

Introduction à la théorie des graphes

- La Recherche Opérationnelle est un outil d'aide à la décision afin de résoudre un problème donné de la façon la plus optimale.
- Transformer un problème du monde réel sous la forme d'un problème mathématique (Modélisation mathématique) en utilisant 2 méthodes :
 - La Programmation Linéaire: transformation sous la forme d'une équation linéaire en prenant en compte les contraintes.
 - Théorie des langages: transformation sous forme de sommet et de relations entre les sommets.
- un graphe G est formé de 2 ensembles:
 - * un ensemble de sommets V . (nb sommets = ordre du graphe).
 - * un ensemble de relations entre les sommets E .

Les graphes non orientés:

où il n'y a pas de sens entre les sommets. Il est constitué de:

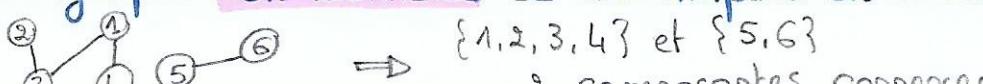
- un ensemble V de sommets.
- un ensemble E d'arêtes.

On note $G = (V, E)$.

- Notes:
- 1 - une arête e est incidente à un sommet v si v est une extrémité de e .
 - 2 - un sommet est isolé s'il n'a pas d'arête incidente.
 - 3 - une arête dont les extrémités sont égales est une boucle.
 - 4 - une arête e est multiple s'il existe une autre arête qui relie les mêmes sommets que e . On dit qu'elles sont parallèles.

Types:

- un graphe est simple si il ne comporte pas d'arête multiple et qu'il n'y a pas de boucle sur un sommet.
- un graphe est multiple si il comporte au moins une arête multiple et un sommet.
- un graphe est connexe s'il existe une chaîne reliant 2 sommets i et j (une chaîne dans G est une succession de sommets adjacents).
- un graphe non connexe se décompose en composantes connexes.



Types de graphes:

- **Graphe complet**: si chaque sommet est relié directement à tous les autres sommets.
- **Graphe biparti**: si ses sommets peuvent être divisés en 2 ensembles X et Y de sorte que toutes les arêtes du graphe relient un sommet de X à un sommet de Y .
- Soient $G(V, E)$ et $G'(V, E')$. G' est un graphe partiel de G si on obtient G' en enlevant une ou plusieurs arêtes à G ($E' \subseteq E$).
- Un sous-graphe de G induit par A (sous-ensemble de sommets A inclus dans V) $G(A, E(A))$ est un graphe de sommets A et d'arêtes $E(A)$ ayant leurs extrémités dans A .

Degré d'un sommet:

on appelle degré du sommet v et on note $d(v)$ le nombre d'arêtes incidentes à ce sommet.

⚠ une boucle sur un sommet compte double.

Théorème 1.1 (Lemme des poignées de main)

La somme des degrés des sommets d'un graphe est égale à deux fois le nombre d'arêtes.

Le degré d'un graphe: est le degré maximum de tous ses sommets

Note: 1- un graphe dont tous les sommets ont le même degré k est dit k -régulier.

- une chaîne est **simple** si chaque arête apparaît au plus une fois.
- une chaîne est **élémentaire** si chaque arête apparaît au plus une fois.
- une chaîne est **appelée cycle** si ses extrémités se coïncident.
- une chaîne est **eulérienne** si elle contient toutes les arêtes une seule fois.
- un cycle **eulérien** est une chaîne eulérienne fermée (sommet 1 = final).
- une chaîne **hamiltonienne** si elle passe par tous les sommets une seule fois.
- un cycle **hamiltonien** est une chaîne hamiltonienne fermée.

