

Recherche Opérationnelle

PAR FRANÇOIS BOCHATAY & CHRISTINA HAUENSTEIN

1 Programmation linéaire

1.1 Formes d'un programme linéaire

Un programme linéaire à deux formes possibles :

1. forme canonique

$$\begin{array}{llll} \text{Maximiser } z = & 5x_1 + x_2 & & \text{problème de maximisation} \\ \text{s.c.} & 2x_1 + x_2 & \leq & 6 \quad \text{contraintes de type } \leq \\ & x_1 + x_2 & \leq & 4 \\ & x_1, x_2 & \geq & 0 \quad \text{variables non-négatives} \end{array}$$

2. forme standard

$$\begin{array}{llll} \text{Maximiser } z = & 5x_1 + x_2 & & \text{problème de maximisation} \\ \text{s.c.} & 2x_1 + x_2 + x_3 & = & 6 \quad \text{contraintes de type } = \\ & x_1 + x_2 + x_4 & = & 4 \\ & x_1, x_2, x_3, x_4 & \geq & 0 \quad \text{variables non-négatives} \end{array}$$

On passe de la forme canonique à la forme standard en rajoutant des variables d'écarts.

1.2 Règles de transformation

Minimisation \Leftrightarrow Maximisation	$\min(z) = \max(\bar{z})$ où $\bar{z} = -z$
$\geq \Leftrightarrow \leq$	$ax \geq b \Leftrightarrow (-a)x \leq (-b)$
Équation \Rightarrow Inéquation	$ax = b \Leftrightarrow \begin{cases} ax \leq b \\ -ax \leq -b \end{cases}$
Inéquation \Rightarrow Équation	$ax \leq b \Leftrightarrow ax + s = b \quad \text{avec } s \geq 0$ $ax \geq b \Leftrightarrow ax - s = b \quad \text{avec } s \geq 0$
Variable réelle \Rightarrow non négative	$x \in \mathbb{R} \Rightarrow \begin{cases} x = x^+ - x^- \\ x^+, x^- \geq 0 \end{cases}$
Variable bornée inférieurement	$x \geq b \Leftrightarrow \begin{cases} x' = x - b \\ x' \geq 0 \end{cases}$
Maximum / Minimum	$z = \min \max \{cx, \dots, c'''x\} \Leftrightarrow \begin{matrix} z = t \\ \text{s.c. } t(\leq \geq) cx \dots \\ t \in \mathbb{R} \end{matrix}$
Valeurs absolues (fct objectif)	$\min(z) = x \Leftrightarrow \begin{matrix} z = t \\ \text{s.c. } x \leq t \\ t \geq 0 \end{matrix}$
Valeurs absolues (contraintes)	$ x \leq b \Leftrightarrow \begin{matrix} x \leq b \\ x \geq -b \end{matrix} \Leftrightarrow \begin{matrix} x \leq b \\ -x \leq b \end{matrix} \quad x \geq b \text{ non-linéaire !}$

1.3 Règles de dualisation

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

Si un programme linéaire primal a n variables et m contraintes, alors le programme linéaire dual a m variables et n contraintes. Les variables basiques du programme linéaire dual sont associées aux variables hors base du programme linéaire primal.

1.4 Tableaux

1.4.1 Construction

Reprenons le problème du point 1.1, sous forme standard

x_1	x_2	x_3	x_4	z	
2	1	1	0	0	6
1	1	0	1	0	4
-5	-1	0	0	1	0

x_1 et x_2 sont les variables de décision, et x_3 et x_4 sont les variables d'écart. La dernière colonne est formée des éléments à droite des contraintes. La dernière ligne est formée des coefficients inversés de la fonction objectif.

1.4.2 Propriétés

α_{11}	\dots	α_{1n}	0	β_1
\vdots	\ddots	\vdots	\vdots	\vdots
α_{m1}	\dots	α_{mn}	0	β_m
$-\gamma_1$	\dots	$-\gamma_n$	1	z

Notation : $\oplus \Leftrightarrow \geq 0$; $\ominus \Leftrightarrow \leq 0$; $-\Leftrightarrow < 0$; $+\Leftrightarrow > 0$

Primal admissible colonne $\beta \oplus$

Dual admissible ligne $\gamma \oplus$

Primal non-borné 1 colonne $\alpha_{ix} \ominus$ et $\gamma_x -$

Dual non-borné 1 ligne $\alpha_{xj} \oplus$ et solution $\beta_j -$

Primal dégénéré variable de base primale = 0

Dual dégénéré variable de base duale = 0

Optimal primal et dual admissible

Les variables de base primales sont les variables correspondant aux colonnes c de la matrice I , et les autres sont les variables de base duales.

1.5 Algorithme du Simplexe

Tout d'abord, il faut formuler le problème de départ **sous forme canonique, puis standard**. De la forme standard, l'on construit le tableau initial correspondant.

Si le tableau initial est admissible, on passe directement à la phase II; autrement il faut effectuer la phase I pour le rendre admissible avant de passer à la phase II.

Règle de Bland

Faire rentrer ou sortir de la base la variable de plus petit indice.

