire :

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{y} &= x_1 y_1 + x_2 y_2 + x_3 y_3 \\ &= (1)(4) + (2)(5) + (3)(6) \\ &= 4 + 10 + 18 \\ &= 32 \end{aligned}$$

Ainsi, le produit scalaire de $\boldsymbol{x}$ et $\boldsymbol{y}$ est $\boxed{32}$.

2.1.2 Dépendance linéaire

Note:-

Un ensemble de vecteur $x^1, \ldots, x^k \in \mathbb{R}^n$ est dit **linéairement dépendant** s'il existe k scalaires $\lambda_1, \lambda_2, \ldots \lambda_k \in \mathbb{R}$ *non nul* tel que :

$$\lambda_1 x^1 + \lambda_2 x^2 + \cdots + \lambda_k x^k = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = 0$$

Dans le contexte d'une matrice, s'il existe une combinaison linéaire de ligne ou de colonne qui peuvent engendrer une autre ligne ou colonne, on dit que la matrice linéairement dépendantes.

2.1.3 Exemples de dépendance

Considérons la matrice suivante : Considérons la matrice suivante :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix}$$

Les lignes de cette matrice sont linéairement dépendantes. La première ligne est une combinaison linéaire de la 2e ligne :

$$\text{Ligne } 2 = \lambda \times \text{Ligne } 1 = 2 \times \text{Ligne } 1$$

D'ailleurs, la 3e ligne est une combinaison linéaire des deux premières lignes :

$$\text{Ligne } 3 = \text{Ligne } 1 + 1 \times \text{Ligne } 2$$

Cela signifie qu'il existe une relation linéaire entre les lignes, donc la matrice **A** est linéairement dépendante.

2.1.4 Exemples d'indépendance linéaire

Considérons la matrice suivante :

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Les lignes de cette matrice sont linéairement indépendantes car aucune des lignes ne peut être obtenue comme une combinaison linéaire des autres lignes. Il s'agit d'une matrice identité, et donc elle est linéairement indépendante.

2.1.5 Matrice carrée

Soit une matrice A :

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \ldots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

alors, on dit que l'élément a_{ij} est à **l'intersection** de la ligne i et de la colonne j de la matrice. Une **matrice carrée** est une matrice $A_{m \times n}$ où $m = n$ et un vecteur de dimension n peut être vue comme une matrice de dimension $1 \times n$. La **matrice identité** I est une **matrice carrée** telle qu'on a :

$$\boxed{\forall a_{ij} \in A_{m \times n} = I, \ [i = j] \implies a_{ij} = 1, \ \textbf{et} \ [i \neq j] \implies a_{ij} = 0}$$

2.2 Multiplication de Matrices

2.2.1 Formule Générale

Soient les matrices $A_{a\times n}$ et $B_{n\times b}$. Le produit matriciel $C = A \times B$ est une matrice de taille $a \times b$, où chaque élément c_{ij} est donné par :

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}, \quad \text{pour } i = 1,\ldots,a \text{ et } j = 1,\ldots,b$$

Autrement dit, l'élément c_{ij} est le produit scalaire de la $i^{\text{ème}}$ ligne de A et de la $j^{\text{ème}}$ colonne de B.

2.2.2 Exemple Concret

Prenons A une matrice 2×3 et B une matrice 3×2 :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

Le produit $C = A \times B$ est une matrice de taille 2×2 :

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Calculons chaque élément de C :

Calcul de c_{11} :

$$\begin{aligned} c_{11} &= a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31} \\ &= (1)(7) + (2)(9) + (3)(11) \\ &= 7 + 18 + 33 \\ &= 58 \end{aligned}$$

Calcul de c_{12} :

$$\begin{aligned} c_{12} &= a_{11} \times b_{12} + a_{12} \times b_{22} + a_{13} \times b_{32} \\ &= (1)(8) + (2)(10) + (3)(12) \\ &= 8 + 20 + 36 \\ &= 64 \end{aligned}$$

Calcul de c_{21} :

$$\begin{aligned} c_{21} &= a_{21} \times b_{11} + a_{22} \times b_{21} + a_{23} \times b_{31} \\ &= (4)(7) + (5)(9) + (6)(11) \\ &= 28 + 45 + 66 \\ &= 139 \end{aligned}$$

Calcul de c_{22} :

$$\begin{aligned} c_{22} &= a_{21} \times b_{12} + a_{22} \times b_{22} + a_{23} \times b_{32} \\ &= (4)(8) + (5)(10) + (6)(12) \\ &= 32 + 50 + 72 \\ &= 154 \end{aligned}$$

Résultat Final : La matrice produit $C = A \times B$ est donc :

$$C = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Ainsi, la multiplication de A et B nous donne la matrice C ci-dessus.

2.3 Transposée d'une Matrice : Formule Générale et Exemple

2.3.1 Formule Générale

Soit A une matrice de taille $m \times n$:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

La **transposée** de A, notée A^T, est une matrice de taille $n \times m$, obtenue en échangeant les lignes et les colonnes de A :

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

Autrement dit, l'élément $(A^T)_{ij} = a_{ji}$ pour $i = 1,\ldots,n$ et $j = 1,\ldots,m$.

2.3.2 Exemple Concret

Prenons A une matrice 2×3 :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

La transposée de A est une matrice 3×2 obtenue en échangeant les lignes et les colonnes :

$$A^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Ainsi, les éléments de A sont réorganisés pour former A^T :

- ▷ a_{11} devient $(A^T)_{11}$, a_{21} devient $(A^T)_{12}$
- ▷ a_{12} devient $(A^T)_{21}$, a_{22} devient $(A^T)_{22}$
- ▷ a_{13} devient $(A^T)_{31}$, a_{23} devient $(A^T)_{32}$

Vérification des éléments :

$$\begin{aligned} (A^T)_{11} &= a_{11} = 1 \\ (A^T)_{12} &= a_{21} = 4 \\ (A^T)_{21} &= a_{12} = 2 \\ (A^T)_{22} &= a_{22} = 5 \\ (A^T)_{31} &= a_{13} = 3 \\ (A^T)_{32} &= a_{23} = 6 \end{aligned}$$

Ainsi, la transposée A^T est :

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Conclusion : La transposition d'une matrice consiste à transformer ses lignes en colonnes (et vice versa), ce qui revient à refléter la matrice par rapport à sa diagonale principale.

2.4 Calcul du Déterminant d'une Matrice : Formule Générale et Exemple

2.4.1 Formule Générale

Le déterminant d'une matrice carrée $n \times n$ est défini de manière récursive en utilisant les mineurs et les cofacteurs. La formule générale pour le déterminant de la matrice A est :

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij})$$

où :

- a_{ij} est l'élément de la $i^{\text{ème}}$ ligne et $j^{\text{ème}}$ colonne de A. - A_{ij} est la matrice de dimension $(n-1) \times (n-1)$ obtenue en retirant la ligne i et la colonne j de A. - Le signe $(-1)^{i+j}$ est le signe du cofacteur, positif si $i + j$ est pair, négatif si $i + j$ est impair.

2.4.2 Exemple Concret avec une Matrice 2×2

Soit la matrice A :

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$$

Le déterminant de A est calculé comme suit :

$$\begin{aligned} \det(A) &= a_{11}a_{22} - a_{12}a_{21} \\ &= (2) \times (4) - (3) \times (1) \\ &= 8 - 3 \\ &= 5 \end{aligned}$$

Ainsi, le déterminant de A est $\boxed{5}$.

2.4.3 Remarque sur les Mineurs et les Cofacteurs

Pour une matrice 2×2, le calcul du déterminant est direct. Pour des matrices de taille supérieure, on utilise les mineurs et les cofacteurs en appliquant le développement par rapport à une ligne ou une colonne.

2.4.4 Exemple avec une Matrice 3×3

Soit la matrice B :

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Le déterminant de B est :

$$\det(B) = b_{11} \det(B_{11}) - b_{12} \det(B_{12}) + b_{13} \det(B_{13})$$

où :

- B_{1j} est la matrice 2×2 obtenue en retirant la première ligne et la $j^{\text{ème}}$ colonne de B. - Par exemple, B_{11} est obtenu en supprimant la première ligne et la première colonne de B.

Calcul détaillé :

Supposons que :

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{bmatrix}$$

Calculons les mineurs :

$$det(B_{11}) = \begin{vmatrix} 4 & 5 \\ 0 & 6 \end{vmatrix} = (4)(6) - (5)(0) = 24$$
$$det(B_{12}) = \begin{vmatrix} 0 & 5 \\ 1 & 6 \end{vmatrix} = (0)(6) - (5)(1) = -5$$
$$det(B_{13}) = \begin{vmatrix} 0 & 4 \\ 1 & 0 \end{vmatrix} = (0)(0) - (4)(1) = -4$$

Calcul du déterminant de B :

$$\begin{aligned} \det(B) &= b_{11} \det(B_{11}) - b_{12} \det(B_{12}) + b_{13} \det(B_{13}) \\ &= (1)(24) - (2)(-5) + (3)(-4) \\ &= 24 + 10 - 12 \\ &= 22 \end{aligned}$$

Ainsi, le déterminant de B est $\boxed{22}$.

2.4.5 Conclusion

Le calcul du déterminant d'une matrice carrée implique l'utilisation des mineurs et des cofacteurs, en développant le déterminant par rapport à une ligne ou une colonne. Pour une matrice $n \times n$, cette méthode est appliquée récursivement jusqu'à obtenir des déterminants de matrices 2×2, qui sont calculés directement.

2.5 Rang et matrices non singulières

Le **rang** d'une matrice est le nombre maximum de lignes (ou colonnes) linéairement indépendantes.

2.5.1 Plein rang

Une matrice $\mathbf{A}_{m \times n}$ est dite de **plein rang** si son rang est égal à $\min(m, n)$.

Exemple : Considérons la matrice **A** suivante de plein rang (rang = 2) :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Cette matrice a deux lignes linéairement indépendantes, donc elle est de plein rang.

2.5.2 Matrice non singulière

Une matrice carrée $\mathbf{A}_{n \times n}$ est dite **non singulière** si :

- ▷ ses lignes (ou colonnes) sont linéairement indépendantes,
- ▷ elle est de plein rang (rang = n),
- ▷ det(**A**,
- ▷ elle possède une inverse $\mathbf{A}^{-1}$ telle que $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

Exemple : Considérons la matrice suivante :

$$\mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}$$

Cette matrice est non singulière car son déterminant est non nul ($\det(\mathbf{B}) = 1$), ses lignes sont linéairement indépendantes, et elle possède une inverse.

3

Programmation Linéaire en Nombres Entiers

3.1 Théorie

- **Introduction** :
  - La programmation linéaire en nombres entiers (PLNE) est utilisée lorsque les variables doivent prendre des valeurs entières.
  - Les solutions fractionnaires ne sont pas acceptables dans certains contextes (exemple : on ne peut pas commander une fraction d'un produit).
- **Modèles Binaires** :
  - Utilisation de variables binaires $x_i \in \{0, 1\}$ pour modéliser des décisions oui/non.
  - Exemples d'applications : sélection de projets, affectation de tâches, problèmes de routage.
- **Problème de Coloration de Graphe** :
  - Objectif : Colorier les sommets d'un graphe de sorte que deux sommets adjacents n'aient pas la même couleur, en minimisant le nombre de couleurs utilisées.
  - **Variables** :

$$c_i = \begin{cases} 1 & \text{si la couleur } i \text{ est utilisée} \\ 0 & \text{sinon} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{si } j \text{ est colorié avec la couleur } i \\ 0 & \text{sinon} \end{cases}$$

  - **Contraintes** :
    - Chaque sommet doit être colorié avec exactement une couleur :

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \in V$$

    - Deux sommets adjacents ne peuvent pas avoir la même couleur :

$$x_{ij}+x_{ik} \leqslant c_i, \quad \forall (j,k) \in E, \quad \forall i = 1, \ldots, m$$

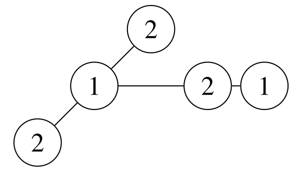
    - Liaison entre x_{ij} et c_i :

$$x_{ij} \leqslant c_i, \quad \forall i = 1, \ldots, m, \quad \forall j \in V$$

  - **Objectif** :

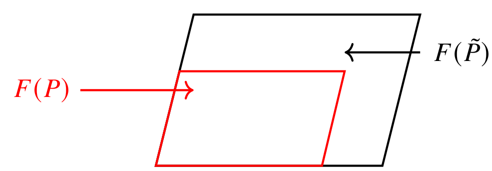
$$\text{Minimiser} \quad \sum_{i=1}^{m} c_i$$

3.1.1 Représentation du réseau



- **Principes Généraux en PLNE** : Soient un problème de **minimisation** en programmation linéaire dénoté P et sa relaxation continue, $\tilde{P}$. La **relaxation continue** est obtenue en **ignorant certaines contraintes** imposées par le contexte de P.
  - **Relaxation Continue** :
    - En relâchant les contraintes d'intégralité, on obtient une borne inférieure sur la valeur optimale.
    - Le domaine faisable de la PLNE est inclus dans celui de sa relaxation :

$$F(P) \subseteq F(\tilde{P})$$



    - Si nous minimisons sur un domaine réalisable plus grand $F(\tilde{P})$, qui inclut le plus petit domaine $F(P)$, la valeur optimale de l'objectif ne peut qu'être meilleure ou demeurer la même.

$$v(\tilde{P}) \leqslant v(P)$$

  - **Optimalité des Solutions Entières** :
    - Si la solution optimale de la relaxation continue est entière, elle est aussi optimale pour le problème entier.
- **Exemple Illustratif** :
  - Considérons le problème :

$$\begin{array}{rl} \text{Minimiser} & z = -x_1 - 5x_2 \\ \text{sous contraintes} & x_1 + 10x_2 \leqslant 20 \\ & x_1 \leqslant 2 \\ & x_1, x_2 \geqslant 0 \text{ et entiers} \end{array}$$

  - La relaxation continue permet de trouver une solution fractionnaire qui sert de borne inférieure.
  - Les solutions obtenues par arrondi ou troncature ne sont pas nécessairement optimales pour le problème entier.