Graphes orientés :

- il contient des arêtes ayant un sens à respecter. il est défini par :
- un ensemble de sommets $V = \{v_1, v_2, \dots\}$
 - un ensemble d'arcs $E = \{e_1, e_2, \dots\}$

un arc e est défini par une paire ordonnée de sommets (sommet initial + sommet final).

degré d'un sommet :

Soit v le sommet. $d(v) = d^+(v) + d^-(v)$

- * $d^+(v)$ est le degré extérieur c'ds le nb d'arcs ayant v comme extrémité initiale.
- * $d^-(v)$ est le degré intérieur c'ds le nb d'arcs ayant v comme extrémité finale.

chemin : (chaîne) .

c'est une succession de sommets adjacents (en respectant le sens d'orientation).

circuit : (cycle)

c'est un chemin dont les sommets de départ et de fin sont les mêmes

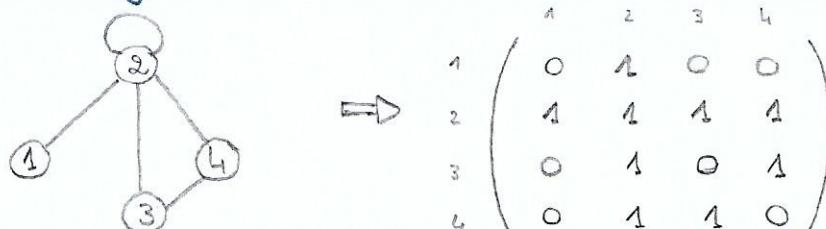
- un graphe orienté est fortement connexe s'il existe un chemin de i vers j et de j vers i .

Représentations non graphiques des graphes :

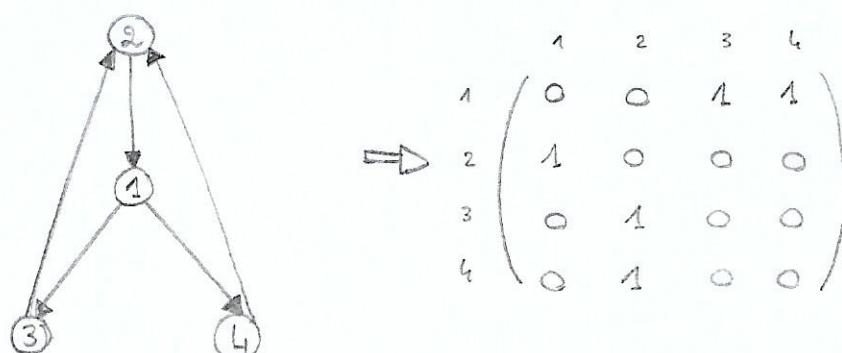
Matrice d'adjacence :

matrice de n lignes et n colonnes ($n = \text{nb de sommets}$).

exemple 1 :



exemple 2 :

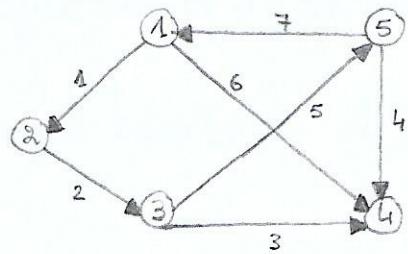


Matrice d'incidence :

matrice de n lignes et y colonnes : $n = \text{nb de sommets}$.
 $y = \text{nb d'arêtes/arc}s.$

- s'il existe une relation directe entre le sommet et l'arc $\rightarrow 1$.
- s'il existe une relation dans le sens inverse entre le sommet et l'arc $\rightarrow -1$.
- s'il n'existe aucune relation $\rightarrow 0$.

exemple :

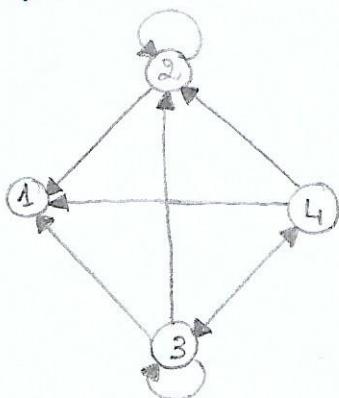


	1	2	3	4	5	6	7
1	1	0	0	0	0	1	-1
2	-1	1	0	0	0	0	0
3	0	-1	1	0	1	0	0
4	0	0	-1	-1	0	-1	0
5	0	0	0	1	-1	0	1

