Recherche opérationelle
Graphes – Réseaux – Flots

Siarry Patrick

ING1 2014

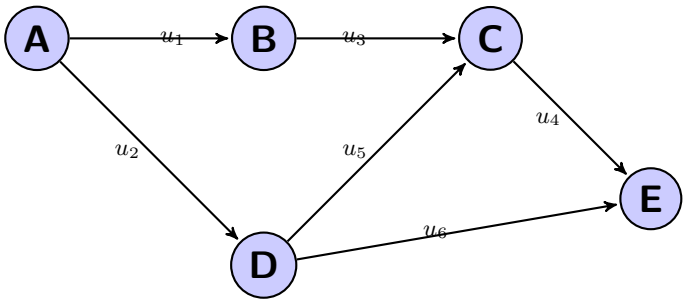
Sources disponibles sur http://ing1.nemunai.re/ ou ing1@nemunai.re

Table des matières

Chapitre 1

Connectivité dans un graphe

1.1 Algorithme de Roy-Warshall



Exercice page 101

Soit M la *matrice binaire* associée à un graphe.

$$M = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Représenter le graphe associé.

M est la matrice $n \times n$ avec n le nombre de sommets (ou nœuds) du graphe.

Ici, on a 5 sommets : A, B, C, D et E.

$$m_{i,j} = 1 \Longleftrightarrow \exists \text{ un arc } (i, j)$$

Donner la matrice d'incidence de ce graphe.

Soit la matrice $A : n \times m$, avec n le nombre de *sommets* du graphe et m le nombre *d'arcs* du graphe.

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{pmatrix} +1 & +1 & & & & \\ -1 & & +1 & & & \\ & & & +1 & -1 & \\ & -1 & & & +1 & +1 \\ & & & -1 & & -1 \end{pmatrix} \end{array}$$

$a_{i,j} = +1$ si i est à l'origine de l'arc j, $a_{i,j} = -1$ si i est à l'extrêmité terminale de j, $a_{i,j} = 0$ sinon.

Calculer M^2, M^3, M^4, ...et interpréter les termes non nuls de ces matrices.

$$M^2 = \begin{bmatrix} 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$a_{3,1} = 2$, cela signifie qu'il existe deux chemins de longueur 2 reliant A à C : ABC et ADC. $a_{5,1} = 2$, cela signifie qu'il existe un chemin de longueur 2 reliant A à E : ADE.

$$M^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad M^4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = M^n \quad \forall n \geq 4$$

Grâce à M^3, on voit qu'il n'existe que deux chemins de longueur trois reliant A à E : $ABCE$ et $ADCE$.

Il n'existe pas de chemin plus grand que 3, car on a une matrice nulle pour M^4.

Calculer les puissances booléennes de M, soit $M^{[2]}$, $M^{[3]}$, $M^{[4]}$, ...

$$M^{[2]} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad M^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dans ce cas, on dit qu'il existe au moins un chemin de longueur deux reliant les points où l'intersection dans la matrice n'est pas nulle.
On a perdu l'information du nombre de chemin, on sait seulement si un chemin existe ou non.

Calculer la fermeture réflexo-transitive du graphe.

$$B = I \dot{+} M \dot{+} M^{[2]} \dot{+} \dots$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ la matrice identitée}$$

En clair : B correspond en fait à un OU logique appliqué à l'ensemble des matrices.

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

Pour chacun des éléments non nuls de la matrice B, on peut dire qu'il existe au moin un chemin (de longueur indéterminée). La matrice B permet donc de savoir si deux sommets sont interconnectés dans le graphe.

Algorithme de Roy-Warshall.

L'algorithme de Roy-Warshall permet de calculer B sans calculer les produits matriciels.

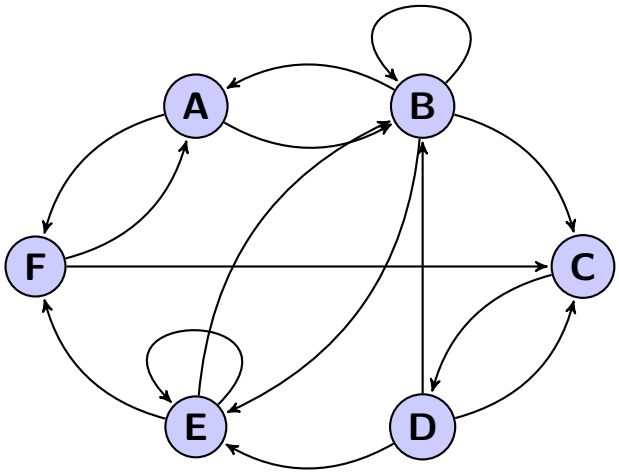
1.1.1 Étapes

- On calcul $B = I \dot{+} M$;
- On calcul l'image de B par l'opérateur θ_1, soit $B = \theta_1(B)$;
- On calcul l'image de B par l'opérateur θ_2 ;
- Ainsi de suite, jusqu'à θ_n.

1.1.2 Opérateur θ_i

- On considère toutes les lignes de B qui ont un 1 en colonne i ;
- On ajoute (addition booléenne $\dot{+}$) à chacune de ces lignes tous les 1 de la ligne i ;

$$B = \theta_1(B) = \begin{bmatrix} 1 & \mathbf{1} & 0 & 1 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{0} & 1 & 0 & 1 \\ 0 & \mathbf{0} & 1 & 1 & 1 \\ 0 & \mathbf{0} & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\theta_2} \begin{bmatrix} 1 & 1 & \mathbf{1} & 1 & 0 \\ 0 & 1 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 0 & \mathbf{1} & 1 & 1 \\ 0 & 0 & \mathbf{0} & 0 & 1 \end{bmatrix} \xrightarrow{\theta_3} \begin{bmatrix} 1 & 1 & 1 & 1 & \mathbf{1} \\ 0 & 1 & 1 & 0 & \mathbf{1} \\ 0 & 0 & 1 & 0 & \mathbf{1} \\ 0 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$



Exercice 2 page 101

Calculer les demi-degrés extérieurs et intérieurs des sommets du graphe

$\Gamma^+(A), \Gamma^+(B), \ldots$; $\Gamma^+(A) = B, F$, $\Gamma^+(B) = A, B, C, E, \ldots$
$\Gamma^-(A), \Gamma^-(B), \ldots$; $\Gamma^-(A) = B, F$, $\Gamma^+(B) = A, B, C, E, \ldots$

Γ^+ : la liste des descendants immédiats de i et Γ^+ : la liste des prédécesseurs immédiats de i.

