

**Définition :** R.O. = résolution de problèmes linéaires courants : allocation de ressources, gestion de production, économie ...

**Terminologie :**

- variables de décision ( $x_i$ )
- fonction objectif ( $z$ )
- contraintes (inéquations)
- contraintes de bornes ( $x_i \geq 0$ )
- solution admissible  $\Leftrightarrow$  satisfait les contraintes
- valeur d'une solution = valeur de la fonction objectif en ce point
- domaine admissible = ensemble des solutions admissibles du problème

**Hypothèses de la PL :** linéarité, proportionnalité, divisibilité des variables, déterminisme des données

**Interprétation géométrique :** intersection de demi-plans  $\Rightarrow$  domaine admissible

**Les différentes formes d'un PL :**

1. forme canonique
2. forme standard ( $\rightarrow$  simplexe !)

On passe de la forme canonique à la forme standard en ajoutant des variables d'écart !

forme canonique :	$\leftrightarrow$	forme standard
* max		* max
* s.c. $\leq$		* s.c. =
* variables $\geq 0$		* variables $\geq 0$
		$\rightarrow$ solution basique

**Forme canonique :**

$$\begin{array}{ll}
 \text{Maximiser} & z = 5x_1 + x_2 \\
 \text{s.c.} & 2x_1 + x_2 \leq 6 \\
 & x_1 + x_2 \leq 4 \\
 & x_1, x_2 \geq 0
 \end{array}$$

**Forme standard :** Maximiser ( $z$ , s.c.  $x_1, x_2, x_3, x_4 \geq 0$ )

$$\begin{array}{rcl}
 \text{avec} & x_3 & = 6 - 2x_1 - x_2 \\
 & x_4 & = 4 - x_1 - x_2 \\
 \hline
 & z & = 0 + 5x_1 + x_2
 \end{array}$$

### Règles de transformation :

- $x \in \mathbb{R} : x = x^+ - x^- \ (x^+, x^- \geq 0)$
- $\min(f(x)) = -\max(-f(x))$
- $\geq \rightarrow \leq$  : multiplier par (-1)
- $ax = b \Leftrightarrow \begin{cases} ax \leq b \\ -ax \leq -b \end{cases}$
- inéquation  $\rightarrow$  équation : on rajoute des variables d'écart  $s \ (s \geq 0)$   $ax \leq b \Leftrightarrow ax + s = b \quad ax \geq b \Leftrightarrow ax - s = b$
- $|x| \leq b \Leftrightarrow \begin{cases} x \leq b \\ x \geq -b \end{cases}$
- $\min z = \max \{c_1 x, \dots, c_n x\} \Leftrightarrow \begin{cases} \min z = t \\ \text{s.c.} \quad t \geq c_i x \\ t \geq 0 \end{cases}$
- $\min |x| \Leftrightarrow \begin{cases} \min t \\ \text{s.c.} \quad t = |x| \end{cases} \Leftrightarrow \begin{cases} \min t \\ \text{s.c.} \quad t \leq x \end{cases}$   
(cas  $t \geq |x|$  non convexe)

**Résultat d'une optimisation linéaire :** Le domaine admissible d'un P.L. peut être :

- vide  $\Leftrightarrow$  pas de solution admissible
- borné (et non vide) ; le problème possède tj au moins une sol. optimale,  $\forall$  fonct. obj.
- non borné. Selon la fonction objectif choisie :
  - le problème peut posséder des solutions optimales
  - il peut exister des sol. admissibles de valeur arbitrairement grande (ou petite)  $\Rightarrow$  pas de sol. optimale finie  $\Rightarrow$  non borné !

**Tableau initial admissible/dégénéré :** admissible si les  $b_i$  sont  $\geq 0$  ( $\oplus$ )

	$x_D$	$x_E$	$z$	
$T_0 =$	$A$	$I$	$0$	$b$
	$-\gamma$	$0$	$1$	$0$

$T_{degenere} =$			*
			0
			*

Dégénéré  $\Rightarrow$  risque de cyclage.

Règle de Bland :

si plusieurs pivots possibles  $\Rightarrow$  prendre celui de plus petit indice.

**Tableau optimal et non borné :**

$x_1$	$\dots$	$x_n$	$z$	
			$0$	$\oplus$
				$:$
				$\oplus$
$\oplus$	$\dots$	$\oplus$	$1$	$*$

$x_1$	$x_k$	$x_n$	$z$	
	$\ominus$			$\oplus$
	$:$		$0$	$:$
	$\ominus$			$\oplus$
$*$	$-$	$*$	$1$	$*$

Si on ne se trouve pas dans un de ces deux cas, on peut effectuer un pivotage t.q. la nouvelle solution basique soit  $\geq$  à la précédente.

