ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Optimisation dans les réseaux

Recherche Opérationnelle

GC-SIE

Le problème de transbordement

Énoncé

$$\min \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij}$$

sous contraintes

$$y_i = s_i \quad \forall i \in \mathcal{N}$$

$$b_{ij} \leq x_{ij} \leq c_{ij} \quad \forall (i,j) \in \mathcal{A}$$

$$y_i = \sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji}$$

Dualité

- Lagrangien

$$L(x,p) = \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} + \sum_{i\in\mathcal{N}} p_i \left(s_i - \sum_{j|(i,j)\in\mathcal{A}} x_{ij} + \sum_{j|(j,i)\in\mathcal{A}} x_{ji} \right)$$

$$= \sum_{(i,j)\in\mathcal{A}} (a_{ij} + p_j - p_i)x_{ij} + \sum_{i\in\mathcal{N}} s_i p_i$$

- **Fonction duale**

$$q(p) = \min_x \{L(x,p) | b_{ij} \leq x_{ij} \leq c_{ij}, (i,j) \in \mathcal{A}\}$$

Dualité

- Comme L(x,p) est séparable en x,

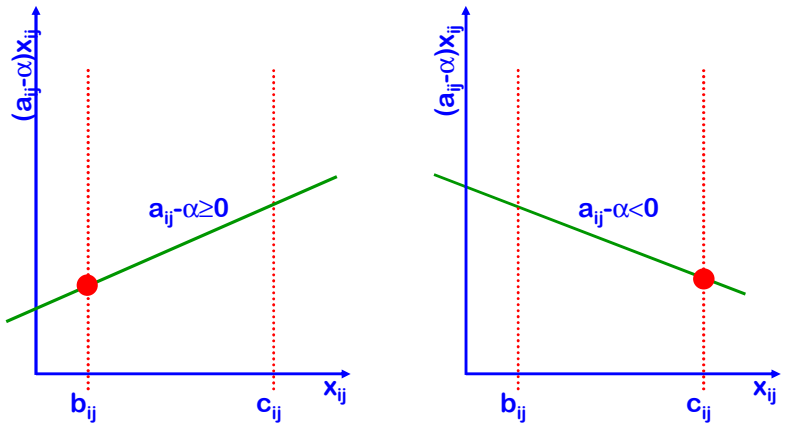
$$q(p) = \sum_{(i,j)\in\mathcal{A}} q_{ij}(p_i - p_j) + \sum_{i\in\mathcal{N}} s_i p_i$$

avec

$$q_{ij}(\alpha) = \min_{b_{ij}\le x_{ij}\le c_{ij}} (a_{ij} - \alpha)x_{ij}$$

$$= \begin{cases} (a_{ij} - \alpha)b_{ij} & \text{si } \alpha \le a_{ij} \\ (a_{ij} - \alpha)c_{ij} & \text{si } \alpha > a_{ij} \end{cases}$$

Dualité



Dualité

Condition des écarts complémentaires (CEC)

- La paire (x,p) vérifie la condition des écarts complémentaires si x vérifie les contraintes de capacité et
- $p_i - p_j \leq a_{ij} \quad \forall (i,j) \in \mathcal{A}$ tel que $x_{ij} < c_{ij}$
- $p_i - p_j \geq a_{ij} \quad \forall (i,j) \in \mathcal{A}$ tel que $b_{ij} < x_{ij}$

Note

- $p_i - p_j = a_{ij} \quad \forall (i,j) \in \mathcal{A}$ tel que $b_{ij} < x_{ij} < c_{ij}$

Dualité

Théorème :

- Un vecteur de flot admissible x* et un vecteur p* satisfont la CEC ssi x* et p* sont solutions primales et duales (resp.) et les coûts optimaux sont égaux.

Dualité

Note :

- Si $b_{ij} = 0$ et $c_{ij} = +\infty$

La CEC

- $p_i - p_j \leq a_{ij}$ $\forall (i,j) \in \mathcal{A}$ tel que $x_{ij} < c_{ij}$
- $p_i - p_j \geq a_{ij}$ $\forall (i,j) \in \mathcal{A}$ tel que $b_{ij} < x_{ij}$

s'écrit

- $p_i - p_j \leq a_{ij}$ $\forall (i,j) \in \mathcal{A}$
- $p_i - p_j = a_{ij}$ $\forall (i,j) \in \mathcal{A}$ tel que $x_{ij} > 0$

Transformations

Mettre les capacités inférieures à 0

$$b_{ij} \leq x_{ij} \leq c_{ij}$$

- Posons $z_{ij} = x_{ij} - b_{ij}$ ou $x_{ij} = z_{ij} + b_{ij}$
- Fonction objectif

$$\sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} = \sum_{(i,j)\in\mathcal{A}} a_{ij}z_{ij} + \cancel{\sum_{(i,j)\in\mathcal{A}} a_{ij}b_{ij}}$$

Transformations

- Contraintes d'offre-demande

$$\sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji} = s_i$$

$$\sum_{j|(i,j)\in\mathcal{A}} z_{ij} - \sum_{j|(j,i)\in\mathcal{A}} z_{ji} = \bar{s}_i$$

$$\bar{s}_i = s_i - \sum_{j|(i,j)\in\mathcal{A}} b_{ij} + \sum_{j|(j,i)\in\mathcal{A}} b_{ij}$$

Transformations

- Contraintes de capacité

$$b_{ij} \le x_{ij} \le c_{ij}$$
$$b_{ij} \le z_{ij}+b_{ij} \le c_{ij}$$
$$0 \le z_{ij} \le c_{ij}-b_{ij}$$

On peut donc supposer $b_{ij} = 0$ $\forall (i,j)\in\mathcal{A}$ sans perte de généralité.

Transformations

Supprimer les contraintes supérieures de capacité

- Idée : ajouter des variables d'écart

$$x_{ij} + z_{ij} = c_{ij} \text{ avec } z_{ij} \geq 0$$

- Fonction objectif : inchangée
- Contraintes de capacité :

$$0 \leq x_{ij} \leq c_{ij}$$

$$x_{ij} \leq c_{ij} \Leftrightarrow z_{ij} \geq 0$$

$$x_{ij} \geq 0 \text{ et } z_{ij} \geq 0$$

Transformations

- Contraintes d'offre-demande

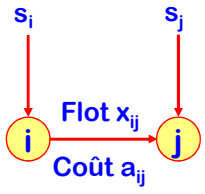
$$\sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji} = s_i$$

$$\sum_{j|(i,j)\in\mathcal{A}} c_{ij} - \sum_{j|(i,j)\in\mathcal{A}} z_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji} = s_i$$

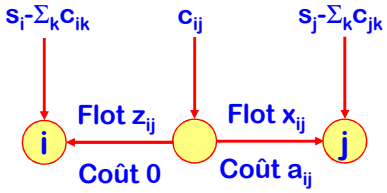
$$- \sum_{j|(i,j)\in\mathcal{A}} z_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji} = s_i - \sum_{j|(i,j)\in\mathcal{A}} c_{ij}$$

Transformations

- Interprétation :



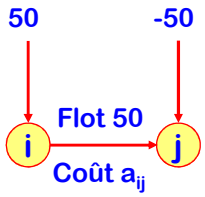
Capacité $[0,c_{ij}]$



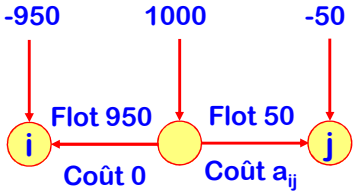
Capacité $[0,\infty[$

Transformations

- Interprétation :



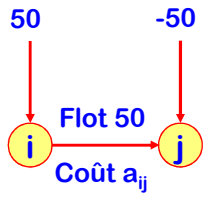
Capacité [0,1000]



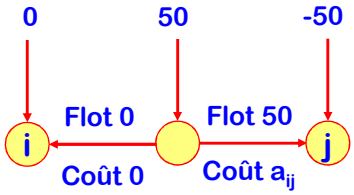
Capacité $[0,\infty[$

Transformations

- Interprétation :



Capacité [0,50]



Capacité [0,∞[

Problème transformé

$$\min \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij}$$

sous contraintes

$$y_i = s_i \quad \forall i \in \mathcal{N}$$

$$x_{ij} \geq 0$$

$$y_i = \sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji}$$

Problème transformé

Attention :

- En l'absence de capacités supérieures, le problème peut être non borné.
- Cela n'arrive cependant pas s'il s'agit d'un problème transformé.
- Le problème est non borné ssi il possède au moins un solution admissible, et s'il existe un cycle avançant de coût négatif.

Méthode du simplexe

Idée : exploiter explicitement la structure de réseau.

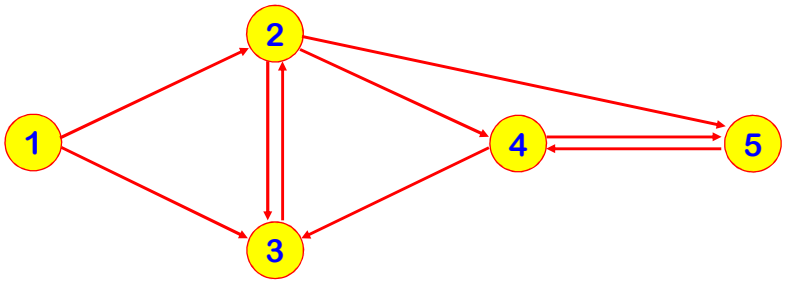
Élément principal : arbre maximal

Définitions :

- Un arbre est un graphe connexe sans cycle
- Un arbre maximal d'un graphe G est un sous-graphe qui soit un arbre et qui inclue tous les nœuds du graphe
- Une feuille est un nœud de degré 1 dans un arbre.

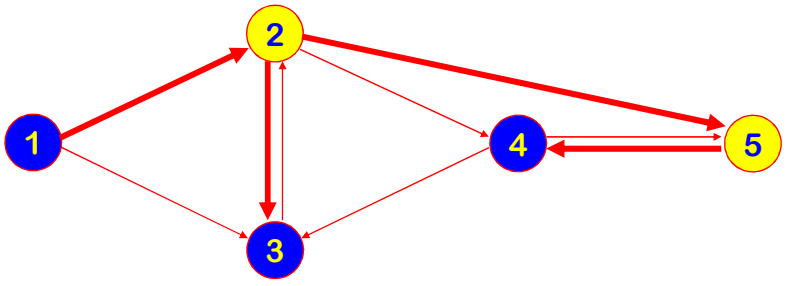
Méthode du simplexe

- Graphe :



Méthode du simplexe

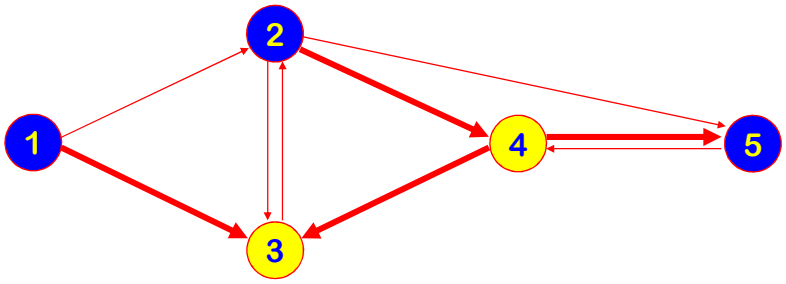
- **Arbre maximal :**



i = feuilles

Méthode du simplexe

- **Arbre maximal :**



i = feuilles

Méthode du simplexe

Propriétés :

- Soit T le sous-graphe d'un graphe à N nœuds.

1. Si T est sans cycle et possède au moins un arc, alors il a au moins une feuille.
2. T est un arbre maximal ssi T est connexe et contient N nœuds et N-1 arcs.
3. Si T est un arbre, il y a un chemin unique reliant deux nœuds i et j de cet arbre.
4. Soit e $\notin$ T un arc dont les extrémités sont dans T. Le graphe T$\cup${e} contient un cycle simple unique, dont e est un arc avançant.
5. Si T est un arbre contenant (i,j), et si (i,j) est supprimé, les arcs restant forment deux arbres disjoints, l'un contenant i l'autre j.

Méthode du simplexe

- La base en programmation linéaire « générale » peut être représentée ici grâce aux arbres maximaux.
- Pour chaque arbre maximal T, il existe un vecteur de flots unique x tel que
  - x vérifie les contraintes de conservation des flots
  - $x_i = 0$ si $i \notin T$

Méthode du simplexe

Procédure :

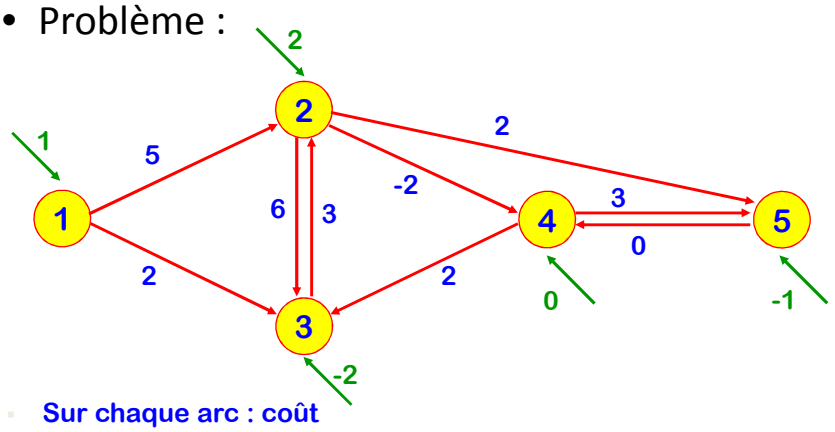
- Soit R=T, x=0, $w_i=s_i$ $\forall i$.
- Pas 1 :
  - choisir une feuille i de R.
  - si (i,j) est l'unique arc incident à i

    $x_{ij}=w_i$ et $w_j=w_j+w_i$
  - si (j,i) est l'unique arc incident à i

    $x_{ij}=-w_i$ et $w_j=w_j+w_i$
- Pas 2 : supprimer i et son arc incident de R. Si R n'a plus que 1 nœud, STOP. Sinon retour au pas 1.

