# Chapitre 3
# Problèmes de Cheminement

# Introduction (1/2)

Le **routage** est un mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur à un ou plusieurs destinataires.

Deux types de machines (composants) du réseau sont considérées: les ***routeurs*** (servent d'intermédiaire dans la transmission d'un message) et les ***hôtes*** *(*émettent ou reçoivent les messages).

Une ***passerelle*** est utilisée lorsque le routeur se trouve entre deux réseaux dépendant d'autorités différentes (le réseau local d'une entreprise et l'Internet).

Chaque routeur maintient une liste (**table de routage**) des réseaux connus ( chacun de ces réseaux est associé à un ou plusieurs routeurs voisins à qui le message peut être passé).

# Introduction (2/2)

**- *OSPF*** (***Open Shortest Path First***) est un protocole de routage interne IP de type « à état de liens » où chaque routeur établit des relations d'adjacence avec ses voisins immédiats en envoyant des messages *hello* à intervalle régulier. Chaque routeur communique ensuite la liste des réseaux auxquels il est connecté par des messages *Link-state advertisements* (LSA) propagés de proche en proche à tous les routeurs du réseau. L'ensemble des LSA forme une base de données de l'état des liens *Link-State Database* (LSDB) pour chaque aire, qui est identique pour tous les routeurs participants dans cette aire. Chaque routeur utilise ensuite l'algorithme de ***Djikstra*** pour déterminer la route la plus courte vers chacun des réseaux connus dans la LSDB.

**- RIP (Routing Information Protocol)** est un protocole de routage IP de type Vector-Distance(à vecteur de distances) s'appuyant sur l'algorithme de détermination des routes décentralisé Belleman-Ford qui permet à chaque routeur de communiquer aux routeurs voisins la métrique, c'est-à-dire la distance qui les sépare d'un réseau IP déterminé quant au nombre de *hops* (sauts).

Plusieurs applications des problèmes de cheminement existent en pratique comme en optimisation de réseaux, programmation dynamique, problème de tournées, traitement du signal etc.

# Généralités (1/2)

Un réseau R=(X, U, d) est un graphe G=(X, U) muni d'une application d: $U \rightarrow \Re$

$u \rightarrow d(u)=d_u$ longueur de l'arc u

Soit C un chemin dans R, la longueur du chemin C est:

$l(C) = \sum_{u \in C} d(u) n_u$ où $n_u$ est le nombre de fois que l'arc u est emprunté.

On définit de la même manière la longueur d'une chaine, cycle et circuit.

Un circuit *C* est dit absorbant si sa longueur est négative ($l(C)<0$).

Soient un sommet source s ($\Gamma$-(s)=$\emptyset$) et un sommet puits p ($\Gamma$+(p)=$\emptyset$)

En pratique l'application l peut être interprétée de différentes façons (coût de transport, coût de construction de l'arc, temps de parcours de l'arc, débit que peut supporter l'arc etc.)

# Généralités (2/2)

Considérons les trois problèmes suivants:

$$(A)\begin{cases} Trouver\ un\ chemin\ élémentaire\ C^* \\ l(C^*) \le l(C)\ \ \forall C \in \zeta \quad où\ \zeta = \{C\ /\ C\ chemin\ élémentaire\ de\ s\ à\ p\} \end{cases}$$

$$(B)\begin{cases} Trouver\ un\ chemin\ élémentaire\ C^* \\ l(C^*) \le l(C)\ \forall C \in \zeta\ où\ \zeta = \{C\,/\,C\ est\ un\ élémentaire\ de\ s\ à\ x, \forall x \in X\} \end{cases}$$

$$(C)\begin{cases} Trouver\ un\ chemin\ élémentaire\ C^* \\ l(C^*) \le l(C)\ \forall C \in \zeta\ où\ \zeta = \{C\,/\,C\ est\ un\ élémentaire\ de\ x\ à\ y, \forall x, y \in X\} \end{cases}$$

On définit également les problèmes A', B', C' en omettant le terme élémentaire.