Choix du pivot :

1. colonne où  $-\gamma_r < 0$  (coût marginal négatif)
2. ligne où  $\min \{ \frac{\beta_i}{\alpha_{ij}} | \alpha_{ij} > 0 \}$  (min des Q.C.)

**Algo. dual du simplexe :** tableaux particuliers

$x_1$	$\dots$	$x_n$	$z$	
$\oplus$	$\dots$	$\oplus$	0	*
			$\vdots$	—
			0	*
$\oplus$	$\dots$	$\oplus$	1	*

Dual non borné

$x_1$	$\dots$	$x_n$	$z$	
$\oplus$	$\dots$	$\oplus$	0	*
			$\vdots$	—
			0	*
*	$\dots$	*	1	*

Sans solution admissible

### Algo. primal du simplexe, phase I et II

1. Problème textuel  $\rightarrow$  formulation sous forme canonique
2. forme standard et tableau initial correspondant  $\rightarrow$  SIMPLEXE
3. – tableau admissible  $\Rightarrow$  phase II  
– tableau non admissible  $\Rightarrow$  phase I (pb aux.  $\rightarrow$  tableau admissible)
4. Phase II : (boucle  $\odot$ )  
– si (optimal ou non borné)  $\Rightarrow$  STOP  
– sinon pivotage :  
– choix du pivot (coût marginal  $< 0$ ,  $\min \{Q.C.\}$ )  
– si dégénéré, risque de cyclage  $\Rightarrow$  règle de Bland

Rem : Algo du Simplexe  $\approx$  algèbre linéaire, sauf que l'on cherche à augmenter  $z$  à chaque pivotage.

### Algorithme du Simplexe : Phase I

**Données :** un tableau non admissible

**Résultat :** un tableau admissible ou un certificat d'absence de solutions admissibles

1. Construire le PL auxiliaire et un tableau initial admissible :  
– ajout de  $x_0$  aux  $b_i < 0$   
– la fct objectif devient  $z' = -x_0$  (à maximiser)  
l'ancienne fct obj. passe dans les contraintes  
– faire entrer  $x_0$  dans la base en pivotant sur  $\alpha_{j0}$   
où  $j = \min\{i | b_i = \min(b_k | b_k < 0)\}$   
( $\Leftrightarrow$  la ligne est déterminée par le plus petit  $b_i < 0$ )
2. Résoudre le PL auxiliaire à l'aide de la phase II de l'algorithme du simplexe et en utilisant la règle de Bland.  
– Si  $z' = 0$  à l'optimum, supprimer les colonnes de  $x_0$ ,  $z'$  et la ligne de  $z'$ . Le tableau restant est admissible pour le problème de départ.  
– Si  $z' < 0$  à l'optimum, le problème de départ n'admet pas de solutions admissibles.

### La dualité en programmation linéaire :

PLP : problème linéaire primal, PLD : problème linéaire dual

Règles de dualisation :

Problème de maximisation :	$\leftrightarrow$	Problème de minimisation
Variable $x_j \geq 0$	$\leftrightarrow$	$j^e$ contrainte de type $\geq$
Variable $x_j \in \mathbb{R}$	$\leftrightarrow$	$j^e$ contrainte de type $=$
Variable $x_j \leq 0$	$\leftrightarrow$	$j^e$ contrainte de type $\leq$
$i^e$ contrainte de type $\leq$	$\leftrightarrow$	variable $y_i \geq 0$
$i^e$ contrainte de type $=$	$\leftrightarrow$	variable $y_i \in \mathbb{R}$
$i^e$ contrainte de type $\geq$	$\leftrightarrow$	variable $y_i \leq 0$

**Algorithme dual du simplexe :**

- Données : un tableau dual admissible
- Résultat : un tableau optimal ou un certificat d'absence de solutions admissibles

Boucle  $\odot$  (1)-(3) :

1. Choix d'une variable sortante = ligne avec  $\beta_i < 0$   
Si il n'existe pas de variable sortante : STOP : le tableau est optimal
2. Choix d'une variable entrante :  
Choix de la colonne maximisant les QC duaux :  $r = \max\{\frac{-\gamma_j}{\alpha_{ij}} | \alpha_{ij} < 0\}$   
Si il n'existe pas de variable entrante : STOP le dual est non borné, et le primal est sans solutions admissibles.
3. Pivoter autour de  $\alpha_{ir}$  . Retourner en (1).