Méthode du simplexe

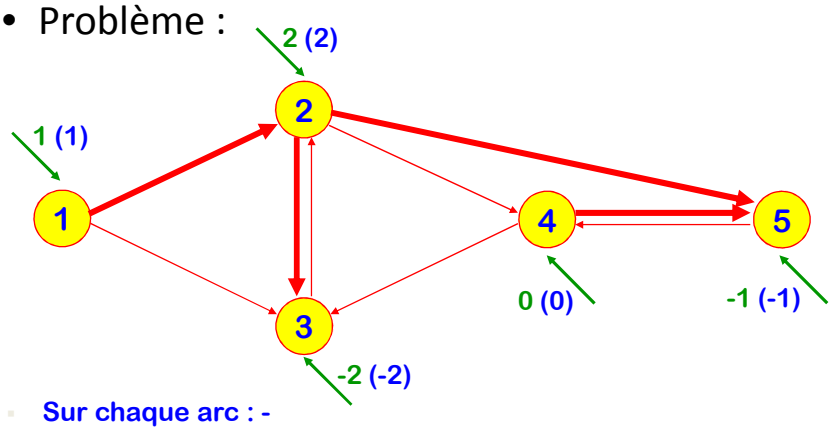
- Problème :



- Sur chaque arc : coût
- Sur chaque nœud : offre

Méthode du simplexe

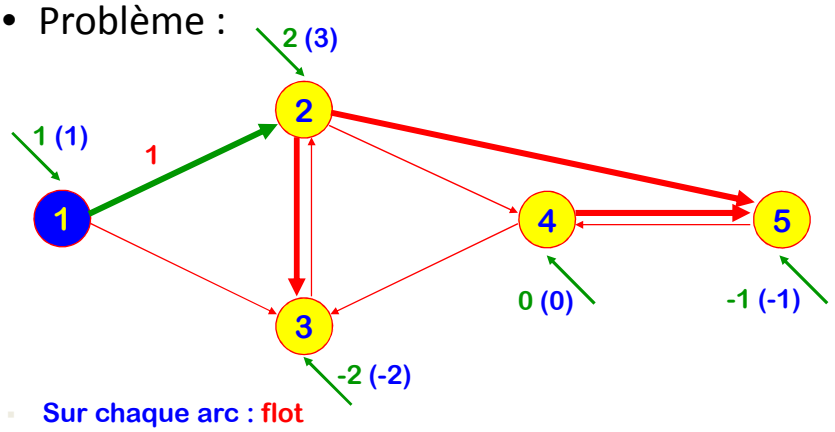
- Problème :



- Sur chaque arc : -
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

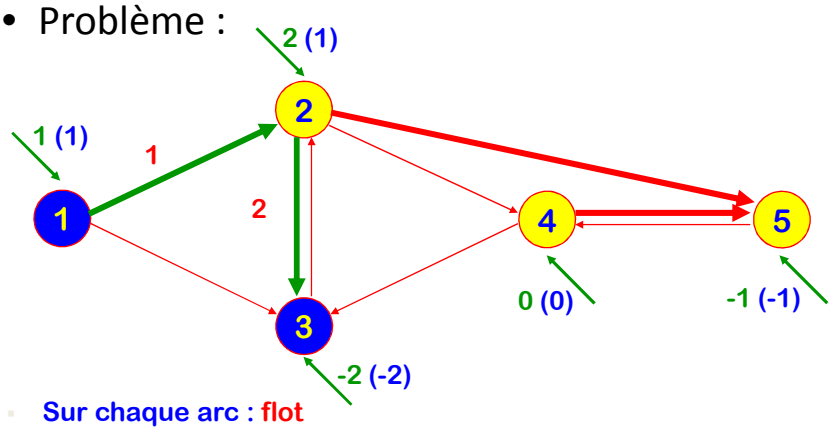
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

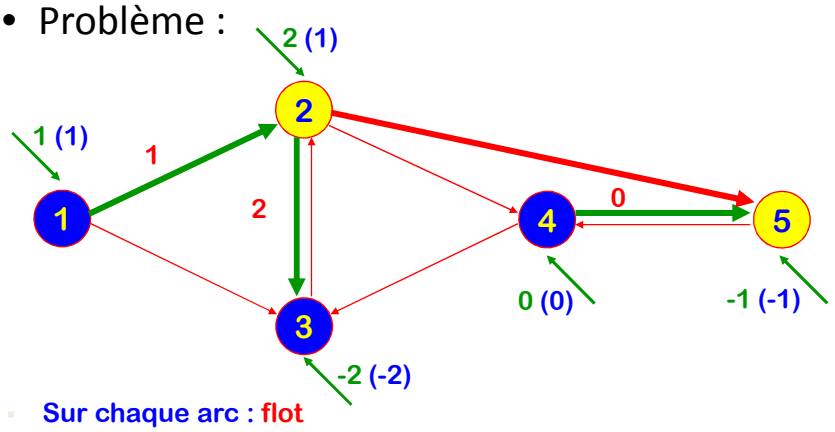
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

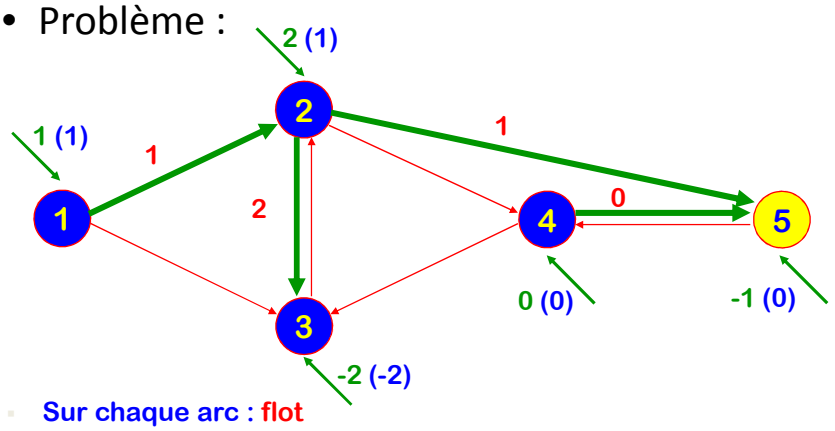
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

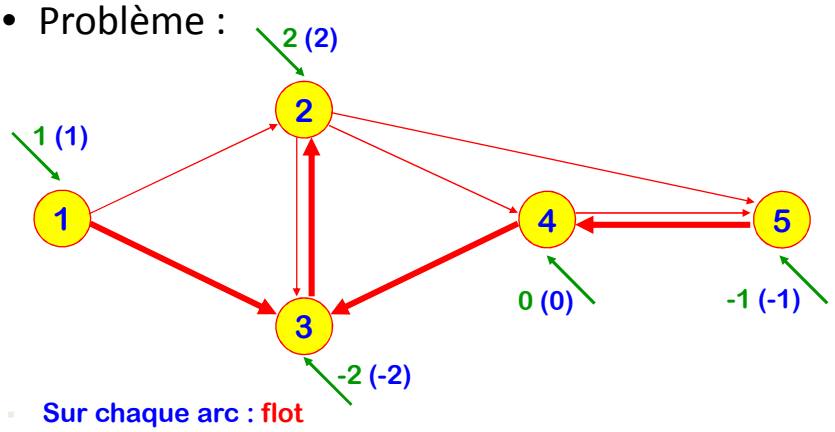
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

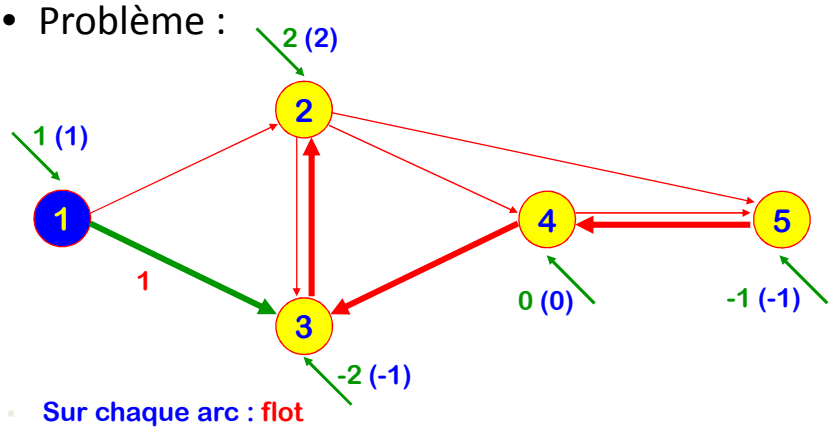
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

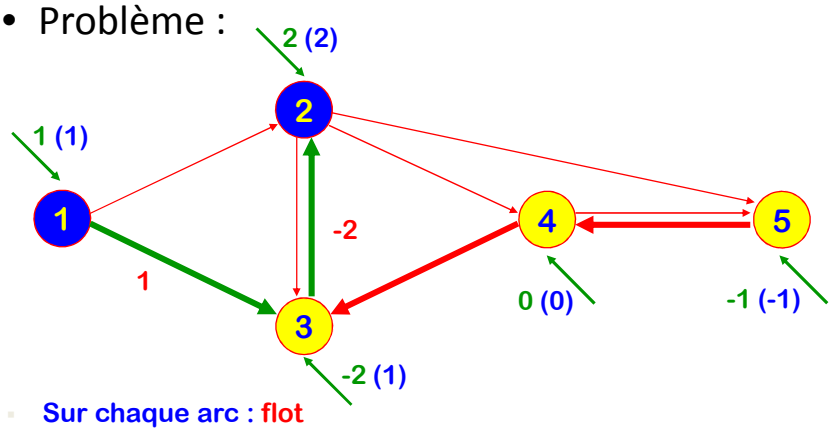
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

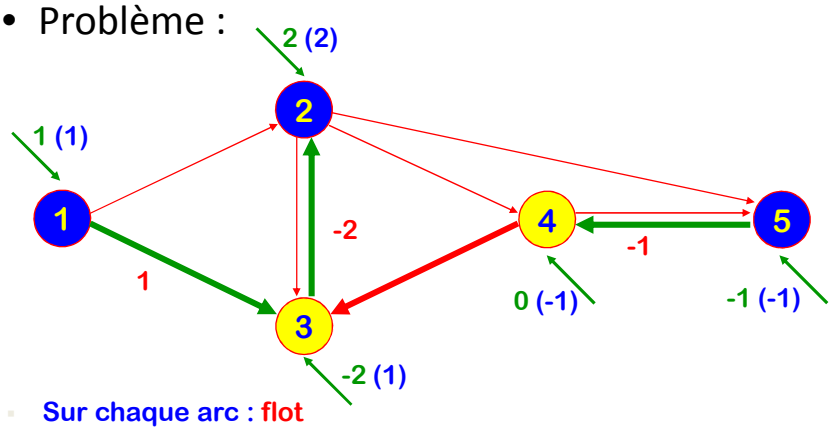
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

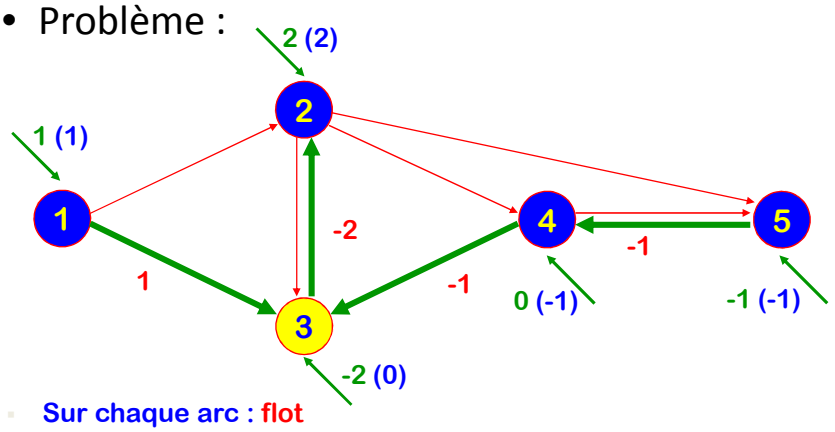
- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

- Problème :



- Sur chaque arc : flot
- Sur chaque nœud : offre (w_i)

Méthode du simplexe

- Ce vecteur de flots est appelé une solution de base
- Si, de plus, le vecteur de flots vérifie les contraintes de capacité $x_{ij} \geq 0$, il est appelé une solution de base admissible.
- Un arbre maximal sera dit admissible si le vecteur de flots correspondant est une solution de base admissible.

