Algorithmique des graphes

Cours 8 – Optimisation de flots

František Kardoš

frantisek.kardos@u-bordeaux.fr

Réseaux de flot

Entrée :
Un graphe orienté G, avec deux sommets spéciaux

- l'origine (une source) s
- la destination (un puits) t

et, pour chaque arc, une capacité

$$c : E(G) \to \mathbb{R}^+$$

Un flot sur un réseau de flot

Un *flot* sur un réseaux de flot est une fonction $\varphi : E(G) \to \mathbb{R}^+$ satisfaisant deux conditions suivantes :

1. les capacités sur les arcs sont respectées

$$\forall e \in E(G) : \varphi(e) \leq c(e);$$

2. la loi de Kirchhoff – pour chaque sommet interne du réseau (sauf la source et la destination) le flot entrant est égal au flot sortant :

$$\forall v \in V(G), v \neq s, v \neq t : \sum_{uv \in E(G)} \varphi(uv) = \sum_{vw \in E(G)} \varphi(vw).$$

Optimisation de flot

La *valeur* $|\varphi|$ d'un flot φ est définie comme la somme des flots sortants de la source s (ce qui est toujours égal à la somme des flots entrants de la destination t).

Objectif : Maximiser le flot.

Flots et coupes

Une *s-t-coupe* dans un réseau de flot G est une partition (X, Y) des sommets de G telle que $s \in X$ et $t \in Y$. La capacité $c(X, Y)$ d'une s-t-coupe (X, Y) est définie comme la somme de capacités des arcs uv tels que $u \in X$ et $v \in Y$.

Lemme

Si φ est un flot et (X, Y) est une s-t-coupe, alors $|\varphi| \leq c(X, Y)$.

Flots et coupes

Une *s-t-coupe* dans un réseau de flot G est une partition (X, Y) des sommets de G telle que $s \in X$ et $t \in Y$. La capacité $c(X, Y)$ d'une s-t-coupe (X, Y) est définie comme la somme de capacités des arcs uv tels que $u \in X$ et $v \in Y$.

Lemme

Si φ est un flot et (X, Y) est une s-t-coupe, alors $|\varphi| \leq c(X, Y)$.

Par conséquence,

$$\max_{\varphi} |\varphi| \leq \min_{(X,Y)} c(X, Y).$$

Flots et coupes

Theorem (Ford et Fulkerson)

$$\max_{\varphi} |\varphi| = \min_{(X,Y)} c(X, Y).$$

Preuve : Par construction.
L'algorithme de Ford-Fulkerson trouve à la fois un flot φ^* et une s-t-coupe (X^*, Y^*) avec

$$|\varphi^*| = c(X^*, Y^*).$$

Algorithme de Ford-Fulkerson

```
1: Marquage()
2: while le sommet destination t est marqué do
3:     Améliorer()
4:     Marquage()
5: end while
```

L'algorithme de Ford-Fulkerson utilise le marquage de sommets pour déterminer, étant donné un flot, s'il existe une coupe empêchant d'améliorer le flot donné ; s'il n'y en a pas, l'algorithme propose une amélioration du flot actuel.

Cette procédure est répétée en boucle jusqu'à ce qu'aucune amélioration ne soit possible.

Algorithme de Ford-Fulkerson : Marquage

Le fait qu'un sommet v est marqué veut dire que pour le flot actuel il existe une modification telle que, exceptionnellement, v devient une seconde destination temporaire pour une moindre quantité de flot ($\varepsilon > 0$).

Initialement, seul le sommet source s est marqué.

Algorithme de Ford-Fulkerson : Marquage

Un arc $e \in E(G)$ est dit *saturé* par un flot φ si $\varphi(e) = c(e)$.

Règles de propagation de marquage :

- Si uv non saturé et que le sommet u est marqué, marquer v.
- Si sur uv il y a un flot non-nul et que le sommet v est marqué, marquer u.

Algorithme de Ford-Fulkerson : Marquage

Un arc $e \in E(G)$ est dit *saturé* par un flot φ si $\varphi(e) = c(e)$.

Règles de propagation de marquage :

- Si uv non saturé et que le sommet u est marqué, marquer v.
- Si sur uv il y a un flot non-nul et que le sommet v est marqué, marquer u.

Pour exécuter la propagation, on peut employer le principe du parcours en largeur ou celui du parcours en profondeur.

Algorithme de Ford-Fulkerson : Marquage

1: **for** chaque sommet v **do**
2: démarquer(v)
3: **end for**
4: marquer(s)
5: enfiler(s)
6: **while** file non vide **do**
7: $u \leftarrow$ tête de la file
8: **for** chaque voisin sortant v de u non-marqué **do**
9: **if** $\varphi(uv) < c(uv)$ **then**
10: marquer(v)
11: enfiler(v)
12: $\pi(v) \leftarrow uv$
13: **end if**
14: **end for**
15: **for** chaque voisin entrant v de u non-marqué **do**
16: **if** $\varphi(vu) > 0$ **then**
17: marquer(v)
18: enfiler(v)
19: $\pi(v) \leftarrow vu$
20: **end if**
21: **end for**
22: défiler(u)
23: **end while**

Algorithme de Ford-Fulkerson : Marquage

À la fin de marquage, si le sommet destination t est marqué, alors il existe une modification du flot actuel permettant d'ajouter encore du flot entre s et t. Puisqu'on a conservé les arcs permettant d'effectuer cette modification, il suffit de retracer un quasi-chemin entre s et t.

Un *quasi-chemin* est une suite d'arcs reliant le sommet source s et le sommet destination t, pas forcément toujours respectant la direction.

Algorithme de Ford-Fulkerson : Améliorer

La deuxième procédure Améliorer()

- identifie un quasi-chemin reliant s et t,
- calcule la valeur maximal de modification possible le long de ce quasi-chemin, et
- effectue cette modification.

Pour le quasi-chemin,

- le flot d'un arc dans le bon sens (non-saturé) est augmenté,
- le flot (non-nul) d'un arc dans le sens opposé est diminué,

d'une même valeur partout.

Algorithme de Ford-Fulkerson : Améliorer

1: $v \leftarrow t$
2: $val \leftarrow \infty$
3: **while** $v \neq s$ **do**
4: **if** $\pi(v) = uv$ **then**
5: $val \leftarrow \min\{val, c(uv) - \varphi(uv)\}$
6: **end if**
7: **if** $\pi(v) = vu$ **then**
8: $val \leftarrow \min\{val, \varphi(uv)\}$
9: **end if**
10: $v \leftarrow u$
11: **end while**
12: $v \leftarrow t$
13: **while** $v \neq s$ **do**
14: **if** $\pi(v) = uv$ **then**
15: $\varphi(uv) \leftarrow \varphi(uv) + val$
16: **end if**
17: **if** $\pi(v) = vu$ **then**
18: $\varphi(vu) \leftarrow \varphi(vu) - val$
19: **end if**
20: $v \leftarrow u$
21: **end while**