Chapitre 2 : Généralisation et Applications des notions de flot maximum/coupe minimum

M1-OPTION RO- PARTIE FLOTS

Université Paul Sabatier

Année 2013-2014

- R Ahuja, TL Magnanti, JB Orlin, and MR Reddy. Applications of Network Optimization, volume 7, chapter 1. Elsevier Science B.V., 1995.
- A. Alj and R. Faure. Guide de la recherche opérationnelle : Les applications. Masson, 1990.
- R. Faure, B. Lemaire, and C. Picouleau. Précis de Recherche Opérationnelle. Dunod, Paris, 2000.
- J.C. Fournier. Théorie des graphes et applications : Avec exercices et problèmes. Collection Informatique. Hermes Science Publications, 2011.
- D. Hochbaum. Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today. Management Science, 50(6) :709–723, 2004.
- E. Tardos J. Kleinberg. Algorithm Design. Addison Wesley, 2005.

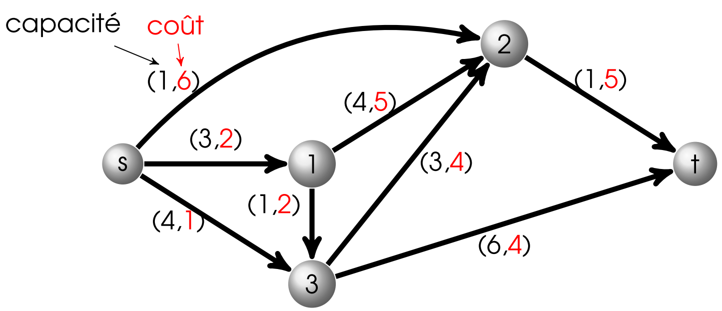
Plan

Pb du flot de coût minimum

$R = (X, U, c, \gamma)$ réseau de transport avec

- $c(u) \geq 0$ capacité de l'arc u et
- $\gamma(u) \geq 0$ coût de passage d'une unité de flux dans l'arc u
- avec $c(u_0) = \infty$ et $\gamma(u_0) = 0$.

Le coût d'un flot φ :$\gamma(\varphi) = \sum_{u \in U} \gamma(u)\varphi(u)$
Trouver un flot compatible de coût minimum. Voire un flot max de coût min.



Recherche d'un flot de coût minimum

Algorithme de Roy (1960) (Roy, 1969), Busacker et Gowen (1961) (Busacker and Gowen, 1961)

- Initialisation : $\varphi \leftarrow (0, \ldots, 0)$ (flot nul)
- Étape courante : Construire le graphe d'écart $G_R(\varphi) = (X, U_\varphi, c_\varphi, \gamma_\varphi)$ avec
  $\gamma_\varphi(u) = \begin{cases} \gamma(u) & \text{si } u \text{ même sens dans } U \text{ et } U_\varphi \\ -\gamma(u) & \text{sinon} \end{cases}$
- Si $\nexists$ chemin $[s,t]$ dans $G_R(\varphi)$
  alors FIN : φ est un flot maximum de coût minimum
  Sinon
  - déterminer un chemin ν_{st} élémentaire st-minimal de $(X, U_\varphi, \gamma_\varphi)$
  - $\varepsilon \leftarrow \min_{u \in \nu_{st}} c_\varphi(u)$
  - $\mu_{st} \leftarrow$ vecteur cycle $\nu_{st} \cup \{u_0\}$
  - $\varphi \leftarrow \varphi + \varepsilon\mu_{st}$ nouveau flot compatible $\gamma(\varphi) \leftarrow \gamma(\varphi) + \varepsilon.\gamma_\varphi(\nu_{st})$ coût du nouveau flot
- réitérer pour obtenir un flot de valeur supérieure

Autre méthode : suppression des circuits négatifs

Théorème (Théorème d'optimalité de Berges)

Soit φ un flot compatible, φ est de coût minimum ssi $G_R(\varphi)$ n'admet aucun circuit de coût négatif.