1.5.1 Phase I

Si le résultat de cette phase est un tableau non-borné, le problème n'admet pas de solutions.

Primal

Donnée : tableau non-admissible

a) Ajout d'un second membre artificiel

1. une colonne 0 à la droite du tableau \Rightarrow primal admissible
2. On applique l'algorithme du Simplexe primal phase II jusqu'à ce que le tableau initial soit dual-admissible ou non-borné.
3. On peut maintenant enlever notre colonne supplémentaire.

b) Construction d'un problème auxiliaire

1. Construire le problème auxiliaire

- ajout d'une variable x_0 valant -1 pour les $\beta_i < 0$
- la fonction objectif devient $\max z' = -x_0$
- faire rentrer x_0 dans la base en pivotant sur α_{j0} où $\beta_j < 0$ est le min de β

2. Appliquer l'algorithme du Simplexe phase II sur le problème auxiliaire en utilisant la règle de Bland.

3. Solutions

- Si $z' = 0$, supprimer tous les ajouts fait au point 1. Le tableau est maintenant admissible.
- Si $z' < 0$, le problème de départ n'admet pas de solutions admissible.

Résultat : tableau admissible ou non-borné

Dual

Donnée : tableau non-admissible

1. On ajoute une fonction objectif artificielle pour rendre le tableau dual admissible \Rightarrow une ligne

$\boxed{0 \cdots 0 \mid 1 \mid 0}$ à la fin du tableau.

2. On applique l'algorithme du Simplexe dual phase II jusqu'à ce que le tableau initial soit primal-admissible ou non-borné.

3. On peut maintenant enlever notre ligne supplémentaire.

Résultat : tableau admissible ou non-borné

1.5.2 Phase II**Primal**

Donnée : tableau primal admissible

1. Si le tableau est optimal ou non-borné \Rightarrow ARRÊT

2. Choix du pivot

- colonne : $-\gamma_j$ le plus négatif (+ règle de Bland si le tableau est dégénéré)
- ligne : $\min \left(\frac{\beta_i}{\alpha_{ij}} \right)$ pour $\alpha_{ij} > 0$ (quotient caractéristique; en cas d'égalité, on prend le max (α_{ij}))

3. Supprimer par opérations sur les lignes les éléments en dessus et en dessous du pivot

4. Retour à 1.

Résultat : tableau optimal ou non-borné

Dual

Donnée : tableau dual admissible

1. Si le tableau est optimal ou non-borné \Rightarrow ARRÊT

2. Choix du pivot

- ligne : $\beta_i < 0$ et règle de Bland
- colonne : $\max \left(\frac{-\gamma_j}{\alpha_{ij}} \right)$ pour $\alpha_{ij} < 0$ (quotient caractéristique dual)

3. Supprimer par opérations sur les lignes les éléments en dessus et en dessous du pivot

4. Retour à 1.

Résultat : tableau admissible ou non-borné

1.6 Analyse de sensibilité

1.6.1 Sensibilité du second membre

Dans quel intervalle δ_i peut varier le $i^{\text{ème}}$ coefficient de β sans que la base optimale ne change ?

$$\delta_i \vec{B}_i^{-1} \geq -\vec{\beta}$$

\vec{B}_i^{-1} est la colonne correspondant à la $i^{\text{ème}}$ variable d'écart du système.

La nouvelle solution optimale en fonction de $b_i \in [\beta_i \pm \delta_i]$ est donnée par

$$\vec{b} = \begin{pmatrix} \beta_1 \\ \vdots \\ b_i \\ \vdots \\ \beta_n \end{pmatrix} \quad x_B^* = \begin{pmatrix} x_1^* \\ \vdots \\ x_n^* \end{pmatrix} = B^{-1} \vec{b} \quad z' = \sum_{i=1}^n y_i b_i \Rightarrow z' \in [z^* + \delta_i y_i]$$

où B^{-1} est la matrice correspondant aux variables d'écart

1.6.2 Sensibilité de la fonction objectif

Dans quelle mesure peut-on modifier c_i le $i^{\text{ème}}$ coefficient de la fonction objectif sans que la base optimale ne change ?

$$-\vec{\gamma}'_H = \vec{c}'_B H - \vec{c}'_H \geq \vec{0}$$

H est la matrice des variables hors base.

Exemple 1. $\max z = 2x_1 + x_2$ de tableau optimal

La base optimale est $B = \{2, 1, 4\}$, $H = \{3, 5\}$, $c_B = (1 \ 2 \ 0)$ dont les entrées correspondent aux coefficients de la fonction objectif dans l'ordre où ils apparaissent dans la base. Et $c_H = (0 \ 0)$, coefficients des variables hors base. Pour trouver c'_B , il faut remplacer la valeur correspondante par une inconnue. Si l'on cherche c_1 , $c'_B = (1 \ c_1 \ 0)$. Il ne reste donc plus qu'à résoudre l'équation.

x_1	x_2	x_3	x_4	x_5	z	
0	1	1	0	-1	0	1
1	0	0	0	1	0	1
0	0	1	1	-2	0	1
0	0	1	0	1	1	3

1.6.3 Ajout de contrainte

Pour ajouter une contrainte dans un tableau optimal, il faut l'exprimer en fonction des variables dans la base optimale. Il faut de plus faire entrer la nouvelle variable d'écart dans la base. Enfin, il faut ré-appliquer l'algorithme du simplexe si nécessaire pour obtenir le nouveau tableau optimal.

