Optimisation et Recherche Operationnelle

Zakaria EJJED

Contents

Thursday January 26 2023 Course

Optimisation:

$\rightarrow$ on cherhce le point $x*$ où l'optimisation (minimum ou maximum selon le cas) de la fonction f est réalisé.

Recherche opérationnelle:

$\rightarrow$ on se trouve dans un contexte industriel dans lequel on cherche à résoudre un problème concret, problèmes que l'on peut représenter comme des problème d'optimisation.

$\Rightarrow$ <u>modélisation</u>: travail de représentation mathématique/informatique d'un problème concret qui est posé. $\rightarrow$ abstraction/simplification.

Exemple: Gestion de stocks.

- **Entrepot:** Chaque jour, q=100 unités sortant; une unité en stock coûte a=0,16€/jour; le coût d'une passation de commande est b=50€, et le réapprovisionnement à lui automatiquement dès que le stock est vide; une commande ne peux axcéder c=400 unités.
  On cherche le volume optimal O^{R} d'une commande.
  $\rightarrow$ modélisation: on cherche à minimiser le coût de gestion de l'entrepôt.

- **Coût de gestion:**

$$\begin{array}{rlcl} & \text{coût de stock} & = & 0,16€/\text{jour/unité} \\ + & \text{coût de commande} & = & 50€ \end{array}$$

→ On expime tout en euros par jour.

- **Coût de stock:**
  On cherche le nombre moyen d'unités dans le stock: on suppose que le stock se vide à un rythme régulier, et de façon continue.
  //1 Le stock moyen est le stock moyen sur un période $\frac{Q}{2}$.

$$\int_0^T Q(1-\frac{t}{T})dT = \frac{Q}{2}$$

Le coût quotidien de gestion de stock est a.$\frac{Q}{2}$€/j.
coût de commande:
$\frac{Q}{100}$ est le temps (en jour) entre deux commandes: on fait une commande, coûte b=50€, tous les $\frac{Q}{100}$ jours, et le coût quotidient moyen de commande est des:

$$\frac{b}{a/q} = \frac{50}{Q/100} = \frac{5000}{Q}$$

Le coût quotidien moyen de gestion de l'entrepôt est de:

$$\begin{array}{l} f(Q) = 0,08Q + \frac{5000}{Q} \\ D = [0,400] \quad Q \leq 400 \quad Q \geq 0 \\ \min \{f(Q)/Q \in D\} \\ \min \{0,08Q + \frac{5000}{Q}/0 \leq 400\} \end{array}$$

On détermine le table de variation de f:

$$\begin{array}{lcl} \forall Q \in]O,400], & & \\ f'(Q) & = & 0,08 - \frac{5000}{Q^2} \\ f'(Q) & = & 0 \\ & & \\ \Leftrightarrow 0,08 - \frac{5000}{Q^2} & = & 0 \\ \Leftrightarrow \frac{5000}{Q^2} & = & 0,08 \\ \Leftrightarrow \frac{Q^2}{5000} & = & \frac{1}{0,08} \\ \Leftrightarrow Q^2 & = & \frac{5000}{0,08} = \frac{500000}{8} = 62500 \\ \Leftrightarrow Q & = & \sqrt{62500} \simeq 250 \text{car Q>0} \end{array}$$

$$\text{si } \sqrt{\frac{2bq}{Q}}$$

...

//TABLEAU DE VARIATION

La minimisation de f est atteinte à 250: le volume optimal de commande est Q*=250, pour un coût quotidien de f(Q*)=40.

