FLOTS - Théorie

Soit R=(V,A) un réseau orienté contenant un sommet s sans prédécesseur et un sommet t sans successeur. On dit que s est une **source** et t est un **puits**. Une **capacité** $c_a=c_{(x,y)}$ est associée à chaque arc a=(x,y).

Un **flot** dans R est une fonction f qui attribue un entier f(a) à tout arc a de R de telle sorte que

$$\sum_{a\in\omega^+(\{v\})} f(a) = \sum_{a\in\omega^-(\{v\})} f(a) \qquad \forall\ v\neq s,t \quad \text{(lois de conservation)}$$

Le flot est dit **compatible** si $0\leq f(a)\leq c_a$ pour tout arc a.

Des lois de conservation, on déduit $\sum_{a\in\omega^+(\{s\})} f(a) = \sum_{a\in\omega^-(\{t\})} f(a)$. Cette valeur est appelée **le flot entre s et t** et sera notée $f_{s\to t}$.

Soit W un sous-ensemble de V tel que $s\in W$ et $t\notin W$. L'ensemble des arcs dans $\omega^+(W)$ c'est-à-dire ceux allant de W vers V-W est appelé une **coupe** et sera notée (W,V-W)

On notera

- C(W,V-W) la **capacité de la coupe** (W,V-W), c'est-à-dire $C(W,V-W)=\sum_{a\in\omega^+(W)} c_a$.
- f(W,V-W) le flot traversant la frontière de W vers V-W, c'est-à-dire $\sum_{a\in\omega^+(W)} f(a)$

Problème
Déterminer un flot compatible dans R qui maximise $f_{s\to t}$. Un tel flot sera dit *maximum*.

Propriété :
On a $f_{s\to t} \leq C(W,V-W)$ pour tout flot compatible f et pour tout coupe (W,V-W)

Preuve
$f_{s\to t} + f(V-W,W) = f(W,V-W) \leq C(W,V-W)$. Le résultat découle alors du fait que $f(V-W,W)\geq 0$.

Théorème (Ford-Fulkerson)
Il existe un flot compatible f et une coupe (W,V-W) tel que $f_{s\to t} = C(W,V-W)$.

Corollaire
Soit f un flot compatible et (W,V-W) une coupe telle que $f_{s\to t} = C(W,V-W)$.
Alors le flot f est maximum et la coupe (W,V-W) est de capacité minimum.

Algorithme de détermination d'un flot maximum

(1) Déterminer un flot compatible f (par exemple f(a)=0 pour tout a)

(2) Construire un réseau R*(f) comme suit :
- R*(f) a exactement les même sommets que R;
- pour tout arc a=(x,y) dans R faire :
  - créer dans R*(f) un arc (x,y) de capacité $c^*_{(x,y)}=c_a-f(a)$ si $c_a>f(a)$
  - créer dans R*(f) un arc (y,x) de capacité $c^*_{(y,x)}=f(a)$ si $f(a)>0$.

(3) S'il n'existe pas de chemin de s à t dans R*(f) alors STOP : le flot f est maximum
Sinon, soit P un tel chemin et soit $\Delta = \min_{(x,y)\in P} c^*_{(x,y)}$. Pour tout arc (x,y) dans P faire :
- Augmenter le flot de Δ unités sur (x,y) si l'arc (x,y) existe dans R
- Diminuer le flot de Δ unités sur (y,x) si l'arc (y,x) existe dans R

Retourner à (2).

A l'étape (3), lorsqu'il n'existe plus de chemin de s à t dans R*(f), on peut déterminer l'ensemble W des sommets x pour lesquels ils existe un chemin de s à x. On a donc $s\in W$ et $t\notin W$. La coupe (W,V-W) est de capacité minimale.

La complexité de cet algorithme dépend des capacités sur les arcs, et pas seulement des nombres de sommets et d'arcs.

Définitions

Un flot f est <u>bloquant</u> si tous les chemins de s à t dans R contiennent un arc a tel que c_a=f(a)

Remarques

Tout flot compatible maximum est bloquant. Mais un flot bloquant n'est pas nécessairement maximum.