Exemple 2.

$$T_{\text{opt}} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline x_1 & x_2 & x_3 & x_4 & x_5 & z & & \\ \hline 1 & 0 & 4 & 1 & 0 & 0 & 8 & \\ \hline 0 & 1 & 3 & 2 & 1 & 0 & 26 & \\ \hline 0 & 0 & 13 & 5 & 1 & 1 & 50 & \\ \hline \end{array}$$

On ajoute la contrainte $2x_1 + 2x_2 \leq 6$, donc $2x_1 + 2x_2 + x_6 = 6$.

Du tableau optimal, on tire que $x_1 = 8 - 4x_3 - x_4$ et $x_2 = 26 - 3x_3 - 2x_4 - x_5$.

Dans la base $B = \{1, 2, 6\}$, notre contrainte devient $-14x_3 - 6x_4 - 2x_5 + x_6 = -62$.

Le nouveau tableau est alors

x_1	x_2	x_3	x_4	x_5	x_6	z	
1	0	4	1	0	0	0	8
0	1	3	2	1	0	0	26
0	0	-14	-6	-2	1	0	-62
0	0	13	5	1	0	1	50

Il suffit alors de rendre ce tableau optimal pour trouver la nouvelle solution

2 Théories des graphes

2.1 Définitions

Graphe $G = (V, E, \varphi)$: (sommets, arêtes/arcs, fonction d'incidence).

Arêtes/Arcs incidents à v : toutes les arêtes/arcs qui sont connectés à v .

Graphe orienté (réseau) \Rightarrow arcs ; Graphe non orienté \Rightarrow arêtes.

Graphe simple = sans boucles (sur lui-même) ni arêtes/arcs multiples.

Multigraphe = avec boucles et/ou arêtes/arcs multiples.

Degré : $\deg(v) = \text{nb d'arêtes/arcs incidents à } v$.

– degré extérieur : $\deg_+(v) (\leftarrow \bullet \rightarrow)$

– degré intérieur : $\deg_-(v) (\rightarrow \bullet \leftarrow)$

Chaîne : suite alternée sommets/arêtes

– Élémentaire : chaque sommet apparaît au plus une fois

– Simple : chaque arête apparaît au plus une fois

Chemin : suite alternée sommets/arcs

Cycle : chaîne dont les extrémités sont confondues

Arbre : multigraphe non orienté sans cycle et connexe

Forêt : ensemble d'arbres non reliés entre eux

Connexité : on peut atteindre n'importe quel sommet

Connexité forte : connexité + les deux sens de circulation existent.

Nombre cyclomatique = $m - n + p$ (m arêtes, n sommets, p composantes connexes)

2.2 Matrice d'adjacence et d'incidence

Soit un graphe ayant n sommets et m arêtes.

2.2.1 Graphe simple non orienté

Matrice d'adjacence sommets-sommets $B: n \times n$

$$b_{ij} = \begin{cases} 1 & \text{si } v_i \text{ et } v_j \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$$

Matrice d'incidence sommets-arêtes $A: n \times m$

$$a_{ij} = \begin{cases} 1 & \text{si } v_i \text{ est incident à } e_j \\ 0 & \text{sinon} \end{cases}$$

2.2.2 Graphe simple orienté

Matrice d'adjacence sommets-sommets $B: n \times n$

$$b_{ij} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \\ 0 & \text{sinon} \end{cases}$$

Matrice d'incidence sommets-arêtes $A: n \times m$

$$a_{ij} = \begin{cases} -1 & \text{si } v_i \text{ est l'extrémité initiale de } e_j \\ 1 & \text{si } v_i \text{ est l'extrémité finale de } e_j \\ 0 & \text{sinon} \end{cases}$$

2.3 Algorithmes

2.3.1 Problème de l'arbre recouvrant de poids minimum

Ce problème est aussi appelé problème de section minimale.

Algorithme de Kruskal

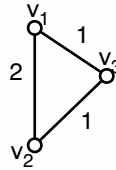
Donnée : graphe connexe avec pondération ($\in \mathbb{R}$) des arêtes

1. Numéroté les arêtes par ordre croissant en fonction de leurs poids
2. Partir d'un arbre solution vide
3. Si e_k ne forme pas de cycle, l'ajouter à l'arbre solution

Résultat : arbre recouvrant de poids total minimum

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	×
⋮	⋮	⋮	⋮



Pour obtenir l'arbre recouvrant de poids maximum, trier dans l'ordre décroissant.

Algorithme de Prim

Donnée : graphe connexe avec pondération ($\in \mathbb{R}$) des arêtes

1. Partir d'un arbre solution vide
2. Choisir arbitrairement un sommet
3. Choisir l'arête la plus économique pour rejoindre un sommet pas encore visité depuis un sommet déjà présent dans l'arbre solution. Ajouter cette arête à l'arbre solution.