Algorithme de Klein 1967 (Klein, 1967) :

- construire un flot φ compatible sur $R \cup \{u_0\}$
- tant que le graphe d'écart admet un circuit négatif μ faire
  - soit k la capacité résiduelle minimale de ce circuit,
  - $\varphi \leftarrow \varphi + k\mu$.

Variante algo polynomial : choisir de supprimer le circuit de coût moyen minimum (= coût du circuit/ nombre d'arcs du circuit).

- complexité $O(nm)$ pour le trouver
- garantit un nombre max de $O(nm^2 logn)$ iterations.
- D'où un algo polynomial en : $O(n^2m^3logn)$.

Circulation avec demande

Un graphe orienté $G = (X, U)$, les arcs $u \in U$ munis de capacités $c(u)$, les sommets $x \in X$ associés à des demandes ($d(x) < 0$ apports ou offre, $d(x) > 0$ demande ou besoin, $d(x) = 0$ transbordement).

Définition

Une circulation est une fonction f qui satisfait :

- $\forall u \in U, \quad 0 \le f(u) \le c(u) \quad$ *(capacité)*
- $\forall x \in X, \quad \sum_{u\in\omega^-(\{x\})} f(u) - \sum_{u\in\omega^+(\{x\})} f(u) = d(x) \quad$ *(conservation)*

Problème : Existe-t'il une circulation étant donné (X, U, c, d) ?

Condition nécéssaire d'existence

somme des demandes = somme des offres

$$\sum_{\{x|d(x)>0\}} d(x) = \sum_{\{x|d(x)<0\}} d(x) = D$$

Circulation avec demande : exemple

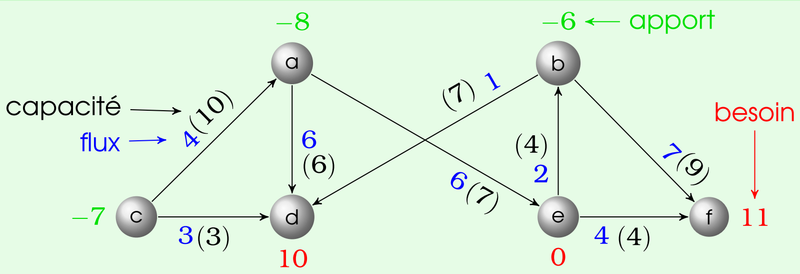
Une circulation est une fonction f qui satisfait :

- $\forall u \in U, \quad 0 \le f(u) \le c(u) \quad$ (capacité)
- $\forall x \in X, \quad \sum_{u\in\omega^-(\{x\})} f(u) - \sum_{u\in\omega^+(\{x\})} f(u) = d(x) \quad$ (conservation)

Condition nécéssaire d'existence :

$$\sum_{\{x|d(x)>0\}} d(x) = \sum_{\{x|d(x)<0\}} d(x) = D$$

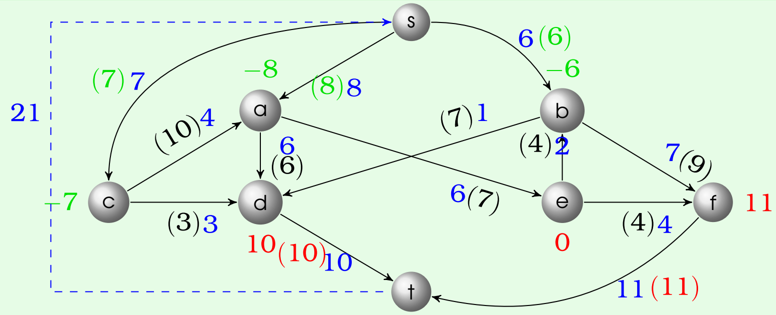
Exemple



Circulation avec demande : formulation en flots

- Ajouter 2 nouveaux sommets : une source s et un puits t.
- Pour chaque sommet x avec $d(x) < 0$, ajouter un arc (s, x) de capacité $-d(v)$.
- Pour chaque sommet x avec $d(x) > 0$, ajouter un arc (x, t) de capacité $d(v)$.
- G admet une circulation ssi G' admet un flot maximum de valeur D.