Si on généralise à des a, b, c, q, quelconques:

$$\begin{array}{lcl}
\forall Q \in]O,0], & & \\
f(Q) & = & a \times \frac{Q}{Q^2} + \frac{bq}{Q} \\
f'(Q) & = & \frac{a}{2} - \frac{bq}{Q^2} \\
 & & \\
f'(Q) & = & 0 \\
 & & \\
\Leftrightarrow \frac{a}{2} - \frac{bq}{Q^2} & = & 0 \\
\Leftrightarrow \frac{bq}{Q^2} & = & \frac{a}{2} \\
\Leftrightarrow \frac{Q^2}{5000} & = & \frac{1}{0{,}08} \\
\Leftrightarrow Q & = & sqrt\frac{2bq}{a} \text{ car } Q > 0
\end{array}$$

//TABLEAU DE VARIATIONS

Algorithm 1 Algo Stock opti (a,b,c,q:réels > 0)→Q*:réel>0

if $\sqrt{\frac{2bq}{a}} \leq c$ **then**
  $Q^* \leftarrow \frac{2bq}{a}$
else
  $Q^* \leftarrow c$
end if

I - Problème d'ordonnancement

On cherche à déterminer l'ordre dans le quel effectuer des tâches, définies par leurs durées, et par des relations de précedence: certaines tâches ne peuvent être effectuées qu'après que d'autres aient été accomplies.

On cherche à déterminer le planning permettant de terminer au plus vite le projet.

On a une série de tâche $\{t_i / i = 1, ..., u\}$, A chaque tâche est associée une durée, ainsi qu'une liste de tâches $\{p_{ij} / j = 1, ..., k_i\}$ qui doivent être terminée avant que la tâche t_i ne puisse commencer.

La fonction objective est la durée du projet, que l'on cherche à minimiser; les contraintes sont les contraintes de précendence; les variables de décision décrivent l'ordre dans lequel on effectue les tâches: c'est le planning de ces tâches, c'est à dire la date de début de chacune des tâches.

La durée totale du projet, fonction objective, est la durée entre le début de la premiète tâche ($\rightarrow t = 0$) et la fin de la dernière.

$t_f =$

la tâche t_i se termine au temps a_i(date de début de la tâche)+d_i(durée de la tâche)

Le projet se termine en même temps que la dernière tâche, au temps:
$\max\{\ a_i + d_i / i = 1, ..., u\ \}$.

On cherche donc à minimiser $\max\{\ a_i + d_i / i = 1, ..., u\ \}$.

On a de plus les contraintes suivantes:

$$\begin{array}{lcl} \forall i = 1, ..., u & & \\ a_i & \geq & 0 \\ \forall i = 1, ..., u & & \forall k \in P_i, \\ a_k + d_k a & \leq & a a_i \end{array}$$

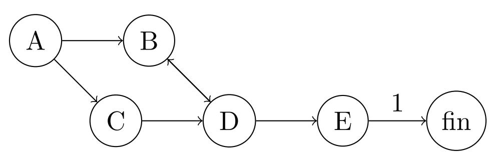
Au final, le problème consiste à trouver des a_i réalisant:

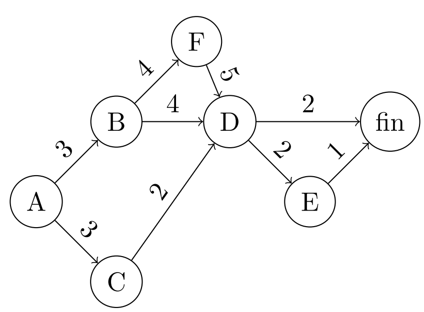
$\min\{\max\{a_i + d_i / i = 1, ..., u\} / \ \forall i, a_i \geq 0; \quad \forall k \in P_i,\ a_i \geq a_k + d_k\}$ contraintes

On peut également modéliser ce problème sous la forme de graphes. Dans la méthode des **potentiels**, chaque tâche est représentée par un sommet et les relations de précédences par des arcs. Un arc joint j à i si $j \in P_i$

$$j \xrightarrow{d_j} i$$

A chaque arc, on associe la durée de la tâche dont il est issu. La longueur d'un chemin représente la durée de tâches qui doivent être executés consécutivement: la durée du projet est donc plus grande que la longueure et n'importe quel chemin dans ce graphe.