**Phase I duale :**

- Données : un tableau non admissible
- Résultat : un tableau admissible ou un certificat d'absence de solutions admissibles

1. Construire le PL auxiliaire et un tableau initial admissible
  - Introduire la variable auxiliaire  $x_0$  dans toutes les contraintes vérifiant  $b_i < 0$ .
  - Ajouter la fonction objectif auxiliaire (à maximiser)  $z' = -x_0$ .
  - Faire entrer  $x_0$  dans la base en pivotant sur  $a_{j0}$  où :

$$j = \min\{i | b_i = \min\{b_k | b_k < 0\}\}$$

2. Résoudre le PL auxiliaire à l'aide de la phase II de l'algorithme du simplexe et en utilisant la règle de Bland.
  - Si  $z' = 0$  à l'optimum, supprimer les colonnes de  $x_0$  et  $z'$  ainsi que la ligne de  $z'$ . Le tableau restant est admissible pour le problème de départ.
  - Si  $z' < 0$  à l'optimum, le problème de départ n'admet pas de solutions admissibles.

$$T = \begin{array}{|c|c|c|c|} \hline \alpha & I & 0 & \beta \\ \hline -\gamma & 0 & 1 & 0 \\ \hline \end{array}$$

**Analyse de sensibilité :** = dans quel intervalle peut varier un coefficient sans que la base optimale change ?

$$\boxed{\delta B_i^{-1} \geq -\beta} \quad (1) \qquad \boxed{z' = z + y_i \cdot \delta} \quad (2)$$

Exemple :

$$T_{opt} = \begin{array}{|c|c|c|c|c|c|c|} \hline x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & z \\ \hline 1 & -1 & 0 & 0 & 1 & 0 & 8 \\ 0 & 22 & 0 & 1 & 1 & 7 & 31 \\ 0 & 2 & 1 & 0 & 0 & 1 & 4 \\ 0 & 5 & 0 & 0 & 1 & 1 & 12 \\ \hline \end{array}$$

$$\text{composante } b_1 : (1) \Leftrightarrow \delta \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \geq - \begin{pmatrix} 8 \\ 31 \\ 4 \end{pmatrix} \Leftrightarrow \delta \geq -31$$

( $b_1$  initial = -5)  $\Rightarrow$  peut varier  $\in [-36, \infty[$  sans modifier la base optimale.

Fonction objectif : (2)  $\Rightarrow 12 + 0 \cdot \delta = 12$

### **Théorie des graphes :**

- Graphe  $G = (V, E, \varphi)$  = triplet (sommets, arêtes, fct.incidence)
- Arc = orienté, arête = non orienté
- Graphe simple = graphe sans boucle ( $\odot$ ) ni arcs/arêtes multiples.
- Degré ( $\deg(v)$ ) = nb d'arêtes/arcs incidents à  $v$   
*Rem* : si un sommet possède une ou plusieurs boucles, chacune apporte une contribution de 2 dans le calcul du degré de ce sommet.
  - degré extérieur :  $\deg_+(v) (\leftarrow \bullet \rightarrow)$
  - degré intérieur :  $\deg_-(v) (\rightarrow \bullet \leftarrow)$
- Rem* : Dans tout graphe, la somme des degrés = pair ( $2 \times$  le nb d'arêtes)
- chaîne = suite alternée de sommets/arêtes (pas orienté !)
- cycle  $\odot$  = chaîne dont les extrémités sont confondues
- chemin = suite alternée sommets/arcs (orienté !)
- arbre = multigraphe non orienté sans cycle et connexe ( $n-1$  arêtes)
- forêt = multigraphe non orienté sans cycle
- ★ Connexité  $\Leftrightarrow \exists$  une chaîne qui va de  $a$  à  $b$
- ★ Connexité forte :  $G$  et  $F$  sont fortement connexes  $\Leftrightarrow$  toute paire de sommets distincts est reliée par un chemin.

Algorithme de marquage d'un multigraphe orienté :

- on note un sommet  $\pm$ . On note ses successeurs +, ses prédécesseurs -
- les sommets marqués  $\pm$  sont fortement connexes.