Résultat : arbre recouvrant de poids total minimum

2.3.2 Problèmes de plus courts chemins

On cherche le plus court chemin d'un sommet source v_s à tous les autres.

Principe d'optimalité de Bellman

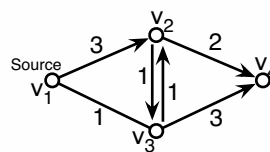
Un plus court chemin est formé de plus courts chemins.

Algorithme générique

1. On part d'un vecteur λ initial : $\lambda_s = 0$ et $\lambda_i = \infty \forall i \neq s$
2. On tient à jour une liste de candidats L , initialement $L = \{v_s\}$.
3. Tant que $L \neq \emptyset$, on retire le sommet v_i de L et 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 3.

Itération	L	λ	Sommet retiré de L
0	$\{v_1\}$	$(0, \infty, \infty, \infty)$	—
1	$\{v_2, v_3\}$	1	v_1
2	$\{v_3, v_4\}$	2	v_2
⋮	⋮	⋮	⋮
5	\emptyset	$(0, 2, 1, 4)$	v_2



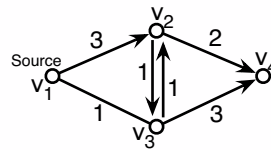
Algorithme de Dijkstra

Données : réseau connexe avec pondération non-négative

1. $\lambda_s = 0$; $\lambda_i = \infty \forall i \neq s$; $p(i) = \text{null}$; $T = V$
2. tant que $T \neq \emptyset$:
 - Soit i le sommet de T de plus petite étiquette λ_i
 - Si un tel sommet n'existe pas ($\lambda_j = \infty \forall j \in T$) : STOP, les sommets encore dans T ne sont pas atteignables depuis s
 - Sinon, retirer i de T et tester tous les successeurs j de i encore dans T :
si $\lambda_j > \lambda_i + c_{ij}$, on pose $\lambda_j = \lambda_i + c_{ij}$ et $p(j) = i$.

Exemple 4.

Itér.	i_{\min}	Étiquettes λ_i / Prédécesseurs $p(i)$				T
0	—	0/null	∞ /null	∞ /null	∞ /null	$\{v_1, v_2, v_3, v_4\}$
1	v_1	0/null	3/ v_1	1/ v_1	∞ /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



Graphes acycliques

Un graphe acyclique (sans-circuit) possède au moins un sommet sans prédécesseur, et au moins un sommet sans successeur. Un graphe est acyclique si et seulement si on peut attribuer à chaque sommet i un nombre $r(i)$, appelé rang, tel que pour tout arc reliant i à j on ait $r(i)$ plus petit que $r(j)$.

Algorithme du rang

Donnée : un graphe orienté acyclique

1. $k = 1$
2. On prend les sommets sans prédécesseurs : rang = k
3. On élimine tous les sommets sans prédécesseurs
4. $k = k + 1$; retour à 2.

Résultat : le rang $r(i)$ minimal

Tri topologique

Donnée : un graphe orienté acyclique

1. $k = 1$
2. On prend un sommet i sans prédécesseurs : $\nu_i = k$
3. On retire le sommet du graphe
4. $k = k + 1$; retour à 2.

Résultat : une numérotation des sommets compatible avec le rang

Plus courts chemins dans les réseaux sans circuit

1. Tri topologique du graphe : sommet 1... n
2. $\lambda_i = 0$ et pour $k = 1 \dots n$ on calcule $\lambda_k = \min \{\lambda_j + c_{jk} \mid j \in \text{Pred}(k)\}$

Application à la gestion de projet

On considère un projet complexe (tâches et ordre défini), on cherche une planification optimale minimisant la durée totale.

Le problème doit être modélisé par un réseau : les sommets représentent les tâches, les arcs les contraintes de précedence entre deux tâches. Le poids de l'arc sortant équivaut à la durée de la tâche. On ajoute des sommets α (de durée nulle et sans prédécesseur) représentant le début des travaux, et ω (de durée nulle et sans successeur) représentant la fin des travaux.

On cherche maintenant le plus long chemin entre α et ω , on utilise pour ce faire l'algorithme du chemin critique.

Algorithme du chemin critique

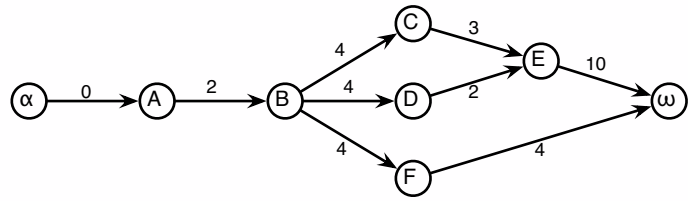
Donnée : réseau trié topologiquement associé à un projet

1. Récurrence en avançant dans le projet
 $\delta_i = 0$
 Pour $k = 2 \dots n$, poser $\delta_k = \max \{ \delta_j + c_{jk} \mid j \in \text{Pred}(k) \}$
2. Récurrence en reculant dans le projet
 $D = \delta_n$ et $\varphi_n = \delta_n$
 Pour $k = n - 1 \dots 1$, poser $\varphi_k = \min \{ \varphi_j - c_{kj} \mid j \in \text{Succ}(k) \}$

Résultat : la durée minimale D du projet et pour chaque tâche i sa date δ_i de début au plus tôt et sa date φ_i de début au plus tard.