Le demi-degré intérieur du sommet $i \leftarrow d'(i)$: le nombre de prédécesseurs immédiats de i, à l'exception de i lui-même.
Le demi-degré extérieur du sommet $i \leftarrow d''(i)$: le nombre de successeurs immédiats de i, à ;'exception de i lui-même.

$\Gamma^+(A) = B, F : d''(A) = 2$
$\Gamma^+(B) = A, B, D, E : d''(B) = 3$

Donner un exemple de chemin simple mais non élémentaire.

Un **chemin est simple** s'il ne passe qu'une fois par l'ensemble de ces arcs. Exemple : $ABCDBE$.

Un **chemin est élémentaire** s'il ne passe qu'une fois par l'ensemble de ces sommets.

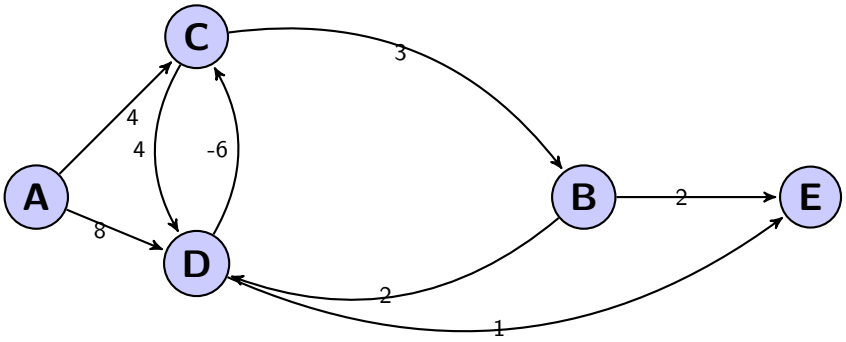
Existe-il un circuit Hamiltonien dans ce graphe ?

Un **circuit** est un chemin qui se ferme sur lui-même.

Un **circuit hamiltonien** est un circuit passant par tous les sommets du graphe.

Problème du voyageur de commerce Le voyageur de commerce visite une fois et une seul chaque ville, puis retourne à la ville de départ.

1.2 Recherche de plus court chemin optimal



1.2.1 Méthode matricielle

Pas de sommet source.

Cette méthode donne en une seule exécution le chemin le plus court ou le chemin le plus long entre deux sommets.

On cherche le chemin de valeur maximale reliant deux sommets.

Initialisation

On considère deux matrices D_0 et Π_0, de dimension $n \times n$, avec n le nombre de sommets (ici 5).

$$D_0 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} -\infty & -\infty & 4 & 8 & -\infty \\ -\infty & -\infty & -\infty & 2 & 2 \\ -\infty & 3 & -\infty & 4 & -\infty \\ -\infty & -\infty & -6 & -\infty & 1 \\ -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix} \end{array} \qquad \Pi_0 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \end{array}$$

$(D_0)_{i,j} = v_{i,j}$ = la valeur de l'arc (i,j) si cet arc est présent
sinon $(D_0)_{i,j} = -\infty$
$(\Pi_{ij}) = i$, $\forall j$.

À chaque étape $k \geq 1$

On calcule ma matrice D_k obtenue en replaçant $(D_{k-1})_{i,j}$ par $\max\{(D_{k-1})_{i,j}, x\}$, x correspond à la somme du k-ième terme de la ligne et du k-ième terme de la colonne où se trouve $(D_k)_{i,j}$.

On itére pour $k = 1, 2, \ldots, n$. Ici $n = 5 \longmapsto$ on lit le résultat cherché par l'intermédiaire des matrices (D_n, Π_n).

$$D_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} -\infty & -\infty & 4 & 8 & -\infty \\ -\infty & -\infty & -\infty & 2 & 2 \\ -\infty & 3 & -\infty & \mathbf{4} & -\infty \\ -\infty & -\infty & -6 & -\infty & 1 \\ -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix} \end{array} \qquad \Pi_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & \mathbf{3} & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \end{array}$$

$\max\{4, -\infty + 8\}$. La première colonne étant composée de $-\infty$, les deux matrices sont identiques. On a $D_0 \equiv D_1$ et $\Pi_0 \equiv \Pi_1$.

$$D_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} -\infty & -\infty & 4 & 8 & -\infty \\ -\infty & -\infty & -\infty & 2 & 2 \\ -\infty & 3 & -\infty & \mathbf{5} & \mathbf{5} \\ -\infty & -\infty & -6 & -\infty & 1 \\ -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix} \end{array} \qquad \Pi_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & \mathbf{2} & \mathbf{2} \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \end{array}$$