3.2 Algorithme Branch-and-Bound

- **Introduction** :
  - Méthode pour résoudre les problèmes de PLNE en explorant systématiquement les sous-problèmes.
  - Combine des techniques de séparation (branching) et d'évaluation (bounding).
- **Branchement (Branching)** :
  - Si la solution optimale de la relaxation est fractionnaire, on crée deux sous-problèmes en imposant des contraintes supplémentaires sur une variable fractionnaire x_j :

$$P_1 : x_j \leqslant \lfloor x_j^* \rfloor$$
$$P_2 : x_j \geqslant \lceil x_j^* \rceil$$

- **Évaluation des Sous-Problèmes (Bounding)** :
  - Résolution de la relaxation continue de chaque sous-problème pour obtenir une borne inférieure.
  - Mise à jour de la borne supérieure $\overline{z}$ lorsque des solutions entières meilleures sont trouvées.
  - **Relation des Bornes** :

$$\overline{z} \geqslant z^* \geqslant z, \quad \forall \text{ sous-problèmes}$$

- **Critères d'Arrêt** :
  - Un sous-problème peut être éliminé si :
    - Son domaine faisable est vide.

$$\overline{z}_{P_i} = \{\}$$

    - La solution optimale est entière et son coût est supérieur ou égal à $\overline{z}$.

$$\forall \ x_{ij} \ : x_{ij} \in N, \ \overline{z}_{P_i} \geqslant \overline{z}_{\text{courrant}}$$

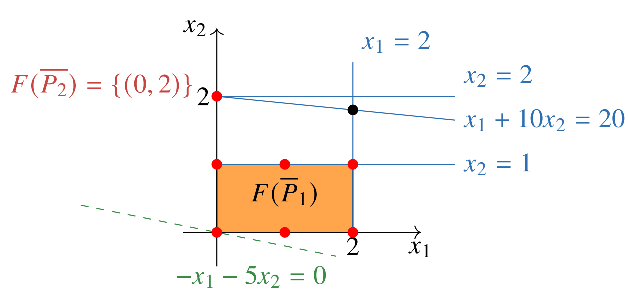
    - La borne inférieure du sous-problème est supérieure ou égale à $\overline{z}$.

$$\overline{z}_{P_i} \geqslant \overline{z}_{\text{courrant}}$$

- **Algorithme (Pseudo-Code)** :
  1. **Initialisation** : Résoudre la relaxation continue du problème initial pour obtenir une borne inférieure z.
  2. **Mise à Jour** :
     - Si la solution est entière, elle devient la borne supérieure $\overline{z}$.
     - Sinon, on branche sur une variable fractionnaire pour créer des sous-problèmes.
  3. **Exploration** :
     - Résoudre les relaxations continues des sous-problèmes.
     - Appliquer les critères d'arrêt pour éliminer des branches.
     - Mettre à jour $\overline{z}$ si une meilleure solution entière est trouvée.
  4. **Itération** : Répéter l'exploration jusqu'à ce que toutes les branches soient explorées ou éliminées.
- **Exemple Illustratif** :
  - Considérons le problème :

$$\begin{array}{rl} \text{Minimiser} & z = -x_1 - 5x_2 \\ \text{sous contraintes} & x_1 + 10x_2 \leqslant 20 \\ & x_1 \leqslant 2 \\ & x_1, x_2 \geqslant 0 \text{ et entiers} \end{array}$$

  - **Étape 1** : Résoudre la relaxation continue pour obtenir $x_1^* = 2, x_2^* = 1.8, z^* = -11$.
  - **Branchement sur** x_2 :
    - **Sous-problème** $P_1 : x_2 \leqslant 1$.
    - **Sous-problème** $P_2 : x_2 \geqslant 2$.
  - **Résolution de** P_1 :
    - Solution entière : $x_1 = 2, x_2 = 1, z = -7$.
    - Mise à jour de $\overline{z} = -7$.
  - **Résolution de** P_2 :
    - Solution entière : $x_1 = 0, x_2 = 2, z = -10$.
    - Mise à jour de $\overline{z} = -10$ car meilleure que la précédente.
  - **Conclusion** : La solution optimale est $x_1 = 0$, $x_2 = 2$, avec $z = -10$.

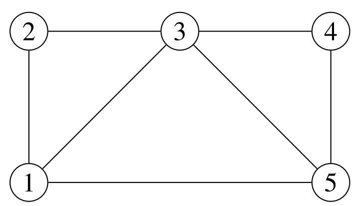


4

Optimisation de Graphes et Réseaux 1

4.1 Notions de Base

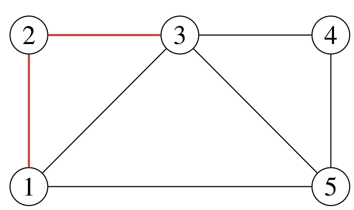
- **Graphe Non Orienté** :
  - ▷ Un graphe $G = (V, E)$ où V est l'ensemble des sommets et E l'ensemble des arêtes.
  - ▷ Une arête (i, j) relie les sommets i et j.
  - ▷ Deux sommets sont **adjacents** s'ils sont reliés par une arête.
  - ▷ Une arête est **incidente** à ses sommets.



- Ici, $V = \{1, 2, 3, 4, 5\}$

- $E = \{(1, 2),\ (1, 3),\ (1, 5), (2, 3), (3, 4), (4, 5)\}$

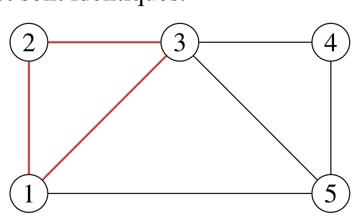
- **Chaînes et Cycles** :
  - ▷ Une **chaîne** est une suite d'arêtes $e_1, e_2, \ldots, e_p$ connectant une séquence de sommets où il existe $p + 1$ **sommets** $v_1, v_2, \ldots, v_{p+1}$



- Ici, $(1, 2), (2, 3)$ est une **chaîne**

- $p = 2,\ e_1 = (1, 2),\ e_2 = (2, 3),\ v_1 = 1, v_2 = 2, v_3 = 3$

  - ▷ Un **cycle** est une chaîne où le premier et le dernier sommet sont identiques.

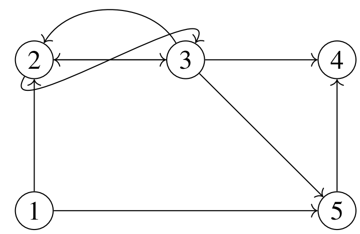


- Ici, $(1, 2), (2, 3), (1, 3)$ est une **cycle**

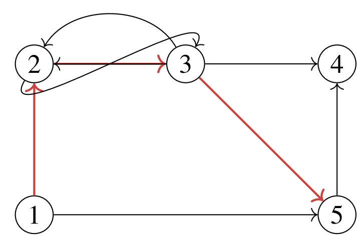
- **Graphe Connexe** :
  - ▷ Un graphe est connexe s'il existe une chaîne entre chaque paire de sommets.

4.1.1 Graphe Orienté et Concepts Associés

- **Graphe Orienté** :
  - ▷ Un graphe $G = (V, A)$ où A est l'ensemble des arcs (arêtes orientées).
  - ▷ Un arc (i, j) va du sommet i au sommet j.



- **Chemins et Circuits** :
  - ▷ Un **chemin** est une suite d'arcs orientés dans la même direction.
  - ▷ Un **circuit** est un chemin fermé (départ et arrivée au même sommet).



4.1.2 Concepts de Flot et Contraintes sur les Arcs

- **Flot sur les Arcs** :
  - ▷ Chaque arc (i, j) a :
    - ▷ Une **borne inférieure** l_{ij} (minimum de flot).
    - ▷ Une **capacité** u_{ij} (maximum de flot).
    - ▷ Un **coût** c_{ij} par unité de flot.
- **Objectif** :
  - ▷ Maximiser le flot total du **source** s au **puits** t.
  - ▷ Minimiser le coût total associé au flot.

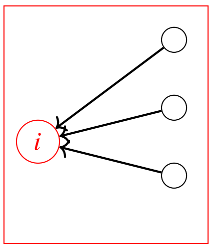
4.2 Problèmes de Flots à Coût Minimal

4.2.1 Notations de Base

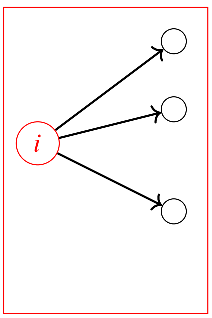
- ▷ $G = (V, A)$: Graphe orienté avec sommets V et arcs A.
- ▷ c_{ij} : Coût par unité de flot sur l'arc (i, j).

- ▷ x_{ij} : Flot sur l'arc (i, j).
- ▷ u_{ij} : Capacité maximale de l'arc (i, j).
- ▷ s : Source du réseau.
- ▷ t : Puits du réseau.
- ▷ B_i : ensemble des arcs entrants dans i : (j, i)
- ▷ P_i : ensemble des arcs sortant de i : (i, j)

4.2.2 Diagramme pour B_i et P_i



B_i



P_i

4.2.3 Formulation Mathématique

$$\text{Minimiser} \qquad z = \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{sous contraintes} \quad \sum_{j\in P_i} x_{ij} - \sum_{j\in B_i} x_{ji} = \begin{cases} d & \text{si } i = s \\ -d & \text{si } i = t \\ 0 & \text{sinon} \end{cases}$$

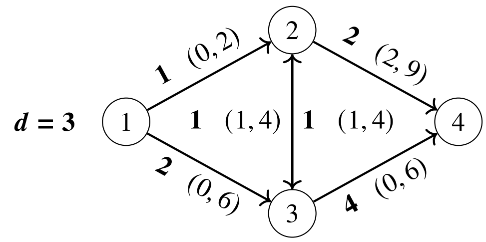
$$l_{ij} \leqslant x_{ij} \leqslant u_{ij}, \quad \forall (i, j) \in A$$

- ▷ P_i : Ensemble des sommets atteints depuis i (arcs sortants).
- ▷ B_i : Ensemble des sommets rejoignant i (arcs entrants).
- ▷ d : Quantité de flot à acheminer de s vers t.

4.2.4 Interprétation

- ► **Conservation du Flot** : Le flot entrant moins le flot sortant est égal à la demande ou l'offre en chaque sommet.
- ► **Respect des Capacités** : Le flot sur chaque arc doit respecter ses bornes.
- ► **Objectif** : Minimiser le coût total du transport du flot.