Exemple 5.

Tâche	α	A	B	C	D	E	F	ω
n° k	1	2	3	4	5	6	7	8
durée d_k	0	2	4	3	2	10	4	0
$\text{Pred}(k)$	-	α	A	B	B	C,D	B	E,F
$\text{Succ}(k)$	A	B	C,D,F	E	E	ω	ω	-
$\delta_k \rightarrow$	0	0	2	6	6	9	6	19
$\varphi_k \leftarrow$	0	0	2	6	7	9	15	19



- la tâche est critique si $\delta_i = \varphi_i$ un retard dans la tâche implique un retard du projet
- un chemin critique va de α à ω en passant uniquement par des tâches critiques
- la longueur du chemin critique est la durée minimale du projet

2.4 Problème du transbordement

2.4.1 Définitions

- Soit un graphe orienté connexe $G = (V, E)$. Soit b une pondération des sommets représentant les offres et les demandes : offre (sommet source) $\Rightarrow -$ et demande (sommet puits) $\Rightarrow +$. Soit c une pondération des arcs représentant les coûts unitaires d'utilisation des arcs.
 La solution basique linéaire du problèmes est l'arbre maximal du réseau.
- Un graphe $G = (V, E)$ est biparti s'il existe une partition de ses sommets en deux ensembles V_1 et V_2 telle que toute arête (i, j) de E ait une extrémité dans V_1 et l'autre dans V_2 .
- Lorsque G est un graphe biparti complet, chaque sommet source étant relié à tous les sommets puits, on parle de problème de transport.
- Si, dans un problème de transport, il y a autant de sommets source que de sommets puits et que toutes les offres et les demandes sont égales à 1, on parle de problème d'affectation.

2.4.2 Algorithme

Donnée: un réseau connexe $R(V, E, b, c)$

1. Construction du problème auxiliaire et de sa solution initiale admissible
2. Résolution du problème auxiliaire à l'aide de l'algorithme graphique du Simplexe
3. Les arcs, et seulement les arcs (les pondérations restent inchangées), de la solution optimale du problème auxiliaire forment la solution-arbre admissible pour le problème initial
4. Résolution du problème initial à l'aide de l'algorithme graphique du Simplexe

Résultat: solution optimale pour le réseau

2.4.3 L'algorithme graphique du Simplexe dans les réseaux

Donnée: réseau connexe avec une solution-arbre admissible T

1. Calcul des solutions primale x et duale y associées à T
2. Recherche d'un arc entrant, s'il n'en existe pas \Rightarrow STOP : solution optimale
3. Recherche d'un arc sortant, s'il n'en existe pas \Rightarrow STOP : pas d'optimum fini
4. Mise à jour de la solution-arbre et retour à 1.

Résultat : un flot de coût total minimum ou la preuve qu'un tel flot n'existe pas

2.4.4 Construction du problème auxiliaire et de sa solution initiale admissible

Donnée: un réseau $R = (V, E, b, c)$ connexe

1. Poser $c'_{ij} = 0 \forall (i, j) \in E$
2. Choisir un sommet source k .
3. Relier chaque sommet source $i (i \neq k)$ à k par un arc artificiel (i, k) de poids $c'_{ik} = 1$ (s'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 $c'_{kj} = 1$ (s'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).

Résultat: un réseau $R' = (V, E', b, c)$ et une solution-arbre $T' = (V, E'_T)$ admissible pour R'

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

Donnée: réseau $R = (V, E, b, c)$ connexe et une solution-arbre T

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
2. Il reste un seul arc dans E_T , disons (i, j) , poser $x_{ij} = b_j$

Résultat : le flot $x: E \rightarrow \mathbb{R}_+$ associé à T et $z = \sum c_{ij}x_{ij}$

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

Données : réseau $R = (V, E, b, c)$ connexe et une solution-arbre T

1. Choisir arbitrairement $i \in V$, poser $y_i = 0$ et poser $W = V \setminus \{i\}$
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 $y_j = y_i - c_{ij}$
 - c) Retirer j de W

Résultats : les prix duaux $y: V \rightarrow \mathbb{R}$ associés à T et $w = \sum b_i y_i$

2.4.7 Recherche d'un arc entrant

Donnée : solution-arbre admissible $T = (V, E_T)$ ainsi que les solutions x et y associées.

1. On passe en revue les arcs $(i, j) \notin E_T$ (hors base) et on teste pour chacun d'eux si la contrainte duale associée $y_j - y_i \leq c_{ij}$ est satisfaite ou non.
2. Dès qu'une contrainte violée est trouvée : STOP, l'arc $e = (i, j)$ en question rentre dans la base.
3. Si toutes les contraintes sont satisfaites, les solutions actuelles sont optimales.