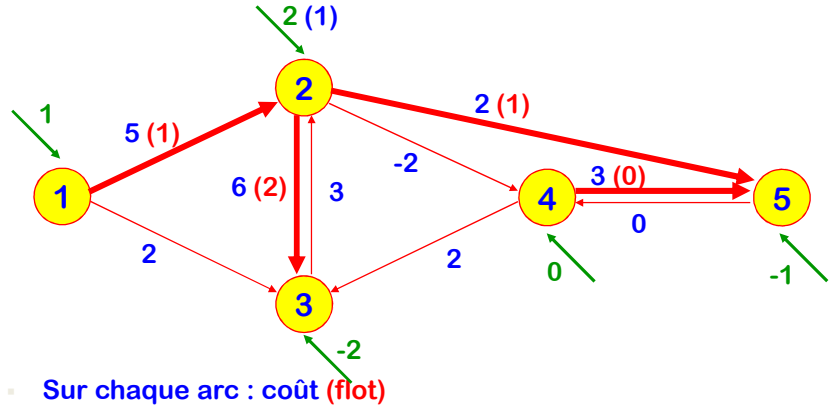
Méthode du simplexe

Aperçu de la méthode

- Soit un arbre maximal admissible initial.
- Chaque itération (pivotage) génère un autre arbre admissible dont le coût n'est pas plus élevé que le précédent.
- Chaque itération est composée de trois opérations principales
  1. Ajout d'un arc à l'arbre afin de former un cycle à coût négatif
  2. Envoyer le plus de flot possible le long de ce cycle, sans violer les contraintes
  3. Supprimer un arc du cycle pour obtenir à nouveau un arbre.

Méthode du simplexe

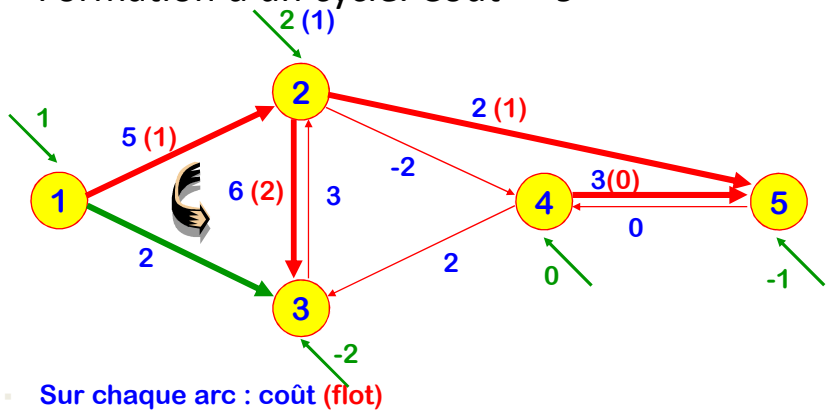
- Arbre maximal initial :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : offre

Méthode du simplexe

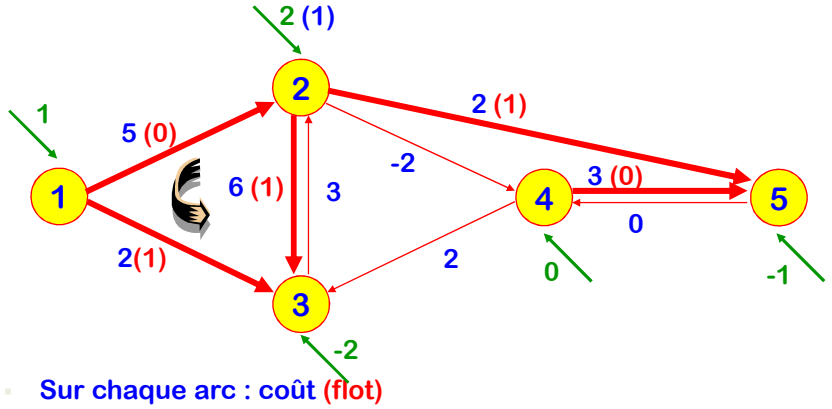
- Formation d'un cycle. Coût = -9



- Sur chaque arc : coût (flot)
- Sur chaque nœud : offre

Méthode du simplexe

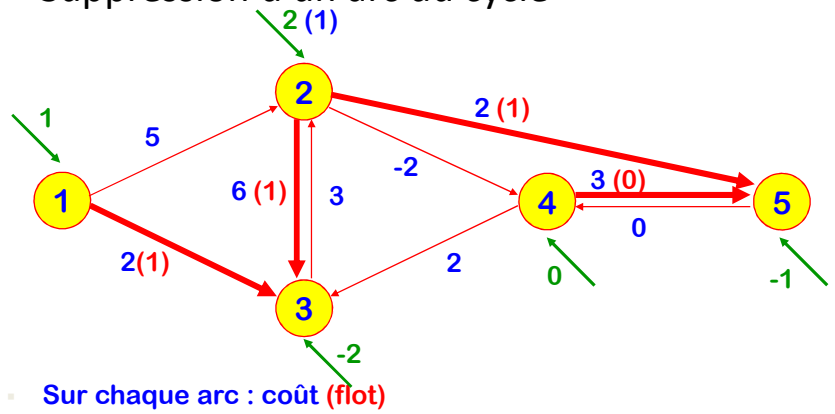
- Envoi d'une unité de flot



- Sur chaque arc : coût (flot)
- Sur chaque nœud : offre

Méthode du simplexe

- Suppression d'un arc du cycle



- Sur chaque arc : coût (flot)
- Sur chaque nœud : offre

Méthode du simplexe

Questions :

- Comment choisir l'arc entrant ?
- Comment choisir l'arc sortant ?
- Comment gérer les cas de dégénérescence ?

Choix de l'arc entrant

- Idée : utiliser la condition des écarts complémentaires
- $p_i - p_j \leq a_{ij} \quad \forall (i,j) \in \mathcal{A}$
- $p_i - p_j = a_{ij} \quad \forall (i,j) \in \mathcal{A}$ tel que $x_{ij} > 0$
- Nous allons affecter des prix p_i aux nœuds tels que

$$p_i - p_j = a_{ij} \quad \forall (i,j) \in T$$

Choix de l'arc entrant

Procédure récursive :

- Soit un nœud r arbitraire (racine).
- p_r est initialisé à une valeur quelconque.
- $p_i = +\infty$, $i \neq r$
- CalculeVoisins(r) ;

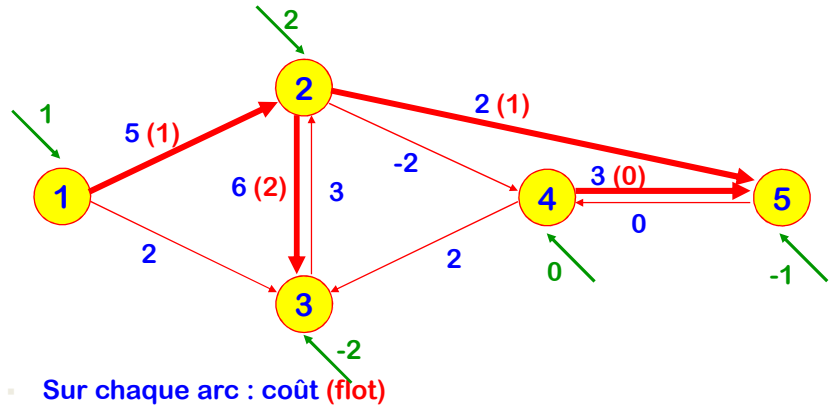
Choix de l'arc entrant

CalculVoisins(i)

- Pour tout j adjacent à i t.q. $p_j = +\infty$
- Si $(i,j) \in T$, alors $p_j = p_i - a_{ij}$
- Si $(j,i) \in T$, alors $p_j = p_i + a_{ij}$
- CalculVoisins(j)

Choix de l'arc entrant

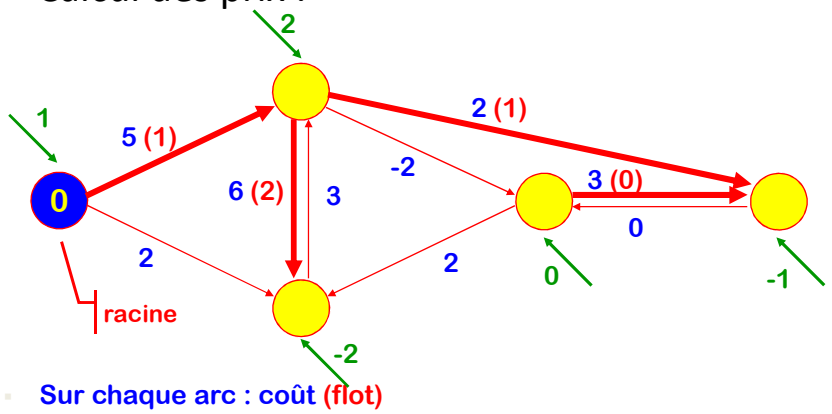
- Arbre maximal :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : offre [pi]

Choix de l'arc entrant

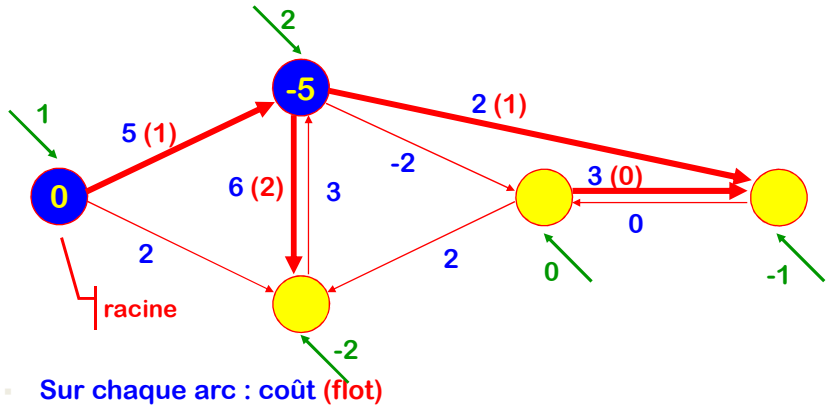
- Calcul des prix :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

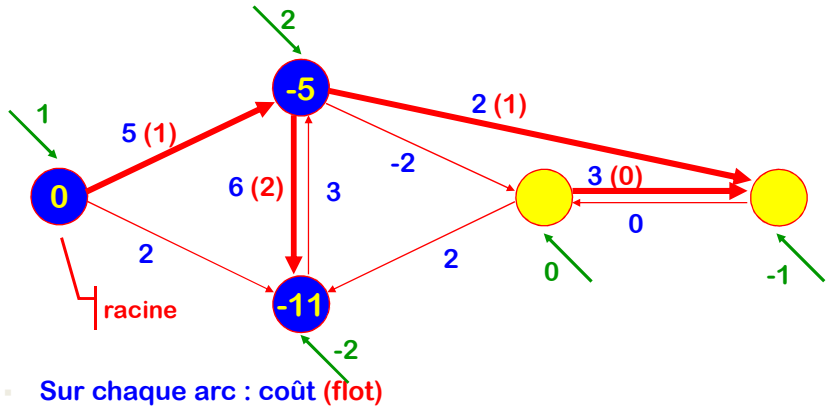
- Arbre maximal :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

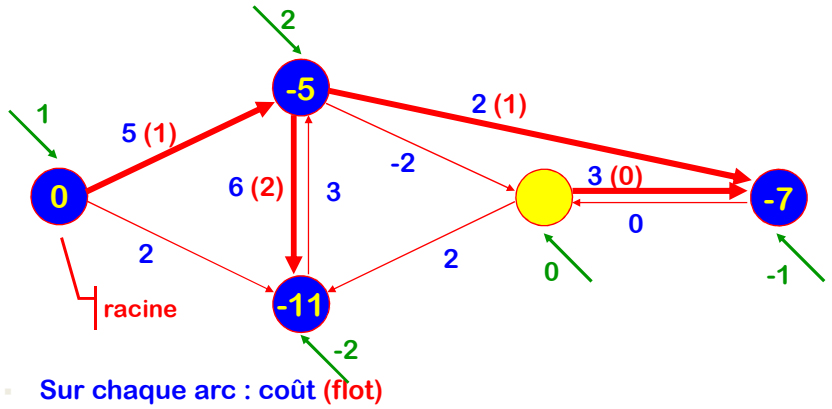
- Arbre maximal :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

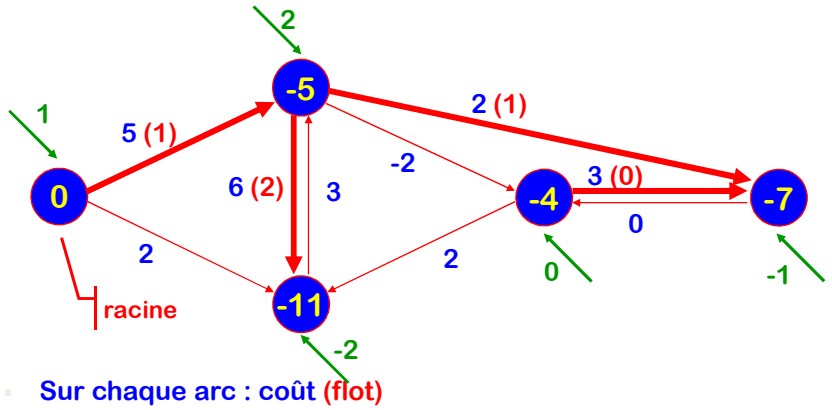
- Arbre maximal :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