4.3 Exemple de Problème à Coût Minimal



- ► **Réseau** :
  - ▷ Sommets : 1 (source), 2, 3 (intermédiaire), 4 (puits).
  - ▷ Arcs avec coûts c_{ij}, capacités u_{ij} et bornes inférieures l_{ij}.
- ► **Objectif** :

  $$\text{Minimiser} \quad z = x_{12} + 2x_{13} + 2x_{23} + 2x_{24} + x_{32} + 4x_{34}$$

- ► **Contraintes de Conservation du Flot** :

$$\begin{cases} x_{12} + x_{13} = 3 & \text{(sommet 1)} \\ -x_{12} + x_{23} + x_{24} - x_{32} = 0 & \text{(sommet 2)} \\ -x_{13} - x_{23} + x_{32} + x_{34} = 0 & \text{(sommet 3)} \\ -x_{24} - x_{34} = -3 & \text{(sommet 4)} \end{cases}$$

- ► **Contraintes de Capacité et Bornes** :

$$0 \leqslant x_{12} \leqslant 2, \quad 0 \leqslant x_{13} \leqslant 4, \quad 0 \leqslant x_{23} \leqslant 6,$$
$$2 \leqslant x_{24} \leqslant 9, \quad 1 \leqslant x_{32} \leqslant 4, \quad 0 \leqslant x_{34} \leqslant 6$$

4.4 Problème de Flot avec Plusieurs Sources et Puits

4.4.1 Formulation Générale

- ▷ **Objectif** :

$$\text{Minimiser} \quad z = \sum_{(i,j)\in A} c_{ij}x_{ij}$$

- ▷ **Contraintes de Conservation du Flot** :

$$\sum_{j\in P_i} x_{ij} - \sum_{j\in B_i} x_{ji} = \begin{cases} d_i & \text{si } i \in S \text{ (sources)} \\ -d_j & \text{si } i \in T \text{ (puits)} \\ 0 & \text{sinon} \end{cases}$$

- ▷ **Contraintes de Capacité** :

$$0 \leqslant x_{ij} \leqslant u_{ij}, \quad \forall (i, j) \in A$$

4.4.2 Interprétation

- Chaque source s_i fournit un flot d_i.
- Chaque puits t_j consomme un flot d_j.
- Le flot total injecté par les sources est égal au flot total absorbé par les puits :

$$\sum_{i \in S} d_i = \sum_{j \in T} d_j$$

4.5 Problème de Flot à Coût Maximal avec Arc Fictif

4.5.1 Approche

- Introduction d'un **arc fictif** (t, s) avec :
  - ▷ Capacité infinie.
  - ▷ Coût $c_{ts} = -1$.
- **Objectif** :

$$\text{Minimiser} \quad z = \sum_{(i,j) \in A} c_{ij} x_{ij} - x_{ts}$$

- Ceci équivaut à **maximiser** x_{ts}, le flot total du réseau.

4.5.2 Formulation Mathématique

$$\begin{aligned}
\text{Minimiser} \quad & z = \sum_{(i,j) \in A} c_{ij} x_{ij} - x_{ts} \\
\text{sous contraintes} \quad & \sum_{j \in P_i} x_{ij} - \sum_{j \in B_i} x_{ji} = 0, \quad \forall i \in V \\
& x_{ij} \geqslant l_{ij}, \quad x_{ij} \leqslant u_{ij}, \quad \forall (i,j) \in A^+ \\
& x_{ts} \geqslant 0
\end{aligned}$$

4.5.3 Interprétation

- L'arc fictif (t, s) permet de transformer le problème en un problème de minimisation avec une fonction objectif linéaire.
- Maximiser le flot sur x_{ts} revient à maximiser le flot total dans le réseau.

4.6 Algorithme de Dijkstra

4.6.1 Introduction

L'algorithme de Dijkstra est une méthode efficace pour trouver les chemins les plus courts depuis un sommet source s vers tous les autres sommets dans un graphe connexe non orienté et pondéré $G = (V, E)$. Les poids des arêtes représentent les distances ou les coûts, et doivent être non négatifs.

4.6.2 Notations et Ensembles

λ_{ij} : Longueur (ou poids) de l'arête entre les sommets i et j, avec $\lambda_{ij} \geqslant 0$.

N_i : Ensemble des voisins (sommets adjacents) du sommet i :

$$N_i = \{j \in V \mid (i, j) \in E\}$$

EM : Ensemble des sommets marqués, c'est-à-dire les sommets pour lesquels le chemin le plus court depuis s a été déterminé.

EM^c : Complémentaire de EM, soit l'ensemble des sommets non marqués :

$$EM^c = V \setminus EM$$

δ_j : Étiquette associée au sommet j, représentant la longueur du chemin le plus court de s à j trouvé jusqu'à présent.

4.6.3 Étapes de l'Algorithme

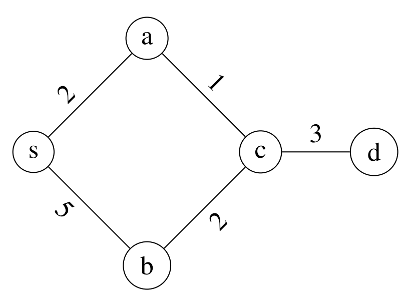
1. **Initialisation** :
   - ▷ Définir $EM = \{s\}$ et initialiser $\delta_s = 0$.
   - ▷ Pour tout $j \in V \setminus \{s\}$, initialiser $\delta_j = +\infty$.
2. **Étape Itérative** : Tant que $EM^c \neq \emptyset$, répéter les sous-étapes suivantes :

   (a) **Trouver le sommet adjacent le plus proche** :
   - ▷ Pour chaque **sommet marqué** $i \in EM$, trouver le sommet j **non marqué** qui est adjacent à i et qui est **le plus proche de** i :

   $$\min_{j \in EM^c \cap N_i} \{\lambda_{ij}\}$$

   (b) **identifier le sommet adjacent** :
   - ▷ Pour chaque $i \in EM$ pour lequel on a trouvé le sommet adjacent j le plus proche, marquer

   $$\boxed{j_i = j}$$



- **Exemple**. Soit $s \in EM$, on a :

$$j_s = \min\{\lambda_{sa}, \lambda_{sb}\} = \min\{2, 5\} = 2$$

- **Identification**. Donc on identifie :

$$j_s = a$$

(c) **Trouver la chaîne la plus courte** :

- Pour chaque sommet **marqué**, trouver la chaîne la plus courte :

$$\Delta = \min_{i \in EM} \{\delta_i + \lambda_{ij_i}\}$$

(d) **Marquer le sommet créant la chaîne la plus courte** :

- Lorsqu'on trouve le **sommet** j créant la chaîne la plus courte, on le marque :

$$\delta_j = \Delta$$

- **Exemple**. Soit $s \in EM$, $j_s = a$, $\delta_s = 0$, on a :

$$\min\{\lambda_{sa}, \lambda_{sb}\} = \min\{0 + 2\} = 2 = \Delta$$

- **Marquage**. Donc on marque :

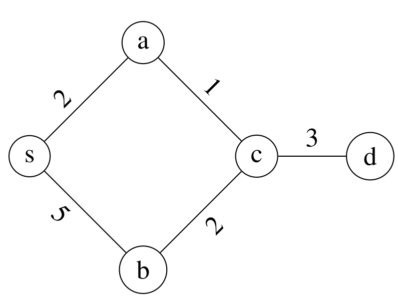
$$\delta_a = 2$$

3. **Terminaison** :
   - Lorsque tous les sommets sont marqués ($EM = V$), l'algorithme s'arrête. Les étiquettes δ_j contiennent alors les longueurs minimales des chemins depuis s vers chaque sommet j.

4.6.4 Remarques Importantes

- L'algorithme de Dijkstra fonctionne correctement uniquement si tous les poids des arêtes λ_{ij} sont non négatifs.
- Il peut être adapté aux graphes orientés en considérant les arcs orientés au lieu des arêtes.
- L'efficacité de l'algorithme peut être améliorée en utilisant des structures de données appropriées, comme des files de priorité (tas).

4.6.5 Exemple Illustratif



- **Initialisation** :
  - $EM = \{s\}$, $\delta_s = 0$, $\delta_a = \delta_b = \delta_c = \delta_d = +\infty$.
- **Itération 1** :
  - ⓢ :

$$\min(\lambda_{sa}, \lambda_{sb}) = \min(\mathbf{2}, 5) = 2 \implies j_s = a$$

  - Chaîne la plus courte :

$$\begin{aligned} \Delta &= \min(\delta_s + \lambda_{sa}) \\ &= \min(0 + 2) = 2 \\ &\implies \delta_a = 2 \end{aligned}$$

- **Itération 2** :
  - $EM = \{s, a\}$, $\delta_s = 0, \delta_a = 2$, $\delta_b = \delta_c = \delta_d = +\infty$.
  - ⓢ :

$$\min(\lambda_{sa}, \lambda_{sb}) = \min(\mathbf{5}) = 5 \implies j_s = b$$

  - ⓐ :

$$\min(\lambda_{ac}, \lambda_{as}) = \min(\mathbf{1}, 2) = 1 \implies j_a = c$$

  - Chaîne la plus courte :

$$\begin{aligned} \Delta &= \min(\delta_s + \lambda_{sb}, \delta_a + \lambda_{ac}) \\ &= \min(0 + 5, 2 + 1) = 3 \\ &\implies \delta_c = 3 \end{aligned}$$

- **Itération 3** :
  - $EM = \{s, a, c\}$, $\delta_s = 0, \delta_a = 2, \delta_c = 3$, $\delta_b = \delta_d = +\infty$.
  - ⓒ :

$$\min(\lambda_{c_b}, \lambda_{cd}) = \min(\mathbf{2}, 3) = 2 \implies j_c = b$$

  - Chaîne la plus courte :

$$\begin{aligned} \Delta &= \min(\delta_c + \lambda_{cb}) \\ &= \min(3 + 2) = 5 \\ &\implies \delta_b = 5 \end{aligned}$$

- **Itération 4** :
  - $EM = \{s, a, c, d\}$, $\delta_s = 0, \delta_a = 2, \delta_c = 3, \delta_b = 5$, $\delta_d = +\infty$.
  - ⓒ :

$$\min(\lambda_{cd}) = \min(\mathbf{3}) = 3 \implies j_c = d$$

  - Chaîne la plus courte :

$$\begin{aligned} \Delta &= \min(\delta_c + \lambda_{cd}) \\ &= \min(3 + 3) = 6 \\ &\implies \delta_d = 6 \end{aligned}$$

Le chemin le plus court de s à d est $\boldsymbol{\delta_d = 6}$, en visitant, dans l'ordre, les noeud s, a, c, d.

5

Optimisation de Graphes et Réseaux 2

5.1 Problème de transport et d'affectation

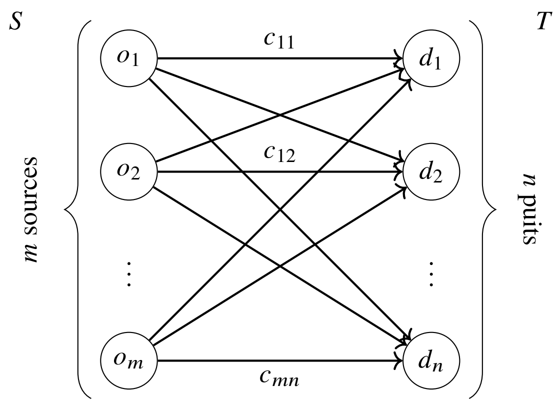
- **Problème d'affectation**
  - Cas particulier du problème de *flot à coût minimal* dans lequel on a plusieurs **sources** et **puits**.
  - Un certain nombre de **ressources** devoient être acheminé des sources **vers** les puits.
- **Sources** $\boldsymbol{i}$
  - Chaque source $i \in S$ **offre** une quantité de ressources $\boldsymbol{o_i}$.
- **Puits** $\boldsymbol{j}$
  - Chaque puits $j \in T$ **demande** une quantité de ressources $\boldsymbol{d_j}$.
- **Ensemble des sommets** $\boldsymbol{V}$
  - Il n'y aucun sommet interminédiaire ; il n'y a que les sommets sources et puits :

$$V = S \cup T$$

- **Quantité de sommets** dans $\boldsymbol{V}$
  - Il y a généralement **plus de demande** que d'offre :

$$|S| = m \leqslant n = |T|$$

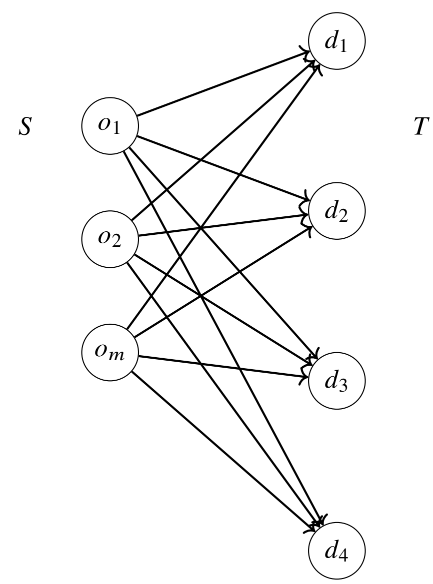
- **Ensemble des arcs** dans $\boldsymbol{A}$
  - Chaque arc $(i, j) \in A$ mène d'une source $i \in S$ à un puits $j \in T$:
- **Poids des arcs** $\boldsymbol{c_{ij}}$
  - À chaque arc $(i, j) \in A$ est associé un **coût de transport** ou d'**association** c_{ij}
- **Bornes** des arcs
  - À chaque arc $(i, j) \in A$ est associé une **bornes** :
  - **Borne inférieure** $l_{ij} = 0$; impose une contrainte de non négativité.
  - **Borne supérieure** $u_{ij} = \infty$



5.1.1 Modèle Mathématique

Pour chaque noeud $i \in S$, il y a n arcs **sortants** potentiels et l'offre pour un noeud i est donné par :

$$o_i = \sum_{j=1}^{n} c_{ij} x_{ij}$$



Exemple

$$o_1 = \sum_{j=1}^{n} c_{1j} x_{1j}$$

Pour chaque noeud $j \in T$, il y a n arcs **entrants** potentiels et la demande pour un noeud i est donné par :

$$d_j = \sum_{i=1}^{m} c_{ij} x_{ij}$$

Exemple

$$d_3 = \sum_{i=1}^{m} c_{i3} x_{i3}$$

Pour chaque arc $(i, j) \in A$, on ne peut pas affecter une quantité négative de ressources x_{ij} :

$$\forall x_{ij} : (i, j) \in A,\ x_{ij} \geqslant 0$$

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

s.a

$$\sum_{j=1}^{n} x_{ij} = o_i, \quad i = 1, \cdots, m$$

$$\sum_{i=1}^{m} x_{ij} = d_i, \quad j = 1, \cdots, n$$

$$x_{ij} \geqslant 0 \quad (i, j) \in A$$

5.1.2 Algorithme Hongrois

5.1.3 Introduction

L'algorithme Hongrois est une **procédure itérative** qui permet de résoudre des problèmes d'affectation. Il permet d'engendrer un **ensemble de solutions optimales** qui minimise le coût global d'affectation.