### **Matrices d'adjacence et d'incidence :**

- $A$  = Matrice d'adjacence sommet-sommet (symétrique, graphe simple, non orienté)
- $B$  = Matrice d'adjacence sommet-sommet (asymétrique, graphe simple, orienté)

$$A_{n \times n} = \begin{pmatrix} \cdots & 1 \\ 1 & \cdots \end{pmatrix} \quad B_{n \times n} = \begin{pmatrix} \cdots & 1 \\ 0 & \cdots \end{pmatrix}$$

- $C$  = Matrice d'incidence sommets-arêtes
- $D$  = Matrice d'incidence sommets-arcs

$$C_{n \times m} = \begin{pmatrix} \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \end{pmatrix} \quad D_{n \times m} = \begin{pmatrix} \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & -1 & \cdots \\ \cdots & 0 & \cdots \end{pmatrix}$$

Fonction d'incidence  $\varphi$  :

$\varphi$	$e_1$	$e_2$	$\dots$
$u(e)$	$v_1$	$v_3$	$\dots$
$v(e)$	$v_2$	$v_2$	$\dots$

### Algorithmes de la théorie des graphes :

- Problème de l'arbre max de poids min.
  - Algorithme de Kruskal :
  - Algorithme de Prim :
- Problèmes de plus courts chemins :
  - Principe d'optimalité de Bellman (un plus court chemin est formé de plus courts chemins )
  - Pondérations non négatives : Dijkstra
  - Les graphes acycliques
    - Fonction rang et tri topologique
    - Plus courts et plus longs chemins dans les graphes acycliques
    - Application à la gestion de projet : méthode PERT

### Algorithme de Kruskal

- Donnée : graphe connexe avec pondération ( $\in \mathbb{R}$ ) des arêtes
- Résultat : arbre max. de poids min.
  1. Numéroté les arêtes par ordre croissant
  2. si  $e_k$  ne forme pas de cycle, on l'ajoute.

Exemple :

$k$	$e_k$	$c(e_k)$	$\in T$
1	$\{v_1, v_3\}$	1	✓
2	$\{v_2, v_3\}$	1	✓
3	$\{v_1, v_2\}$	2	✗
...	...	...	...

Rem : Pour obtenir un arbre max de poids max, on trie les arêtes par ordre décroissant.

### Algorithme de Prim : (variante de l'algo. de Kruskal)

1. On choisit arbitrairement un sommet
2. On fait croître l'arbre en le connectant aux nouveaux sommets de la manière la plus économique à chaque fois.

### Chaîne de section minimale reliant 2 sommets :

(section = valeur max des poids des arêtes)

THM : l'arbre max de poids min fournit aussi des chaînes de section minimale entre toutes les paires de sommets !!!

### Plus courts chemins dans les réseaux : (graphe orienté + pondération)

Rem : la longueur d'un chemin  $\neq$  nombre d'arcs, mais = somme des poids des arcs !!!

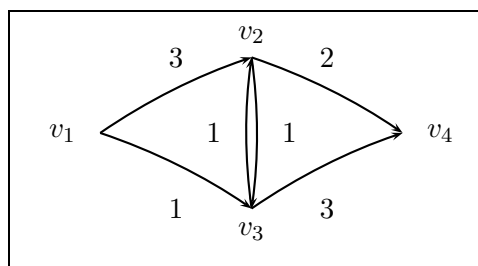
Principe d'optimalité de Bellman : un plus court chemin est formé de plus courts chemins.

### Algorithme générique : plus court chemin d'un sommet-source ( $v_s$ ) à tous les autres

1. Vecteur longueur  $\lambda$  (initial  $\lambda_s = 0, \lambda_i = \infty$ )
2. mise à jour des candidats  $L$  (initial  $L = \{v_s\}$ )
3. tant que  $L \neq \emptyset$  on retire le sommet  $i$  de  $L$ , on teste ses successeurs  $j$  :  
si  $\lambda_j > \lambda_i + c_{ij}$ , on pose  $\lambda_j = \lambda_i + c_{ij}$  et on introduit  $j$  dans  $L$

Exemple : (sens de travail :  $\leftarrow$ )

Itération	Candidats $L$	étiquettes $\lambda$	Sommet retiré de $L$
0	$\{v_1\}$	$(0, \infty, \infty, \infty)$	-
1	$\{v_2, v_3\}$	$(0, 3, 1, \infty)$	$v_1$
2	$\{v_3, v_4\}$	$(0, 3, 1, 5)$	$v_2$
...	...	...	...