Exemple



Circulation avec demandes et bornes inférieures

Un graphe orienté $G = (X, U)$, les arcs $u \in U$ munis de capacités $c(u)$ et de bornes inférieures $b(u)$, les sommets $x \in X$ associés à des demandes.

Définition

Une circulation dans (X, U, b, c, d) est une fonction f qui satisfait :
- $\forall u \in U, \quad b(u) \leq f(u) \leq c(u)$ *(capacité)*
- $\forall x \in X, \quad \sum_{u \in \omega^-(\{x\})} f(u) - \sum_{u \in \omega^+(\{x\})} f(u) = d(x)$ *(conservation)*

Problème : Existe-t'il une circulation étant donné (X, U, b, c, d) ?

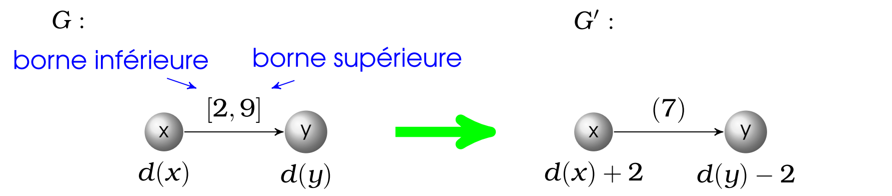
Transformation du problème

Idée : Considérer les bornes inférieures comme des demandes :
- envoyer $b(u)$ unités sur chaque arc u
- mettre à jour les demandes aux deux extrémités de u

Transformation du problème de circulation avec bornes inférieures

Transformation du problème

Idée : Considérer les bornes inférieures comme des demandes :
- envoyer $b(u)$ unités sur chaque arc u
- mettre à jour les demandes aux deux extrémités de u

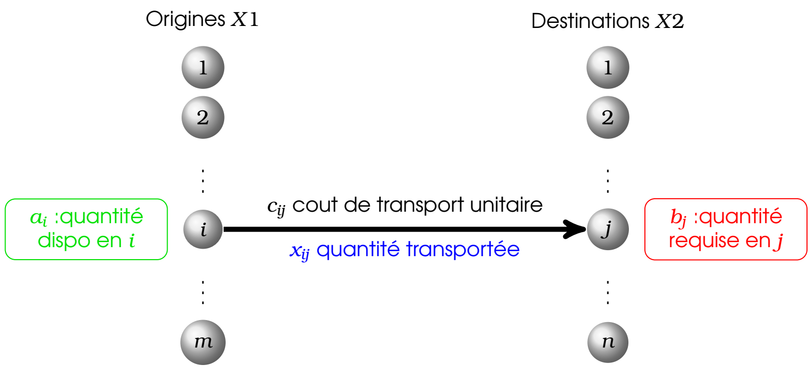


Propriété

Il existe une circulation dans G ssi il existe une circulation dans G'.

Programme de transport

Graphe biparti complet :



$$\boxed{\sum_{i=1}^m a_i = \sum_{j=1}^n b_j}$$

Déterminer les $x_{ij} \geq 0$ t.q. : $\forall i, \sum_{j=1}^n x_{ij} = a_i$ et $\forall j, \sum_{i=1}^m x_{ij} = b_j$ et $\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ soit minimale.

Programme de transport : exemple

Une fabrique de conserves expédie des caisses vers des dépôts. Il y a trois usines I, II et III et cinq dépôts A, B, C, D et E. Les coûts d'expédition sont donnés dans le tableau ci-dessous, ainsi que les quantités disponibles et demandées.