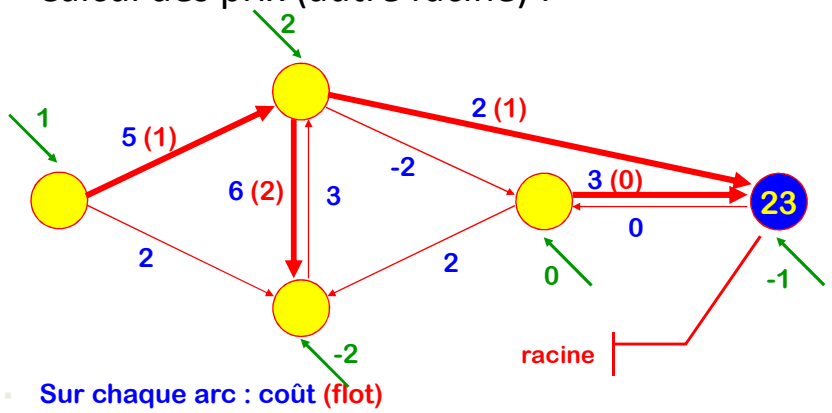
- Arbre maximal :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

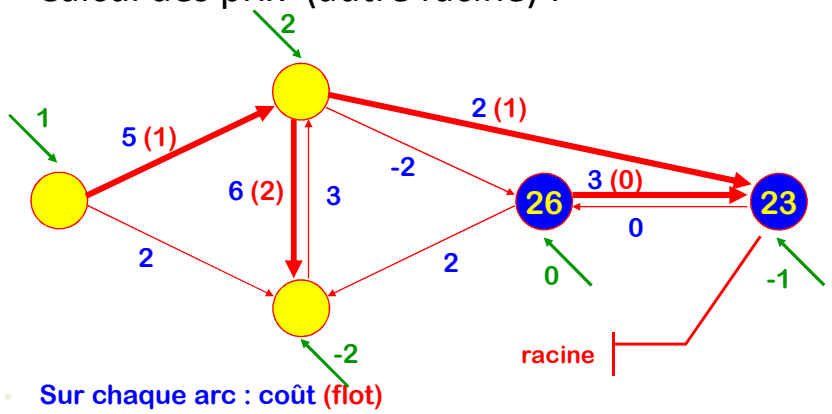
- Calcul des prix (autre racine) :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

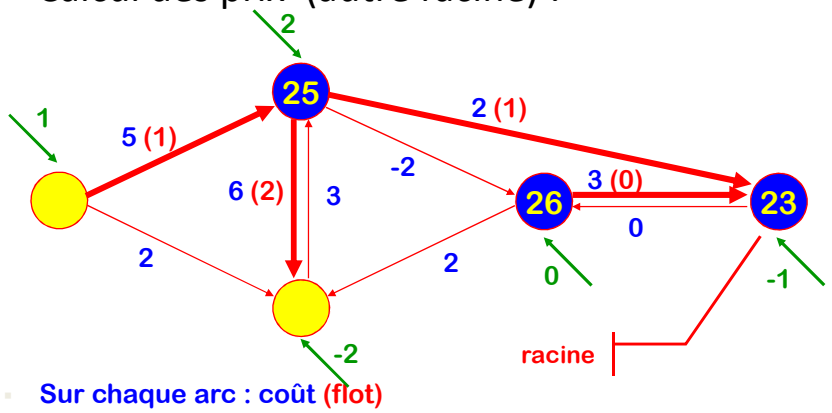
- Calcul des prix (autre racine) :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

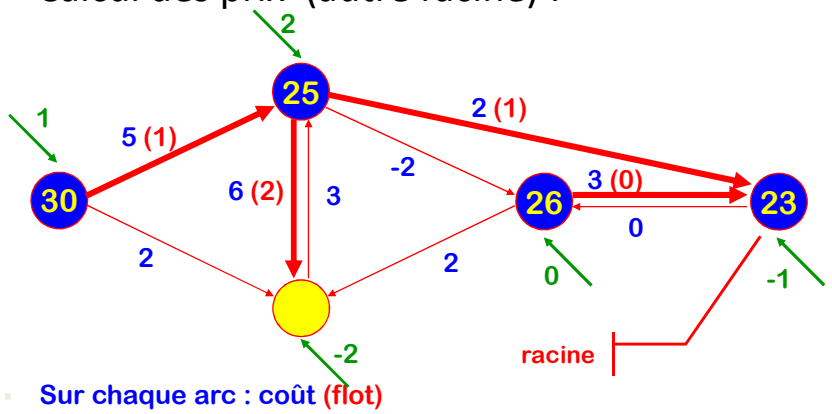
- Calcul des prix (autre racine) :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

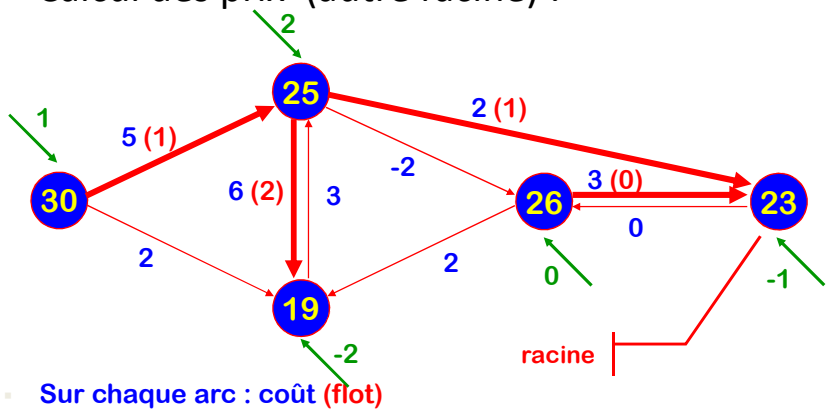
- Calcul des prix (autre racine) :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

- Calcul des prix (autre racine) :



- Sur chaque arc : coût (flot)
- Sur chaque nœud : prix

Choix de l'arc entrant

Notes :

- Les prix dépendent
  - du choix de la racine
  - du prix de la racine
- La différence de prix entre deux nœuds quelconques est indépendantes des décisions liées à la racine.
- Notamment, la quantité

$$r_{ij} = a_{ij} + p_j - p_i$$

dépend uniquement de l'arbre maximal T

Choix de l'arc entrant

- Par définition des prix,

$$r_{ij} = a_{ij} + p_j - p_i = 0 \text{ si } (i,j) \in T$$

- Si, de plus, $r_{ij} \geq 0 \ \forall (i,j) \in \mathcal{A}$, la CEC est vérifiée
  - $p_i - p_j \leq a_{ij} \ \ \forall (i,j) \in \mathcal{A}$
  - $p_i - p_j = a_{ij} \ \ \forall (i,j) \in \mathcal{A}$ tel que $x_{ij} > 0$
- Dans ce cas, x est une solution optimale du primal, et p du dual.
- r_{ij} est appelé le coût réduit de (i,j)

Choix de l'arc entrant

- Dans le cas contraire, il existe (k,l) tel que
  - (k,l) $\in \mathcal{A}$,
  - (k,l) $\notin$ T
  - $r_{kl} = a_{kl} + p_l - p_k < 0$
- Si l'on rajoute (k,l) à l'arbre, on crée un cycle. Par convention, (k,l) doit être avançant dans le cycle.

Choix de l'arc entrant

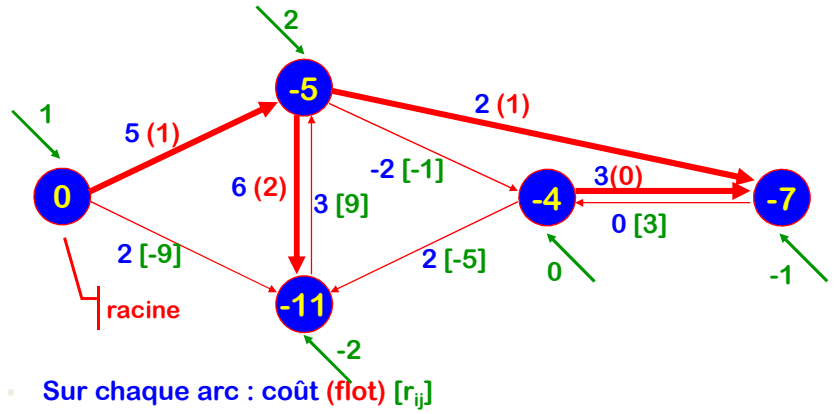
- Coût du cycle formé :

$$\begin{aligned}
&\sum_{(i,j)\in C^+} a_{ij} - \sum_{(i,j)\in C^-} a_{ij} \\
&= \sum_{(i,j)\in C^+} (a_{ij} + p_j - p_i) - \sum_{(i,j)\in C^-} (a_{ij} + p_j - p_i) \\
&= \sum_{(i,j)\in C^+} r_{ij} - \sum_{(i,j)\in C^-} r_{ij} \\
&= r_{kl}
\end{aligned}$$

Cycle à coût négatif

Choix de l'arc entrant

- Solution de base admissible :



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Cycle

- Soit C le cycle formé par T et (k,l)
- Si C^- est vide, tous les arcs sont orientés comme (k,l)
- C est donc un cycle à coût négatif, le long duquel le flot peut être augmenté arbitrairement.
- Le problème est donc non-borné.

Cycle

- Si C^- n'est pas vide, notons

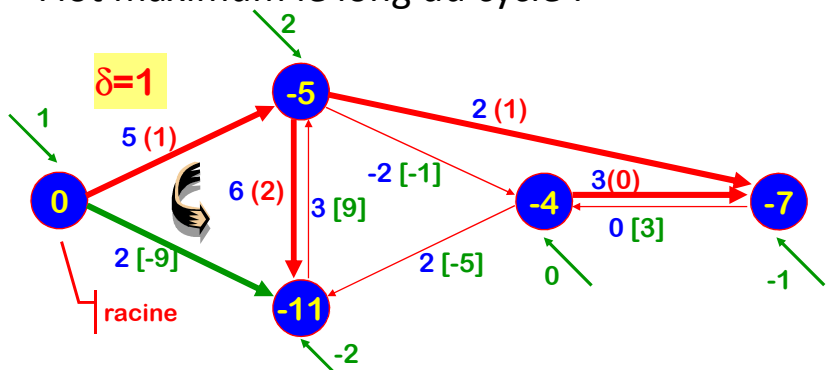
$$\delta = \min_{(i,j)\in C^-} x_{ij}$$

le plus petit flot sur les arcs reculant.

- Il n'est pas possible d'envoyer plus que δ unités de flots sans violer les contraintes de non négativité.

Cycle

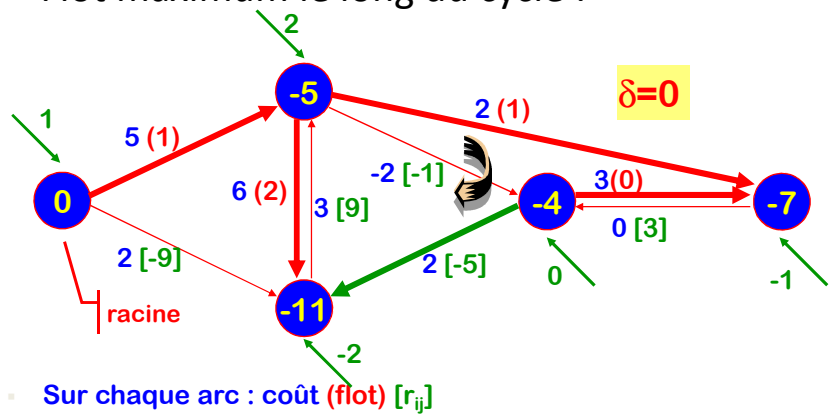
- Flot maximum le long du cycle :



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Cycle

- Flot maximum le long du cycle :



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Choix de l'arc sortant

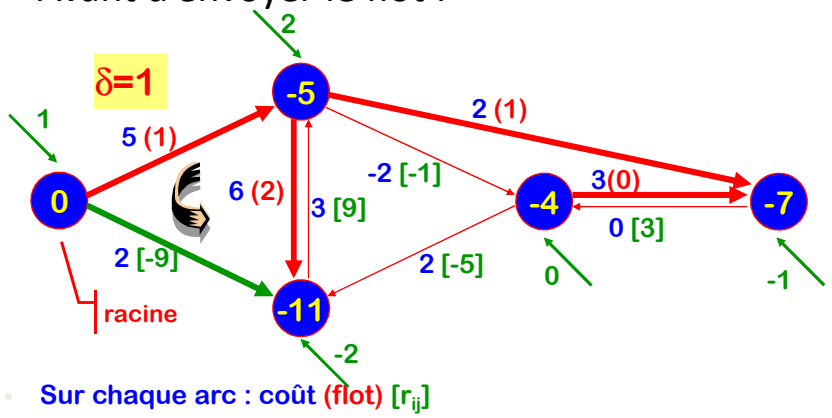
- Le nouveau vecteur de flot sera

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

- **Tout arc (i,j) du cycle tel que $x_{ij}^{+} = 0$ est candidat pour sortir.**

Choix de l'arc sortant

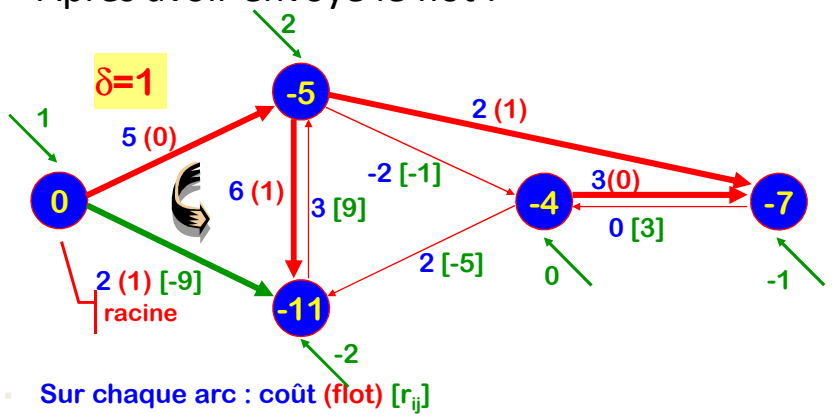
- Avant d'envoyer le flot :