- ▷ **Initialisation** :puits Soit un problème d'affectation impliquant $m \in S$ **sources** et $n \in T$ **puits**, produire une matrice $B_{n \times n}$.
    - ► Placer les sources en **colonnes**
    - ► Placer les puits en **rangées**
    - ► S'il y a plus de **noeud de demande** que d'offre, créer une offre fictive en introduisant une rangée de poids nul.

Exemple

	j_1	j_2	j_3	j_4
i_1	6	M	3	5
i_2	8	4	9	6
i_3	7	5	2	8
i_f	0	0	0	0

- ▷ **Étape 1.**
    - ► Dans chacune des **rangées**, identifier **le coût minimal**
    - ► **Soustraire** celui-ci de chacun des coûts dans la rangée correspondante.
- ▷ **Étape 2.**
    - ► Dans chacune des **colonnes**, identifier **le coût minimal**
    - ► **Soustraire** celui-ci de chacun des coûts dans la rangée

  correspondante.
- ▷ **Étapes 3.**
    - ► Si une **affectation de coût 0** est possible :
        - ▷ Procéder à cette affectation optimale
        - ▷ Évaluer son véritable coût avec la matrice des coût originale.

Note:-

Une **affectation de coût 0** est possible si le nombre minimum de lignes horizontales **et ou** verticales permettant de couvrir **tous les 0 dans la matrice** des coûts correspond à la dimension de cette matrice, c'est-à-dire n.

Exemple

	j_1	j_2	j_3
i_1	0	M	0
i_2	4	0	3
i_3	0	0	0

Il faut au minimum **3 lignes** horizontales et ou verticales pour couvrir tous les 0. La matrice est de dimension $B_{3 \times 3}$ Une affectation de coût 0 est donc possible.

- ▷ **Étape 4.**
    - ► Ajouter des 0 dans la matrice de coût :
        - ▷ Identifier le plus petit coût non couvert par une ligne.
        - ▷ Soustraire ce coût de chacun des coûts non couverts par une ligne.
        - ▷ Ajouter ce coût à chacun des coûts couverts par une ligne horizontale et verticale.
- ▷ **Étape 5.**
    - ► Retourner à l'Étape 3.

À la fin de chaque itération, tant que le nombre lignes horizontales et ou verticales nécessaire pour couvrir tous les 0 est inférieur à la dimension de la matrice, on retourne à l'étape 1 (après avoir visité l'étape 3 pour vérifier la condition d'arrêt).

6

Médecins à l'hôpital

Exercice 1 Affectation à l'hôpital

Un hôpital a **quatre patients** nécessitant des traitements spécifiques pour des maladies différentes. Il y a **trois médecins spécialisés** disponibles pour traiter ces patients. Pour chaque paire de patient et de médecin, l'hôpital attribue un **score** qui correspond à la durée estimée du traitement (en heures). Ces scores se trouvent dans le tableau suivant :

	Médecin 1	**Médecin 2**	**Médecin 3**
Patient 1	6	8	7
Patient 2	–	4	5
Patient 3	3	9	2
Patient 4	5	6	8

Résoudre ce problème à l'aide de l'algorithme Hongrois. N'oubliez pas d'indiquer clairement la solution optimale et la valeur optimale.

6.0.1 Organisation de la matrice

Pour appliquer l'algorithme Hongrois, nous devons obtenir une matrice carrée $B_{n\times n}$. Pour indiquer l'incompatibilité du médecin $i = 1$ avec la patient $j = 2$, nous introduisons un coût arbitrairement grand $M \to \infty$. Pour rendre la matrice carrée, nous introduisons une 4$^{\text{e}}$ rangée représentant un médecin fictif i_f. Chaque poids fictif x_{fj} a une valeur de 0.

	j_1	j_2	j_3	j_4
i_1	6	M	3	5
i_2	8	4	9	6
i_3	7	5	2	8
i_f	0	0	0	0

Étape 1. Dans chacune des rangées, identifier le coût minimal et soustraire celui-ci de chacun des coûts dans la rangée correspondante.

	j_1	j_2	j_3	j_4	
i_1	6	M	3	5	$\leftarrow$ -3
i_2	8	4	9	6	$\leftarrow$ -4
i_3	7	5	2	8	$\leftarrow$ -2
i_f	0	0	0	0	$\leftarrow$ -0

	j_1	j_2	j_3	j_4
i_1	3	M	0	2
i_2	4	0	5	2
i_3	5	3	0	6
i_f	0	0	0	0

Étape 2. Dans chacune des colonnes, identifier le coût minimal et soustraire celui-ci de chacun des coûts dans la colonne correspondante.

Note:-
Le coût minimal pour chaque colonne est de 0, le tableau reste tel quel

Étape 3. Si une affectation de coût 0 est possible, procéder à cette affectation optimale et évaluer son véritable coût avec la matrice des coûts originale.

	j_1	j_2	j_3	j_4
i_1	3	M	0	2
i_2	4	0	5	2
i_3	5	3	0	6
i_f	0	0	0	0

Il faut un minimum de $3 < 4$ lignes horizontales et ou verticales pour couvrir les 0.

Étape 4. a) Identifier le plus petit coût non couvert par une ligne

	j_1	j_2	j_3	j_4
i_1	3	M	0	2
i_2	4	0	5	2
i_3	5	3	0	6
i_f	0	0	0	0

Étape 4. b) Soustraire ce coût de chacun des coûts non couverts par une ligne.

	j_1	j_2	j_3	j_4
i_1	0	M	0	2
i_2	4	0	5	2
i_3	0	0	0	6
i_f	0	0	0	0

Étape 4. c) Ajouter ce coût à chacun des coûts couverts par une ligne **horizontale et verticale** .

	j_1	j_2	j_3	j_4
i_1	0	M	0	2
i_2	4	0	3	2
i_3	0	0	0	6
i_f	0	0	0	3

Étape 3. Si une affectation de coût 0 est possible, procéder à cette affectation optimale et évaluer son véritable coût avec la matrice des coûts originale.

	j_1	j_2	j_3	j_4
i_1	0	M	0	2
i_2	4	0	3	2
i_3	0	0	0	6
i_f	0	0	0	3

Il faut un minimum de $3 < 4$ lignes horizontales et ou verticales pour couvrir les 0.

Étape 4. a) Identifier le plus petit coût non couvert par une ligne

	j_1	j_2	j_3	j_4
i_1	0	M	0	2
i_2	4	0	3	2
i_3	0	0	0	6
i_f	0	0	0	3

Étape 4. b) Soustraire ce coût de chacun des coûts non couverts par une ligne.

	j_1	j_2	j_3	j_4
i_1	0	M	0	0
i_2	4	0	3	0
i_3	0	0	0	4
i_f	0	0	0	1

Étape 4. c) Ajouter ce coût à chacun des coûts couverts par une ligne **horizontale et verticale** .

	j_1	j_2	j_3	j_4
i_1	0	M	0	0
i_2	4	0	3	0
i_3	0	0	0	4
i_f	0	0	0	1

Étape 3. Si une affectation de coût 0 est possible, procéder à cette affectation optimale et évaluer son véritable coût avec la matrice des coûts originale.

	j_1	j_2	j_3	j_4
i_1	0	M	0	0
i_2	4	0	3	0
i_3	0	0	0	4
i_f	0	0	0	1

Nous avons l'affectation suivante :

$$\begin{aligned} i_1 &\longrightarrow j_4 \\ i_2 &\longrightarrow j_2 \\ i_3 &\longrightarrow j_3 \\ i_f &\longrightarrow j_1 \end{aligned}$$

$$z = 5 + 4 + 2 = 11$$

Le médecin 1 traitera le patient 4, le médecin 2 traitera le patient 2, et le médecin 3 traitera le patient 3, pour un temps d'attente total de 11 h.

7

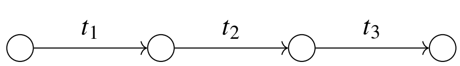
Optimisation de Graphes et Réseau 3

7.1 Réseau PERT/CPM

- **Définition** :
  - ▷ **PERT** ou *Program Evaluation Review* et **CPM**, *Critical Path Method*, sont des approches qui permettent de résoudre des problèmes dans lesquels la précédances des tâches est importante.
- **Exemple** :
  - ▷ Un réseau PERT/CPM pourrait être schématisé pour représenter un système avec les contraintes suivantes :
  - Il faut monter la charpente avant d'installer les panneaux de gypse
  - Par contre, il est possible de peinturer l'extérieur de la maison touten peinturant les murs intérieurs
  - ▷ Il faut donc **identifier les tâches critiques**
  - ▷ Les tâches critiques sont celles qui ne peuvent pas être retardées sans quoi elles **retarderaient l'ensemble du projet**.

7.1.1 Réseau PERT/CPM simple

- **Arcs** : tâches t_n
- **Sommets** : étape d'avancement du projet



Le réseau illustre les contraintes selon lesquelles la tâche 1 doit être **complétée avant** la tâche 2, et la tâche 2 doit être complétée avant la tâche 3.

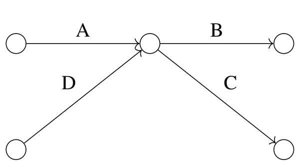
7.1.2 Principe de représentation

Selon les instructions logiques de précédance, il faut s'assurer que le réseau représente correctement la **relation stricte** entre chaque tâche.

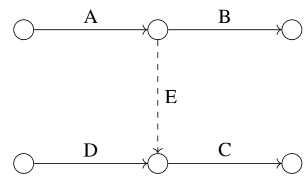
Soit la relation suivante :

$$\boxed{A < B, A < C, D < C}$$

Le réseau suivant est **incorrect**, puisqu'il force une relation stricte de précédance $D < B$, alors qu'elle n'a pas été mentionnée dans l'instruction logique :



Pour corriger cette erreur, on peut introduire un **arc fictif** E de **poids nul** :

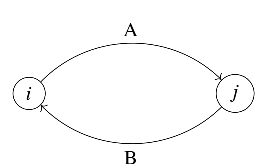


- ▷ $A < B$
  - La tâche A doit être complété avant la tâche B.
- ▷ $D < C$
  - La tâche D doit être complété avant la tâche C.
- ▷ $A < D \mid D < A$
  - Aucune précédance de commencement n'est spécifiée, les deux tâches A et D peuvent être complétée **simultanément**.
- ▷ $E < C$ && $D < C$
  - Les tâches E et D doivent se terminer pour réaliser C. Puisque A précède E, cela implique que A précède C
- ▷ $E < D \mid D < E$
  - Si la tâche A se termine avant la tâche D, la tâche E qui se termine instantanément pourrait précéder la tâche D. Autrement, D précède E : $D < E$.
- ▷ $C \leqslant B$
  - Pour réaliser C, il faut que A, D et (instantanément) E se terminent. Au mieux, C peut se produire en même temps que B, pourvu que A et D se réalisent en même temps.
- ▷ $D \leqslant B$
  - Si A se réalise rapidement, B peut précéder la complétion de D. Si non, D, précèdera la complétion de B.

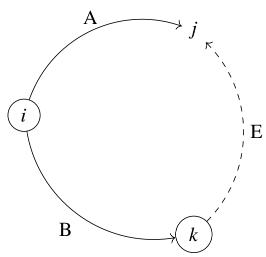
Bien qu'il y ait plusieurs contraintes implicites, il n'y a pas de contrainte stricte entre D et B. Notre réseau PERT/CPM répésente donc fidèlement les relations $A < B, A < C, D < C$.

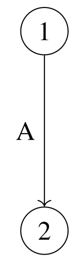
Note:-

Dans un réseau PERT/CPM, il faut être en mesure d'identifier une tâche correspondant à un arc sans **ambiguité** quant au sommet d'origine et au somme des destination. Ainsi, les **arcs parallèles ne sont pas permis**.