Dans Π_2 : $(\Pi_2)_{i,j} = (\Pi)_{2,j}$

$$D_3 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} -\infty & \mathbf{7} & 4 & \mathbf{9} & \mathbf{9} \\ -\infty & -\infty & -\infty & 2 & 2 \\ -\infty & 3 & -\infty & 5 & 5 \\ -\infty & \mathbf{-3} & -6 & \mathbf{-1} & 1 \\ -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix} \end{array} \qquad \Pi_3 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 1 & \mathbf{3} & 1 & \mathbf{2} & \mathbf{2} \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 2 & 2 \\ 4 & \mathbf{3} & 4 & \mathbf{2} & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \end{array}$$

$$D_4 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} -\infty & 7 & 4 & 9 & \mathbf{10} \\ -\infty & \mathbf{-1} & \mathbf{-4} & 2 & \mathbf{3} \\ -\infty & 3 & \mathbf{-1} & 5 & \mathbf{6} \\ -\infty & -3 & -6 & -1 & 1 \\ -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix} \end{array} \qquad \Pi_4 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 1 & 3 & 1 & 2 & \mathbf{4} \\ 2 & \mathbf{3} & \mathbf{4} & 2 & \mathbf{4} \\ 3 & 3 & \mathbf{4} & 2 & \mathbf{4} \\ 4 & 3 & 4 & 2 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \end{array}$$

La dernière ligne étant composée de $= -\infty$, $D_5 \equiv D_4$ et $\Pi_5 \equiv \Pi_4$.

Lecture du résultat via D_5 et Π_5

On lit dans la case $(D_n)_{i,j}$ la valeur du chemin le plus long reliant i à j. Exemple : le chemin le plus long allant de A à E est de longueur 10.

$(\Pi_n)_{i,j}$ est le prédécesseur immédiat de j, soit j'.
$(\Pi_n)_{i,j'}$ est le prédécesseur immédiat de j', soit j'', ...

Remarque Le même algorithme procure également les chemins les plus courts en appliquant deux changements :

- **Initialisation :** on initialise la matrice à $+\infty$ au lieu de $-\infty$.
- **À l'itération k :** on remplace l'opération max par min.

Chapitre 2

Problèmes d'ordonnancement de projets

Le but est de trouver l'ordonnancement au plus tôt de toutes les tâches.

On représente généralement ce genre de problème sous la forme de diagrammes de Gantt calés à gauche.

Le deuxième objectif est de déterminer le chemin critique : tout retard sur une tâche critique se représente sur la durée totale du projet.

Pour les tâches non-critiques, on peut calculer la marge totale et la marge libre.

Il existe deux méthodes principales pour calculer les crontraites de type *potentiel* : la méthode tâche et la méthode PERT (qui ne sera pas traitée dans ce cours).

Les seules contraintes sont du type : « la tâche B ne peut démarrer que 3 jours après la tâche A », contraintes de succession.

Remarque les contraintes les plus dures ne sont pas prises en compte : on parle de type cumulatif.

Lorsque les ressources sont limités, les interactions sont très fortes entre les tâches.

2.1 Méthode portentiels-tâches

Exemple On a un problème à 7 tâches (a, b, c, d, e, f et g) ; les contraintes sont de type portentiel seulement.

Soit le tableau des contraintes suivantes :

Tâches	**Durées**	**Contraintes**
a	6	
b	3	
c	6	
d	2	b achevée
e	4	b achevée
f	3	a et d achevées
g	1	c, e et f achevées

2.1.1 Ordonancement au plus tôt – Algorithme de Bellman

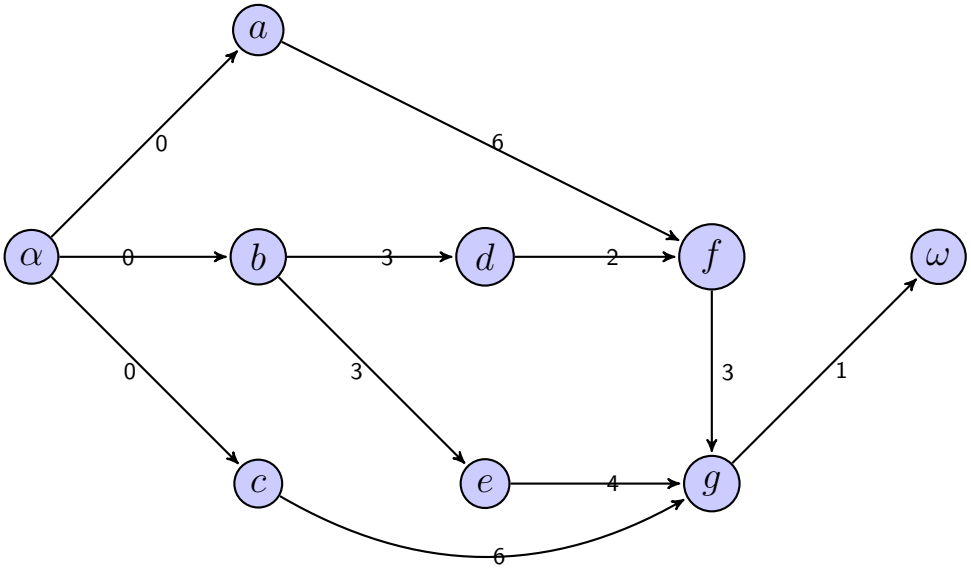
On peut représenter le tableau des contraintes à l'aide d'un graphe, le graphe *potentiels-tâches*, dans lequel :

- un sommet est associé à chaque tâche ;
- un sommet α que l'on associe au début du projet ;
- un sommet ω qui sera la fin du projet.

On aura un arc i, j si la tâche j ne peut démarrer qu'après la tâche i. Il s'agit d'une contrainte d'antécédence entre tâches.

La valuation de l'arc i, j : le plus souvent, c'est la durée de la tâche i. La tâche j ne peut démarrer que lorsque la tâche i est achevée : entre le début de la tâche j et le début de la tâche i qui la précéde s'écoule le temps d'exécution de la tâche i.

Remarque On peut rencontrer le contraire : « j ne peut démarrer que 2 jours après le début de i ». Dans ce cas, l'arc i, j sera valué par 2.



La durée minimale correspond au chemin le plug long entre α et ω.