Recherche d'un plus long chemin dans un graphe (problème de maximisation): max$\{f\}$ = -min$\{-f\}$

//COURBE

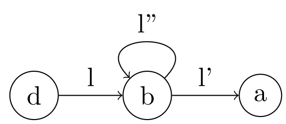
L'algo de Dijkstra ne peut s'appliquer car il nécessite des valuations positives.
$\Rightarrow$ algo de **Bellman-Ford**.

La durée totale du projet est donc minorée par la longueure du plus long chemin dans ce graphe; en fait, on peut la rendre égale à cette longueuer.

$\rightarrow$ recherche d'un plus long chemin dans un graphe.

$\hookrightarrow$ revient à chercher le plus court chemin dans ce graphe dans lequel on a remplacé chaque valuation par son opposé.

Le problème du plus court chemin n'a pas de solution en présence d'un cycle de poids <0;



$$l + l' > l + l' + l'' > l + l' + 2l'' > ... > l + l' + kl'' \quad \forall k$$

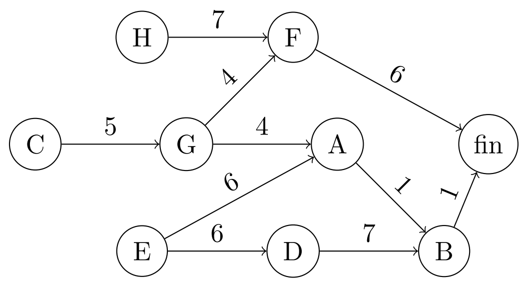
$\rightarrow$ pas de plus court chemin, parce qu'une fois qu'on en a un, on peut toujours en construire un encore plus court.

DAG: directed acyclic graph, graphe orienté acyclique.

$\Rightarrow$ algo de **Ford**
On prend les noeuds dans un ordre garantissant qu'in noeud est traité après tous ses prédécesseurs (cet ordre existe sinon le graphe contiendrait un cycle) traiter les noeud i consiste à fixer sa date à $max\{a_j + d_j / j \in P\}$

Par exemple:

Tâche	Description	Durée	Précédence
A	teinture	1	E,G
B	confection	1	A,D
C	étude de marché	5	-
D	commande fermeture éclaire	7	E
E	commande tissu	6	-
F	impression catalogue	6	G,H
G	choix coloris	4	C
H	contact imprimeur	7	-



Graphe des potentiels

$\max\{a_j + d_j / j \ inP\}$

$\searrow$

Tâche	Date au plus tôt	Date au plus tard	Marge totale	Marge libre
C	0	min{5-5}=0	0	0
E	0	min{13-6,7-6}=1	1	0
H	0	min{5-7}=2	2	2
D	max{0+6}=6	min{14-7}=7	1	0
G	max{0+5}=5	min{13-4,9-4}=5	0	0
A	max{5+4,0+6}=9	min{14-1}=13	4	3
F	max{5+4,0+7}=9	min{15-6}=9	0	0
B	max{9+1,6+7}=13	min{15-1}=14	1	1
fin	max{5+6,13+2}=15	=15	0	0

$\downarrow$

Planning optimal

Les tâches C,G et F doivent être exécutées consécutivement, et elles prennent à elles 3 15j: le projet ne peut pas durer moins, et comme le planning trouvé prend 15j, il est optimal.

On appelle date au plus tard la date avant laquelle une tâche doit commencer au risque de mettre tout le projet en retard.

Une tâche ayant une marge totale de 0 est dite critique: il existe un chemin partant d'un sommet sans prédécesseur et allant au noeud “***fin***” en ne passant que par des tâches critiques.

La marge libre est le retard que peut prendre une tâche tout en permettant aux tâches suivantes de commencer à leur date au plus tôt (=conserver leur marge).