On peut désambiguié le réseau en introduisant un arc fictif E de poids null :





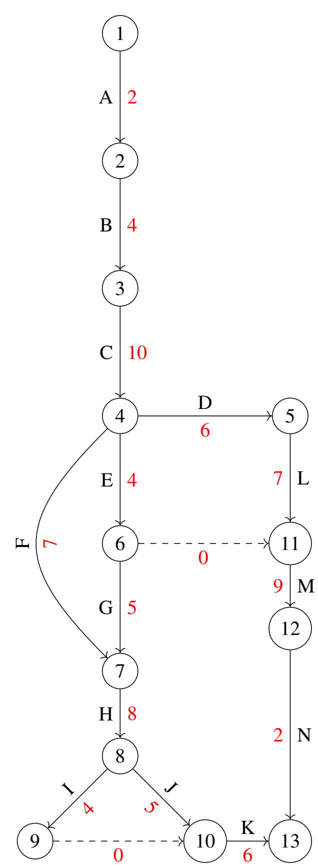
7.2 Résolution d'un réseau CPM

7.2.1 Graphe du réseau

À partir d'information tabuliées, il faut représenter graphiquement le réseau en respectant les contraintes de précédance, en introduisant des **arcs fictifs** lorsque nécessaire.

Tâche	Durée	Préd.
A	2	-
B	4	A
C	10	B
D	6	C
E	4	C
F	7	C
G	5	E
H	8	F, G
I	4	H
J	5	H
K	6	I, J
L	7	D
M	9	E, L
N	2	M

TABLE 7.1 – Tableau des tâches avec durée et prédécesseurs



Principe : Le temps de complétion le plus tôt d'une tâche i dépend du temps de complétion le plus tôt de chaque tâche j précédant i et du de la durée de la tâche associée à l'arc t_{ji}

Soit ET_{debut} ou *earliest time*, le temps de complétion le plus tôt de la tâche précédant la tâche A, nous avons :

$$ET_{debut} = 0$$

Cela est dû au fait que cette tâche est associée au sommet 1.

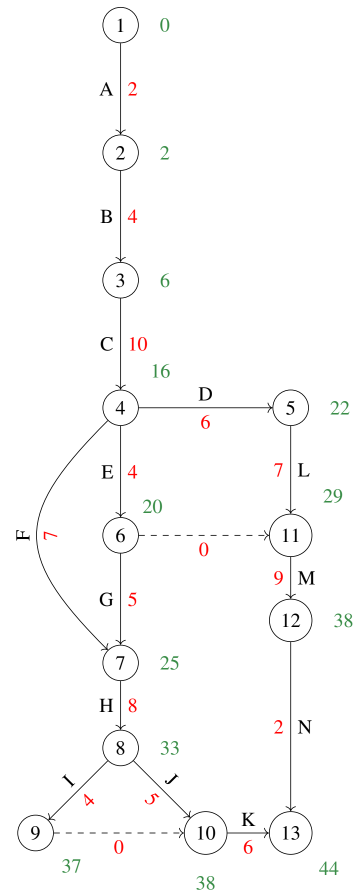
Étape i	$j \in B_i$	$ET_j + t_{ji}$	ET_i
1	-	-	0
2	1	$0+2$	2
3	2	$2+4$	6
4	3	$6+10$	16
5	4	$16+6$	22
6	4	$16+6$	20
7	$\begin{cases}4\\6\end{cases}$	$\begin{cases}16+7\\20+5\end{cases}$	25
8	7	$25+8$	33
9	8	$33+4$	37
10	$\begin{cases}8\\9\end{cases}$	$\begin{cases}33+4\\37+0\end{cases}$	38
11	$\begin{cases}5\\6\end{cases}$	$\begin{cases}22+7\\22+0\end{cases}$	29
12	11	$29+9$	38
13	$\begin{cases}10\\12\end{cases}$	$\begin{cases}38+6\\38+2\end{cases}$	44

TABLE 7.2 – Table des temps de complétions les plus tôt

Ainsi, de façon générale on a :

$$\boxed{ET_i = \max_{j \in B_i}\left\{ET_j + t_{ji}\right\}}$$

Pour **remplir le tableau des temps de complétions le plus tôt**, il faut débuter en haut du tableau en posant $ET_{debut} = E_{T_1} = 0$. Ensuite, on trouve le E_T pour les tâches i succéssive en applicant la formule récursivement :



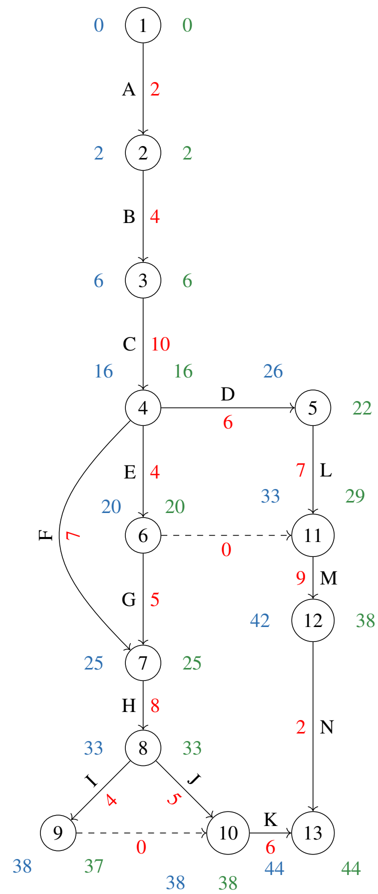
7.2.2 Représentation des temps les plus tard

Les succésseurs de i ont besoin que i se termine le plus rapidement possible, pour ne pas retarder l'ensemble du projet. Le temps le plus tard auquel i peut donc être complété, LTi sans affecter le projet dépend du temps de complétion le plus petit d'un successeur de i,

$$\boxed{LT_i = \max_{j \in P_i}\left\{LT_j - t_{ji}\right\}}$$

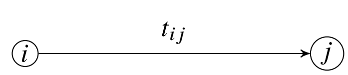
Étape i	$j \in P_i$	$LT_j + t_{ji}$	LT_i
13	-	-	44
12	13	44 - 2	42
11	12	42 - 9	33
10	13	44 - 6	38
9	10	38 - 0	38
8	10 9	38 − 5 38 − 4	33
7	8	33 - 8	25
6	11 7	33 − 0 25 − 5	20
5	11	37 - 7	26
4	7 6 5	25 − 7 20 − 4 26 − 6	16
3	4	16 - 10	6
2	3	6 - 4	2
1	2	2 - 2	0

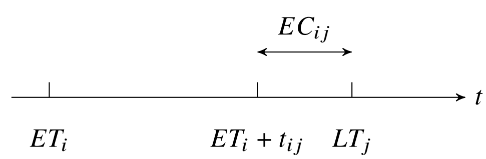
TABLE 7.3 – Projet de construction d'une maison

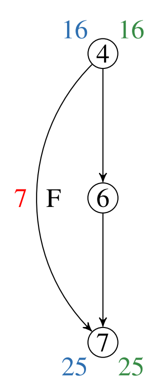


L'écart pour la tâche associée à l'arrête ij est le délai maximal permettant de compléter la tâchge sans retarder l'achèvement du projet à son temps le plus tôt.

$$\boxed{EC_{ij} = LT_j - (ET_i + t_{ij})}$$

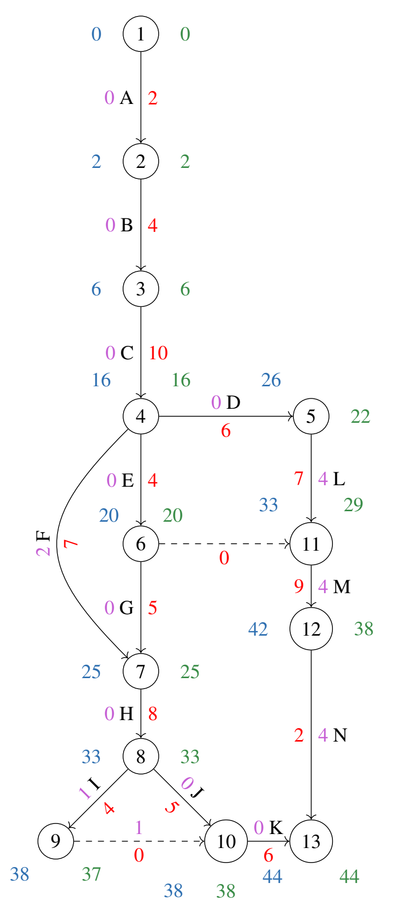


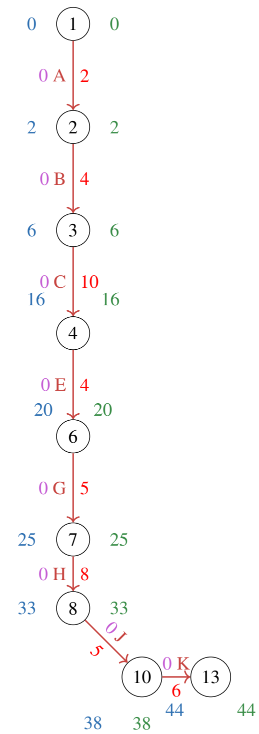




Tâche (i, j)	**Écart** $LT_j - (ET_i + t_{ij})$
(1, 2)	$2 - (0 + 2) = 0$
(2, 3)	$6 - (2 + 4) = 0$
(3, 4)	$16 - (6 + 10) = 0$
(4, 5)	$26 - (16 + 6) = 4$
(4, 6)	$20 - (16 + 4) = 0$
(4, 7)	$25 - (16 + 7) = 2$
(6, 7)	$25 - (20 + 5) = 0$
(7, 8)	$33 - (25 + 8) = 0$
(8, 9)	$38 - (33 + 4) = 1$
(8, 10)	$38 - (33 + 5) = 0$
(10, 13)	$44 - (38 + 6) = 0$
(5, 11)	$33 - (22 + 7) = 4$
(11, 12)	$42 - (29 + 9) = 4$
(12, 13)	$44 - (38 + 2) = 4$
(6, 11)	$33 - (20 + 0) = 13$
(9, 10)	$38 - (37 + 0) = 1$

TABLE 7.4 – Écart $LT_j - (ET_i + t_{ij})$ pour chaque tâche (i, j)





Les tâches d'écart nul sont critiques car tout délai dans leur réalisation **entraîne un retard** dans l'achèvement du projet au temps le plus tôt.

7.2.3 Chemin critique

Le chemin critique est un chemin du sommet début au sommet fin le long duquel toutes les tâches (arcs) ont un écart nul.

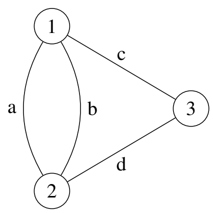
Optimisation de Graphes et Réseaux 4

8.1 Abre partiel de poids minimal

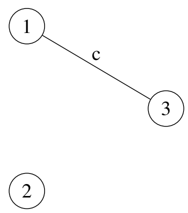
8.1.1 Théorie

▹ **Définitions et rappels** :

- Un **graphe connexe** est un graphe dans lequel il existe une chaîne reliant chaque paire de sommets distincts.

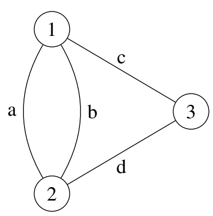


Connexe

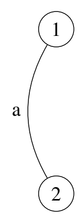


Non connexe

- Un **sous-graphe** G' d'un graphe G contient un sous-ensemble des sommets et arêtes de G.

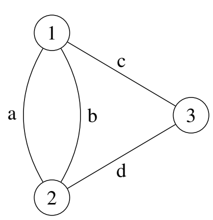


Graphe G

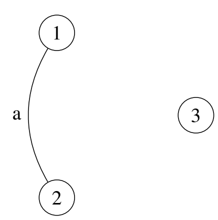


Sous-graphe G'

- Un **graphe partiel** est un sous-graphe qui contient tous les sommets de G.

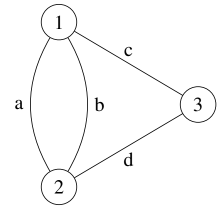


Graphe G

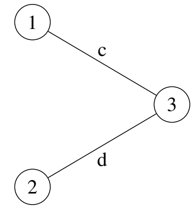


Graphe partiel G'

- Un **arbre** est un graphe connexe sans cycle. Il a exactement $n-1$ arêtes pour n sommets.
- Un **arbre partiel** ou arbre **de recouvrement** est un sous-graphe connexe et sans cycle contenant tous les sommets du graphe. Autrement dit, c'est un arbre qui se trouve à être un également un graphe partiel.



Graphe G

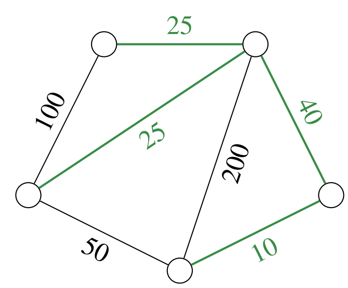


Arbre partiel G'

8.1.2 Analyse des arbre partiels

▹ **Arbre partiel de poids minimal** :

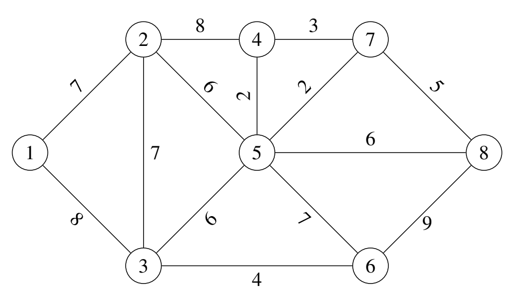
- Un **arbre partiel de poids minimal** est un arbre de recouvrement dont la somme des poids des arêtes est minimale.
- Problème concret : Minimiser le coût de connexion des habitations par des lignes électriques en identifiant un arbre partiel de poids minimal.