2.1.2 Algorithme de Bellman

On commence par dresser le tableau suivant :

	0	0	0	0	3	3	6	9	10
Tâche	α	a	b	c	d	e	f	g	ω
Prédécesseurs		0 α : 0	0 α : 0	0 α : 0	0 b : 3	0 b : 3	$\psi = 0$ a : 6 $\chi = 3$ d : 2	0 c : 6 3 e : 4 6 f : 3	9 g : 1

Initialisation Le projet démarre à la date 0. Il s'agit de la date de début au plus tôt de α ; ici 0.

Deuxième phase Mouvement de haut en bas.
Dès que l'on connaît une date de début au plus tôt (1religne du tableau), on la répercute dans la deuxième ligne du tableau partout où la tâche concernée apparaît.

Ensuite mouvement de bas en haut.
Dès que l'on connaît les dates de début au plus tôt de tous les prédécesseurs d'une tâche, on dit que la colonne est complète, et on calcule la date de début au plus tôt de la tâche en tête de colonne (première ligne) par la formule de Bellman :

$$t_j = \max[t_i + v_{ij}] \qquad i \in (\Gamma^{-1}(j))$$

Par exemple, dans la colonne f, ψ et χ sont t_i et 6 et 2 sont v_{ij}

– t_j : la date de début au plus tôt de j ;
– v_{ij} : valuation de l'arc (i, j) ;
– Γ^{-1} : prédécesseur de j.
Ensuite
Ensuite

L'ordonnancement au plus tôt est donc la durée minimale du projet, soit la date au plus tôt de $\omega = 10$ jours.

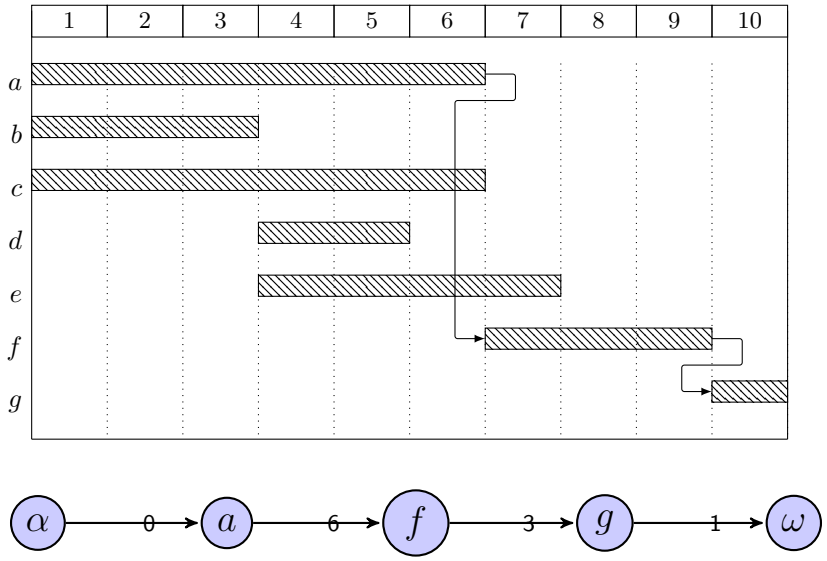


FIGURE 2.1 – Chemin critique

On s'intéresse à l'ordonancement au plud tard, pour calculer les marges de toutes les tâches non critiques.

Hypothèse La durée minimale du projet est respectée.
Pour les tâches critiques, la date de début au plus tard est égale à la date de début au plus tôt.
Quelles sont les dates de début au plus tard des tâches non critiques ?

	0	0	1	3	4	5	6	9	10
Tâche	α	a	b	c	d	e	f	g	ω
Successeurs	0 a : 0 1 b : 0 3 c : 0	6 f : 6	4 d : 3 5 e : 3	9 g : 6	6 f : 2	9 g : 4	9 g : 3	10 ω : 1	

Pour les tâches critiques :

$$t_a^* = t_a = 0$$
$$t_f^* = t_f = 6$$
$$t_g^* = t_g = 9$$

Pour les tâches non-critiques : la *marge totale* $= M_i = t_i^* - t_i$, soit :

$$M_b = 1$$
$$M_c = 3$$
$$M_d = 1$$
$$M_e = 2$$

La *marge libre*, m_i, est le délai dont on peut retarder la tâche i sans conséquence sur ses successeurs.

$$m_i = \min(t_j - t_i - v_{ij}) \qquad j \in \Gamma(i)$$

$$m_b = \min_{d,e}\{t_d - t_b - v_{bd}; t_e - t_b - v_{be}\}$$
$$= \min_{d,e}\{3 - 0 - 3; 3 - 0 - 3\} = 0$$

2.1.3 Deuxième application : algorithme de Ford

Cet algorithme permet de trouver le chemin le plus long, ou le chemin le plus court entre un sommet particulier (la racine) et tous les autres sommets du graphe.

Initialisation On numérote les sommets, le sommet racine prennant le numéro 1. On essaye de respecter autant que possible la numérotation topologique des sommets. Les arcs de numérotation non topologique risquent d'entrainer des retours en arrière ce qui impliquerait des retards.

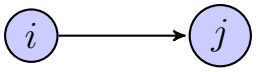


FIGURE 2.2 – Représentation topologique ; ici $i < j$

Voir graphe page 106.

On cherche le chemin le plus court allant du sommet 1 vers tous les autres sommets.