Liste d'adjacences :

- Donner à chacun des sommets la liste des sommets auxquels il est adjacent (auquel il est une extrémité initiale).
- Si un sommet est une extrémité finale pour les autres arcs, on l'appelle puit.
- Si un sommet est une extrémité initiale pour les autres arcs, on l'appelle source.

exemple :



1	\rightarrow			
2	\rightarrow	2	1	
3	\rightarrow	3	1	4
4	\rightarrow	1	2	3

Colonisation d'un graphes:

- une arête ne doit pas avoir 2 sommets de la même couleur pour qu'ils soient indépendants. s'ils ont la même couleur c'est qu'ils sont dépendants.
(les arêtes représentent les conditions à respecter).
 - une k-Colonisation d'un graphe G est une application C qui associe à chaque sommet une couleur de l'ensemble $\{1, 2, \dots, k\}$ tq les sommets voisins aient des couleurs différentes.
- ⚠ - La notion de colonisation n'est définie que pour les graphes sans boucle.
- On cherche généralement à déterminer une colonisation en utilisant aussi peu de couleurs que possible: Problème d'optimisation classique.
 - Le plus petit nb de couleurs nécessaires pour coloniser un graphe est le nombre chromatique de G : $\chi(G) (\leq n)$.

un graphe planaire peut être représenté sur un plan tq tout arc (ou arête) ne se croise pas avec un autre.

- Lorsque le graphe est planaire, $\chi(G) \leq 4$.

Algorithme de Glouton:

Principe: Faire , étape par étape, un choix optimum local:

Début

Pour i allant de 1 à n

Affecter à i si la plus petite couleur non affectée à ses voisins déjà colonisés .

Fin Pour

Afficher le nombre de couleur utilisées

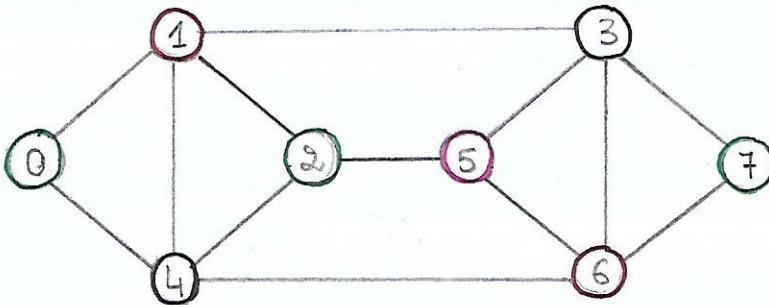
Fin

un algorithme de glouton se termine en n étapes. il nécessite au plus $d + 1$ couleurs avec d . le degré du graphe.

Algorithme Welch & Powell :

1. On donner les sommets du graphe suivant un ordre décroissant des degrés $\rightarrow d(x_1) \geq d(x_2) \geq d(x_3) \dots \geq d(x_n)$.
2. Affecter une couleur c_1 au sommet x_1 et attribuer la même couleur aux reste des sommets non colorés et non adjacents. d'une façon cumulative et suivants des degrés décroissants.
3. S'il reste des sommets non colorés, attribuer une nouvelle couleur au premier sommet non coloré et répéter la procédure.
4. Condition d'arrêt: tous les sommets sont colorés.

exemple:



Sommets	1	3	4	6	2	5	0	7
Degrés	4	4	4	4	3	3	2	2
Iter 1	C_1				C_1			
Iter 2	X	C_2	C_2	X				
Iter 3	X	X	X	X	C_3	C_3	C_3	C_3
Iter 4	X	X	X	X	X	X	C_4	X

Problème du plus court chemin

Graphe valué :

Soit un graphe orienté $G = (V, E)$. On associe à chaque arc AB un $n \in \mathbb{R}$ appelé longueur ou coût de l'arc et on note $C_{AB} = n$.

Rq : un coût peut être positif, négatif ou nul.