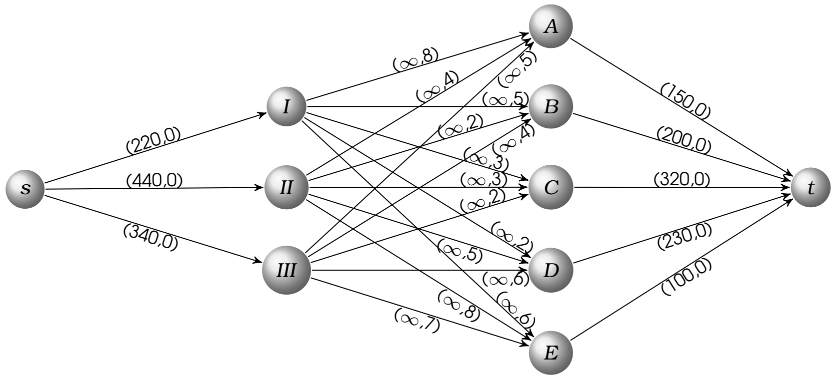
coût (c_{ij})	Destinations					
Origine	A	B	C	D	E	disponibles (a_i)
I	8	5	3	2	6	220
II	4	2	3	5	8	440
III	5	4	2	6	7	340
demandées (b_j)	150	200	320	230	100	

Déterminer le nombre de caisses que chacune des usines doit expédier vers chacun des dépôts de façon à minimiser le coût de transport total.

Résolution d'un programme de transport

On peut le résoudre de deux façons :

- Programmation linéaire.
- Algorithme de Roy (ou Busacker et Gowen)



NB. si l'offre et la demande sont de 1 alors c'est un pb d'affectation.

Tensions

La notion duale de flot est celle de tension. Les cocycles étant remplacés par des cycles.

Définition

Une tension (ou différence de potentiel) est un vecteur θ de $\mathbb{Z}^m$ t.q. pour tout cycle élémentaire μ, on ait :

$$\sum_{u\in\mu^+} \theta(u) = \sum_{u\in\mu^-} \theta(u)$$

On obtient de façon duale tous les résultats des flots :

- le vecteur nul est une tension,
- tout vecteur cocycle est une tension,
- toute combinaison linéaire de tension est une tension
- ...

Comment connaître tous les cycles élémentaires ?

- On prend un arbre partiel H de $G = (X, U)$ (avec $|X| = n$ et $|U| = m$)
- Tout arc ajouté crée un cycle unique :
- $\Rightarrow n - 1$ arcs dans l'arbre donc $m - n + 1$ cycles élémentaires indépendants
- Les autres sont des combinaisons linéaires de ces cycles.
- Il suffit de vérifier sur ces cycles élémentaires de base.

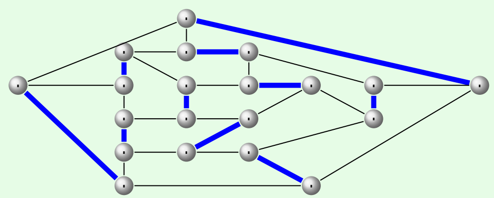
Couplage

Aparier des éléments étant données des contraintes de compatibilités, chaque élément ne peut avoir au plus qu'un seul correspondant.

Définition (Couplage)

- *Entrée : un graphe non orienté $G = (X, U)$.*
- *$V \subseteq U$ couplage si chaque sommet est présent au plus une fois dans les arêtes de V.*
- *Couplage maximum : couplage de cardinalité maximum.*

Exemple



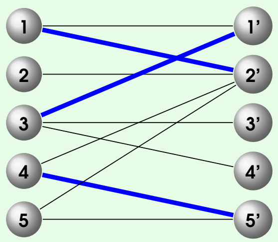
Couplage dans un graphe biparti

Aparier des éléments de deux ensembles distincts.