***<u>Remarque</u>***: Trouver un plus long chemin  dans R revient à trouver un plus court chemin dans R'=(X, U, -*l*).

# Quelques résultats théoriques (1/4)

***Proposition***: Soit R=(X, U, l) un réseau ayant un circuit absorbant *C* alors il n'existe pas de plus court chemin de s à p.

En effet, soit un sommet $x \in C$ avec s, $p \notin C$. Considérons les chemins $C_{sx}$ de s à x et $C_{xp}$ de x à p. $C_{sx} \cup C_{xp}$ *et* $C_{sx} \cup (k\ C) \cup C_{xp}$ *sont des c*hemins de s à p et on a:

$l(C_{sx} \cup (k\ C) \cup C_{xp}) = l(C_{sx}) + k(C) + l(C_{xp}) < l(C_{sx}) + l(C_{xp})$.

Il n'existe donc pas de chemin de s à p car le minimum est atteint à l'infini.

***Théorème 1***: Le problème (A') possède une solution si et seulement si l'ensemble Y des descendants de s et ascendants à p soit non vide et que le sous réseau $R_Y$ construit sur Y ne possède pas de circuit absorbant. Si tel est le cas toute solution de (A) est solution de (A')

# Quelques résultats théoriques (2/4)

***<u>Théorème 2</u>***: Le problème (B) possède une solution si et seulement si s est racine et R ne possède pas de circuit absorbant. Si tel est le cas toute solution de (B) est solution de (B’).

***<u>Théorème 3</u>***: Le problème (C) possède une solution si et seulement si le graphe G est fortement connexe et R ne possède pas de circuit absorbant. Si tel est le cas toute solution de (C) est solution de (C’).

***<u>Théorème 4</u>***: Soit R=(X, U, l) un réseau et C un plus court chemin de s à p dans R. Soit $C_{xy}$ une portion du chemin C comprise entre x et y alors cette portion Cxy est aussi plus court chemin entre x et y.

> ***<u>Principe de Bellman</u>*: Toute sous politique d’une politique optimale et elle-même optimale**

En effet:

Supposons qu’il existe un autre chemin $C'_{xy}$ compris entre x et y vérifiant $l(C'_{xy}) < l(C_{xy})$. Or $C= C_{sx} \cup C_{xy} \cup C_{yp}$ et $C'= C_{sx} \cup C'_{xy} \cup C_{yp}$ sont des chemins de s à y avec $l(C')= l(C_{sx}) \cup l(C'_{xy}) \cup l(C_{yp}) < l(C)$ d’où C n’est plus un plus court chemin de s à p. D’où la contradiction.

# Quelques résultats théoriques (3/4)

***Proposition***: Soient R=(X, U, l) un réseau sans circuit absorbant et admettant s comme racine. On associe à tout sommet x de X la distance $\pi$(x) représentant la longueur d'un plus court chemin de s à x dans R (potentiel).

On a: $\forall u \in U, \pi(T(u)) - \pi(I(u)) \leq l(u)$ avec I et T sont les applications représentant les extrémités d'arcs.

En effet,

Soient u un arc quelconque de U, $C_{sI(u)}$ le plus court chemin de s à I(u) (et $C_{sT(u)}$ le plus court chemin de s à T(u). En d'autres termes: $l(C_{sI(u)}) = \pi(I(u))$ et $l(C_{sT(u)}) = \pi(T(u))$ .

On a: $C_{sI(u)} \cup \{u\}$ est un chemin de s à T(u), donc

$l(C_{sI(u)} \cup \{u\}) \geq l(C_{sT(u)}) \Leftrightarrow l(C_{sI(u)}) + l(u) \geq l(C_{sT(u)}) \Leftrightarrow \pi(I(u)) + l(u) \geq \pi(T(u))$ ou encore $\pi(T(u))\ \pi(I(u)) \leq l(u)$.

# Quelques résultats théoriques (4/4)

***Proposition***: Le problème de la recherche des plus courts chemins de s à x revient dans le réseau R ayant s comme racine et ne possédant pas de circuit absorbant revient à déterminer les plus courtes distances issues de s ( potentiels).

On suppose $\pi(s)=0$.

En effet, considérons le graphe partiel $H=(X, V)$ avec $V=\{u\in U / \ \pi(T(u)) \ \pi(I(u)) = l(u)\}$.