### Algorithme de Dijkstra

- Donnée : réseau connexe avec pondération non-négative
- Résultat : plus court chemin de  $v_s$  à  $v_i$  + prédécesseur  $p(i)$ 
  1.  $\lambda_s = 0 \quad \lambda_i = \infty \quad p(i) = NULL$
  2. tant que  $T \neq \emptyset : (T = L \cup \{j | \lambda_j = \infty\})$ 
    - soit  $i$  le sommet de plus petite étiquette  $\lambda_i$
    - si un tel sommet n'existe pas : STOP (pas atteignable)
    - sinon retirer  $i$  de  $T$  et tester les successeurs  $j$  :  
si  $\lambda_j > \lambda_i + c_{ij} \Rightarrow \lambda_j = \lambda_i + c_{ij}$  et  $p(j) = i$

Exemple : (même graphe que précédent !)

Itér.	$i$ min	Étiquette $\lambda_i$ / prédécesseur $p(i)$				T
0	-	0/null	$\infty$ /null	$\infty$ /null	$\infty$ /null	$\{v_1, v_2, v_3, v_4\}$
1	$v_1$	0/null	3/ $v_1$	1/ $v_1$	$\infty$ /null	$\{v_2, v_3, v_4\}$
2	$v_3$		2/ $v_3$	1/ $v_1$	4/ $v_3$	$\{v_2, v_4\}$
3	$v_2$		2/ $v_3$		4/ $v_3$	$\{v_4\}$
4	$v_4$				4/ $v_3$	$\emptyset$

### Les graphes acycliques : (= sans circuit)

Un graphe acyclique possède au moins un sommet sans prédécesseur et un sans successeur.

Graphe acyclique  $\Leftrightarrow$  on peut attribuer à chaque sommet  $i$  son rang : ( $\forall$  arc  $(i, j)$  on a  $r(i) < r(j)$ )

### Algorithme du rang :

- Donnée : graphe orienté acyclique
- Résultat :  $\forall$  sommet  $i$  un rang  $r(i)$  minimal.
  1.  $k = 1$  (initialisation)
  2. on prend les sommets sans prédécesseurs : rang =  $k$
  3. on élimine tous les sommets sans prédécesseurs,  $k = k + 1$  puis  $\odot$  (2)

### Tri topologique :

- Donnée : graphe orienté acyclique
- Résultat : numérotation des sommets compatible avec le rang
  1.  $k = 1$  (initialisation)
  2. soit un sommet  $v_i$  sans prédécesseur : on pose  $v_i = k$
  3. on retire le sommet du graphe,  $k = k + 1$  puis  $\odot$  (2)

### Plus courts chemins dans les réseaux sans circuit :

1. tri topologique du graphe : sommets 1..n
2.  $\lambda_1 = 0$  et pour  $k \in [2..n]$  on calcule  $\lambda_k = \min\{\lambda_j + c_{jk} | j \in Pred(k)\}$

**Application à la gestion de projet :** (méthode du chemin critique)projet complexe (tâches + ordre défini)  $\Rightarrow$  planification optimale

On modélise le problème par un réseau :

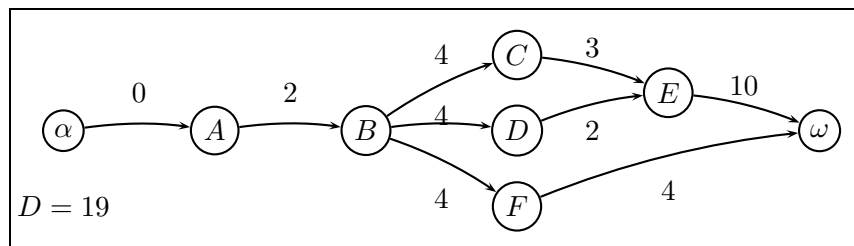
- sommets = tâches, arcs = contraintes de précédence
- poids de l'arc = durée de la tâche
- $\alpha$  = sommet sans prédécesseur, début des travaux, durée nulle
- $\omega$  = sommet sans successeur, fin des travaux

Exemple :

Code	Tâche	Durée	Préd.
A	Choix des stations	2	-
B	Accord administratif	4	A
C	Commande des décodeurs	3	B
D	Installation des antennes	2	B
E	Installation des décodeurs	10	C,D
F	Modification facturation	4	B

**Algorithme du chemin critique :**

- Donnée : réseau-projet trié topologiquement
  - Résultat : durée minimale du projet + planification
1. Récurrence en avançant dans le projet :  $\delta_1 = 0$ ,  
pour  $k \in [2..n]$  poser  $\delta_k = \max\{\delta_j + c_{jk} | j \in \text{Pred}(k)\}$
  2. Récurrence en reculant dans le projet :  $D = \delta_n$ ,  $\varphi_n = \delta_n$   
pour  $k \in [n - 1..1]$  poser  $\varphi_k = \min\{\varphi_j - c_{jk} | j \in \text{Succ}(k)\}$