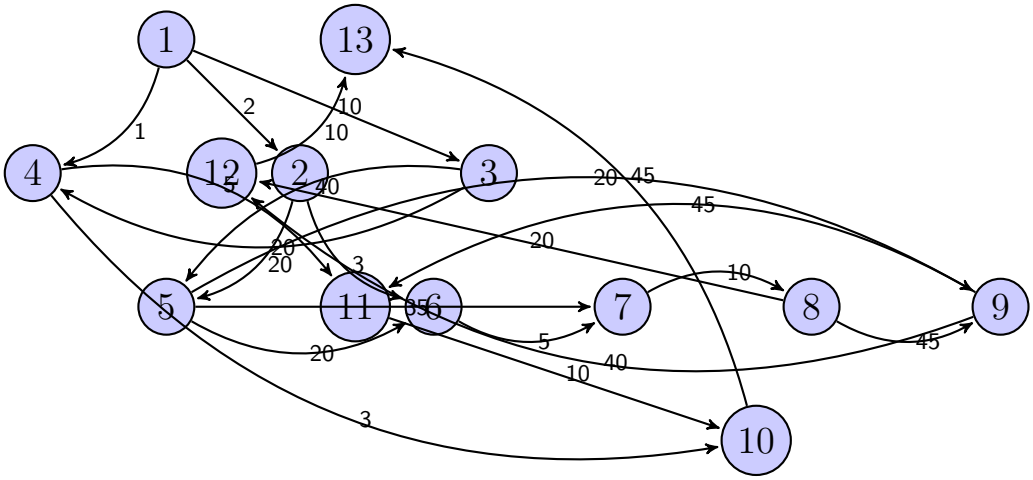


FIGURE 2.3 – Graphe page 106

Marquage des sommets On affecte au sommet 1 la valeur $\lambda_1 = 0$ et à tous les autres sommets la valeur $\lambda_i = +\infty$.

- λ_i évolue en cours d'algorithme ;
- le dernier λ_i affecté à i donne la longueur du chemin le plus court allant du sommet 1 au sommet i.

On examine à tour de rôle tous les arcs partant du sommet i, en commençant par $i = 1$. Pour ne pas en oublier, on fait cet examen dans l'ordre croissant de l'extrémité j. Au moment de l'examen de l'arc (i, j) : $v(i, j)$ est la valuation de l'arc (i, j).

Si $v(i, j) < \lambda_i - \lambda_j$, alors on remplace λ_j par $\lambda_i + v(i, j)$.

Si à un moment donné, on a $i > j$ et un changement de valeur pour λ_j, il faut recommencer une partie des opérations, en repartant du sommet $i = j$. On appel cela un retour en arrière.

Par exemple examen de l'arc $(11, 10)$.

Cet examen risque d'entraîner le changement de valeur de λ_{10}. Alors, il faut reprendre l'examen de l'arc $(10, 13)$ avec la nouvelle valeur λ_{10}.

Go ! examen de l'arc $(1, 2)$. $2 < \infty - 0$? Oui $\Rightarrow$ on remplace λ_2 par $\lambda_1 + v(1, 2) = 0 + 2 = 2$.

Remarque Dans cet exemple, pas de retour en arrière, bien qu'un arc ait ses extrêmités avec une numérotation non topologique. L'examen de l'arc $(11, 10)$ n'a pas entraîné la modification de λ_{10}. On ne doit pas reaire l'examen de l'arc $(10, 13)$.

Phase 2 identification du chemin solution.

On par de l'extrêmité terminale soit i. Le prédécesseur de i est le sommet i tel que :

$$\lambda_i - \lambda_{i'} = v(i', i)$$

2.2 Programme de transport optimal

Définition Soit un graphe biparti avec `m` les origines ou dépôts et `n` les destinataires ou clients.

On veut acheminer des marchandises depuis les dépôts jusqu'aux clients.

Hypothèses le *coût unitaire* de transport est donné depuis chaque dépôt vers chaque client : C_{ij} = coût de transport d'une tonne depuis `i` vers `j` $\rightarrow [C]$.

$$X = \begin{array}{c} \\ I \\ II \\ III \\ IV \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & \end{array} \\ \begin{pmatrix} & & 18 & & & & 18 \\ 9 & 11 & 7 & & & 5 & 32 \\ & & 3 & 6 & 5 & & 14 \\ & & & & 9 & & 9 \\ 9 & 11 & 28 & 6 & 14 & 5 & 73 \end{pmatrix} \end{array}$$

- On connaît la disponibilité totale de chaque fournisseur : a_i.
- On connait la demande totale de chaque client : b_j.
- $\sum_{origines}$ disponnibilités des dépôts = $\sum_{client}$ demandes des clients

Remarque On peut toujours se ramener à ce cas, quitte à créer une origine fictive ou un client fictif pour absorber la différence entre $\sum_{disponibilité}$ et $\sum_{demandes}$.

On cherche un *plan de transport*, soit une matrice $[X]$ $n \times m \rightarrow x_0$ = quantité transportée depuis i vers j.

Matric X 4/6 ?

Un plan de transport est valide s'il respecte certaines conditions :

1. **contraintes de ligne :** la quantité livrée par chaque dépôt est égale exactement à la disponibilité.

$$\sum_{n}^{j=1} xij = a_i \qquad \forall i = 1, 2, \ldots, m$$

2. **contraintes de colonne :** la quantité reçue par chaque client est exactement égale à sa demande.

$$\sum_{m}^{i=1} x_{ij} = b_j \qquad \forall j = 1, 2, \ldots, n$$

Parmi tous les plans de transport $X = [x_{ij}]$ valides (respectant toutes les contraintes), on cherche le plan de transport de coût minimal.

Le coût d'un plan de transport donné est :

$$z = \sum_{m}^{i=1} \sum_{n}^{j=1} x_{ij} C_{ij}$$

Évaluation de la difficulté du problème.

- Nombre d'inconnues : $n \times m$, ici 24 inconnues, la matrice X ;
- nombre d'équations imposées (contraintes) : $n + m = 10$ équations ;
- nombre d'équations indépendantes : $n + m - 1$, ici 9 ;

$$\sum \text{disponnibilités des dépôts} = \sum \text{demandes des clients}$$

Définition Un plan de transport est dit non dégénéré (ou solution de base) si l'excès d'inconnues par nombre d'équations indépendantes est nul.

On cherche un plan de transport ayant $n \times m - (m + m - 1)$ variables nulles (15 variables nulles, 9 variables non nulles).