1) Un plus court chemin C de s à x dans R est aussi un chemin de (X,V). En effet, $x\in X$ et $C_{sx}$ un plus court chemin de s à x dans R. Soit $u \in C_{sx}$ d'après le théorème 4 u est un plus court chemin de I(u) à T(u). D'où $\pi(T(u)) \ \pi(I(u)) = l(u)$ donc $u\in V$. Comme u est quelconque sur $C_{sx}$ alors $C_{sx} \subseteq V$.

2) Soit $x\in X$ et $C_{sx}$ un chemin de s à x dans H et supposons que $C_{sx} = (s, u_1, x_1, u_2, \ldots, x_{p-1}, u_p, x)$. Alors $l(C_{sx}) = \pi(x_1) - \pi(s) + \pi(x_2) - \pi(x_1) + \ldots + \pi(x) - \pi(x_{p-1}) = \pi(x) - \pi(s) = \pi(x)$. Donc $C_{sx}$ est un plus court chemin de s à x dans R.

# Algorithmes de résolution (1/6)

Soit R=(X, U, l)un réseau et on souhaite chercher l'arborescence des plus courts chemins issus de s. Il existe plusieurs algorithmes dont voici les plus importants:

***Algorithme de Bellman***

Hypothèse: le graphe G=(X,U) ne contient pas de circuit

Principe: Calculer de proche en proche les plus courtes distances issues de s. On ne calcule la plus courte distance de s à un sommet x que si l'on a déjà calculé celles de ses prédécesseurs.

Notations:

S: ensemble des sommets pour lesquels la plus courte distance est calculée

$S^-$ est l'ensemble complémentaire de S dans X ($S^-$= X-S)

A: ensemble des arcs appartenant à l'arborescence

$\pi$: application « potentiel »

# Algorithmes de résolution (2/6)

## ***Enoncé de l'algorithme de Bellman***

(0) Poser $S=\{s\}$ $A=\varnothing$ $\pi(s)=0$

(1) Tester s'il existe un sommet $y \in S^-$ tq $\Gamma^-(y) \subset S$

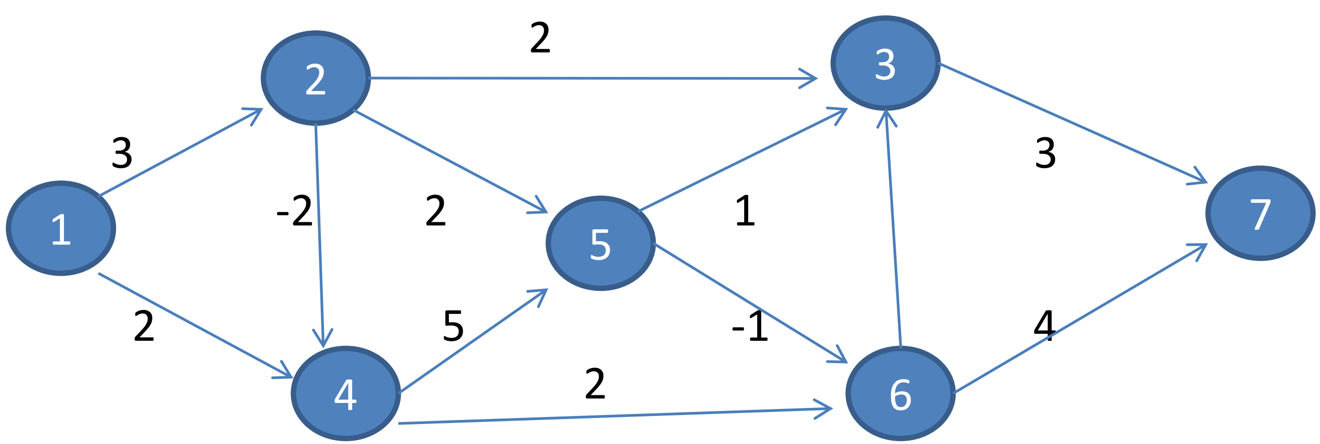
Si oui, aller en 2)