Tâche	$\alpha$	A	B	C	D	E	F	$\omega$
N° $k$	1	2	3	4	5	6	7	8
Durée $d_k$	0	2	4	3	2	10	4	0
Préd. $(k)$	-	$\alpha$	A	B	B	C,D	B	E,F
Succ. $(k)$	A	B	C,D,F	E	E	$\omega$	$\omega$	-
$\delta_k \rightarrow$	0	0	2	6	6	9	6	19 (= D !)
$\varphi_k \leftarrow$	0	0	2	6	7	9	15	19

- tâche critique si  $\varphi_i = \delta_i$  (retard dans la tâche  $\Rightarrow$  retard du projet)
- chemin critique ( $\alpha \rightarrow \omega$ ) = chemin composé de tâches critiques
- longueur du chemin critique = durée minimale du projet

**Problème du transbordement :** Soit un graphe orienté et connexe

Pondération des sommets : offres = (-) , demandes = (+)

Pondération des arcs = coûts unitaires d'utilisation

Solution basique du PL de transbordement  $\Leftrightarrow$  arbre max. du réseau**Algo. graphique du simplexe dans les réseaux :** (phase II primale)

- Donnée : réseau connexe, solution-arbre admissible
- Résultat : un flot de coût min ou la preuve qu'un tel flot n'existe pas.



1. Calcul des solutions primale  $x$  et duale  $y$
2. Recherche d'un arc entrant. Si il n'en existe pas : STOP (sol.optimale)
3. Recherche d'un arc sortant. Si il n'en existe pas : STOP (pas d'optimum fini)
4. Mise à jour de la solution-arbre et retour en 1  $\odot$

#### Calcul de la solution primale associée à $T = (V, E_T)$

- Donnée : réseau connexe  $R = (V, E, b, c)$  ( $b$  = pond.des sommets,  $c$  = pond.des arcs), solution-arbre  $T = (V, E_T)$
  - Résultat : le flot  $x : E \rightarrow \mathbb{R}_+$  associé à  $T$ . (Et  $z = \sum c_{ij}x_{ij}$ )
1. Tant que  $|E_T| > 1$  faire
    - (a) Trouver un sommet pendant  $j$  de  $T$ . Soit  $e$  le seul arc incident avec  $j$  dans  $T$  et  $i$  l'autre extrémité de  $e$ .
    - (b) Si  $e = (i, j)$  poser  $x_{ij} = b_j$ , sinon poser  $x_{ji} = -b_j$
    - (c) Poser  $b_i = b_i + b_j$ .
    - (d) Retirer  $e$  de  $E_T : E_T = E_T \setminus \{e\}$
  2. Il reste un seul arc dans  $E_T$ , disons  $(i, j)$ , poser  $x_{ij} = b_j$

#### Calcul de la solution duale associée à $T = (V, E_T)$

- Donnée : réseau connexe  $R = (V, E, b, c)$ , solution-arbre  $T = (V, E_T)$
  - Résultat : les prix duaux  $y : V \rightarrow \mathbb{R}$  associés à  $T$ . (Et  $w = \sum b_i y_i$ )
1. Choisir arbitrairement  $i \in V$  et poser  $y_i = 0$  et  $W = V \setminus \{i\}$ .
  2. Tant que  $W \neq \emptyset$  faire
    - (a) Trouver un arc  $e \in E_T$  avec une extrémité  $j$  dans  $W$  et une extrémité  $i$  dans  $\bar{W} = V \setminus W$
    - (b) Si  $e = (i, j)$  poser  $y_j = y_i + c_{ij}$ , sinon poser  $y_j = y_i - c_{ij}$ .
    - (c) Retirer  $j$  de  $W : W = W \setminus \{j\}$

#### Recherche d'un arc entrant

- Donnée : solution-arbre admissible  $T = (V, E_T)$  ainsi que sol. primale ( $x$ ) et duale ( $y$ ) associées
- Résultat : arc entrant dans la base
- On passe en revue les arcs  $(i, j)$  hors base ( $\Leftrightarrow \notin E_T$ ) et on teste pour chacun d'eux si la contrainte duale associée  $y_j - y_i \leq c_{ij}$  est satisfaite ou non.
- contrainte violée  $\Rightarrow$  l'arc va entrer dans la base
- si toutes les contraintes sont satisfaites  $\Rightarrow$  solution optimale.