Méthode de recherche du plan de transport de coût minimal (ou plan de transport optimal) 2 phases :

1. **Heuristique de la différence maximale (ou de Balas-Hammer) :** cette méthode fournie une solution de base de bonne qualité, de manière constructive, servant de base pour la phase 2.
2. En partant du plan de transport procuré par la phase 1, on cherche par itération successives le plan de transport optimal.

Phase 1 On calcule pour chaque *rangée* (ligne ou colonne) la différence entre le coût unitaire le plus petit et le coût unitaire qui lui est immédiatement supérieur.

	1	2	3	4	5	6	disponnibilité	Δ_e lignes
I	12	27	61	49	83	35	18	15
II	23	39	78	28	65	42	32	5
III	67	56	92	24	53	54	14	29
IV	71	43	91	67	40	49	9	3
demande	9	11	28	6	14	5	73	
Δ_c	11	12	17	4	13	7		

- On identifie la rangée (ligne ou colonne) correspondant à la différence maximale (ici la ligne III).
- On repère, dans cette rangée, la relation (i, j) de coût unitaire le plus faible (ici la relation (III, 4)).

– On affecte à cette relation la quantité de marchandise la plus élevée possible, c'est-à-dire, le minimum du couple (disponibilité, demande) associé à cette case : $\min(6, 16) = 6$.

Cette opération a pour effet de saturer une rangée (ici la colonne 4), et on actualise la disponibilité ou la demande restante (ici la disponibilité de III : $14 \to 8$). On a éliminé une rangée, et cela a permis de fixer une variable x_{ij} :

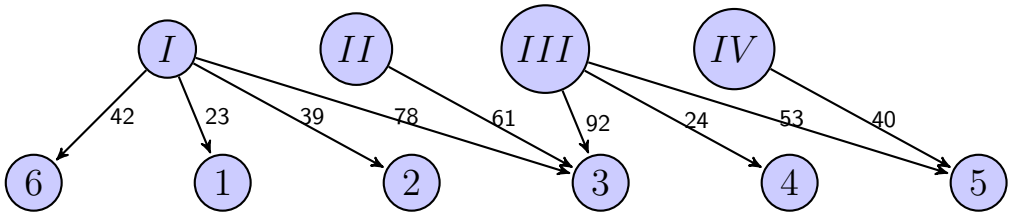
	1	2	3	5	6	disponnibilité	Δ_e lignes
I	12	27	61	83	35	18	15
II	23	39	78	65	42	32	16
III	67	56	92	53	54	8	1
IV	71	43	91	40	49	9	3
demande	9	11	28	14	5	67	
Δ_c	11	12	17	13	7		

On itère le processus jusqu'à ce que tout le tableau soit barré ; à la dernière itération (ici là huitième), deux variables sont affectées d'un seul coup.

Remarque si deux différences maximales sont égales, on a plusieurs solution. Ici à la quatrième itération, on a deux différences maximales égales à 13 : on choisira arbitrairement la liaison $(IV, 5)$.

$z = 3535$ solution de base, de bonne qualité, mais pas optimale.

2e phase recherche de la solution optimale : on construit le graphe biparti associé à la solution de base précédente.



On value ce graphe avec les coût unitaires.

On affect aux différents sommets un potentiel :

– potentiel 0 à l'origine de l'arc de coût max, ici le sommet III ;

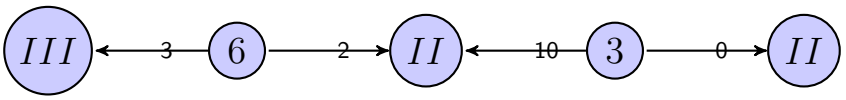
– différence de potentiel arc (i, j) : $v_j - v_i = C_{ij}$.

On calcule le coût marginal de chacune des liaisons (i, j) inutilisées dans la solution de base : $S_{ij} = v_i - v_j + c_{ij}$. On cherche s'il existe une liaison inutilisée de coût marginal négatif.

$$S_{I,1} = 31 - 37 + 12 = 6$$

$$S_{I,2} = 31 - 53 + 27 = 5$$

$$S_{III,6} = 0 - 56 + 54 = -2$$



Ici un seul coût marginal est inférieur à 0 ; la solution analysée n'est pas optimale. On utilise une chaine améliorante.

$$X = \begin{array}{c} \\ I \\ II \\ III \\ IV \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} & & 18 & & & & 18 \\ 9 & 11 & 10 & & & 2 & 32 \\ & & 0 & 6 & 5 & 3 & 14 \\ & & & & 9 & & 9 \\ 9 & 11 & 28 & 6 & 14 & 5 & 73 \end{pmatrix} \end{array}$$

On cherche la quantité déplaçable la plus élevée possible. On augmente d'une même quantité α tous les arcs à l'endroit et on diminue de α tous les arcs à l'envers :

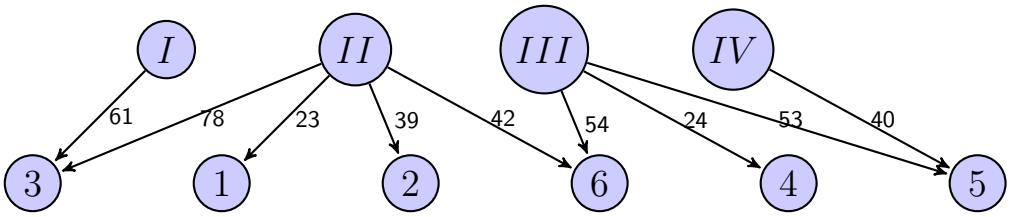
$$\alpha = 3$$

Le coût de cette solution est de $z = 18 * 61 + 9 * 23 + \ldots$.

$$z = 3235 + S_{III,6} * \alpha$$

$$z = 3529$$

2e itération On construit le graphe associé au plan de transport amélioré.



$$S_{I,1} = 17 - 23 + 12 = 6$$

$$S_{I,2} = 17 - 39 + 27 = 5$$

Tous les S_{ij} des liaisons inutilisées sont supérieur à 0, on s'arrête car le plan de transport analysé est optimal.