Définition (Couplage dans un graphe biparti)

- *Entrée : un graphe non orienté biparti* $G = (X \cup Y, U \subseteq X \times Y)$.
- $V \subseteq U$ *couplage si chaque sommet est présent au plus une fois dans les arêtes de* V*. Couplage maximum = cardinalité maximum.*
- *Couplage parfait :* $|V| = |X| = |Y|$*. Chaque sommet apparaît exactement une fois dans* V*.*

Exemple

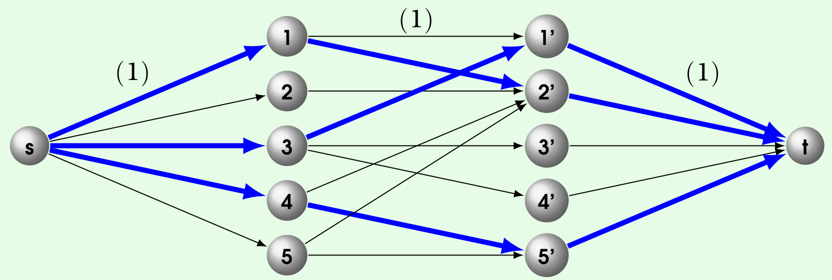


Couplage : Traduction en flot maximum

Créer un graphe orienté $R = (X \cup Y \cup \{s\} \cup \{t\}, U')$.

U' :
- Chaque arête de X vers Y donne un arc (capacité = ∞ ou 1).
- Les arcs de s vers tous les sommets de X de capacité 1.
- Les arcs de tous les sommets de Y vers t de capacité 1.

Exemple



Flot de valeur 3 (non maximum).

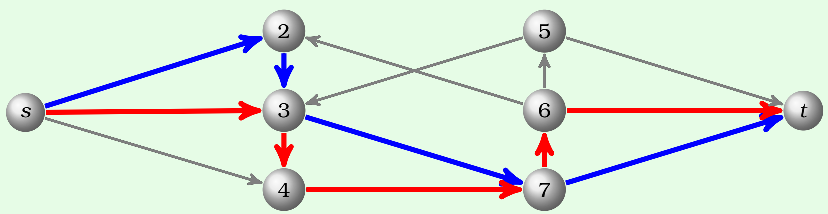
Problème des chemins disjoints

Soit $G = (X, U)$ et deux sommets s et t, trouver un nombre maximal de chemins arcs-disjoints de s à t.

Définition (Chemins arcs-disjoints)

Deux chemins sont arcs-disjoints s'ils n'ont pas d'arc en commun.

Exemple Réseau de communication



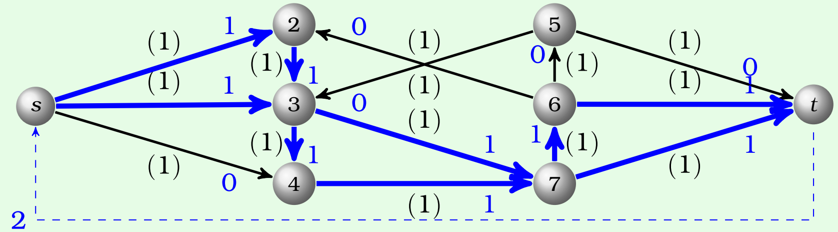
Chemins arcs-disjoints : Formulation en flots

Soit $G = (X, U)$ et deux sommets s et t, trouver un nombre maximal de chemins arcs-disjoints de s à t.
Attribuer une capacité de 1 à chaque arc.

Propriété

Le nombre de chemins arcs disjoints est la valeur du flot maximum.

Exemple Réseau de communication



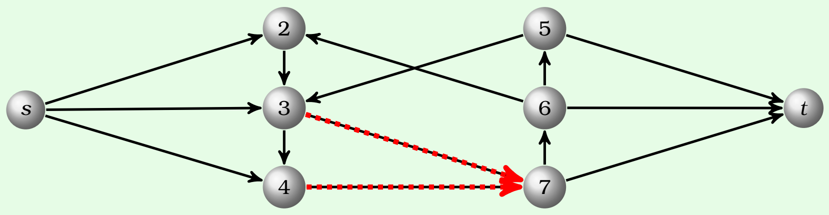
Connectivité

Soit $G = (X, U)$ et deux sommets s et t, trouver un nombre minimal d'arc dont la suppression déconnecte s de t.