**Recherche d'un arc sortant** Si on ajoute à la solution-arbre actuelle l'arc entrant  $e = (i, j)$ , on forme un cycle unique  $C$ . Utilisant l'orientation de  $(i, j)$  pour définir le sens de parcours de  $C$ , on divise les arcs de  $C$  en 2 ensembles disjoints  $C^+$  (même orientation de  $(i, j)$ ) et  $C^-$ .

- Si  $C^- = \emptyset$  STOP : pas d'optimum fini
- Sinon soit  $\Delta = \min\{x_{kl} | (k, l) \in C^-\}$  et  $s$  un arc pour lequel le minimum est atteint. L'arc  $s$  quitte la base.

#### Mise à jour des solutions

- La nouvelle solution-arbre est donnée par  $E_T = E_T \cup \{e\} \setminus \{s\}$
- La solution basique primale ne change que sur les arcs de  $C$  :
  - $x_{ij} = x_{ij} + \Delta$  si  $(i, j) \in C^+$
  - $x_{ij} = x_{ij} - \Delta$  si  $(i, j) \in C^-$
  - $x_{ij} = x_{ij}$  si  $(i, j) \notin C$

**Dégénérescence** (si il existe des arcs dans  $E_T$  le long desquels des quantités nulles sont transportées, risque de cyclage)

Version graphique de la règle de Bland :

- Tester les arcs dans l'ordre lexicographique et faire entrer le premier arc dont la contrainte est violée
- Si la qté  $\Delta$  est transportée le long de plusieurs arcs de  $C^-$  faire sortir l'arc le plus petit dans l'ordre lexicographique.

### Calcul d'une solution-arbre admissible (Phase I)

On va définir un problème auxiliaire

- possédant toujours des solutions admissibles
- possédant toujours un optimum fini
- possédant un optimum fini si et seulement si le problème de départ possède au moins une solution admissible.

### Construction du problème aux. et d'une sol. initiale admissible

- Donnée : Un réseau  $R = (V, E, b, c)$  connexe.
- Résultat : Un réseau  $R' = (V, E', b, c)$  et une solution-arbre  $T' = (V, E'_T)$  admissible pour  $R'$ 
  1. Poser  $c'_{ij} = 0$  pour tout  $(i, j) \in E$
  2. Choisir un sommet source, disons  $k$
  3. Relier chaque sommet source  $i (\neq k)$  à  $k$  par un arc artificiel  $(i, k)$  de poids  $c'_{ik} = 1$  (si il existe déjà un arc de  $i$  à  $k$  dans  $R$  ne pas le rajouter)
  4. Relier  $k$  à chaque sommet puits  $j$  par un arc artificiel  $(k, j)$  de poids  $c'_{kj} = 1$  (si il existe déjà un arc de  $k$  à  $j$  dans  $R$  ne pas le rajouter)
  5. Poser  $E'_T = \{(i, k) | i \text{ sommet source}\} \cup \{(k, j) | j \text{ sommet puits}\}$  et compléter  $E'_T$  jusqu'à obtenir un arbre maximal (si nécessaire).

### Chaînes de Markov (= CM)

- processus stochastique (= aléatoire) :  $\{X_t, t \in T\}$  avec  $X_t$  = état du processus au temps  $t$
- si l'ensemble des états (S) est fini  $\Rightarrow$  chaîne.
- processus à temps continu ( $T = \mathbb{R}_+$ ) ou à temps discret ( $T = \mathbb{Z}_+$ )
- évolution du système = trajectoire
- processus Markovien  $\Leftrightarrow$  à l'instant  $t$   $X_t$  résume à lui seul l'historique du système  $\Rightarrow X_t$  suffit à déterminer l'état futur
- CM à temps discret homogène  $\Leftrightarrow$  indép. de l'origine du temps
- matrice de transition  $P \rightarrow$  graphe représentatif
- $P$  stochastique  $\Leftrightarrow (p_{ij} \geq 0 \text{ et } \sum \text{lignes} = 1 \forall \text{ ligne})$
- Probabilité de transition de  $i$  à  $j$  en  $m$  étapes =  $(P^m)_{ij}$  (avec  $P^0 = I$  et  $P^1 = P$ )
- composantes fortement connexes  $\rightarrow$  classes
- graphe réduit ( $G_R$ )  $\Leftrightarrow$  classe devient un seul état  $\Rightarrow G_R$  forcément sans circuit !
- états : transitoires, persistants, absorbants (= classe persistante à un seul él.)
- CM à une classe = irréductible, sinon réductible
- CM absorbante si tous les états persistants le sont
- période  $d = \text{PGDC des longueurs des circuits}$ . apériodique  $\Leftrightarrow d = 1$
- distribution invariante (ou stationnaire) :  $\pi \cdot P = \pi$  et  $\pi \cdot 1 = 1$
- nb moyen de transitions entre 2 visites successives de l'état  $i = 1/\pi_i$
- chaîne irréductible et apériodique = ergodique
- ergodique  $\Leftrightarrow$  1 seule classe persistante + états persistants = apériodiques
- matrice de transition canonique :  $P = \left( \begin{array}{c|c} I & 0 \\ \hline R & Q \end{array} \right)$ .