Remarque S'il existe un S_{ij} de liaison inutilisée qui soit égal à 0, cela signifie qu'il existe une autre solution de même coût.

$$X = \begin{array}{c} \\ I \\ II \\ III \\ IV \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & \end{array} \\ \begin{pmatrix} & & 18 & & & & 18 \\ 9 & 11 & 10 & & & 2 & 32 \\ & & & 6 & 5 & 3 & 14 \\ & & & & 9 & & 9 \\ 9 & 11 & 28 & 6 & 14 & 5 & 73 \end{pmatrix} \end{array}$$

Chapitre 3

Problèmes de flot maximal

Il existe 2 types d'applications :
- type réseau (de communication, routier, fluvial) : comment faire passer les éléments pour acheminer le débit maximun tout en respectant toutes les contraintes de capacité ?
- type affectation, par exemple un graphe biparti.

Définition Un *réseau de transport* est un graphe orienté sans boucle qui comporte une entrée (ou une source s) et une sortie (ou puits p).

Chaque arc u est doté d'une capacité $c(u) \geq 0$, représenté []
On s'intéresse à une fonction, appelée le flot.

On associe à chaque arc une fonction $f(u)$ = flux transitant dans l'arc$u$. $flot = \text{flux}_{\text{tous les arcs}}\ f(u)$: inconnue $c(u)$: donnée. C'est ce que l'on appel un *flot*.

Celui-ci doit respecter 2 types de contraintes :
1. $\forall u : 0 \leq f(u) \leq c(u)$
2. $\forall$ noeud x dans le réseau, la loi de Kirshoff est vérifiée. (si $f(u) = c(u)$, on dit que l'arc u est saturé).

$$\sum \text{flux entrants} = \sum \text{flux sortants}$$

Un flot est valide si et seulement si, il respecte les deux conditions ci-dessous. exemple : le flot nul.

Définition une *coupe* sépare la source du puits.

On appel S l'ensemble de sommets incluant obligatoirement s et excluant obligatoirement p.

$\bar{S}$ est l'ensemble de sommets complémentaires.

$(S, \bar{S})$ est une coupe séparant la source du puits.

Propriété $\forall$ coupe $(S, \bar{S})$ considérée, la valeur du flot mesurée au niveau de cette coupe est la même : c'est $val(f)$

Définition La capacité d'une coupe se définit par :

$$c(S, \bar{S}) = \sum c(u)_{arcsallantdeSversS\bar{}}$$

La valeur d'un flot ($val(f)$) se définit de 3 manières différentes :

1. Valeur à la source (hypothèse : pas de flux entrants dans s)

$$val(f) = \sum_{\text{arcs issus de } s} f(u)$$

2. Valeur au puits p (pas de flux sortants de p)

$$val(f) = \sum_{\text{flux des arcs entrants dans } p} f(u)$$

3. Valeur au niveau d'une coupe $(S, \bar{S})$ quelconque.

$$val(f) = \sum_{\text{arcs allant de } S \text{ vers } \bar{S}} f(u) \quad \sum_{\text{arcs allant de } \bar{S} \text{ vers } S} f(u)$$

Résultats théoriques

1. $\forall$ coupe $(S, \bar{S})$ considérée, la valeur du flot mesurée au niveau de cette coupe est la même ;
2. $\forall$ flot f et $\forall$ coupe $(S, \bar{S})$, établi dans un réseau de transport : $Val(f) \leq c(S, \bar{S})$.
   Si on trouve un flot f et une coupe $(S, \bar{S})$ tels que : $Val(f) = c(S, \bar{S})$, alors le flot est de valeur maximale, c'est le flot optimal

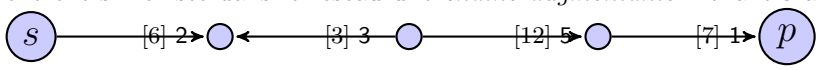
3.1 Théorème du float maximal

Le flot de valeur maximum est identique à la coupe de capacité minimale.
La coupe $(S, \bar{S})$ de capacité minimale respecte les propriétés suivantes :
– Tout arc u allant de S vers $\bar{S}$ dont $f(u) = c(u)$ est dit saturé.
– Tout arc u allant de $\bar{S}$ vers S est de flux nul ($f(u) = 0$).
Lorsque le flot est maximal (et que la coupe de $(S, \bar{S})$ est de capacité minimale) : Pour tout arc de $(S, \bar{S})$, il est saturé : $f(u) = c(u)$ Pour tout arc de $(\bar{S}, S)$, il est de flux nul : $f(u) = 0$

3.2 Algorithme de Ford-Fulkerson

Cet algorithme a pour but de déterminer le flot optimal dans un réseau de transport donné.

- On part d'un flot initial établi dans le réseau (soit un flot établi à la main, soit un flot nul) ;
- on cherche s'il existe dans le réseau une *chaîne augmentante* reliant s à p :



- Cette chaîne est dite *augmentante* si et seulement s'il existe un nombre positif α que l'on peut ajouter au flux de tous les arcs à l'endroit (sans aller au delà de leur saturation) et retrancher de tous les arcs à l'envers (sans aller en-dessous du flux nul). Ici : $\alpha_{\max} = 3$ et $\alpha_{\min} = (\min(c(u) - f(u)) ; \min f(u)$ des arcs à l'envers)
- Exploiter une chaine augmentante, c'est rechercher le α maximum.

$$\alpha = [Min(c(u) - f(u)); Minf(u)]$$

3.2.1 Principe de l'algorithme de Ford-Fullkerson

- On part d'un flot initial ;
- On cherche s'il existe une chaîne augmentante reliant s à p.
  Si oui : on exploite cette chaîne, on augmente α de la valeur du flot et on recommence cette étape.
  Si non : on a atteint le flot optimal.