Pour déterminer le chemin le plus court, il faut que :

$$l(u^*) = \sum_u l(u) \text{ soit minimale}$$

chemin allant
de i vers j

Les algorithmes diffèrent selon :

- les propriétés des graphes : - $l(u) \geq 0$ pour tous les arcs.
ou - $l(u)$ quelconque.
ou - graphe sans circuit (chemin fermé)
 - Recherche du PCC d'un sommet aux autres.
 - Recherche du PCC entre tous les couples.
- le problème considéré :

Condition nécessaire :

Le PCC a une solution ssi il n'existe pas dans le graphe un circuit de longueur strictement négative (appelé circuit absorbant).

Principe d'optimalité :

Tout chemin optimal est constitué de sous-chemins optimaux.

Algorithme de Dijkstra :

Il sert à résoudre le problème du PCC : il calcule les PCC à partir d'une source vers les autres sommets dans un graphe orienté pondéré par des réels positifs.

- * $d(j)$ est la longueur du PCC reliant le sommet i à j .
- les sommets du graphe sont partitionnés en 2 sous-ensembles :
 - * P : les sommets (j) dont les $d(j)$ sont fixés d'une façon permanente.
 - * T : les sommets (j) dont les $d(j)$ sont fixés d'une façon temporaire.
- (nb d'itérations = nb sommets - 1)

(Pseud-code)

Etape 0: Initialisation

$$d(1) = 0$$

Faire $\forall j \in \{2, \dots, n\}$

$$d(j) = \begin{cases} C_{1j} & \text{si il existe un arc direct entre 1 et } j \\ \infty & \text{si non} \end{cases}$$

$PCC(j) = (1, j)$, le plus court chemin qui relie 1 et j .

Fin.

$$P = \{1\} \text{ et } T = \{2, \dots, n\}$$

Etape 1: Choisir d'un $d(k)$ optimal:

déterminer le noeud k tq $d(k) = \min_{j \in T} d(j)$

$$\text{Faire: } P = P \cup \{k\} \text{ et } T = T / \{k\}$$

Fin.

Si $T = \emptyset$ alors Fin

Sinon

Etape 2: Mise à jour des $d(j)$

Faire: $\forall (j) \in T$ et successeur de (k)

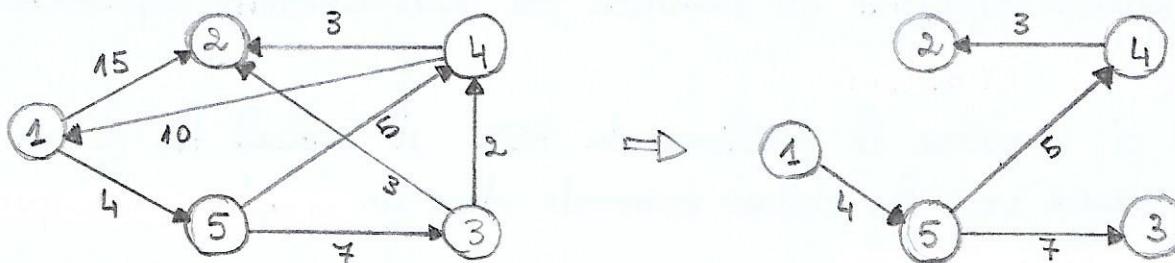
Si $d(k) + C_{kj} < d(j)$ alors $d(j) = d(k) + C_{kj}$

$$PCC(j) = PCC(k) \cup (k, j)$$

Fin.

Repéter à l'étape 1.

Exemple :



Par rapport au sommet 1:

	It(0)	It(1)	It(2)	It(3)	It(4)	
	$d(j)$	$PCC(j)$	$d(j)$	$PCC(j)$	$d(j)$	$PCC(j)$
2	15	(1,2)	15	(1,2)	12	(1,5,4,2)
3	∞	(1,3)	11	(1,5,3)	11	(1,5,3)
4	∞	(1,4)	9	(1,5,4)	-	-
5	4	(1,5)	-	-	-	-
P	{1}	{1,5}	{1,5,4}	{1,5,4,3}	{1,5,4,3,2}	

Algorithme de Ford-Bellman:

- même principe que Djikstra, mais celui-ci autorise la présence des arcs de poids négatif.