2.4.8 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 deux ensembles disjoints C^+ (même orientation que (i, j)) et C^- .

1. Si $C^- = \emptyset$: STOP, pas d'optimum fini
2. 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.

2.4.9 Mise à jour des solutions

1. La nouvelle solution-arbre est donnée par $E_T = E_T \cup \{e\} \setminus \{s\}$
2. La solution basique primale ne change que sur les arcs de C :

$$x_{ij} = \begin{cases} x_{ij} + \Delta & \text{si } (i, j) \in C^+ \\ x_{ij} - \Delta & \text{si } (i, j) \in C^- \\ x_{ij} & \text{si } (i, j) \notin C \end{cases}$$

3. La solution basique duale ne change que de la manière suivante.

$$y_d = y_d + \varepsilon$$

où d est le sommet commun entre les arcs e et s , et

$$\varepsilon = \begin{cases} -y_d + y_a + c_{ad} & \text{si } e = (a, d) \text{ et } s = (b, d) \\ y_a - y_d - c_{da} & \text{si } e = (d, a) \text{ et } s = (d, b) \end{cases}$$

2.4.10 Dégénérescence

Si la solution basique primale est dégénérée, c-à-d s'il existe des arcs dans E_T le long desquels des quantités nulles sont transportées, il y a risque de cyclage. Pour éviter cela, recourons à la règle de Bland graphique :

- Tester les arcs dans l'ordre lexicographique et faire entrer le premier arc dont la contrainte duale associée est violée.
- Si la quantité Δ est transportée le long de plusieurs arcs de C^- , faire sortir l'arc le plus petit dans l'ordre lexicographique.

3 Chaînes de Markov

3.1 Définitions

- Un processus stochastique est une collection de variables aléatoires $\{X_t, t \in T\}$ avec X_t = état du processus au temps t ($X_t \in S$).
- Un processus stochastique dont l'ensemble des états S est fini ou dénombrable est appelé une chaîne.
- Un processus est à temps continu lorsque l'ensemble T est non dénombrable : $T = \mathbb{R}_+$
- Un processus est à temps discret lorsque l'ensemble T est fini ou dénombrable : $T = \mathbb{Z}_+$
- Un processus est markovien si, pour tout instant t , l'état courant X_t résume, à lui seul, tout l'historique du système susceptible d'influencer son évolution future.
- Une évolution du système est appelée trajectoire.

3.2 Chaînes de Markov à temps discret

3.2.1 Matrice de transition

La matrice de transition P peut être représentée par un graphe représentatif G .

P est stochastique $\Leftrightarrow p_{ij} \geq 0 \ \forall i, j$; $\sum_j p_{ij} = 1 \ \forall i$

Probabilité de transition de i à j en m étapes est donnée par l'élément ij de la matrice P élevée à la puissance m : $(P^m)_{ij}$ avec $P^0 = I$ et $P^1 = P$.

3.2.2 Classification

1. Déterminer les classes en calculant les composantes fortement connexes du graphe G de la chaîne.
2. Construire le graphe réduit G_R formées des classes de G et en déduire la classification des classes et des états.

Une **classe** est:

- **persistante** si elle n'a pas de sommet successeur
- **transitoire** si elle a au moins un sommet successeur
- **absorbante** si elle est persistante et composée d'un seul état

Un **état** est:

- **persistant** s'il appartient à une classe persistante
- **transitoire** s'il appartient à une classe transitoire
- **absorbant** s'il appartient à une classe absorbante ($p_{ii} = 1$)

3. Classifier la chaîne :

- Elle est **irréductible** si elle n'a qu'une seule classe, sinon elle est réductible
- Elle est **absorbante** si tous ses états persistants sont absorbants
- Elle est **ergodique** si elle est irréductible et apériodique

3.2.3 Étude des classes persistantes

Pour chaque classe persistante

1. Déterminer sa **période** :

La période est le plus grand commun diviseur des longueurs du circuit (nb d'arcs)

La période $d > 1 \Rightarrow$ la classe est de période d . Si $d = 1$ la classe est apériodique

2. Calculer sa **distribution stationnaire** :

$$\vec{\pi} = (\pi_1 \quad \dots \quad \pi_n) \quad \left\{ \begin{array}{l} \vec{\pi} P = \vec{\pi} \\ \vec{\pi} \vec{1} = 1 \end{array} \right.$$

P est ici la matrice de transition de la chaîne où les composantes qui ne se rapportent pas à la classe ont été annulées.

Cette distribution est unique si la chaîne est ergodique.

3.2.4 Étude des états transitoires

1. Contracter les classes persistantes afin d'obtenir une chaîne absorbante (facultatif)
2. Mettre la matrice de transition de la chaîne sous **forme canonique**:
Réordonner les sommets dans la matrice :
 - les sommets persistants sont numérotés en premier.
 - les sommets d'une classe sont numérotés consécutivement.