- $N = (I - Q)^{-1}$  matrice fondamentale
  - $n_{ij}$  = nombre moyen de périodes séjournées dans l'état transitoire  $j$  en partant de  $i$
  - nombre moyen de transitions avant d'atteindre un état absorbant en partant de  $i = \sum$  des termes de la  $i^e$  ligne de  $N$
- $B = N \cdot R$   $b_{ij}$  = proba d'être absorbé par l'état  $j$  en partant de  $i$

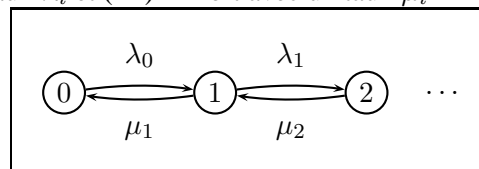
### Chaînes de Markov à temps continu

- Matrice  $P$  = probas de transition avec  $p_{ij}(t) = P[X_t = j | X_0 = i]$
- $P$  stochastique  $\Leftrightarrow (p_{ij} \geq 0 \text{ et } \sum_j p_{ij} = 1)$  Hyp :  $P(0) = I$
- temps de séjour  $\tau_i$  est une variable exponentielle de paramètre  $\alpha_i$
- chaîne de Markov à temps discret sous-jacente = matrice  $Q$ 
  - CMTD :  $(P, \pi(0)) \quad q_{ii} = 0 \Leftrightarrow \alpha_i > 0$
  - CMTC :  $(Q, \alpha, \pi(0)) \quad q_{ii} = 1 \Leftrightarrow \alpha_i = 0$
- loi exponentielle = la seule loi continue sans mémoire ( $E(x) = 1/\lambda$  et  $Var(x) = 1/\lambda^2$ )
- chaîne régulière  $\Leftrightarrow$  nb de transitions fini dans un temps fini. Toute chaîne avec un nb d'états fini = régulière
- matrice génératrice  $A$  (matrice d'intensité) (approche événementielle) :

$$a_{ij} = \begin{cases} -\alpha_i & \text{si } i = j \quad (\text{intensité de passage}) \\ \alpha_i q_{ij} & \text{si } i \neq j \quad (\text{intensité de transition}) \end{cases}$$

avec  $a_{ij} > 0$  si  $i \neq j$  et  $\sum_j a_{ij} = 0 \rightsquigarrow$  graphe représentatif de  $A$

- équations de Kolmogorov :
  - équations du futur :  $P'(t) = A \cdot P(t)$
  - équations du passé :  $P'(t) = P(t) \cdot A$
- distribution stationnaire :  $\pi \cdot A = 0$  et  $\pi \cdot 1 = 1$  ( $\rightsquigarrow$  solution unique si la chaîne est ergodique, càd récurrente non nulle)
- processus de naissance et de mort : CMTC où les seules transitions possibles depuis l'état  $i$  sont  $\rightarrow (i+1)$  = naissance avec un taux  $\lambda_i$  et  $(i-1)$  = mort avec un taux  $\mu_i$



- la matrice sous-jacente  $Q$  est donnée par :

$$Q = \begin{cases} q_{i,i-1} = \frac{\mu_i}{\lambda_i + \mu_i} \\ q_{i,i+1} = \frac{\lambda_i}{\lambda_i + \mu_i} \end{cases}$$

- cas particuliers :
  - processus de Poisson = naissance pur à tx constant =  $\lambda$
  - file  $M/M/1$  : tx de naissance (=  $\lambda$ ) et de mort (=  $\mu$ ) = const.

**Loi d'Erlang** Somme de  $k$  variables aléatoires  $X_k$  iid.  $\sim E(\lambda) \Rightarrow c_s^2 = \frac{1}{k}$

**Loi exponentielle** = la seule loi sans mémoire !!

$$X \sim E(\lambda) \quad Y \sim E(\mu) \quad P[X < Y] = \frac{\lambda}{\lambda + \mu}$$