Algorithme de construction d'un flot bloquant

Poser f(a) :=0 pour tout arc a et aller à *INITIALISATION*

INITIALISATION
v :=s; P :=∅; aller à *AVANCE*

AVANCE
Si v n'a pas de successeur alors aller à *RECUL*
Sinon, soit w un successeur de v.
Poser P :=P∪{(v,w)}; v :=w;
Si w≠t aller à *AVANCE* sinon aller à *AUGMENTATION*

AUGMENTATION
Calculer $\Delta = \min_{a \in P} \{c_a - f(a)\}$ et poser f(a) :=f(a)+Δ pour tout a∈P;
Ôter les arcs tels que c_a=f(a) et retourner à *INITIALISATION*

RECUL
Si v=s STOP
Sinon
soit (x,v) le dernier arc de P. Ôter (x,v) de P et ôter du réseau tous les arcs entrant en v;
poser v :=x et aller à AVANCE

Complexité

Supposons que R=(V,A) soit mémorisé à l'aide de listes de successeurs pour chaque sommet.

- Chaque appel à AVANCE et à INITIALISATION se fait en temps constant (indépendant de $|V|$ et $|A|$).
- Chaque appel à AUGMENTATION demande O($|V|$) opérations pour modifier le flot et O($|V|$) opérations pour chaque suppression d'arc.
- Chaque appel à RECUL demande O($|V|$) opérations pour chaque suppression d'arc (le reste prend un temps constant)
- On fait appel O($|A|$) fois à AUGMENTATION, à RECUL et à INITIALISATION car entre chaque appel on supprime au moins un arc.
- Entre deux augmentations, on fait O($|V|$) appels à AVANCE (car si on recule de v, on ne peut plus revenir à v).

Sans compter les suppressions d'arcs, la complexité de cet algorithme est donc O($|A|$) pour les INITIALISATION, O($|V||A|$) pour les AVANCE, O($|V||A|$) pour les AUGMENTATION, O($|A|$) pour les RECUL, ⇒ total en O($|V||A|$).

Chaque suppression d'arc se fait en O($|V|$) et on en supprime O($|A|$) ⇒ total en O($|V||A|$) pour les suppressions d'arcs.

La complexité de cet algorithme est **O($|\mathbf{V}||\mathbf{A}|$)**

Définitions

Un réseau est dit <u>en couches</u> s'il existe une partition de ses sommets en sous-ensembles $V_1,\ldots,V_r$ tel que chaque arc de ce réseau relie 2 sommets de 2 sous-ensembles consécutifs V_i et V_{i+1}.

L'algorithme ci-dessous construit à partir de R un réseau en couches $\overline{R}$ contenant tous les plus courts chemins de s vers les autres sommets de R, les distances sur les arcs étant unitaires (on s'intéresse donc au nombre minimum d'arcs pour se rendre de s vers x).

Algorithme de création d'un réseau en couches

(1) V_1 :={s}; M :={s}; i :=1;
(2) Si $\omega^+(M)$ est vide alors STOP
Sinon
rajouter à $\overline{R}$ tous les arcs de $\omega^+(M)$, poser V_{i+1} :={x tel que ∃ (y,x)∈ $\omega^+(M)$}, M :=M∪V_{i+1}, i :=i+1 et répéter (2)

Propriété

Soit f un flot compatible dans R. Soit $\bar{f}$ un flot bloquant dans $\overline{R^*(f)}$.

Soit f' le flot obtenu à partir de f comme suit : pour tout arc (x,y) dans $\overline{R^*(f)}$ tel que $\bar{f}$ (x,y)>0 faire

- f'(x,y)=f(x,y)+ $\bar{f}$ (x,y) si (x,y) est un arc de R
- f'(y,x)=f(y,x) - $\bar{f}$ (x,y) si (y,x) est un arc de R

Alors $\overline{R^*(f)}$ a strictement moins de couches que $\overline{R^*(f')}$

Algorithme de Dinic pour la détermination d'un flot maximum.

(1) Déterminer un flot compatible f (par exemple f=0);

(2) Construire $\overline{R^*(f)}$;

(3) S'il n'existe pas de chemin de s à t dans $\overline{R^*(f)}$ alors STOP

Sinon déterminer un flot bloquant $\bar{f}$ dans $\overline{R^*(f)}$, et pour tout (x,y) dans $\overline{R^*(f)}$ faire

f'(x,y)=f(x,y) + $\bar{f}$ (x,y) si (x,y) est un arc de R

f'(y,x)=f(y,x) - $\bar{f}$ (x,y) si (y,x) est un arc de R

Retourner à (2)

Complexité

Etant donné que le nombre de couches augmente à chaque passage dans (2), et que ce nombre est borné supérieurement par $|V|$, on passe O($|V|$) fois dans (2). Si on mémorise R=(V,A) avec les listes de successeurs de chaque sommet, la complexité de cet algorithme est O($|V|^2|A|$)

Problème de circulation avec bornes inférieures

Considérons un réseau quelconque (on ne suppose donc plus l'existence d'une source ou d'un puits). Supposons qu'à chaque arc a de R, on associe non seulement une capacité c_a, mais également une borne inférieure l_a.