(Pseudo-code)

Etape 0: Initialisation

$$d^1(1) = 0$$

Faire $\forall j \in \{2, \dots, n\}$

$$d^1(j) = \begin{cases} C_{1j} & \text{si l'arc } 1,j \in \text{graphe} \\ \infty & \text{si non} \end{cases}$$

Fin.

$$k = 1$$

Etape 1: Mise à jour des $d^k(j)$

Faire $\forall (l)$

$$d^{k+1}(j) = \min [d^k(j), \min (d^k(l) + C_{lj})]$$

avec (l) prédecesseur de (j) .

Fin.

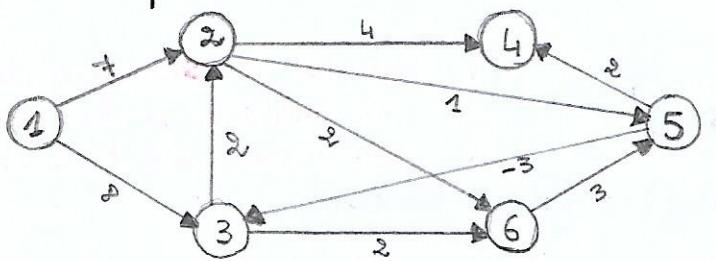
Etape 2: Test de convergence :

Si $d^{k+1}(j) = d^k(j)$, alors Fin (optimalité)

Sinon Si $k = n - 1$ alors il existe un circuit de Pongueurs \Leftrightarrow
sinon optimalité

Sinon faire $k = k + 1$ et aller à l'étape (1).

Exemple:



Critère d'arrêt:
2 itérations égales.

$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$d^k(j)$	PCC(j)	$d^k(j)$	PCC(j)	$d^k(j)$
1 0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)
2 + (1,2)	7 (1,2)	7 (1,2)	7 (1,2)	7 (1,2)
3 8 (1,3)	8 (1,3)	5 (1,2,5,3)	5 (1,2,5,3)	5 (1,2,5,3)
4 ∞ (1,4)	11 (1,2,4)	10 (1,2,5,4)	10 (1,2,5,4)	10 (1,2,5,4)
5 ∞ (1,5)	8 (1,2,5)	8 (1,2,5)	8 (1,2,5)	8 (1,2,5)
6 ∞ (1,6)	9 (1,2,6)	9 (1,2,6)	7 (1,2,5,3,6)	7 (1,2,5,3,6)

(il n'y a pas d'ensemble permanent, on peut faire un mag de tous les sommets)

Problème de flux maximal dans les réseaux

On appelle réseau de transport un graphe orienté avec chaque arc de graphe muni d'une capacité > 0 et constitué par une source et un puit.

Le problème de flux dans un réseau:

représente la circulation d'une matière sur les arcs d'un graphe.
il s'agit d'optimiser le mouvement.

Réseau:

Soit $G(V, E)$ un graphe orienté, on associe à chaque arc une capacité de transport C_e : c'est un réseau.

Dans un réseau $G(V, E)$ l'ensemble des sommets V est composé de:

- Source du réseau : sommet de degré intérieur nul.
- Puit / destination / sortie : sommet de degré extérieur nul.
- Nœuds de transit : autres sommets du réseau.

* Flux : quantité de matière circulant sur un arc e notée φ_e .

* Flot : vecteur dont les composantes sont les flux sur les arcs du graphe, noté $\Phi = (\varphi_1, \varphi_2, \dots, \varphi_n)$.

* Flot canalisé : (ou flot admissible) vérifie les deux conditions suivantes :

$$0 < \varphi_e \leq C_e$$

et
$$\sum_{i \in P(j)} \varphi(i, j) = \sum_{i \in S(j)} \varphi(i, j)$$

Loi de conservation de la matière à travers un nœud de transit.

avec $P(j)$: flux entrant et $S(j)$: flux sortant.