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Choix de l'arc sortant

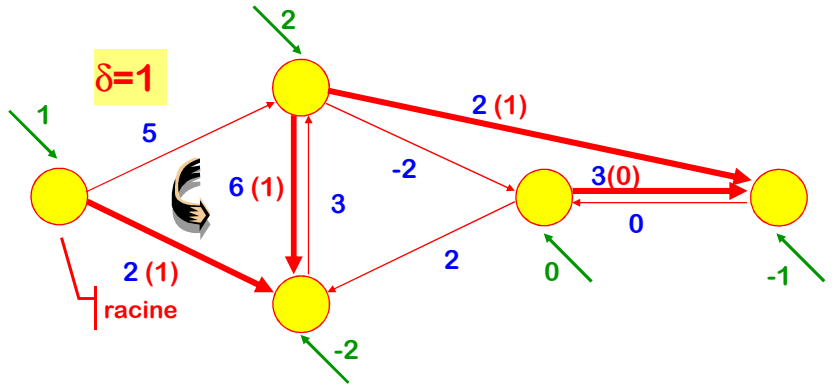
- Après avoir envoyé le flot :



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Choix de l'arc sortant

- Nouvelle solution de base admissible :



- Sur chaque arc : coût (flot)

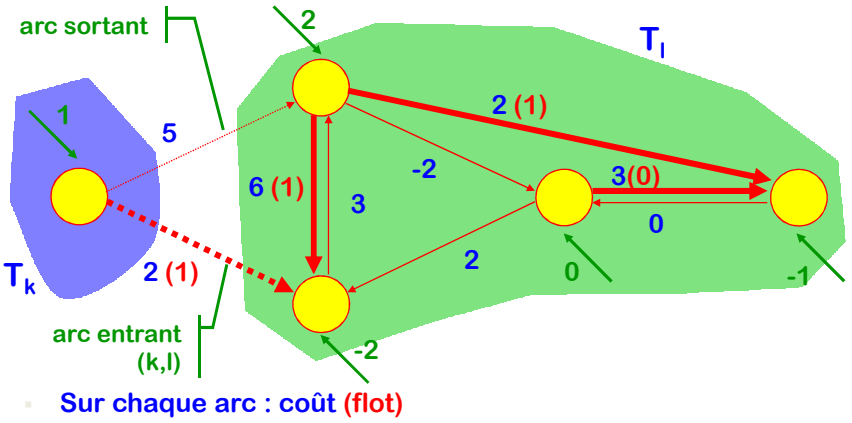
Mise à jour des prix

- Soit (k,l) l'arc entrant
- Soit e l'arc sortant
- Soit

$$T^+ = T + (k,l) - e$$

  l'arbre correspondant à la nouvelle base.
- Considérons le sous-graphe T-e.
- Il est composé de deux arbres :
  - T_k contient le nœud k
  - T_l contient le nœud l

Mise à jour des prix



Mise à jour des prix

Méthode 1

- $p_i^+ = p_i$ si $i \in T_k$
- $p_i^+ = p_i - r_{kl}$ si $i \in T_l$

Méthode K

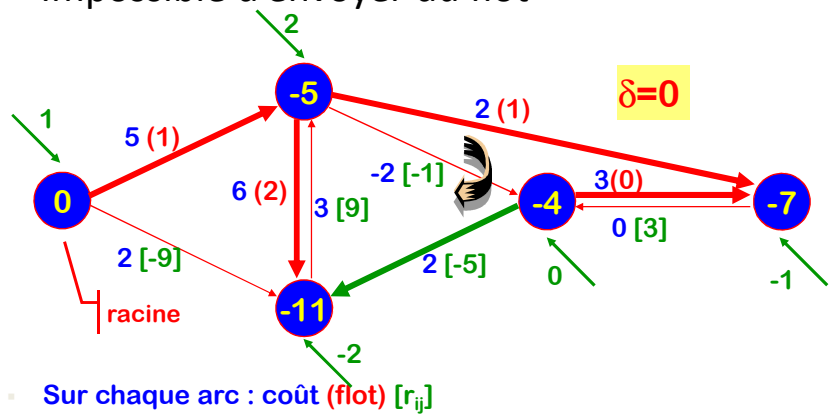
- $p_i^+ = p_i + K$ si $i \in T_k$
- $p_i^+ = p_i - r_{kl} + K$ si $i \in T_l$

Méthode 2 [$K = r_{kl}$]

- $p_i^+ = p_i + r_{kl}$ si $i \in T_k$
- $p_i^+ = p_i$ si $i \in T_l$

Dégénérescence

- Impossible d'envoyer du flot



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Dégénérescence

- On change de base mais la fonction objectif ne change pas.
- Risque que l'algorithme cycle.
- Pour garantir l'absence de cyclage, on introduit la notion de base fortement admissible.

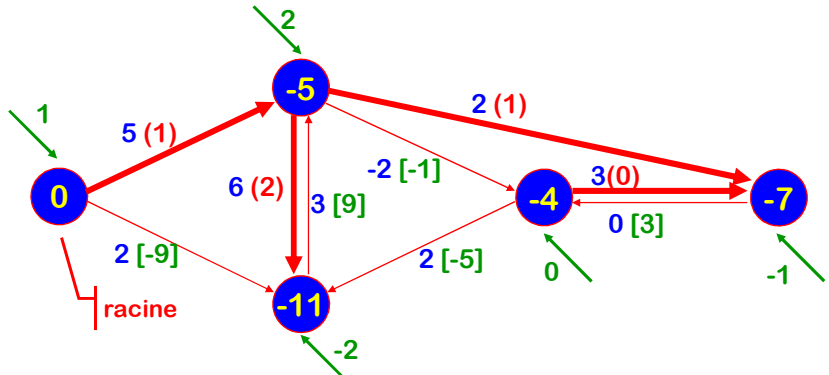
Dégénérescence

Définition

- Soit T un arbre admissible.
- Soit r la racine de l'arbre.
- On dit que (i,j) ∈ T s'écarte de la racine si le seul chemin entre r et j passe par i.
- Un arbre admissible T est fortement admissible si tous les arcs (i,j) tels que $x_{ij} = 0$ s'écartent de la racine.

Dégénérescence

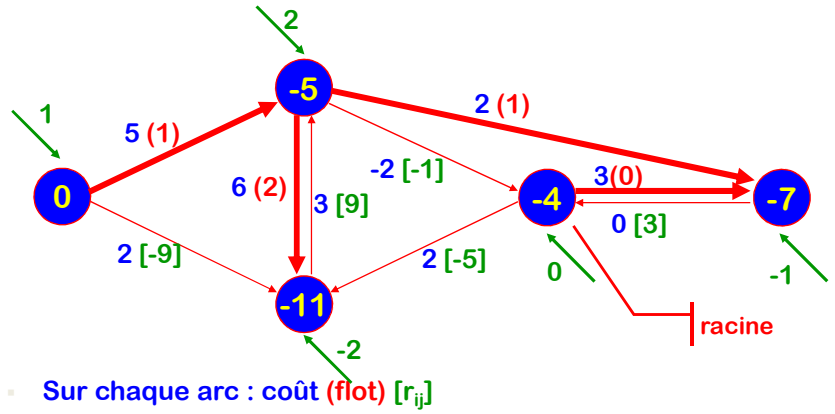
- Arbre admissible, mais pas fortement



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Dégénérescence

- Arbre fortement admissible



- Sur chaque arc : coût (flot) [r_{ij}]
- Sur chaque nœud : prix

Dégénérescence

Théorème

- Si les arbres admissibles générés par la méthode du simplexe sont tous fortement admissibles, alors tous ces arbres sont distincts.
- Dans ce cas, l'algorithme effectuera un nombre fini d'itérations.

Dégénérescence

- Supposons que l'on dispose au début d'un arbre fortement admissible.
- Il faut s'arranger pour que le nouvel arbre produit soit aussi fortement admissible.
- Il faut choisir l'arc sortant de manière appropriée.

Dégénérescence

Procédure de choix de l'arc sortant

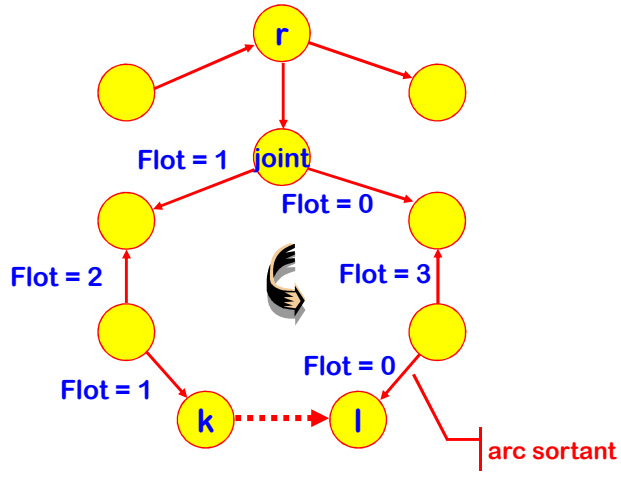
- Soit T un arbre fortement admissible
- Soit (k,l) l'arc entrant
- Soit C le cycle formé par T et (k,l)
- Supposons que C^- est non vide.
- Soit $\delta = \min_{(i,j)\in C^-} x_{ij}$
- Soit $C^* = \{(i,j) \in C^- \mid x_{ij} = \delta\}$ l'ensemble des candidats à sortir.

Dégénérescence

Procédure de choix de l'arc sortant (suite)

- Le joint de C est le premier nœud du cycle sur le chemin entre r et k.
- Choisir comme arc sortant le premier arc de C* rencontré lorsque l'on parcourt le cycle en partant du joint.
- Dans ce cas, le nouvel arbre sera également fortement admissible.

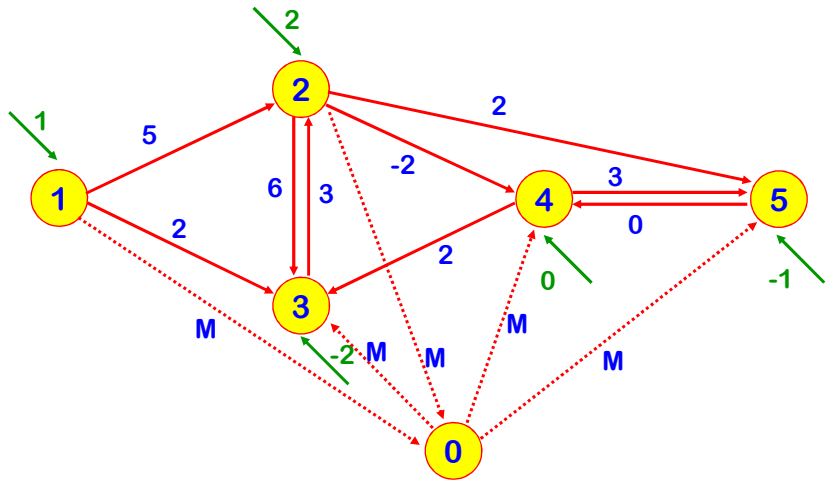
Dégénérescence



Initialisation

- Comment trouver un premier arbre fortement admissible ?
- Idée :
  - ajouter un nœud artificiel 0 avec $s_0=0$
  - ajouter un arc entre 0 et chaque nœud i
    - Arc (i,0) si $s_i > 0$
    - Arc (0,i) si $s_i \leq 0$
  - coût des arcs artificiels : M > 0, très grand

Initialisation

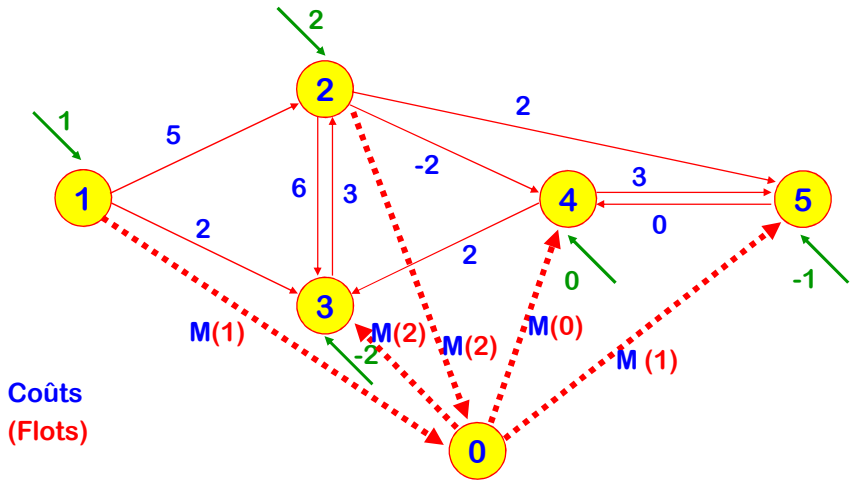


Initialisation

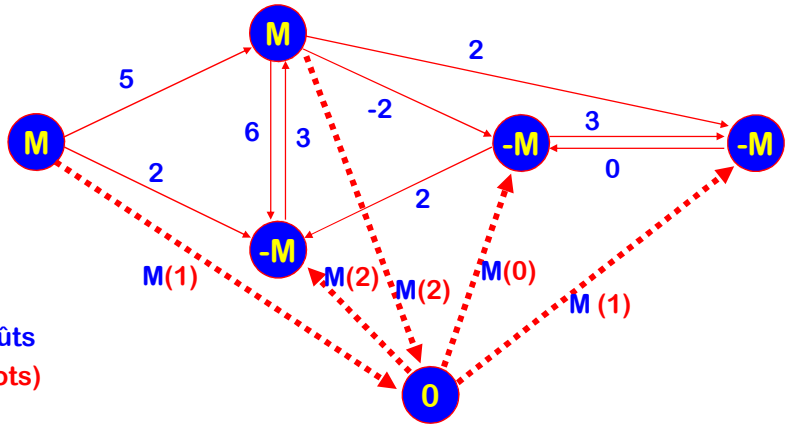
Arbre initial :