Poids total : 25 + 25 + 40 + 10 = 100

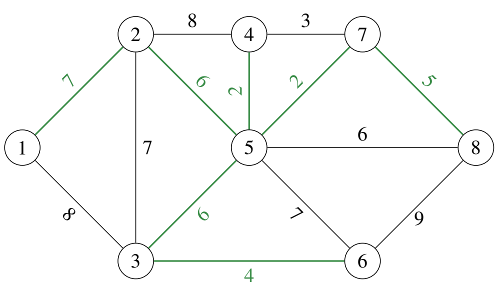
8.1.3 Principe de l'algorithme de Kruskal

- Algorithme dit vorace ou *greedy* pour trouver un arbre partiel de poids minimal.



1. Initialiser l'ensemble $\Omega = \emptyset$ et un compteur $k = 0$.
2. Choisir l'arête de plus petit poids dans $E \setminus \Omega$ qui ne forme pas de cycle avec les arêtes de Ω.
3. Ajouter cette arête à Ω et incrémenter k.
4. Répéter jusqu'à obtenir $n-1$ arêtes.

$(4, 5)$	$k = 1$
$(5, 7)$	$k = 2$
$(4, 7)$	ignoré
$(3, 6)$	$k = 3$
$(7, 8)$	$k = 4$
$(2, 5)$	$k = 5$
$(3, 5)$	$k = 6$
$(5, 8)$	ignoré
$(1, 2)$	$k = 7$
$(2, 3)$	ignoré
$(5, 6)$	ignoré
$(1, 3)$	ignoré
$(2, 4)$	ignoré
$(6, 8)$	ignoré



9

Programmation dynamique déterministe

9.1 Introduction

▷ **Contexte**

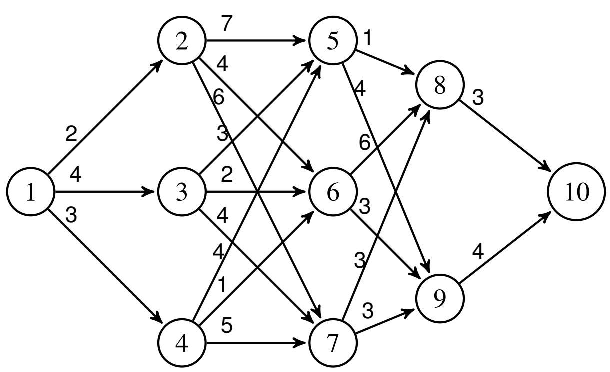
- ► S'applique aux problèmes où la solution correspond à **une suite de décision**.

▷ **Objectif**

- ► Réduire le problème original en sous-problèmes.
- ► Chaque sous-problème doit être plus simple à résoudre.

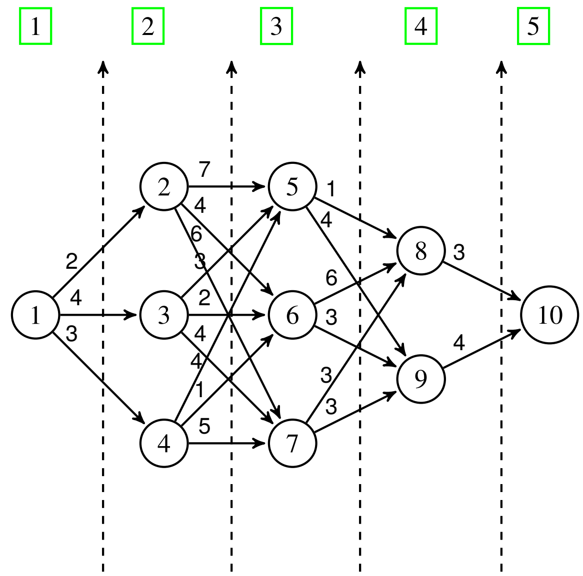
▷ **Application**

- ► Exploiter la solution des sous-problèmes les plus simples afin de produire la solution optimale des sous-problèmes complexes.
- ► Résoudre ainsi les sous-problèmes complexe pour obtenir la **solution optimale du problème original**



9.2 Exemple chemin le plus court

L'algorithme de Dijkstra permettrait de trouver le chemin le plus court. Cependant, la structure du graphe se prette bien à l'utilisation d'une approche de **programmation dynamique**



Il est possible de diviser le graphes en **étapes**. Pour se rendre de la ville 1 à la ville 10, il est nécessaire de visiter **au moins une vielle par étape**.

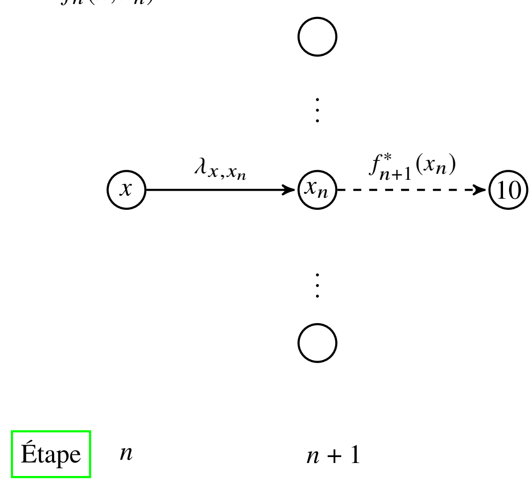
9.3 Notation

▷ **Nombre d'étapes**
- ► n

▷ **Ensemble des villes**
- ▷ $A = A_1 \cup A_2 \cup \cdots \cup A_n$

► **Distance minimale** pour se rendre de $x \in A_n$ à 10
- ► $f_n^*(x)$

▷ **Distance minimale** pour se rendre de $x \in A_n$ à 10, **en passant par $x_n \in A_{n+1}$**
- ► $f_n(x, x_n)$



À chaque étape n, il existe une quantité finie de villes $x_n \in A_{n+1}$ auxquelles on peut aller. La quantité $f_n(x, x_n)$ fait référence à la distance la plus courte possible pour se rendre à fin à partir de x, **en respectant la condition de passer par x_n**.

Ainsi, la distance minimale pour se rendre de $x \in A_n$ à fin en passant par $x_n \in A_{n+1}$ dépend aussi de la distance du déplacement λ_{x,x_n} :

$$f_n(x, x_n) = \lambda_{x,x_n} + f^*_{n+1}(x_n)$$

Note:-

Par définition, la distance minimale pour se déplacer de $x_n \in A_{n+1}$ à 10 est donné par :

$$f^*_{n+1}(x_n)$$

Puisqu'il y a plusieurs choix de villes $x_n \in A_{n+1}$, la distance minimale de x à fin en passant par une des villes $x_n \in A_{n+1}$ dépendant du déplacement optimal $f^*_{n+1}(x_n)$:

$$x^*_n = x_n : \min\{f_{n+1}(x_n)\} = f^*_{n+1}(x_n)$$

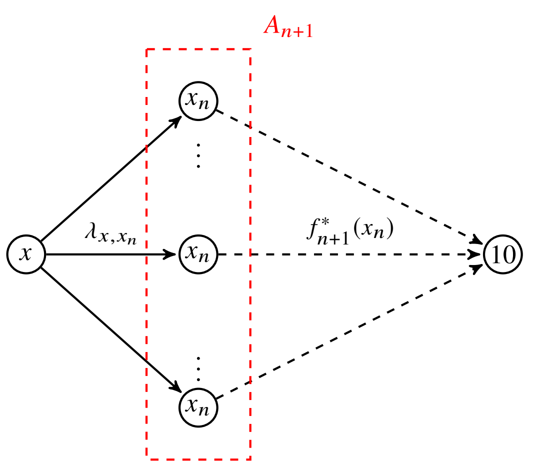
Autrement dit, le x_n optimal, soit x^*_n, est la ville $x_n \in A_{n+1}$ qui minimise le déplacement de $x_n \to fin$.

Ainsi, nous pouvons reformuler $f^*_n(x)$:

$$\min_{x_n \in A_{n+1}} \{f_n(x, x_n)\} = f_n(x, x^*_n)$$

Nous pouvons alors généraliser ce résultat. Nous observons que pour tout n représentant une ville ou une étape, la distance minimale pour aller d'une ville $x \in A_n$ à une ville de l'étape suivant $x_n \in A_{n+1}$ dépend du déplacement optimal λ_{x,x_n} et du déplacement optimal $f^*_{n+1}(x_n)$:

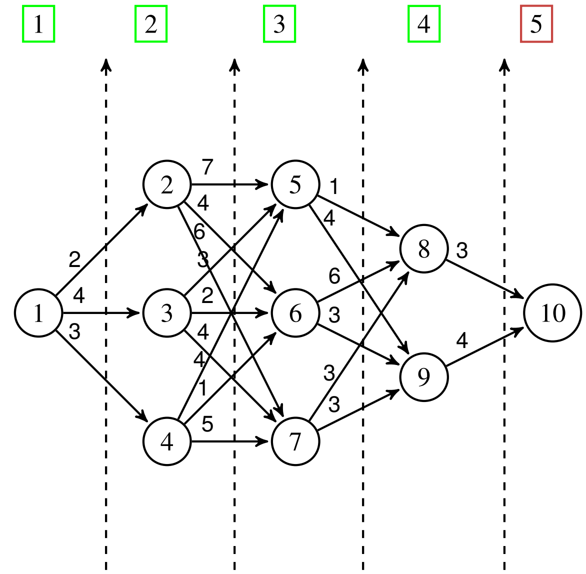
$$\boxed{f^*_n(x) = \min_{x_n \in A_{n+1}} \{\lambda_{x,x_n} + f^*_{n+1}\}}$$



9.3.1 Principe

L'équation ci-haut est une **formule de récurrence**. Pour connaître $f^*_n(x)$ où $x \in A_n$, il faut connaitre $f^*_{n+1}(x_n), x_n \in A_{n+1}$. Pour résoudre ce problème, on commence donc de la fin, puisqu'on sait que $f^*_5(x) = 0$.

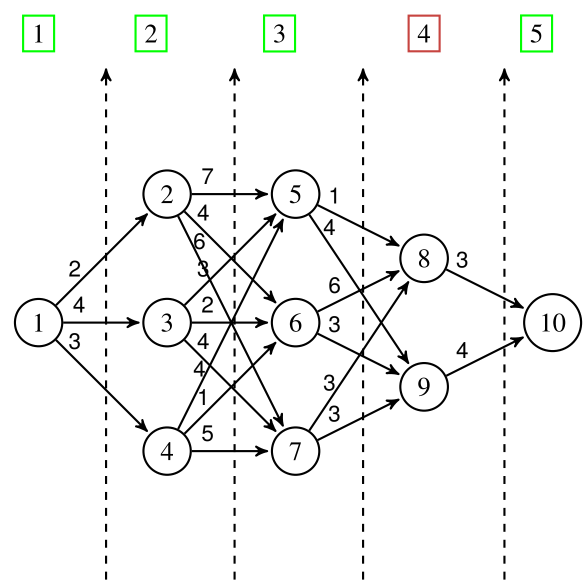
9.3.2 Étape $n = 5$



$$f^*_5(10) = 0$$

x	$f^*_5(x)$	x^*_5
10	0	-

9.3.3 Étape $n = 4$



$$\begin{aligned} f^*_4(8) &= \min_{x_n \in A_5} \{\lambda_{8,x_n} + f^*_5(x_n)\} \\ &= \lambda_{8,10} + f^*_5(10) = 3 + 0 = 3 \implies x^*_4 = 10 \\ f^*_4(9) &= \min_{x_n \in A_5} \{\lambda_{9,x_n} + f^*_5(x_n)\} \\ &= \lambda_{9,10} + f^*_5(10) = 4 + 0 = 4 \implies x^*_4 = 10 \end{aligned}$$