Arc de retour :

c'est un arc imaginaire dont la valeur sera appelée valeur de retour : afin de vérifier que le flot entrant au puit est égal au flot sortant de la source.

* Valeur de flot : quantité totale qui transverse le réseau.

* Arc saturé : si $\varphi_e = C_e$.

* Flot complet : si tout chemin de s_0 (source) vers s_m (puit) passe au moins par un arc saturé (on dit que le chemin est saturé).

Problème du flot maximal :

détermination d'un flot sur G , compatible avec les capacités, et dont le flux Φ_0 sur l'arc de retour n_0 est le plus grand possible.

Algorithme de Ford-Fulkerson : (dans un réseau formé de $n+1$ noeuds)

1^{ere} étape : saturer tous les chemins de n_0 à n_n .

2^{eme} étape : saturer toutes les chaînes entre n_0 et n_n .

Etape 1 : Saturation de tous les chemins de n_0 vers n_n :

$$\Phi_0 = 0, \text{ flot initial} = 0$$

A partir du flot nul, on obtient un flot complet

1) Γ : chemin de n_0 vers n_n

2) Calculer $\Delta = \min_{u_j \in \Gamma} \{C(u_j) - \Phi(u_j)\}$

3) $\Phi(u_j) = \begin{cases} \Phi(u_j) + \Delta & \text{si } u_j \in \Gamma \\ \Phi(u_j) & \text{sinon} \end{cases}$

4) Répéter l'étape 1 tant qu'il y a des chemins non saturés.

Etape 2 : Saturation de toutes les chaînes entre n_0 et n_n :

1) Υ : chaîne entre n_0 et n_n

2) calculer $\Delta = \min \{\Delta_1, \Delta_2\}$

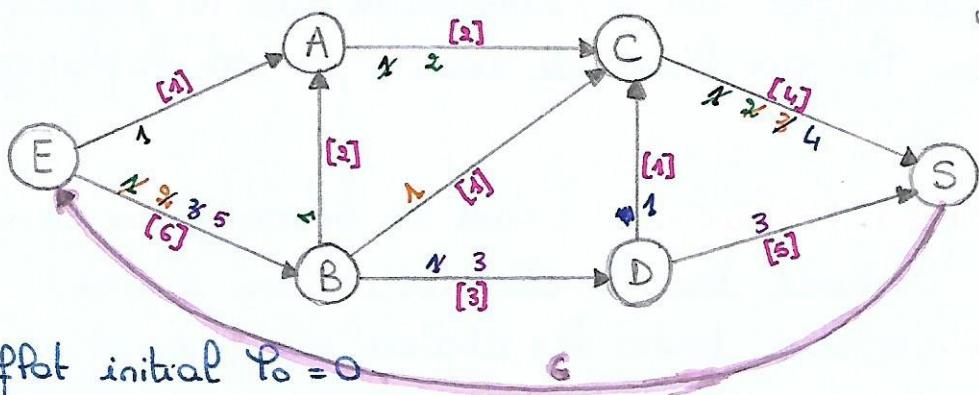
avec $\Delta_1 = \min_{u_j \in \Upsilon} \{C(u_j) - \Phi(u_j)\}$, u_j dans le sens direct de parcours de la chaîne Υ .

$\Delta_2 = \min_{u_j \in \Upsilon} \{\Phi(u_j)\}$, u_j dans le sens contraire de parcours de la chaîne Υ .

3) $\Phi(u_j) = \begin{cases} \Phi(u_j) + \Delta & \text{si } u_j \in \Upsilon \text{ dans le sens direct} \\ \Phi(u_j) - \Delta & \text{si } u_j \in \Upsilon \text{ dans le sens contraire} \end{cases}$

4) Répéter l'étape 2 tant qu'il y a des chaînes pas saturées.