- Uniquement les arcs artificiels.
- Racine : nœud 0
- Par construction, les arcs transportant un flot nul s'éloignent de la racine
- Il s'agit donc bien d'un arbre fortement admissible

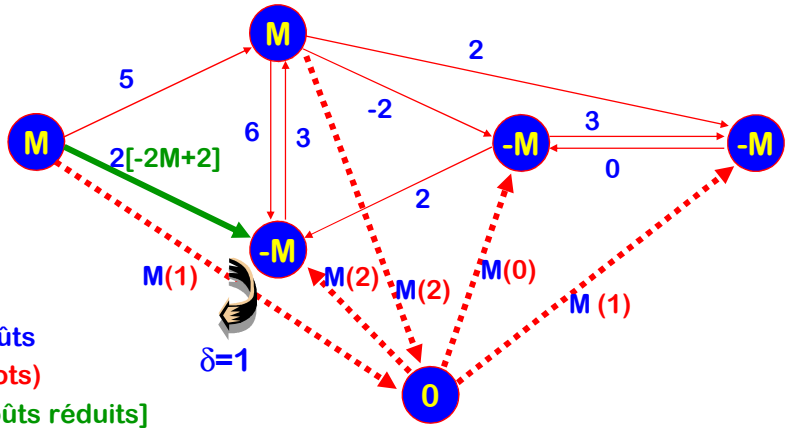
Initialisation



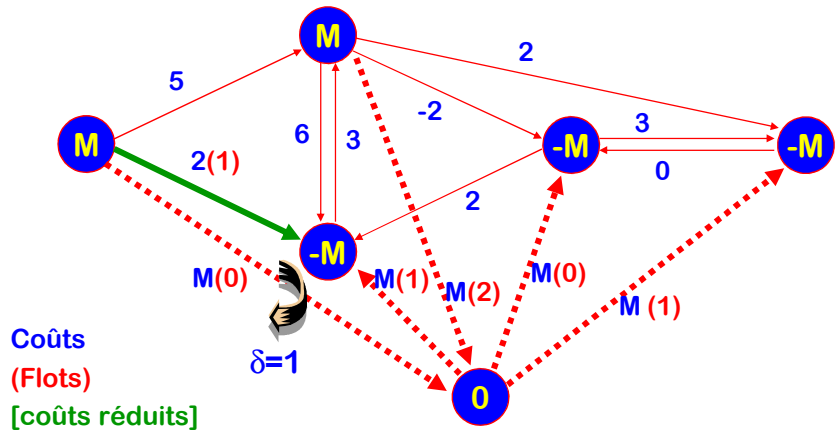
Calcul des prix



Coûts réduits

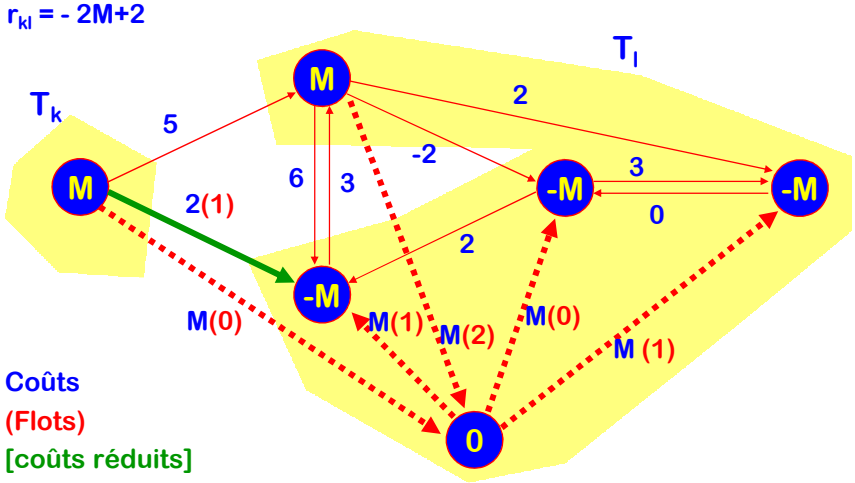


Mise à jour des flots

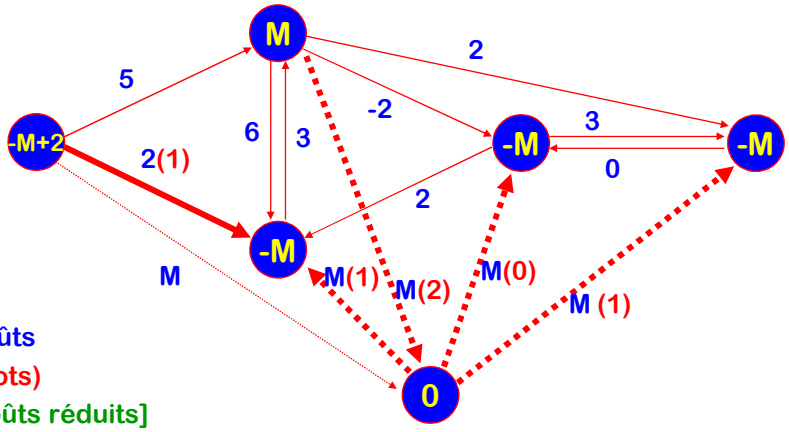


Mise à jour des prix

r_{kl} = - 2M+2

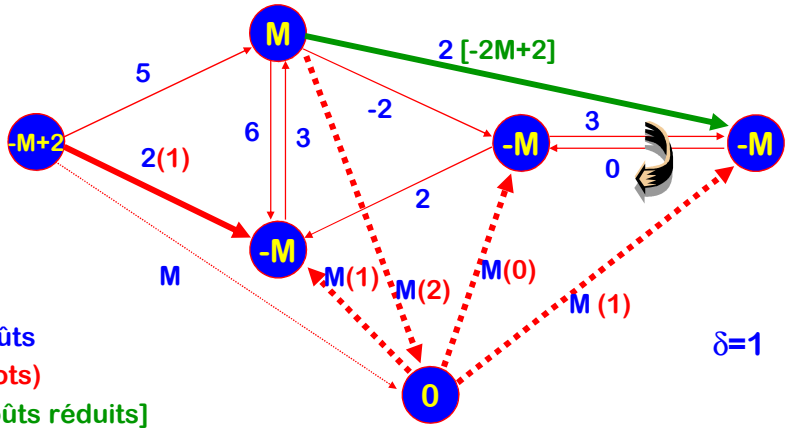


Nouvel arbre



Coûts
(Flots)
[coûts réduits]

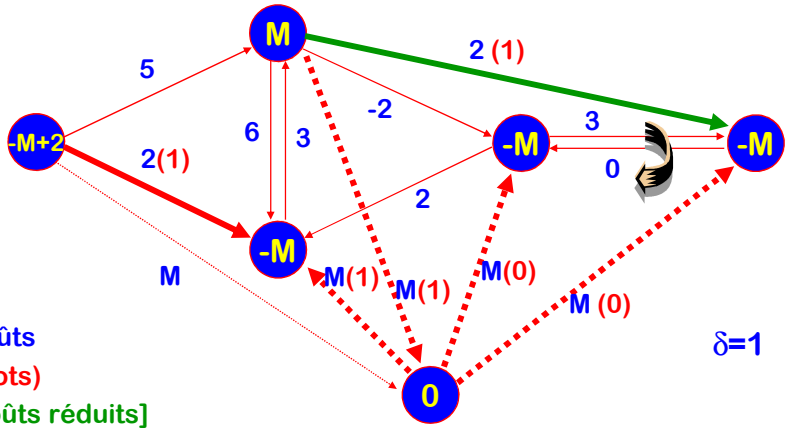
Coûts réduits



Coûts
(Flots)
[coûts réduits]

δ=1

Mise à jour des flots

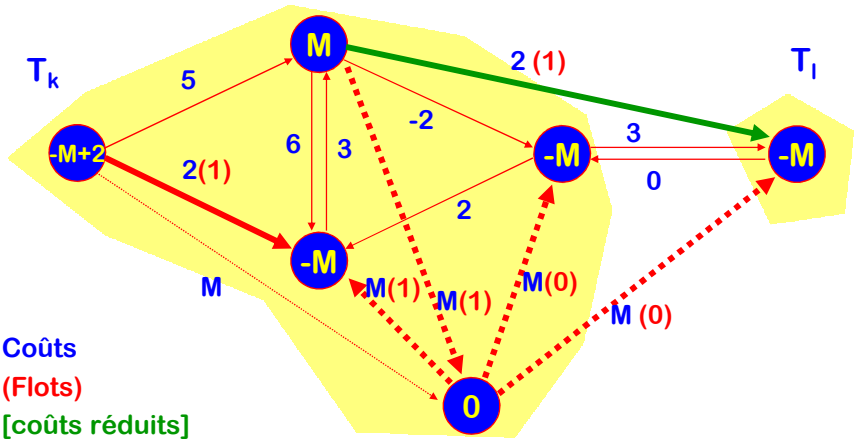


Coûts
(Flots)
[coûts réduits]

δ=1

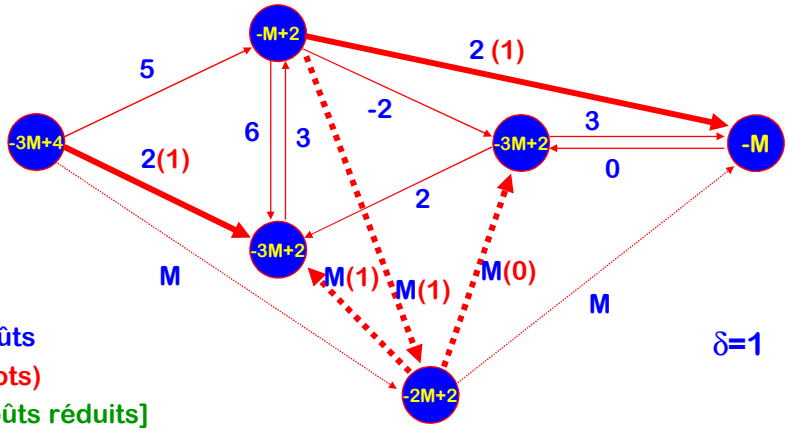
Mise à jour des prix

r_{kl} = - 2M+2



Coûts
(Flots)
[coûts réduits]

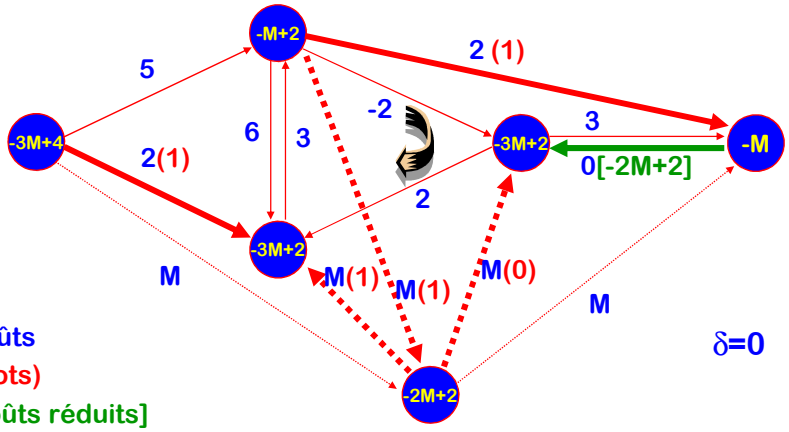
Nouvel arbre



Coûts
(Flots)
[coûts réduits]