Sinon, terminer soit S=X ou s n'est pas racine

2) Calculer $\pi(y) = \underset{\{u \in U \text{ et } T(u)=y\}}{\text{Min}} \{\pi[I(u)]+l(u)\} = \pi[I(\hat{u})] + l(\hat{u})$

Poser $A = A \cup \{\hat{u}\}$, $S = S \cup \{y\}$

***Exemple 1***: Soit le Réseau:



# Algorithmes de résolution (3/6)

***Algorithme de Djikstra***

***Hypothèse***: Les longueurs des arcs du graphe G=(X,U) sont positives ou nulles

***Principe***: Calculer de proche en proche les plus courtes distances issues de s. Les sommets s'introduisent dans S selon l'ordre croissant de leurs distances à s

***Notations***:

Même signification pour S, $S^-$ A et $\pi$

$\sigma$: dernier sommet inséré dans S.

***Enoncé:***

0) Poser $S=\{s\}$ $A=\emptyset$ $\pi(s)=0$ $\pi(x)= +\infty$ $\sigma=s$

1) Examiner tous les arcs de type $\hat{u}=(\sigma, y)$ avec $y\in S^-$

   Tester si $\pi(\sigma)+ l(\sigma, y)< \pi(y)$

   alors poser $\pi(y)= \pi(\sigma)+l(\sigma, y)$ $A(y)=\hat{u}$ et aller en 2)

2) Choisir $z \in S^-$ $\pi(z)= \text{Min}\{\pi[x), x \in S^-\}$

   Tester si $\pi(z)= +\infty$,

   si oui, terminer s n'est pas racine

   si non, poser $\sigma=z$ $S= S \cup \{z\}$, aller en 3)

3) Tester si S=X

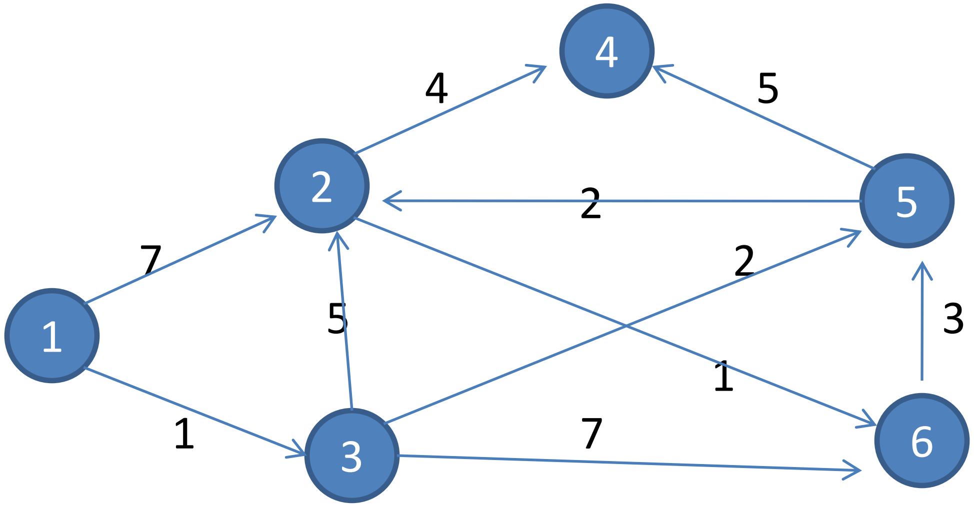
   Si oui, terminer

   Si non, aller en 1)

Remarque: La complexité de cet algorithme est en $O(n^2)$.

# Algorithmes de résolution (4/6)

Exemple 2: Soit le réseau R suivant




1) Vérifier l'existence de solutions aux problèmes A, B et C
2) Résoudre par Djikstra

# Algorithmes de résolution (5/6)

## Algorithme Général de Ford

***Principe:***

Démarrer d'une arborescence réalisable de départ et l'améliorer d'itération en itération pour obtenir l'arborescence optimale où

Met en évidence l'existence de circuit absorbant auquel cas le problème n'a pas de solution.

***Enoncé:***

(0) Démarrer avec une arborescence réalisable A (utiliser Djikstra), s la racine et $\pi$ le vecteur des plus courtes distances issues de s. Les arcs de A sont marqués.