Si la chaîne à k classe persistante, la matrice sous forme canonique à la forme suivante :

$$P' = \left(\begin{array}{ccc|c} P_1 & & 0 & 0 \\ & \ddots & & \vdots \\ 0 & & P_k & 0 \\ \hline R_1 & \dots & R_k & Q \end{array} \right) \text{ si la chaîne est absorbante : } P' = \left(\begin{array}{c|c} I & 0 \\ \hline R & Q \end{array} \right)$$

3. Calculer la **matrice fondamentale** $N = (I - Q)^{-1}$ et la **matrice des probabilités d'absorption** $B = NR$.

3.2.5 Interprétation

- La probabilité d'être dans l'état i est donnée par π_i .
- Le nombre moyen de transitions entre 2 visites successives de l'état i est donné par $\frac{1}{\pi_i}$.
- Le nombre moyen de périodes séjourné dans l'état transitoire j partant de l'état transitoire i , avant absorption, est l'élément n_{ij} de la matrice N .
- Le nombre moyen de transitions avant d'atteindre un état absorbant en partant de l'état transitoire i est la somme des éléments de la i^e ligne de N .
- La probabilité d'être absorbé par l'état j en partant de l'état transitoire i est donnée par l'élément b_{ij} de la matrice de B .

3.3 Chaînes de Markov à temps continu

3.3.1 Probabilités et matrices de transition

Les probabilités de transition au temps t :

$$p_{ij}(t) = P[X_t = j | X_0 = i]$$

La matrice de transition au temps t :

$$P(t) = (p_{ij}(t))$$

On suppose toujours $P(0) = I$

3.3.2 Classification

La classification est la même que pour une chaîne de Markov à temps discret augmentée des définitions suivantes.

Classe et état

Dans les chaînes à temps continu, les classes et états persistants sont appelés **récurrents**.

Chaîne

Elle est **régulière** ssi le nombre de transition qu'elle effectue dans un intervalle de temps fini est fini. Toute chaîne ayant un nombre fini d'état est régulière.

Une chaîne est irréductible si tous ses états communiquent deux à deux.

Une chaîne ergodique est aussi appelée **récurrente non nulle**.

3.3.3 Structure

- Le temps de séjour τ_i dans un état i est une variable aléatoire exponentielle de paramètre λ_i ne dépendant que de i .
- La probabilité de passage quittant l'état i pour aller en j , notée q_{ij} est indépendante de t et de τ_i .
- La suite des états visités par un chaîne de Markov à temps continu forme une chaîne de Markov à temps discret, appelée chaîne induite ou sous-jacente et donnée par sa matrice de transition $Q = q_{ij}$ avec $q_{ii} = 0$.

3.3.4 Loi exponentielle

La loi exponentielle est la seule loi continue sans mémoire.

Soit X une variable aléatoire exponentielle de paramètre λ ($\lambda > 0$) :

$$P[X \leq x] = F(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

La **densité** de X est $f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$

L'**espérance** de X est $E[X] = \frac{1}{\lambda}$ et sa **variance** $\text{Var}[X] = \frac{1}{\lambda^2}$

i est un état absorbant ssi i est une variable aléatoire exponentielle de paramètre $\lambda_i = 0$. Si ce paramètre égal à l'infini alors la variable ne prend qu'une valeur: 0, c'est un état dont on sort tout de suite

$P[X \leq x]$ donne la **probabilité de sortir de l'état** pour $t \leq x$, et $E[X]$ donne le **temps moyen de séjour**

3.3.5 Matrice génératrice

Soit la matrice A (matrice d'intensité):

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

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

3.3.6 Matrice de transition de la chaîne sous-jacente

Le matrice de transition Q de la chaîne de Markov à temps discret sous-jacente peut être retrouvée à l'aide de la matrice génératrice à l'aide des règles suivantes:

$$q_{ij} = \begin{cases} -\frac{a_{ij}}{a_{ii}} & \text{si } a_{ii} < 0 \text{ et } j \neq i \\ 0 & \text{si } a_{ii} = 0 \text{ et } j \neq i \\ 0 & \text{si } a_{ii} < 0 \text{ et } j = i \\ 1 & \text{si } a_{ii} = 0 \text{ et } j = i \end{cases}$$

3.3.7 Équations de Kolmogorov

- Équation du futur : $P'(t) = AP(t)$
- Équation du passé : $P'(t) = P(t)A$

3.3.8 Distribution

Distribution initiale : $\vec{\pi}(0)$

Probabilité d'observer le processus dans l'état i au temps t : $\vec{\pi}(t) = \vec{\pi}(0)P(t)$

Distribution stationnaire :

$$\begin{cases} \vec{\pi}^T A = \vec{0}^T \\ \vec{\pi}^T \vec{1} = 1 \end{cases}$$

Ce système a une solution unique si la chaîne est ergodique (récurrente non nulle).

Taux de transition hors de l'état i : $\pi_i a_{ii}$

Taux de transition dans l'état i : $\sum_{j \neq i} \pi_j a_{ji}$

Proportion de temps passé dans l'état i : π_i

Espérance du temps entre deux visites $\frac{1}{\pi_i a_{ii}}$

3.3.9 Processus de naissance et de mort

Depuis i , les transitions se font que vers $i - 1$ (mort) ou $i + 1$ (naissance).