Un **flot** dans R est une fonction f qui attribue un entier f(a) à tout arc a de R de telle sorte que

$$\sum_{a\in\omega^+(\{v\})} f(a) = \sum_{a\in\omega^-(\{v\})} f(a) \qquad \forall \text{ sommet } v \quad \text{(lois de conservation)}$$

Le flot est dit **compatible** si $l_a \le f(a) \le c_a$ pour tout arc a. On parle dans ce cas de problème de **circulation** car la loi de conservation doit être vérifiée pour tout sommet de V (il n'y a plus d'exception pour une source et un puits).

On peut ramener le modèle précédent à celui-ci en posant l_a=0 pour tout arc a, et en rajoutant un arc du puits t vers la source s, de capacité $c_{(t,s)}=\infty$.

Remarquons que le flot nul (f(a)=0 pour tout a) n'est plus nécessairement compatible.

Problème

Déterminer un flot compatible dans R

Définition

Soit f un flot pas nécessairement compatible, et soit a un arc dans R.

- La <u>déficience de a</u>, notée $D_f(a)$ est définie comme étant égale à max{l_a-f(a),f(a)-c_a,0}.
- La <u>déficience totale</u>, notée D_f est égale à $\sum_{a\in A} D_f(a)$.

Un flot est donc compatible si et seulement si sa déficience totale est nulle. Étant donnée un sous-ensemble W de sommets, on définit maintenant non seulement C(W,V-W)= $\sum_{a\in\omega^+(W)} c_a$ mais également L(W,V-W)= $\sum_{a\in\omega^+(W)} l_a$.

Théorème

Il existe un flot compatible dans R $\Leftrightarrow$ $C(W,V\text{-}W) \geq L(V\text{-}W,W)$ pour tout W non vide strictement inclus dans V.

Etant donné un flot f dans R, on va construire un réseau R*(f) de manière similaire à ce qui a été décrit précédemment :

- R*(f) a exactement les même sommets que R;
- pour tout arc a=(x,y) dans R faire :
  . créer dans R*(f) un arc (x,y) de capacité $c^*_{(x,y)}=c_a-f(a)$ si $c_a>f(a)$
  . créer dans R*(f) un arc (y,x) de capacité $c^*_{(y,x)}=f(a)-l_a$ si $f(a)>l_a$.

Algorithme de détermination d'un flot compatible

(1) Choisir un flot f quelconque (par exemple f(a)=0 pour tout a).

(2) Si f est compatible STOP.

(3) S'il existe un arc a=(x,y) tel que $f(a)<l_a$ alors aller à (4) sinon il existe un arc a=(x,y) tel que $c_a<f(a)$ et aller à (5)

(4) S'il n'existe pas de chemin dans R*(f) allant de y vers x alors STOP, il n'existe pas de flot compatible dans R.

Sinon, soit P un tel chemin et soit $\Delta = \min\left\{\min_{(v,w)\in P} c^*_{(v,w)}, l_a - f(a)\right\}$. Pour tout arc (v,w) dans P faire :

- Augmenter le flot de Δ unités sur (v,w) si l'arc (v,w) existe dans R
- Diminuer le flot de Δ unités sur (w,v) si l'arc (w,v) existe dans R
- Poser $f(a):=f(a)+\Delta$.

Retourner à (2)

(5) S'il n'existe pas de chemin dans R*(f) allant de x vers y alors STOP, il n'existe pas de flot compatible dans R.

Sinon, soit P un tel chemin et soit $\Delta = \min\left\{\min_{(v,w)\in P} c^*_{(v,w)}, f(a) - c_a\right\}$. Pour tout arc (v,w) dans P faire :

- Augmenter le flot de Δ unités sur (v,w) si l'arc (v,w) existe dans R
- Diminuer le flot de Δ unités sur (w,v) si l'arc (w,v) existe dans R
- Poser $f(a):=f(a)-\Delta$.

Retourner à (2)

Flot maximum entre une source et un puits, avec bornes inférieures sur les flots

Revenons au modèle précédent dans lequel le réseau a une source s et un puits t, mais supposons que chaque arc a possède non seulement une borne supérieure c_a mais également une borne inférieure l_a sur la quantité de flot pouvant y circuler.

Si on veut déterminer un flot maximum entre s et t, on peut rajouter un arc de t vers s de capacité infinie. Puis, on peut déterminer un flot compatible grâce à l'algorithme ci-dessus. On peut ensuite à nouveau ôter l'arc de t vers s et appliquer l'algorithme de la page (1), avec la nouvelle définition du réseau R*(f) (c'est-à-dire que si (x,y) est un arc de R avec $f(a)>l_a$, alors on crée un arc (y,x) de capacité $c^*_{(y,x)}=f(a)-l_a$ dans R*(f).)