(1) Chercher un arc u=(i, j) $\notin$ A et vérifiant $\delta = [\pi(j) - \pi(i)] - l(u) > 0$

Si un tel arc n'existe pas, terminer A est optimale

Tester si $A \cup \{u\}$ crée un circuit

Si oui, terminer, il est absorbant

Si non (un cycle), aller en 2)

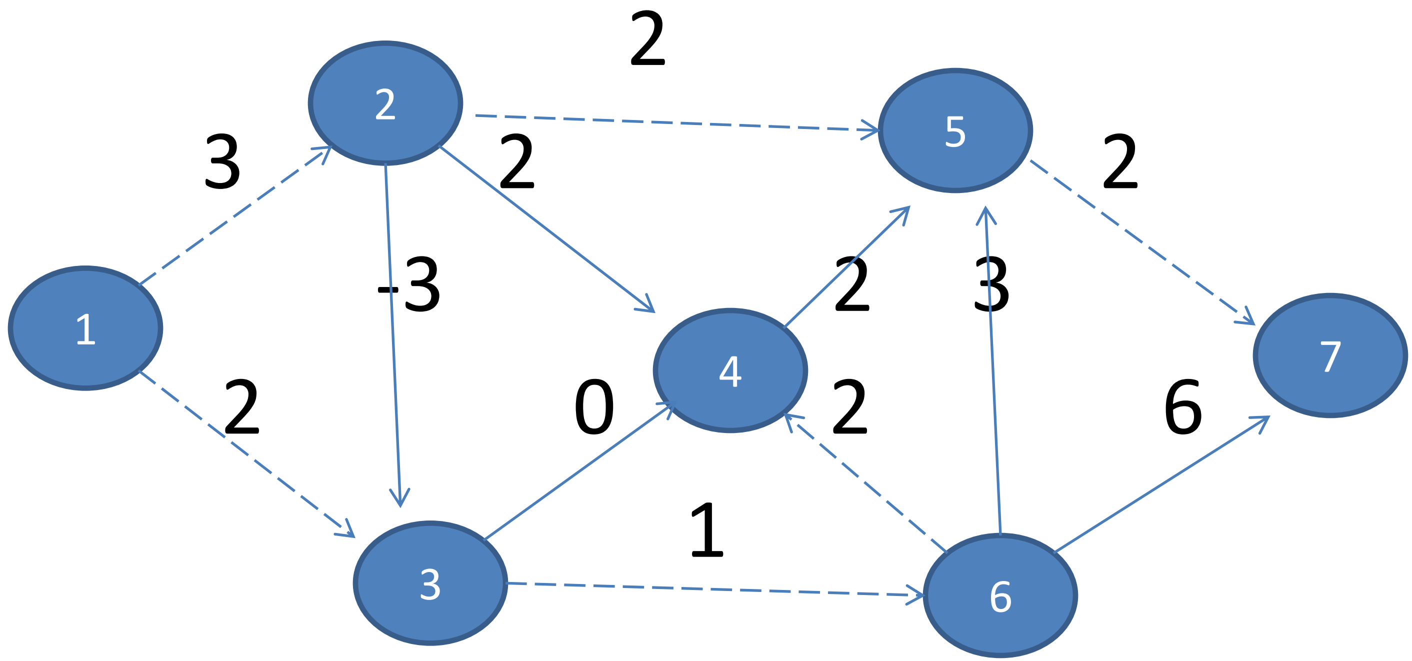
(2) Soit v un arc de A tel que T(v)=j. Poser $A = A - \{v\} \cup \{u\}$

Soit le sous ensemble de sommets $S = \{j\} \cup \{x \in X / \text{ x est descendant de j dans } (X,A)\}$.

Poser $\forall y \in S \; \pi(y) = \pi(y) - \delta$ et aller en 1)

# Algorithmes de résolution (6/6)

Exemple 3: Soit le réseau R




L'arborescence réalisable de départ est formée des arcs en pointillée. Dessiner cette arborescence et améliorer la par l'algorithme général. Donner l'arborescence optimale.