Taux de naissance : λ_i

Taux de mort : μ_i

Le processus reste dans l'état i pendant une durée aléatoire exponentielle de paramètre $\alpha_i = \lambda_i + \mu_i$.

Lorsqu'il quitte l'état i , il se retrouve dans l'état $i - 1$ ou $i + 1$ avec une probabilité

$$q_{i,i-1} = \frac{\mu_i}{\lambda_i + \mu_i} \qquad q_{i,i+1} = \frac{\lambda_i}{\lambda_i + \mu_i}$$

Processus de Poisson

C'est un processus de naissance pur à taux constant : $\lambda_i = \lambda$ et $\mu_i = 0 \quad \forall i$.

3.3.10 Files d'attentes

Ajouts

- Le temps moyen d'exécution d'une tâche régie par une variable aléatoire est l'espérance de cette dernière.
- Dans M/G/1 : $C_S^2 = \frac{\text{Var}[S]}{E[S]^2} = \mu^2 \text{Var}[S]$
- Espérances et variances

$$\begin{array}{lll} X \sim U(a, b) & E[X] = \frac{a+b}{2} & \sigma_X^2 = \frac{(a+b)^2}{12} \\ Y \sim \varepsilon(\lambda) & E[Y] = \frac{1}{\lambda} & \sigma_Y^2 = \frac{1}{\lambda^2} \end{array}$$

4 Programmation dynamique

4.1 Modélisation

Système dynamique

- Étapes N
- État x_k
- Ensemble S_k des états à l'étape k
- État initial x_1
- État terminal x_{N+1}
- Décision u_k
- Ensemble C_k des décisions possibles à l'étape k
- Ensemble $U_k(x_k)$ des décisions admissibles dans l'état x_k à l'étape k
- fonction de transfert

Fonction coût

- fonction de coût $g_k(x_k, u_k, \omega_k)$
- coût terminal $g_{N+1}(x_{N+1})$
- coûts totaux: $g_{N+1}(x_{N+1}) + \sum_{k=1}^N g_k(x_k, u_k, \omega_k)$

4.2 Résolution

$$J_k(x_k) = \max_{u_k \in U_k(x_k)} (g_k(x_k, u_k)) + J_{N+1}(x_{k+1})$$

x_k	$J_k(x_k, 0)$	$J_k(x_k, 1)$	$J_k(x_k)$	$\mu_k^*(x_k) = u_k^*$
0	0	—	0	0
1	0	—	0	0
2	0	—	0	0
3	0	—	0	0
4	0	—	0	0
5	0	15	15	1

Est le tableau de base pour la résolution des problèmes de programmation dynamique. La 1ère colonne contient l'état possible au moment k . Les colonnes suivantes contiennent le coût en fonction de la décision prise ; l'avant dernière colonne contient le coût maximum que l'on peut obtenir en k ; la dernière colonne donne le nombre d'objets choisis pour avoir un coût maximum.

4.3 Algorithme de Wagner Whitin

Soient d_t la demande, K_t les coûts fixes, c_t les coûts unitaires de production et h_t ceux de stockage durant la période t .

- calcul des c_{ij} , le coût engendré par une production à la période i de la quantité nécessaire à satisfaire la demande jusqu'à la période j :

$$c_{ij} = K_i + c_i \sum_{t=i}^j d_t + \sum_{t=i}^{j-1} h_t \sum_{k=t+1}^j d_k$$

- calcul du coût optimal pour les périodes i à N via les équation de récurrence suivantes:

$$\begin{aligned} J_i &= \min_{i \leq j \leq N} (c_{ij} + J_{j+1}) \quad i = N, \dots, 1 \\ J_{N+1} &= 0 \end{aligned}$$

Le coût optimal est donné par J_1

Exemple 6. Soient les données et les c_{ij}

t	d_t	K_t	c_t	h_t	$i \setminus j$	1	2	3	4
1	4	40	9	3	1	76	220	490	666
2	12	20	7	6	2	—	104	299	435
3	15	30	5	4	3	—	—	105	177
4	8	50	4	—	4	—	—	—	82

Les tables optimales sont alors

$$J_4 = 82 + 0 = 82 \quad J_3 = \min \begin{cases} 105 + 82 = 187 \\ 177 + 0 = 177 \end{cases}$$

$$J_2 = \min \begin{cases} 104 + 177 = 281 \\ 299 + 82 = 381 \\ 435 + 0 = 435 \end{cases} \quad J_1 = \min \begin{cases} 76 + 281 = 357 \\ 220 + 177 = 397 \\ 490 + 82 = 572 \\ 666 + 0 = 666 \end{cases}$$

On a alors

i	4	3	2	1
J_i	82	177	281	357
$\mu^*(i)$	4	4	2	1

Et le plan optimal est

$$\begin{aligned} x_1 &= d_1 = 4 \\ x_2 &= d_2 = 12 \\ x_3 &= d_3 + d_4 = 23 \\ x_4 &= 0 \end{aligned}$$