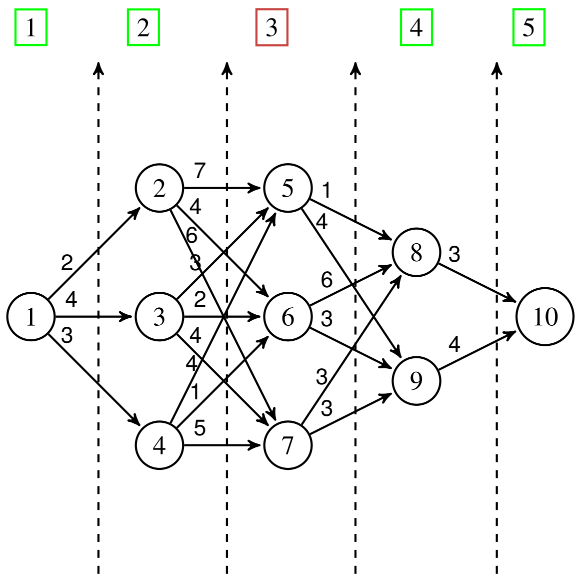
x	$f_5^*(x)$	x_5^*
10	0	-

x	$f_4^*(x)$	x_4^*
8	3	10
9	4	10

x	$f_4^*(x)$	x_4^*
8	3	10
9	4	10

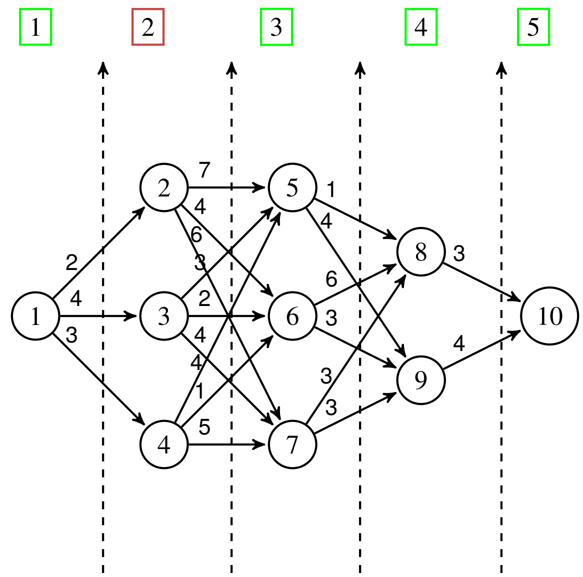
x	$f_3^*(x)$	x_3^*
5	4	8
6	7	9
7	6	8

9.3.4 Étape $n = 3$



$$\begin{aligned}
f_3^*(5) &= \min_{x_n \in A_4} \{\lambda_{5,x_n} + f_4^*(x_n)\} \\
&= \min\{\lambda_{5,8} + f_4^*(8), \lambda_{5,9} + f_4^*(9)\} \\
&= \min\{1 + 3, 4 + 4\} \\
&= 4 \implies x_3^* = 8 \\
f_3^*(6) &= \min_{x_n \in A_4} \{\lambda_{6,x_n} + f_4^*(x_n)\} \\
&= \min\{\lambda_{6,8} + f_4^*(8), \lambda_{6,9} + f_4^*(9)\} \\
&= \min\{6 + 3, 3 + 4\} \\
&= 7 \implies x_3^* = 9 \\
f_3^*(7) &= \min_{x_n \in A_4} \{\lambda_{7,x_n} + f_4^*(x_n)\} \\
&= \min\{\lambda_{7,8} + f_4^*(8), \lambda_{7,9} + f_4^*(9)\} \\
&= \min\{3 + 3, 3 + 4,\} \\
&= 6 \implies x_3^* = 8
\end{aligned}$$

9.3.5 Étape $n = 2$

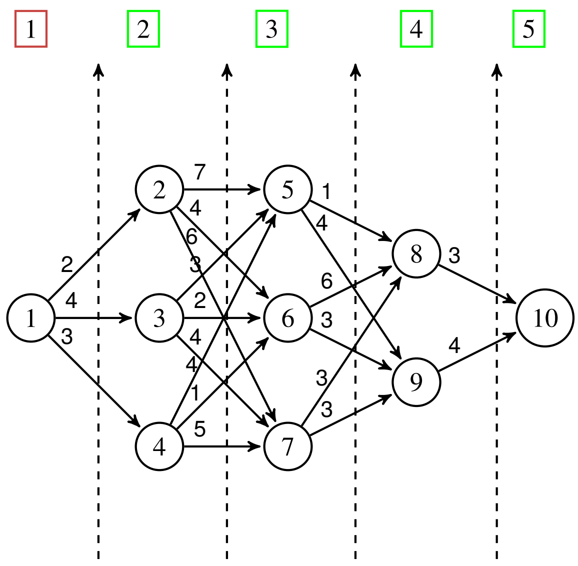


$$\begin{aligned}
f_2^*(2) &= \min_{x_n \in A_3} \{\lambda_{2,x_n} + f_3^*(x_n)\} \\
&= \min\{\lambda_{2,5} + f_3^*(5), \lambda_{2,6} + f_3^*(6), \lambda_{2,7} + f_3^*(7),\} \\
&= \min\{7 + 4, 4 + 7, 6 + 6\} \\
&= 11 \implies x_2^* = 5 \text{ou } 6 \\
f_2^*(3) &= \min_{x_n \in A_3} \{\lambda_{3,x_n} + f_3^*(x_n)\} \\
&= \min\{\lambda_{3,5} + f_3^*(5), \lambda_{3,6} + f_3^*(6), \lambda_{3,7} + f_3^*(7),\} \\
&= \min\{3 + 4, 2 + 7, 4 + 6\} \\
&= 7 \implies x_2^* = 5 \\
f_2^*(4) &= \min_{x_n \in A_3} \{\lambda_{4,x_n} + f_3^*(x_n)\} \\
&= \min\{\lambda_{4,5} + f_3^*(5), \lambda_{4,6} + f_3^*(6), \lambda_{4,7} + f_3^*(7),\} \\
&= \min\{4 + 4, 1 + 7, 5 + 6\} \\
&= 8 \implies x_2^* = 5 \text{ou } 6
\end{aligned}$$

x	$f_3^*(x)$	x_3^*
5	4	8
6	7	9
7	6	8

x	$f_2^*(x)$	x_2^*
2	11	5, 6
3	7	5
4	8	5, 6

9.3.6 Étape $n = 1$

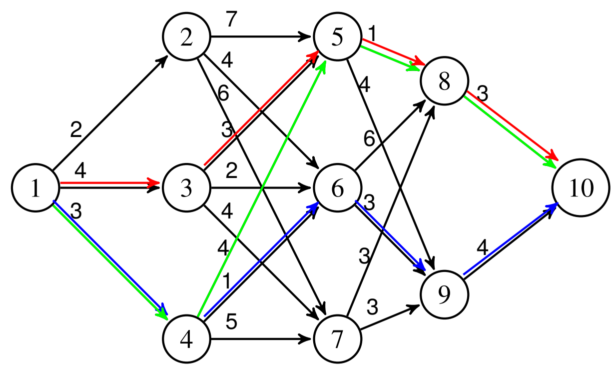


$$
\begin{aligned}
f_1^*(2) &= \min_{x_n \in A_2} \left\{ \lambda_{1,x_n} + f_2^*(x_n) \right\} \\
&= \min\left\{ \lambda_{1,2} + f_2^*(2), \lambda_{1,3} + f_2^*(3), \lambda_{1,4} + f_2^*(3), \right\} \\
&= \min\left\{ 2 + 11, 2 + 11, 3 + 8 \right\} \\
&= 11 \implies x_1^* = 3 \text{ou } 4
\end{aligned}
$$

x	$f_2^*(x)$	x_2^*
2	11	5, 6
3	7	5
4	8	5, 6

Il existe donc trois chemins différents qui engendrent un déplacement minimal. La approche de programmation dynamique permet de tous les identifier

x	$f_1^*(x)$	x_1^*
1	11	3, 4



9.4 Principe général

- **Étapes n**
  - Chaque système a un certain nombre d'étapes n
- **États s_n**
  - À chaque étape n, il existe un ensemble d'états possibles $A_n = \{s_n\}$
- **Variable de décision x_n**
  - À chaque étape n, une décision x_n lorsque le système est dans un des états $s_n \in A_n$ **entraîne une transition** de $s_)n$ vers $s_{n+1} \in A_{n+1}$

9.4.1 Formule de récurrence

Elle permet de calculer la déc ision optimale pour chaque étaat $s_n \in A_n$, étant donné la décision optimale pour chaque état $s_{n+1} \in A_{n+1}$ connu.

9.5 Exercice d'affectation

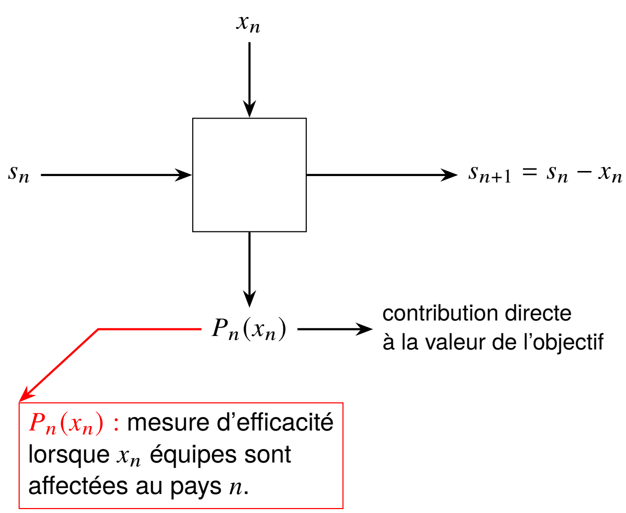
- On dispose de **5 équipes médicales** que l'on doit affecter à **3 pays**.
- Une mesure d'efficacité est fournie. Elle dépend du nombre d'équipes médicales affectées à un pays

- ▷ On veut **déterminer le nombre d'équipes médicales à affecter** afin de maximiser l'**efficacité totale**.

# Équipes	Pays 1	Pays 2	Pays 3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

Puisqu'il y a 3 pays, on peut donc considérer qu'il y a $n = 3$ **étapes d'affectations**. La mesure d'efficacité lorsque x_n équipes sont affectées au pays n est donnée par :

$$P_n(x_n)$$



9.5.1 Formule de récurrence

On a donc la valeur $f_n(s_n, x_n)$ qui correspond à l'efficacité maximale qu'on peut atteindre à partir de l'état s_n à l'étape n, **si on prend la décision d'affecter x_n équipes au pays n** :

$$f_n(s_n, x_n) = P_n(x_n) + f^*_{n+1}(s_n - x_n)$$

Note:-

Le diagramme d'état montre que, à l'état s_n, lorsqu'on affecte x_n équipes au pays n, il rest $s_n - x_n$ équipes disponibles affectables :

$$s_{n+1} = s_n - x_n$$

Ainsi, l'**efficacité maximale** qu'on peut atteindre à partir de l'état s_n à l'étape n est donnée par :

$$f^*_n(s_n) = \max_{0 \leqslant s_n \leqslant s_n} \{f_n(s_n, x_n)\} = f_n(s_n, x^*_n)$$

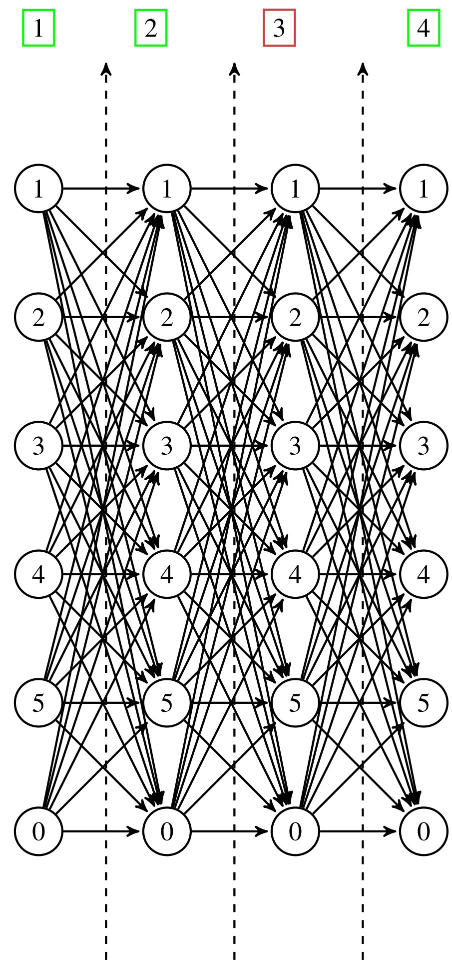
On a donc la formule de récurrence :

$$f^*_n(s_n) = \max_{0 \leqslant x_n \leqslant s_n} \{P_n(x_n) + f^*_{n+1}(s_n - x_n)\}$$

9.5.2 Étape fictive

Pour calculer $f^*_n(s_3)$, il faut qu'il y ait un $f^*_n(s_{n+1}) = f^*_n(s_4)$ qui intervient dans le calcul. On introduit donc une étape fictive s_4 :

$$f^*_4(s_4) = 0, s_4 = \{0, 1, 2, 3, 4, 5\}$$



9.5.3 Étape $n = 3$

$$\begin{aligned}
f_3^*(0) &= \max_{0\leqslant x_n\leqslant 0}\left\{P_3(x_3) + f_4^*(0-x_3)\right\}\\
&= P_3(0) + f_4^*(0) = 0+0 = 0\\
&= 50\\
&\implies x_3^*\Big|_{s=0} = 0\\
f_3^*(1) &= \max_{0\leqslant x_n\leqslant 1}\left\{P_3(x_3) + f_4^*(1-x_3)\right\}\\
&= \max\left\{P_3(0) + f_4^*(1), P_3(1) + f_4^*(0)\right\}\\
&= \max\left\{0+0, 50+0\right\}\\
&\implies x_3^*\Big|_{s=1} = 1\\
f_3^*(2) &= \max_{0\leqslant x_n\leqslant 2}\left\{P_3(x_3) + f_4^*(2-x_3)\right\}\\
&= \max\left\{P_3(0) + f_3^*(2), P_3(1) + f_3^*(1), +P_3(2) + f_3^*(0)\right\}\\
&= \max\left\{0+70, 20+50, 45+0\right\}\\
&= 90\\
&\implies x_3^*\Big|_{s=1} = 0 \text{ ou } 1
\end{aligned}$$