δ=1

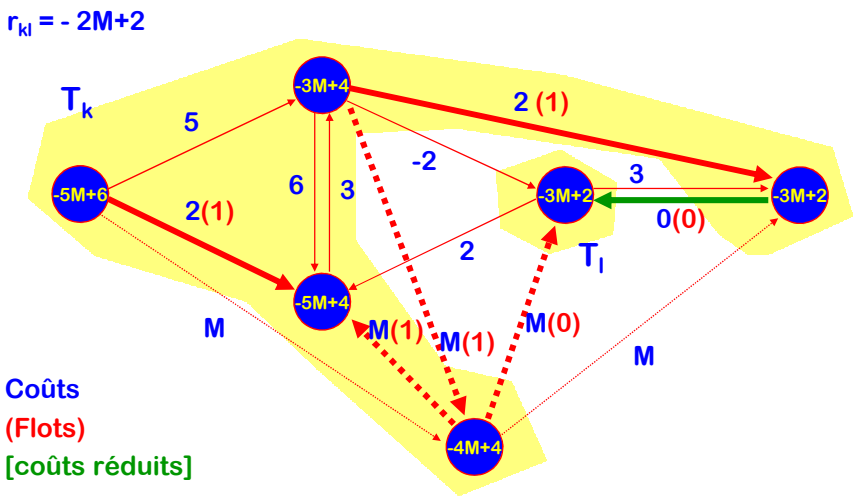
Coûts réduits



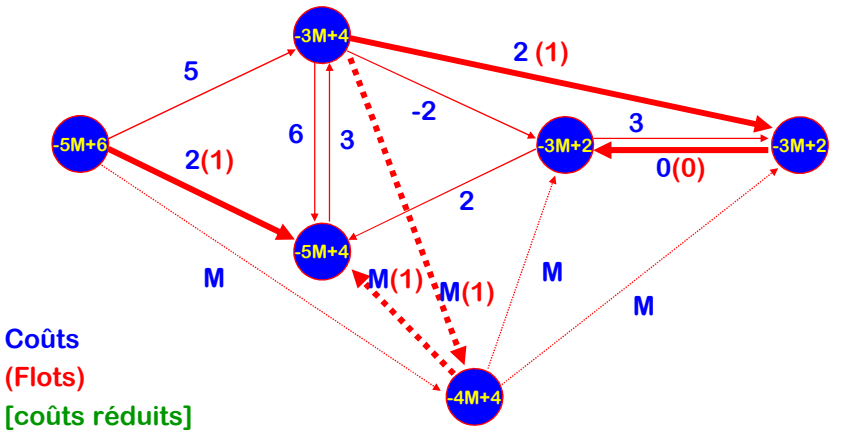
Coûts
(Flots)
[coûts réduits]

δ=0

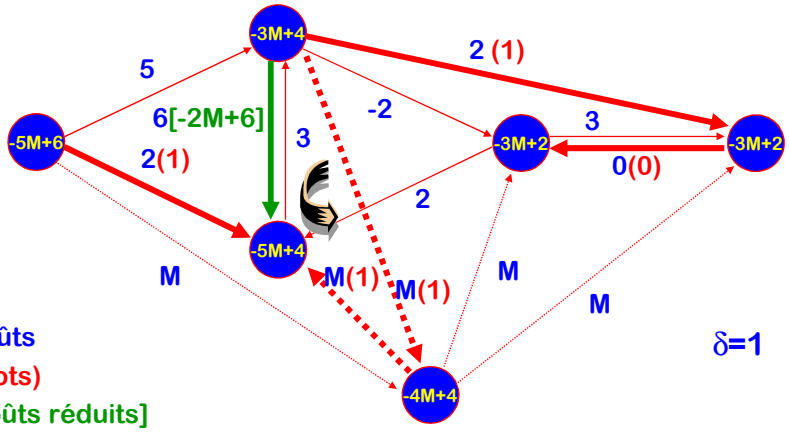
Mise à jour des prix



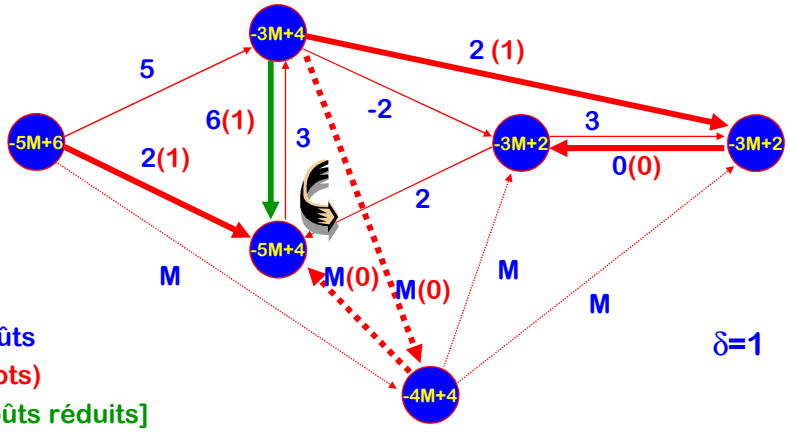
Nouvel arbre



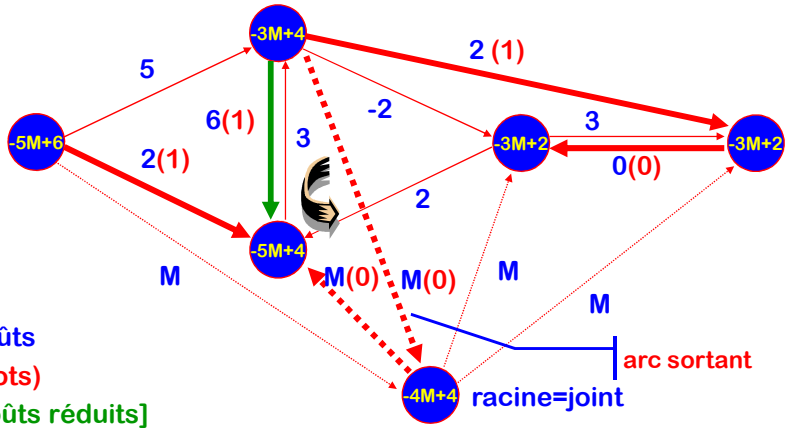
Coûts réduits



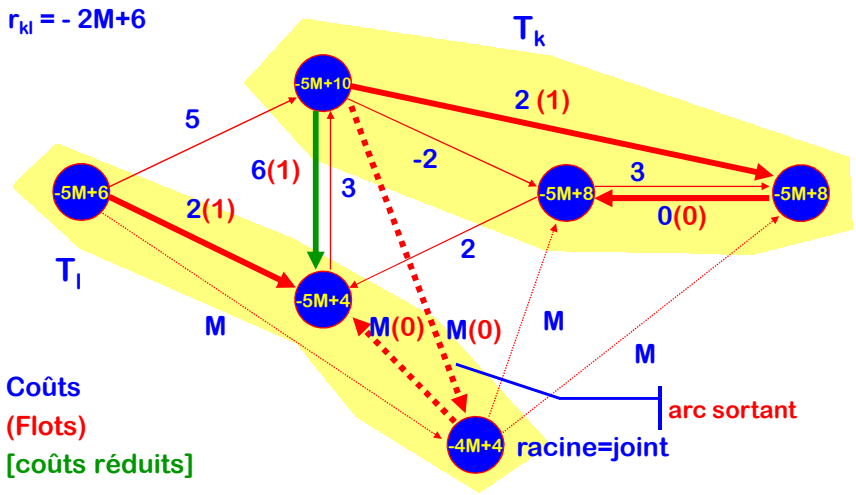
Mise à jour des flots



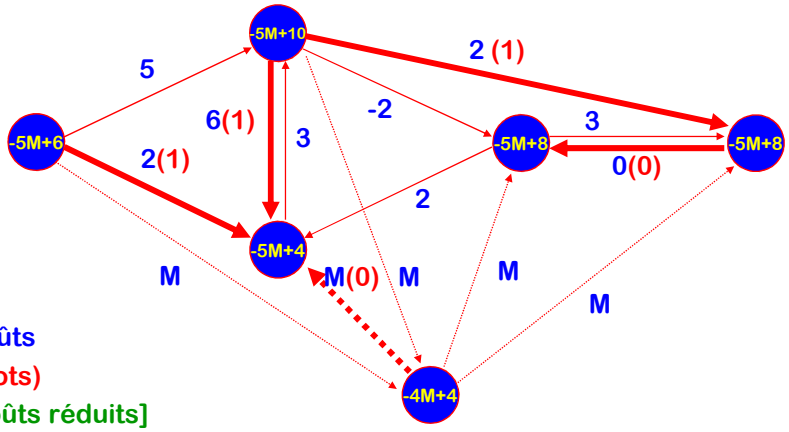
Choix de l'arc sortant



Mise à jour des prix

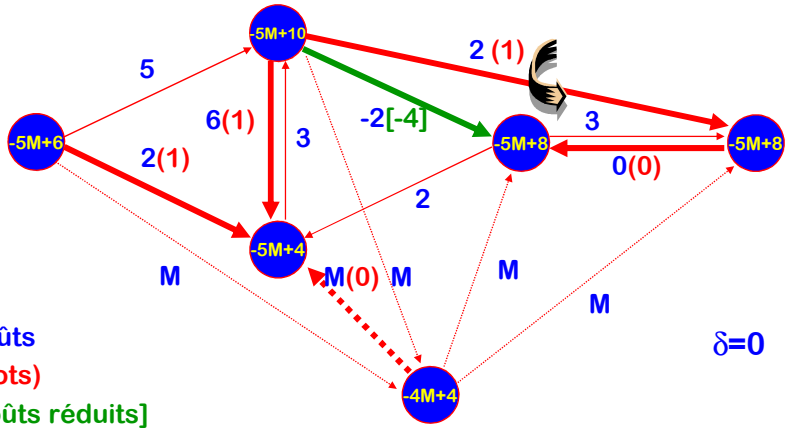


Nouvel arbre



Coûts
(Flots)
[coûts réduits]

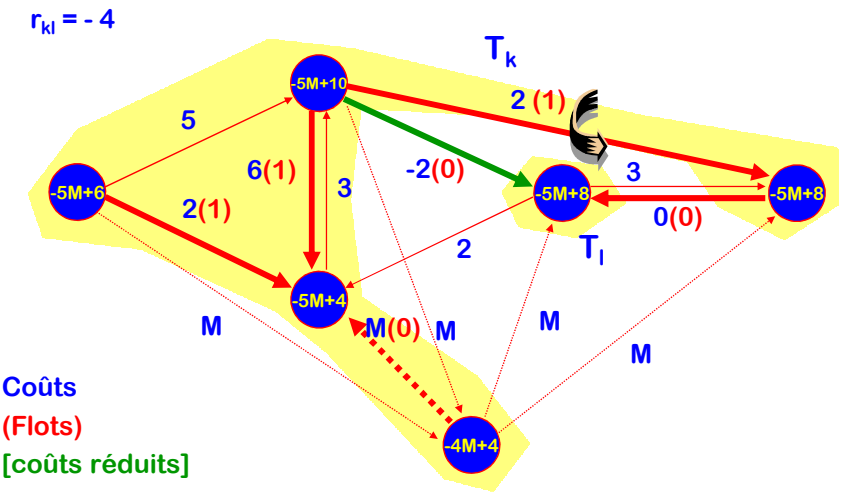
Coûts réduits



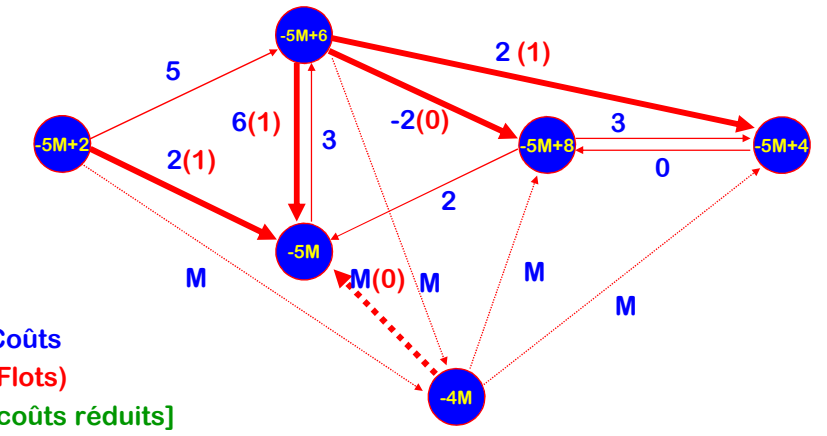
Coûts
(Flots)
[coûts réduits]