3.2.2 Recherche d'une chaîne augmentante dans le réseau

Procédure de marquage des sommets.

- marquer (on note + à côté du sommet) l'entrée du réseau ;
- **marquage en avant :** marquer ($+I$ à côté du sommet J) l'extrémité terminale J de tout arc (I, J) non saturé et dont l'extrémité initiale est déjà marquée ;
- **marquage en arrière :** marquer ($-L$ à côté du sommet J) l'extrémité initiale K de tout arc (K, L) de flux non nul dont l'extrémité terminale L est déjà marquée.

Ces deux phaes de marquage avant et en arriere sont alternées jusqu'au blocage.

Si au moment du blocage, on constate que l'on a pu marqué p, cela signifie que le flot considéré n'est pas encore optimal. On se sert alors des marques pour identifier les différents sommets constituant la chaîne augmentante.

Si au moment du blocage, on constate que l'on a pas pu marqué p, dans ce cas le flot considéré est optimal.

Remarque Si l'on considère la dernière itération de marquage réalisé (où l'on n'a pas pu marqué p), on a la coupe $(S, \bar{S})$ de capacité minimale où S est constitué de la liste des sommets marqués.

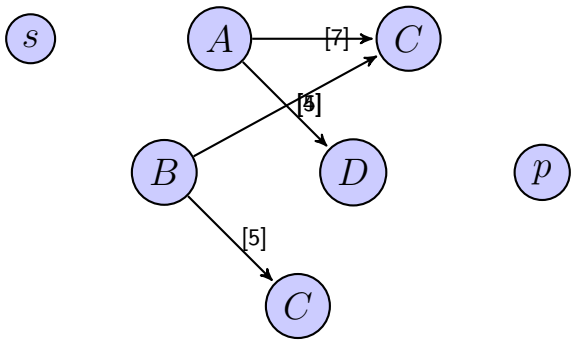
On véfifie que la valeur de cette capacité est égale à la valeur du flot obtenu : le flot maximal = la coupe de capacité minimale.

Remarque Pour diminuer le nombre d'itération de l'algorithme de Ford-Fullkerson, on s'efforce généralement de partir d'un *flot initial complet* (tout chemin de s à p comporte au moins un arc saturé).

Attention Ne pas confondre les problèmes de flot optimal avec les problèmes de plans de transport de coût minimal.

3.3 Application (poly. p103)

- 2 ports A et B : quantités 10 et 10 ;
- marchandises demandée dans 3 ports C, D et E selon les quantités 9, 12 et 7.



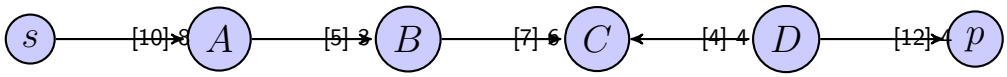
1. Peut-on satisfaire toutes les demandes ?
   Non, car l'offre est inférieure à la demande :
   offre totale $= 10 + 10 = 20$; demande totale $= 9 + 12 + 7 = 28$
2. Comment organiser les expéditions de façon à louer un maximun de marchandise ?

- transformer ce problème en un problème de recherche du flot maximal dans un réseau de transport ;
- résoudre par l'algo de Ford-Fulkerson ;
- **initialisation :** on s'efforce de démarrer l'algo à partir d'un flot complet (tout chemin de s à p comporte au moins 1 arc saturé) :
  - flux 7 de A vers C ;
  - flux 9 de C vers p ;
  - Kirschoff au noeud C : flux 2 de B vers C (OK car ¡ capacité de 5) ;
  - flux 10 de s vers A ;
  - Kirschoff en A : flux 3 de A vers D (OK car ¡ capacité de 4) ;
  - Kirschoff en D : flux 3 de D vers p (OK car ¡ capacité de 12) ;

– on sature $B - E$;
– Kirschoff au nœud E : flux de 5 sur $E - p$ (OK car ¡ capacité de 7) ;
– Kirshchoff au noeud B : flux de 7 sur $s - B$ (OK car ¡ capacité de 10).

On a un flot complet de valeur 17.

On a pu marquer p : le flot considéré n'est pas optimal :



Exploitation de la chaîne augmentente avec α : le flot passe de 17 à 18.
– Le nouveau flot est encore complet ;
– le nouveau flot est de valeur 18 ($10 + 8$ ou $9 + 4 + 5$) ;

2eitération : on ne peut pas marquer p, donc le flot optimal est de valeur 18.

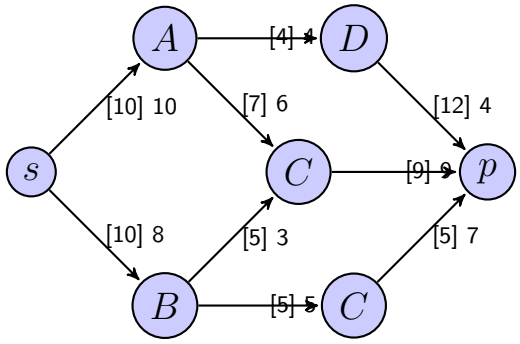
La coupe où S équivaut à l'ensemble des sommets marqués en dernière itération.

La capacité de la coupe $(S, \bar{S})$ est : $c(C,p)+c(A,D)+c(B,E) = [9]+[4]+[5] = 18$

On a trouvé une coupe dont la capacité est alors à la valeur d'un flot.

C'est que le flot de valeur maximum et la coupe est de la capacité minimal.

Remarque Tous les arcs de $(S, \bar{S})$ sont saturés (tous les arcs de $(\bar{S}, S)$ sont de flux nul)



Flot maximal de 4 (En partant du flot nul)