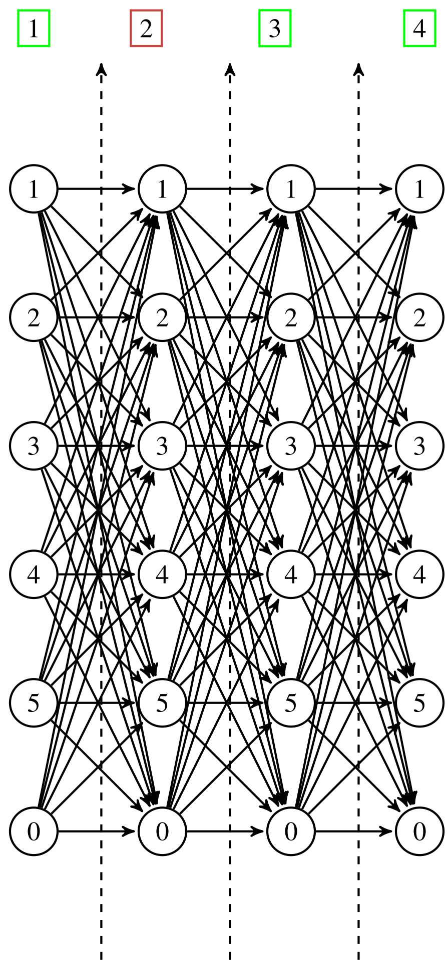
$$\begin{aligned}
f_3^*(3) &= \max_{0\leqslant x_3\leqslant 3}\left\{P_3(x_3) + f_4^*(3-x_3)\right\}\\
&= \max\left\{P_3(0) + f_4^*(3), P_3(1) + f_4^*(2), P_3(2) + f_4^*(1), P_3(3) + f_4^*(0)\right\}\\
&= \max\left\{0+0, 50+0, 70+0, 80+0\right\}\\
&= \max\left\{0, 50, 70, 80\right\}\\
&= 80\\
&\implies x_3^*\Big|_{s=3} = 3
\end{aligned}$$

$$\begin{aligned}
f_3^*(4) &= \max_{0\leqslant x_3\leqslant 4}\left\{P_3(x_3) + f_4^*(4-x_3)\right\}\\
&= \max\left\{P_3(0) + f_4^*(4), P_3(1) + f_4^*(3), P_3(2) + f_4^*(2), P_3(3) + f_4^*(1), P_3(4) + f_4^*(0)\right\}\\
&= \max\left\{0+0, 50+0, 70+0, 80+0, 100+0\right\}\\
&= \max\left\{0, 50, 70, 80, 100\right\}\\
&= 100\\
&\implies x_3^*\Big|_{s=4} = 4
\end{aligned}$$

$$\begin{aligned}
f_3^*(5) &= \max_{0\leqslant x_3\leqslant 5}\left\{P_3(x_3) + f_4^*(5-x_3)\right\}\\
&= \max\left\{P_3(0) + f_4^*(5), P_3(1) + f_4^*(4), P_3(2) + f_4^*(3), P_3(3) + f_4^*(2), P_3(4) + f_4^*(1), P_3(5) + f_4^*(0)\right\}\\
&= \max\left\{0+0, 50+0, 70+0, 80+0, 100+0, 130+0\right\}\\
&= \max\left\{0, 50, 70, 80, 100, 130\right\}\\
&= 130\\
&\implies x_3^*\Big|_{s=5} = 5
\end{aligned}$$

s_3	$f_3^*(s_3)$	x_3^*
0	0	0
1	50	1
2	70	2
3	80	3
4	100	4
5	130	5

9.6 Étape $n = 2$



$$
\begin{aligned}
f_2^*(0) &= \max_{0 \leqslant x_2 \leqslant 0} \left\{ P_2(x_2) + f_3^*(0 - x_2) \right\} \\
&= P_2(0) + f_3^*(0) \\
&= 0 + 0 \\
&= 0 \\
&\implies x_2^*\Big|_{s=0} = 0
\end{aligned}
$$

$$
\begin{aligned}
f_2^*(1) &= \max_{0\leqslant x_2\leqslant 1}\left\{P_2(x_2) + f_3^*(1-x_2)\right\} \\
&= \max\left\{P_2(0) + f_3^*(1), P_2(1) + f_3^*(0)\right\} \\
&= \max\left\{0+50, 20+0\right\} \\
&= 50 \\
&\implies x_2^*\Big|_{s=1} = 0
\end{aligned}
$$

$$
\begin{aligned}
f_2^*(2) &= \max_{0\leqslant x_2\leqslant 2}\left\{P_2(x_2) + f_3^*(2-x_2)\right\} \\
&= \max\left\{P_2(0) + f_3^*(2), P_2(1) + f_3^*(1), P_2(2) + f_3^*(0)\right\} \\
&= \max\left\{0+70, 20+50, 45+0\right\} \\
&= 70 \\
&\implies x_2^*\Big|_{s=2} = 0
\end{aligned}
$$

$$
\begin{aligned}
f_2^*(3) &= \max_{0\leqslant x_2\leqslant 3}\left\{P_2(x_2) + f_3^*(3-x_2)\right\} \\
&= \max\left\{P_2(0) + f_3^*(3), P_2(1) + f_3^*(2), P_2(2) + f_3^*(1), P_2(3) + f_3^*(0)\right\} \\
&= \max\left\{0+80, 20+70, 45+50, 75+0\right\} \\
&= 95 \\
&\implies x_2^*\Big|_{s=3} = 1
\end{aligned}
$$

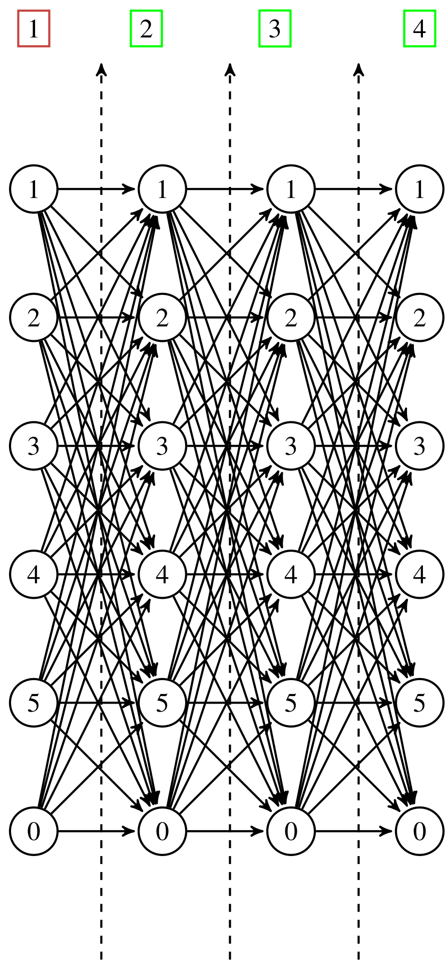
$$
\begin{aligned}
f_2^*(4) &= \max_{0\leqslant x_2\leqslant 4}\left\{P_2(x_2) + f_3^*(4-x_2)\right\} \\
&= \max\left\{P_2(0) + f_3^*(4), P_2(1) + f_3^*(3), P_2(2) + f_3^*(2), P_2(3) + f_3^*(1), P_2(4) + f_3^*(0)\right\} \\
&= \max\left\{0+100, 20+80, 45+70, 75+50, 110+0\right\} \\
&= 125 \\
&\implies x_2^*\Big|_{s=4} = 4
\end{aligned}
$$

$$
\begin{aligned}
f_2^*(5) &= \max_{0\leqslant x_2\leqslant 5}\left\{P_2(x_2) + f_3^*(5-x_2)\right\} \\
&= \max\left\{P_2(0) + f_3^*(5), P_2(1) + f_3^*(4), P_2(2) + f_3^*(3), P_2(3) + f_3^*(2), P_2(4) + f_3^*(1), P_2(5) + f_3^*(0)\right\} \\
&= \max\left\{0+130, 20+100, 45+80, 75+70, 110+50, 150+0\right\} \\
&= 160 \\
&\implies x_2^*\Big|_{s=5} = 5
\end{aligned}
$$

s_2	$f_2^*(s_2)$	x_2^*
0	0	0
1	50	0
2	70	0, 1
3	95	2
4	125	3
5	160	4

s_3	$f_3^*(s_3)$	x_3^*
0	0	0
1	50	1
2	70	2
3	80	3
4	100	4
5	130	5

9.6.1 Étape $n = 1$



$$
\begin{aligned}
f_1^*(0) &= \max_{0\leqslant x_1\leqslant 0}\left\{P_1(x_1) + f_2^*(0-x_1)\right\}\\
&= P_1(0) + f_2^*(0)\\
&= 0 + 0\\
&= 0\\
&\implies x_1^*\Big|_{s=0} = 0
\end{aligned}
$$

$$\begin{aligned} f_1^*(1) &= \max_{0\leqslant x_1\leqslant 1}\{P_1(x_1)+f_2^*(1-x_1)\} \\ &= \max\{P_1(0)+f_2^*(1), P_1(1)+f_2^*(0)\} \\ &= \max\{0+50, 45+0\} \\ &= 50 \\ &\Longrightarrow x_1^*\Big|_{s=1} = 0 \end{aligned}$$

$$\begin{aligned} f_1^*(2) &= \max_{0\leqslant x_1\leqslant 2}\{P_1(x_1)+f_2^*(2-x_1)\} \\ &= \max\{P_1(0)+f_2^*(2), P_1(1)+f_2^*(1), P_1(2)+f_2^*(0)\} \\ &= \max\{0+70, 45+50, 70+0\} \\ &= 95 \\ &\Longrightarrow x_1^*\Big|_{s=2} = 1 \end{aligned}$$

$$\begin{aligned} f_1^*(3) &= \max_{0\leqslant x_1\leqslant 3}\{P_1(x_1)+f_2^*(3-x_1)\} \\ &= \max\{P_1(0)+f_2^*(3), P_1(1)+f_2^*(2), P_1(2)+f_2^*(1), P_1(3)+f_2^*(0)\} \\ &= \max\{0+95, 45+70, 70+50, 90+0\} \\ &= 115 \\ &\Longrightarrow x_1^*\Big|_{s=3} = 1 \end{aligned}$$

$$\begin{aligned} f_1^*(4) &= \max_{0\leqslant x_1\leqslant 4}\{P_1(x_1)+f_2^*(4-x_1)\} \\ &= \max\{P_1(0)+f_2^*(4), P_1(1)+f_2^*(3), P_1(2)+f_2^*(2), P_1(3)+f_2^*(1), P_1(4)+f_2^*(0)\} \\ &= \max\{0+125, 45+95, 70+70, 90+50, 105+0\} \\ &= 140 \\ &\Longrightarrow x_1^*\Big|_{s=4} = 4 \end{aligned}$$

$$\begin{aligned} f_1^*(5) &= \max_{0\leqslant x_1\leqslant 5}\{P_1(x_1)+f_2^*(5-x_1)\} \\ &= \max\{P_1(0)+f_2^*(5), P_1(1)+f_2^*(4), P_1(2)+f_2^*(3), P_1(3)+f_2^*(2), P_1(4)+f_2^*(1), P_1(5)+f_2^*(0)\} \\ &= \max\{0+160, 45+125, 70+95, 90+70, 105+50, 120+0\} \\ &= 170 \\ &\Longrightarrow x_1^*\Big|_{s=5} = 5 \end{aligned}$$

$n=1$

s_1	$f_1^*(s_1)$	x_1^*
0	0	0
1	50	0
2	95	1
3	115	1
4	140	4
5	170	**1**

$n=2$

s_2	$f_2^*(s_2)$	x_2^*
0	0	0
1	50	0
2	70	0, 1
3	95	2
4	125	**3**
5	160	4

$n=3$

s_3	$f_3^*(s_3)$	x_3^*
0	0	0
1	50	**1**
2	70	2
3	80	3
4	100	4
5	130	5

9.6.2 Politique de décision

$$x_1^* = 1 \longrightarrow x_2^* = 3 \longrightarrow x_3^* = 1$$
$$= 45 + 75 + 50$$

# Équipes	Pays 1	Pays 2	Pays 3
0	0	0	0
1	$\boxed{45}$	20	$\boxed{50}$
2	70	45	70
3	90	$\boxed{75}$	80
4	105	110	100
5	120	150	130