δ=0

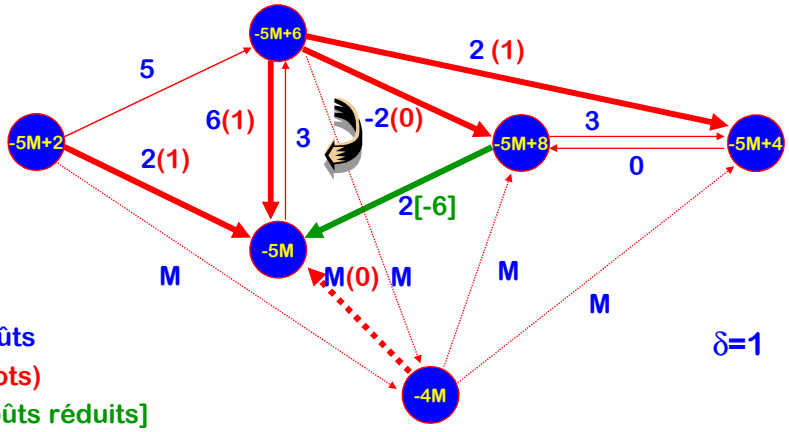
Mise à jour des prix



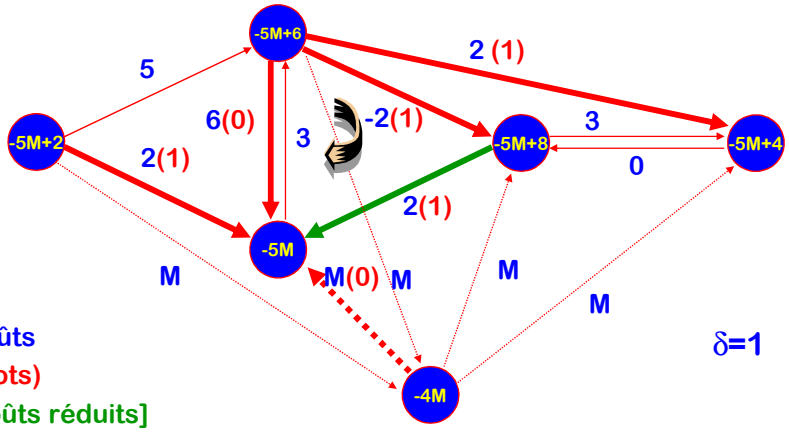
Nouvel arbre



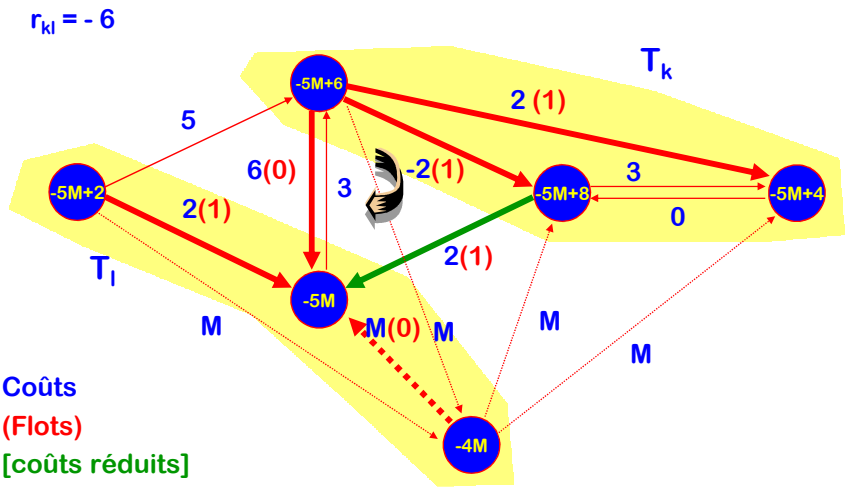
Coûts réduits



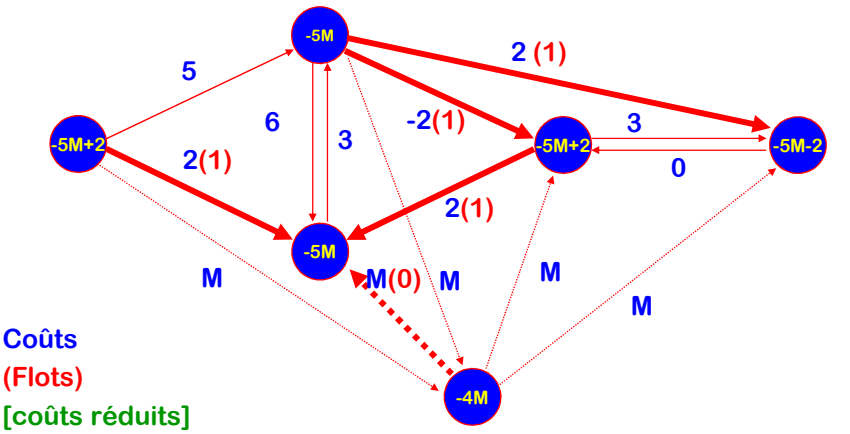
Mise à jour des flots



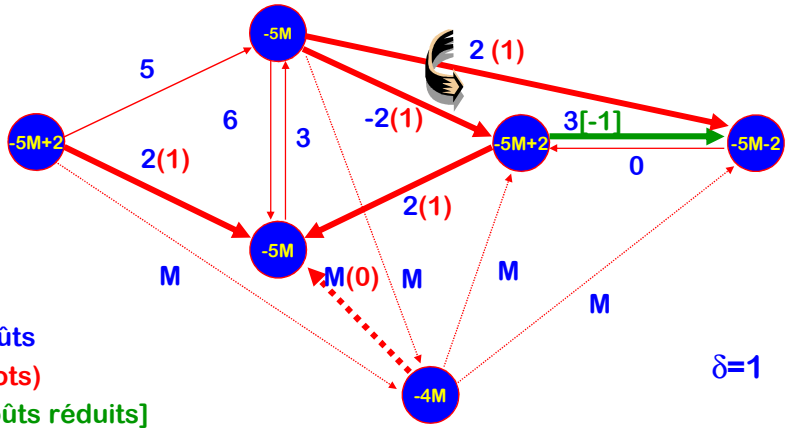
Mise à jour des prix



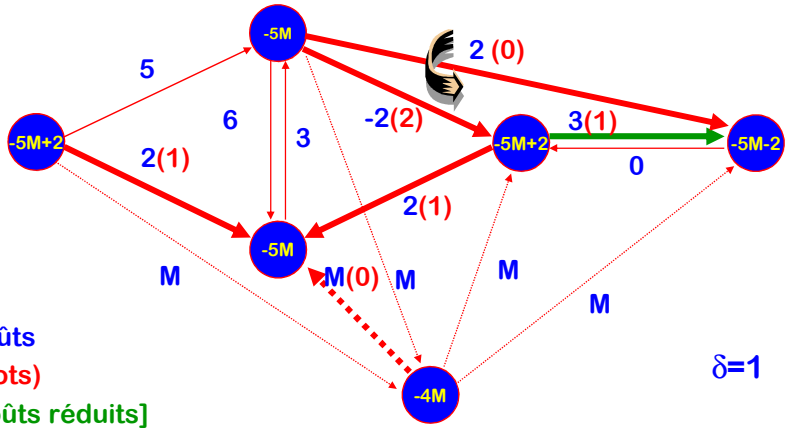
Nouvel arbre



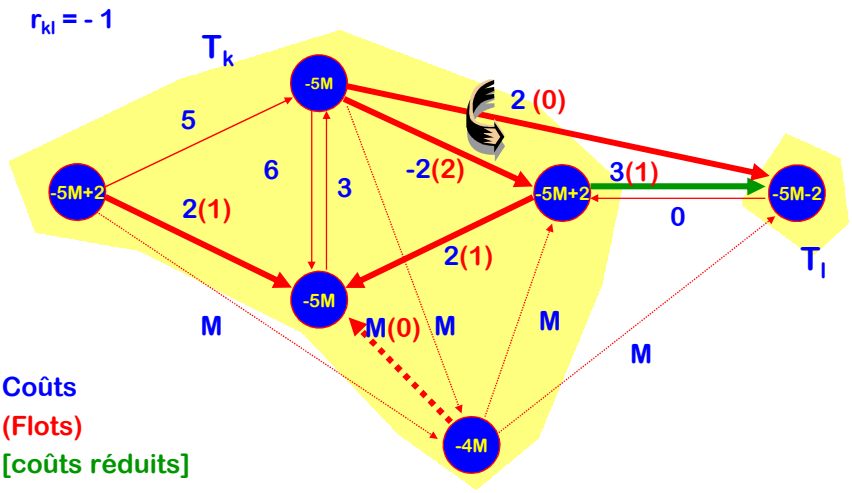
Coûts réduits



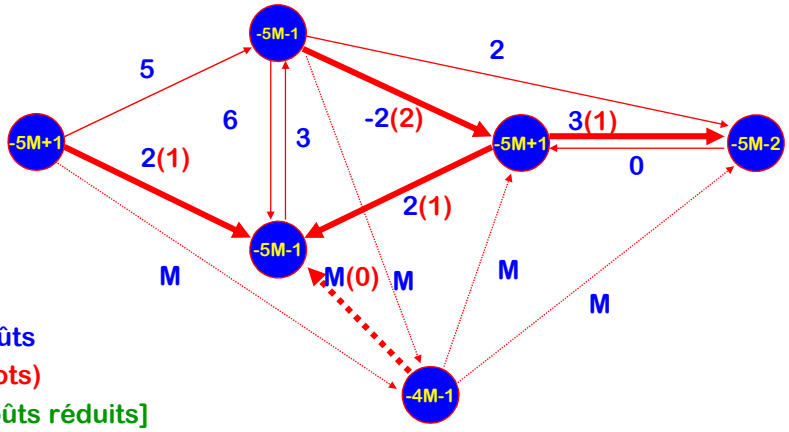
Mise à jour des flots



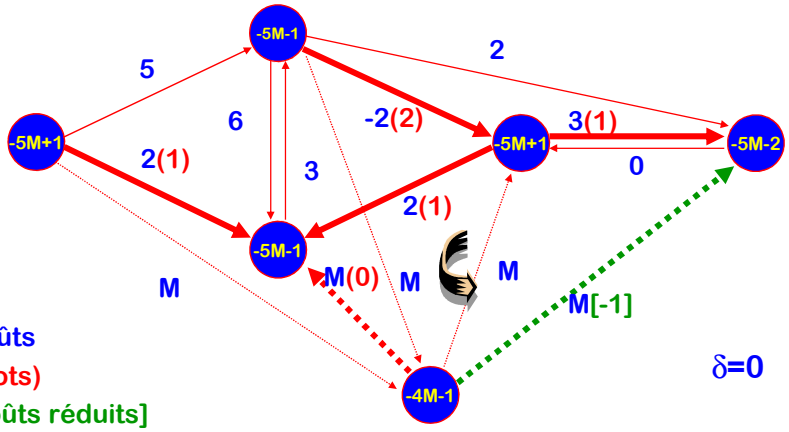
Mise à jour des prix



Nouvel arbre



Coûts réduits

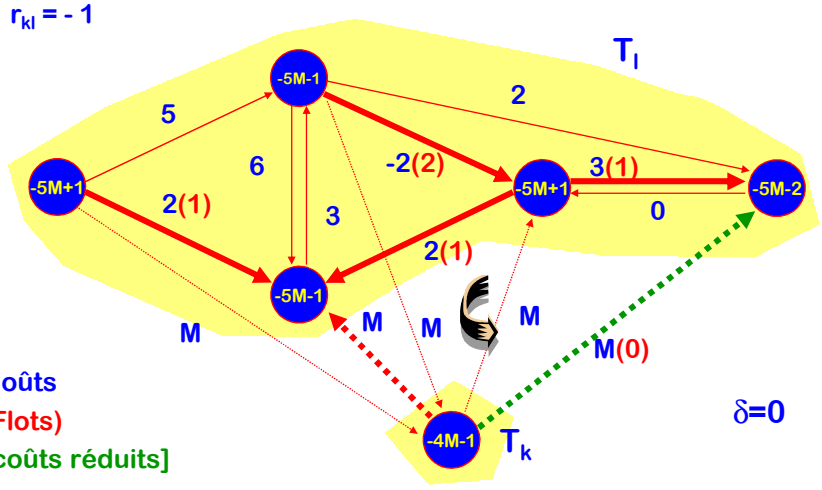


Coûts
(Flots)
[coûts réduits]

δ=0

Mise à jour des prix

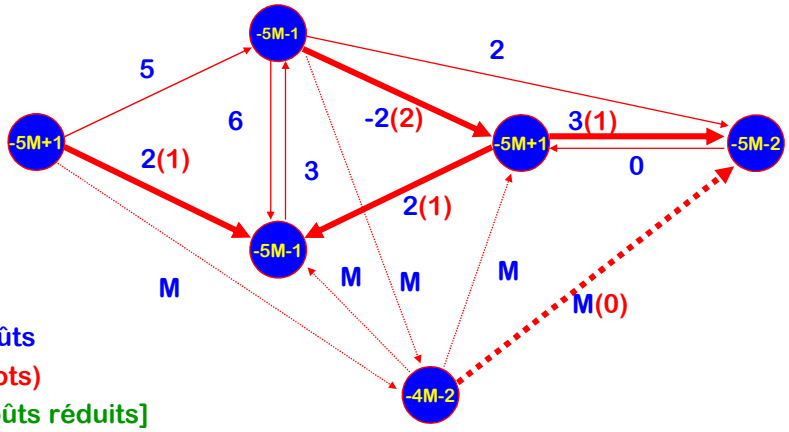
$r_{kl} = -1$



Coûts
(Flots)
[coûts réduits]

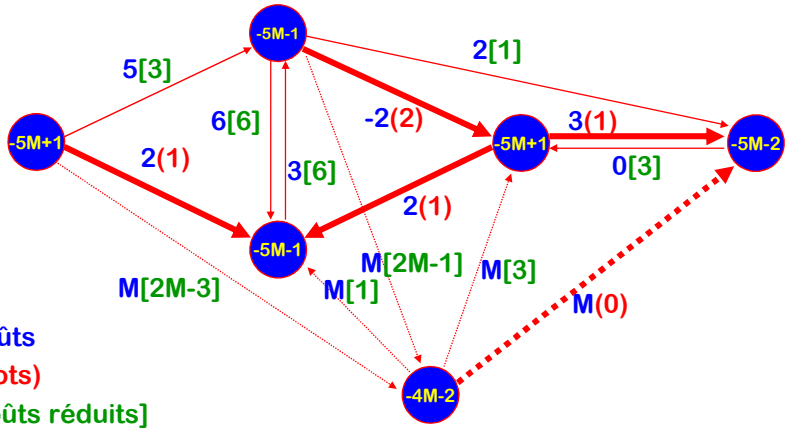
δ=0

Nouvel arbre



Coûts
(Flots)
[coûts réduits]

Coûts réduits : optimum



Coûts
(Flots)
[coûts réduits]

Propriété d'intégralité

- Si les s_i sont tous entiers, alors la solution optimale primale sera aussi entière.
- De plus, si le prix arbitraire initial de la racine est entier, et que tous les coefficients de coût sont entiers, alors la solution optimale duale sera aussi entière.