Exemple :



$\Phi = \Phi_{\text{max}} = \text{somme de fflat sur les arcs finaux}$

Flat entrant :
 $X_{EA} + X_{EB}$

Flat sortant :
 $X_{CS} + X_{DS}$

Le fflat initial $\Phi_0 = 0$

Etape 1 : Saturation des chemins de E vers S.

- * E - A - C - S
- * E - B - A - C - S
- * E - B - C - S
- * E - B - D - C - S
- * E - B - D - S

(les fflats sont initialisés à 0).

Chemin 1 : E - A - C - S :

$$\left. \begin{array}{l} EA : 1 - 0 = 1 \\ AC : 2 - 0 = 2 \\ CS : 4 - 0 = 4 \end{array} \right\} \Delta = 1 \rightarrow \left. \begin{array}{l} EA = 1 - 1 = 0 \\ AC = 2 - 1 = 1 \\ CS = 4 - 1 = 3 \end{array} \right\} \begin{array}{l} EA \text{ est saturé} \\ \text{alors ce chemin} \\ \text{est saturé.} \end{array}$$

Chemin 2 : E-B-A-C-S :

$$\left. \begin{array}{l} EB : 6 - 0 = 6 \\ BA : 2 - 0 = 2 \\ AC : 2 - 1 = 1 \\ CS : 4 - 1 = 3 \end{array} \right\} \Delta = 1 \rightarrow \left. \begin{array}{l} EB = 6 - 1 = 5 \\ BA : 2 - 1 = 1 \\ AC : 2 - (1+1) = 0 \\ CS : 4 - (1+1) = 2 \end{array} \right\} \begin{array}{l} AC \text{ est saturé} \\ \text{alors ce chemin} \\ \text{est saturé.} \end{array}$$

Chemin 3 : E-B-C-S :

$$\left. \begin{array}{l} EB : 6 - 1 = 5 \\ BC : 1 - 0 = 1 \\ CS : 4 - 2 = 2 \end{array} \right\} \Delta = 1 \rightarrow \left. \begin{array}{l} EB : 6 - 2 = 4 \\ BC : 1 - 1 = 0 \\ CS : 4 - 3 = 1 \end{array} \right\} \begin{array}{l} BC \text{ est saturé} \\ \text{alors ce chemin} \\ \text{est saturé.} \end{array}$$

Chemin 4 : E-B-D-C-S :

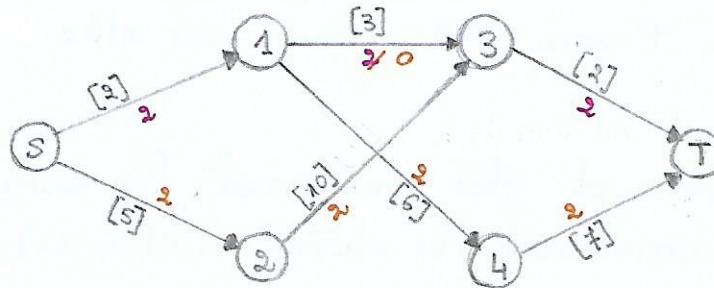
$$\left. \begin{array}{l} EB : 6 - 2 = 4 \\ BD : 3 - 0 = 3 \\ DC : 1 - 0 = 1 \\ CS : 4 - 3 = 1 \end{array} \right\} \Delta = 1 \rightarrow \left. \begin{array}{l} EB : 6 - 3 = 3 \\ BD : 3 - 1 = 2 \\ DC : 1 - 1 = 0 \\ CS : 4 - 4 = 0 \end{array} \right\} \begin{array}{l} DC \text{ et } CS \text{ sont saturés} \\ \text{alors ce chemin est} \\ \text{saturé.} \end{array}$$

Chemin 5 : E-B-D-S :

$$\left. \begin{array}{l} EB : 6 - 3 = 3 \\ BD : 3 - 1 = 2 \\ DS : 5 - 0 = 5 \end{array} \right\} \Delta = 2 \rightarrow \left. \begin{array}{l} EB : 6 - 5 = 1 \\ BD : 3 - 3 = 0 \\ DS : 5 - 2 = 3 \end{array} \right\} \begin{array}{l} BD \text{ est saturé} \\ \text{alors ce chemin} \\ \text{est saturé.} \end{array}$$

Ensuite !