Définition

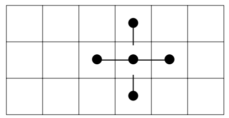
Un ensemble d'arc $V \subseteq U$ déconnecte s de t si tout chemin de s à t utilise au moins un arc de V

Exemple Réseau de communication



Propriété (Menger 1927)

Segmentation d'image



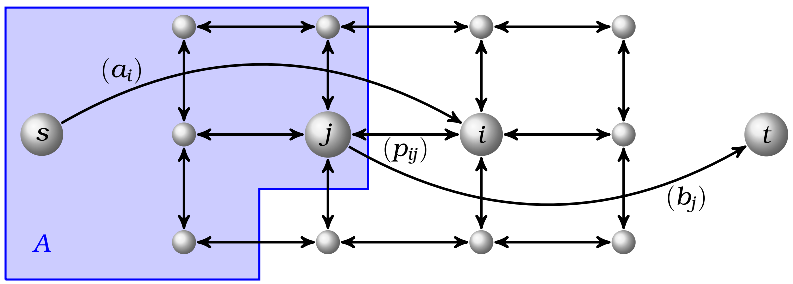
- étiqueter chaque pixel en arrière-plan ou premier-plan
- $a_i \geq 0$ proba que le pixel i soit du premier-plan
- $b_i \geq 0$ proba que le pixel i soit en arrière-plan
- $p_{ij} \geq 0$ pénalité de séparation des voisins (quand deux voisins ne sont pas dans le même plan)

But :

- précision : si $a_i > b_i$ préférer i en premier-plan
- lissage : si beaucoup de voisins en premier-plan préferer le premier-plan
- trouver une partition (A,B) qui maximise : $\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in U, |A \cap \{i,j\}|=1} p_{ij}$.

Segmentation d'image en termes de flots

- Maximiser $\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in U, |A \cap \{i,j\}|=1} p_{ij}$ avec (A,B) = partition des sommets
- revient à Minimiser $\sum_{i \notin A} a_i + \sum_{j \notin B} b_j + \sum_{(i,j) \in U, |A \cap \{i,j\}|=1} p_{ij}$.



- $\omega^+(A)$ a pour capacité : $\sum_{i \notin A} a_i + \sum_{j \in A} b_j + \sum_{(i,j) \in U, |A \cap \{i,j\}|=1} p_{ij}$.
- Trouver une coupe de capa min => trouver un flot max.

Conclusion sur les flots

- Beaucoup de problèmes peuvent se formuler en terme de flots (pbs d'affectation, couplage, pb d'expédition idéale, fournisseurs clients, transports de marchandise etc.), il faut savoir les formaliser.
- Connaître l'algorithme de Ford-Fulkerson pour obtenir un flot maximal (il existe d'autres algorithmes).
- Lien entre coupe de capacité minimale et flot maximum.
- Pour minimiser les coûts de transports, utiliser le graphe d'écart et l'algorithme de Roy (Busacker et Gowen) à partir du flot nul.

Références I

Ahuja, R., Magnanti, T., Orlin, J., and Reddy, M. (1995).
Applications of Network Optimization, volume 7, chapter 1.
Elsevier Science B.V.

Alj, A. and Faure, R. (1990).
Guide de la recherche opérationnelle : Les applications.
Masson.

Busacker, R. and Gowen, P. (1961).
A procedure for determining minimal-cost network flow patterns.
Technical Report ORO-15, Operational Research Office, John Hopkins University.

Faure, R., Lemaire, B., and Picouleau, C. (2000).
Précis de Recherche Opérationnelle.
Dunod, Paris.

Références II

Fournier, J. (2011).
Théorie des graphes et applications : Avec exercices et problèmes.
Collection Informatique. Hermes Science Publications.

Hochbaum, D. S. (2004).
Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today.
Management Science, 50(6) :709–723.

Klein, M. (1967).
A primal method for minimal cost flows with applications to the assignment and transportation problems.
Management Science, 14 :205–220.

Kleinberg, J. and Tardos, E. (2005).
Algorithm Design.
Addison Wesley.

Références III

Roy, B. (1969).
Algèbre moderne et théorie des graphes.
Dunod, Paris.