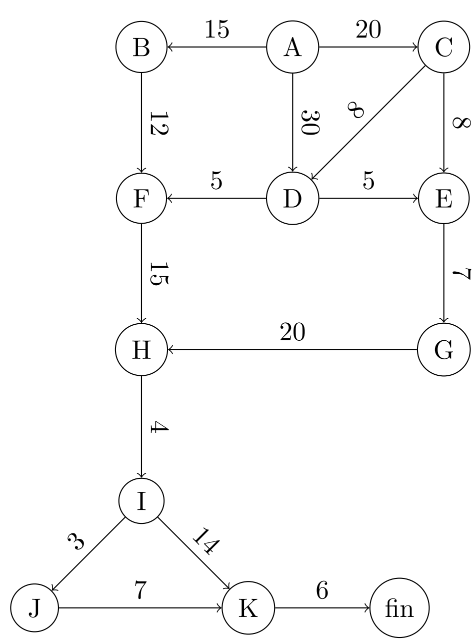
Quelques formules:

$$\begin{aligned}
\text{tôt}_i &= \max\{\text{tôt}_j + d_j / j \in P_i\} \\
&\quad (\text{avec } \max \varnothing = 0) \\
\text{tard}_i &= \min\{\text{tard}_j - d_i / i \in P_j\} \\
\text{marge}_i &= \text{tard}_i - \text{tôt}_i \\
\text{libre}_i &= \min\{\text{tôt}_j - d_i - \text{tôt}_i / i \in P_j\}
\end{aligned}$$

Thursday January 26 2023 Exercises

Exercice 1

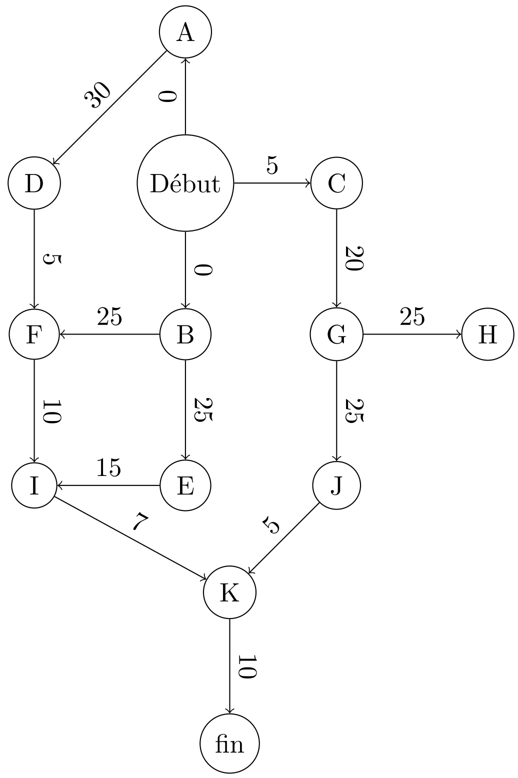
Tâche	Durée	Antériorité
A	30	-
B	12	A+15j
C	8	A+20j
D	5	A et C
E	7	C et D
F	15	B et D
G	20	E
H	4	F et G
I	14	H
J	7	I+3j
K	6	I et J



Tâche	Date au plus tôt	Date au plus tard	Marge totale	Marge libre
A	0	0	0	0
B	15	35	20	8
C	20	22	2	2
D	30	30	0	0
E	35	35	0	0
F	35	47	12	12
G	42	42	0	0
H	62	62	0	0
I	66	66	0	0
J	69	73	4	0
K	80	80	0	0
fin	86	86	0	0

Exercice 2

Tâche	Durée	Antériorité
A	30	-
B	25	-
C	20	5j après début
D	5	A
E	15	B
F	10	B,D
G	25	C
H	12	G
I	7	E,F
J	5	G
K	10	I,J



Tâche	Date au plus tôt	Date au plus tard	Marge totale	Marge libre
Début	0	0	0	0
A	0	0	0	0
B	0	0	0	0
C	5	5	0	0
D	30	33	3	0
E	25	33	8	5
F	35	38	3	0
G	25	25	0	0
H	50	50	0	0
I	45	48	3	3
J	50	50	0	0
K	55	55	0	0
fin	65	65	0	0

II - Programmation linéaire

Algo du simplexe

III - Séparation et Evaluation