Exemple 2 (avec chaînes) :



$\Phi_{initial} = 0$

Etape 1: Recherche de chemins:

- * $S - 1 - 3 - T$
- * $S - 1 - 4 - T$
- * $S - 2 - 3 - T$

$$\begin{aligned} \Phi(S1) &= 2 - 0 = 2 \\ \Phi(13) &= 3 - 0 = 3 \\ \Phi(3T) &= 2 - 0 = 2 \end{aligned} \quad \left. \begin{aligned} \Delta &= 2 \\ (\text{flat} = 2) \end{aligned} \right\} \rightarrow \begin{aligned} \Phi(S1) &= 2 - 2 = 0 \\ \Phi(13) &= 3 - 2 = 1 \\ \Phi(3T) &= 2 - 2 = 0 \end{aligned} \quad \left. \begin{aligned} &\rightarrow S1 \text{ et } 3T \text{ sont saturés} \\ &\text{donc ce chemin est saturé.} \end{aligned} \right.$$

Tous chemins passant par $S1$ ou $3T$ est saturé \rightarrow il n'y a pas d'autres.

Etape 2: Recherche de chaînes non saturées:

- * $S - 2 - 3 - 1 - 4 - T$

$$\begin{aligned} \Phi(S2) &= 5 - 0 = 5 \\ \Phi(23) &= 10 - 0 = 10 \\ \Phi(14) &= 6 - 0 = 6 \\ \Phi(4T) &= 7 - 0 = 7 \\ \Phi(31) &= 2 \rightarrow \Delta_2 = 2 \end{aligned} \quad \left. \begin{aligned} \Delta_1 &= 5 \\ \min(\Delta_1, \Delta_2) &= 2 \rightarrow \Phi_{max} = 2 + 2 \\ \rightarrow \text{flat} &= 2 \end{aligned} \right\}$$

Problème de coupe minimale :

une coupe $[S, T]$ est définie par un ensemble S contenant la source (s) et un ensemble T contenant la cible (t) tq $S \cap T = \emptyset$.

on dit que l'ensemble des arcs :

$\omega^+(S) = \{u \in U \text{ tq origine de } u \text{ dans } S \text{ et non son extrémité}\}$
 est une coupe séparant S et T (les sommets ne sont pas forcément liés)
 La capacité de la coupe $[S, T]$ est donnée par :

$$\sum_{i \in S, j \in T} C_{ij}$$

exemple d'après le schéma du haut de la page :

$$S = \{s, 1, 2\} \text{ et } T = \{3, 4, t\} \quad (\text{choisis au hasard})$$

$$\rightarrow \omega^+(S) = \{(s, 1), (s, 2), (1, 3), (1, 4), (2, 3)\} \quad \text{car les 2 extrémités } \in S$$

$$\rightarrow \text{Capacité } ([S, T]) = C_{13} + C_{14} + C_{23} = 3 + 6 + 10 = 19.$$

Pour toute coupe $[S, T]$: $\Phi_{\text{max}} \leq \text{Capacité } [S, T]$.

Si $\text{Capacité } [S, T] = \Phi_{\text{max}}$ cette coupe est dite minimale.

Recherche de la coupe minimale :

Suite à la recherche du flot maximal, les sommets de S sont ceux qui constituent les chemins (chaînes) non saturés issus de (s) .

Pour l'exemple précédent :

$S = \{2, 3, s\}$ et $T = \{1, 4, t\} \rightarrow$ c'est la coupe minimale.

$$\omega^+(s) = \{(s, 1), (3, t)\}$$

$$\rightarrow C[S, T] = C_{(s, 1)} + C_{(3, t)} = 4 = \Phi_{\text{max}}$$



Somme des capacités
des arcs dont le
sommets $\in S$ et
l'extrémité $\in T$ et
qui sont directs.