Flot à coût minimum

On considère un réseau quelconque (sans source, ni puits) avec bornes supérieures c_a et bornes inférieures l_a sur les quantités de flot pouvant circuler sur les arcs. De plus, on suppose qu'un coût d_a est associé à chaque unité de flot circulant sur l'arc a. Le coût d'un flot est alors le coût total des unités de flot circulant sur les arcs.

Problème

Soit (x,y) un arc particulier du réseau et soit p un entier strictement positif.
Déterminer un flot compatible de coût minimum tel que f(x,y)=p

Etant donné un flot f, on va à nouveau considérer le réseau R*(f), mais en y rajoutant des coûts.

- Pour un arc a=(x,y) dans R avec $f(a)<c_a$ on crée un arc (x,y) dans R*(f) de capacité $c^*_{(x,y)}=c_a-f(a)$ et de coût $d^*_{(x,y)}=d_a$
- Pour un arc a=(x,y) dans R avec $l_a<f(a)$ on crée un arc (y,x) dans R*(f) de capacité $c^*_{(y,x)}=f(a)-l_a$ et de coût $d^*_{(y,x)}= -d_a$

Algorithme de construction d'un flot de coût minimum avec f(x,y)=p

(1) Déterminer un flot compatible tel que f(x,y)=p. Pour ce faire, on peut par exemple fixer $l_{(x,y)}=c_{(x,y)}=p$.
Si un tel flot n'existe pas, STOP

(2) Si R*(f) ne contient aucun circuit de coût négatif alors le flot est de coût minimum
Sinon soit C un tel circuit et soit $\Delta = \min_{(v,w)\in C} c^*_{(v,w)}$. Pour tout arc (v,w) dans C faire :

- Augmenter le flot de Δ unités sur (v,w) si l'arc (v,w) existe dans R
- Diminuer le flot de Δ unités sur (w,v) si l'arc (w,v) existe dans R

Retourner à (2)

Revenons au premier modèle, c'est-à-dire avec une source s, un puits t, et <u>aucune borne inférieure de flot</u> sur les arcs. Supposons que l'on veuille déterminer un flot de p unités de s vers t qui soit de coût minimum. On peut rajouter l'arc (t,s) de capacité infinie et de coût 0, et rechercher avec l'algorithme ci-dessus un flot de coût minimum avec f(t,s)=p. On peut cependant préférer l'un des deux algorithmes suivants.

Algorithme 1

(1) Considérer le flot nul f (c'est-à-dire tel que f(a)=0 pour tout arc a) et poser F :=0;

(2) Déterminer R*(f).

(3) S'il n'existe pas de chemin de s à t dans R*(f) alors STOP, on ne peut pas faire passer p unités de s vers t.

Sinon soient P un chemin de coût minimum et $\Delta = \min\left\{p - F, \min_{(x,y)\in P} c^*_{(x,y)}\right\}$. Pour tout arc (x,y) dans P faire :

- Augmenter le flot de Δ unités sur (x,y) si l'arc (x,y) existe dans R
- Diminuer le flot de Δ unités sur (y,x) si l'arc (y,x) existe dans R
- Poser F :=F+ Δ; si F=p STOP (le flot est de coût minimum), sinon retourner à (2)

Algorithme 2

Cet algorithme fonctionne exactement comme précédent avec une seule différence : les coûts dans R*(f).
Notons f^0, f^1, … les différents flots construits par l'algorithme ci-dessus, et notons $d^i_{(x,y)}$ le coût sur (x,y) dans $R^*(f^i)$.
Pour le flot initial f^0 (qui est nul) on considère les coûts définis ci-dessus, c'est-à-dire $d^0_{(x,y)}=d^*_{(x,y)}$.
Pour un flot f^i avec i>0, on définit :

$$d^i_{(x,y)} = \begin{cases} d^{i-1}_{(x,y)} + pcc^{i-1}(x) - pcc^{i-1}(y) & \text{si } (x,y) \text{ existait dans } R^*(f^{i-1}) \\ 0 & \text{sinon} \end{cases}$$

où $pcc^{i-1}(v)$ est la longueur du plus court chemin de s vers v dans $R^*(f^{i-1})$

L'avantage du deuxième algorithme est qu'il ne manipule que des distances strictement positives. On peut donc utiliser l'algorithme de Dijkstra pour déterminer les plus courts chemins de s vers t.