Optimisation Combinatoire

Chien-Chen Huang

26 septembre 2024

Table des matières

1 Max-Flow (min-cut)

1.1 Max-Flow

Définition 1.1 Soit $G = (V, E)$ un graphe orienté, $c : E \to \mathbb{R}^+$ une fonction de capacité et $s, t \in V$ deux sommets terminaux, $f : E \to \mathbb{R}^+$ est un flot si :

1. $0 \leq f(e) \leq c(e), \forall e \in E$
2. $\sum_{e \in \delta^-(v)} f(e) = \sum_{e \in \delta^+(v)} f(e), \forall v \in V \setminus \{s, t\}$ (Conservation du Flot).

On va s'intéresser au problème suivant :

MAX-FLOW	
Entrée:	$G = (V, E)$ un graphe orienté, $c : E \to \mathbb{R}^+$ une fonction de capacité et $s, t \in V$ deux sommets terminaux
Sortie:	Une fonction de flot f maximale, c'est à dire avec un volume maximal : $\sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e)$

Théorème 1.1 On rappelle qu'obtenir un flot maximal est équivalent à obtenir une coupe de poids minimal.

On définit pour cela le réseau résiduel d'une fonction de flot f :

Définition 1.2 Étant donné un flot f, pour chaque arrête $e = (u, v) \in E$, on définit :

$$(u, v) \in F \text{ si } c(e) - f(e) > 0$$
$$(v, u) \in F \text{ si } f(e) > 0$$

Dans le premier cas, on définit $u(e) = c(e) - f(e)$. Dans le deuxième cas on définit $u(e) = f(e)$.

Proposition 1.1 Si on pousse de u à v, i.e. si $f(((u, v)) > 0$, alors (v, u) doit apparaître dans le réseau résiduel.

On propose alors l'algorithme de Ford-Fulkerson, se basant sur des chemins augmentants pour résoudre le problème :

Algorithme 1 Ford-Fulkerson

Tant que $G(f)$ a un chemin de s à t noté p, on pousse le plus possible le long du chemin p.

Théorème 1.2 Si l'algorithme de Ford-Fulkerson termine, alors f est un flot maximal.

Démonstration. Soit $U \subseteq V$ l'ensemble des sommets atteignables depuis s dans $G(f)$. On a par 1.1

$$\begin{aligned}\sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) &= \sum_{e \in \delta^+(U)} f(e) - \sum_{e \in \delta^-(U)} f(e) \\ &= \sum_{e \in \delta^+(U)} c(e) = \text{cut size de U}\end{aligned}$$

■

Toutefois la complexité de cet algorithme dépend de la valeur du flot maximum, et celui-ci ne termine même pas pour des réels.

On va chercher un algorithme dont la complexité n'en dépend pas (dit fortement polynomial), ce qui nous amène à la notion de pré-flot :

Définition 1.3 Soit $G = (V, E)$ un graphe orienté, $c : E \to \mathbb{R}^+$ une fonction de capacité et $s, t \in V$ deux sommets terminaux, $f : E \to \mathbb{R}^+$ est un pré-flot si :

1. $0 \leq f(e) \leq c(e), \forall e \in E$
2. $\text{exces}(v) = \sum_{e \in \delta^-(v)} f(e) \geq \sum_{e \in \delta^+(v)} f(e), \forall v \in V \setminus \{s, t\}$.

On va essayer de construire un algorithme dont le principe est cette fois ci d'avancer

Définition 1.4

- Un sommet $v \in V \setminus \{s, t\}$ est dite *actif* si $\text{exces}(v) > 0$.
- Un étiquetage des sommets $d : V \to \mathbb{N}$ est valide si $\forall (u, v) \in G(f)$ (pour f un pré-flot), on a : $d(u) \leq d(v) + 1$.
- Une arête $(u, v) \in G(f)$ est admissible si $d(u) = d(v) + 1$.

On obtient alors l'algorithme suivant, proposé originellement par Andrew Goldberg en 1989 :

On va donc prouver la correction de cet algorithme. Pour cela, on se base sur les deux lemmes suivants :

Lemme 1.1 Si v est actif, alors v a un chemin orienté vers s dans $G(f)$.

Démonstration. Soit $X \subseteq V$ l'ensemble des sommets ayant un chemin vers s dans $G(f)$. Par l'absurde, il existe $w \in V \setminus X$ actif. On a alors :

$$0 < \sum_{v \in V \setminus X} \left(\sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) \right) = \sum_{e \in \delta^-(v \setminus x)} f(e) - \sum_{e \in \delta^+(V \setminus X)} f(e)$$

Algorithme 2 Push-Relabel

Initialisation : On pose $\forall e \in \delta^+(s), f(e) = c(e)$, sinon $f(e) = 0$. On pose $d(s) = n, d(v) = 0 \forall v \neq s$.

Boucle Tant qu'il existe un sommet actif v, on effectue deux actions :

`Push` S'il existe $(u, v) \in G(f)$ admissible, on pousse $\min(\text{exces}(u), u(e))$ selon l'arête (u, v).

`Relabel` On pose $d(u) = \min_{v|(u,v)\in G(f)} d(v) + 1$

Or, $\sum_{e\in\delta^-(v\setminus x)} f(e) = 0$, d'où le résultat. ■

Lemme 1.2 Étant donné un chemin P de u à v, alors, $|P| \geq d(u) - d(v)$, si d est valide.

Démonstration. Si $P = v_0 v_1 \cdots v_x$, puisque d est valide : $d(v_i) \leq d(v_{i+1}) + 1$ pour tout i. D'où, $d(v_0) \leq d(v_x) + x$. D'où le résultat. ■

On obtient un corollaire très utile :

Corollaire 1.1 Pour tout n, $d(u) \leq 2n - 1$.

Démonstration. Si u est actif et $d(u) = 2n$, tout chemin de u à s est de longueur au moins n, ce qui est impossible puisque $|V| = n$. ■

Théorème 1.3 — Correction de Push-Relabel Quand l'algorithme 2 s'arrête, on obtient un max-flow.

Démonstration. Il n'y a jamais de chemin de s à t dans $G(f)$ par 1.2. Par ailleurs, il n'y a pas de sommet actif à l'arrêt de l'algorithme, ce qui signifie qu'on a bien un véritable flot. La preuve de correction de 1 s'applique donc. ■

Théorème 1.4 — Complexité de Push-Relabel L'algorithme 2 s'arrête en temps $\mathcal{O}(V^2E)$.

Démonstration. On a toujours 3 opérations :

- Le `Relabel` qui prend un temps $\mathcal{O}(V^2)$ (au plus $(n-2) \times (2n-1)$ opérations).
- Le `Push Saturant` (push qui permet à $f((u, v))$ d'atteindre $c((u, v))$). Celui-ci va supprimer l'arête (u, v) de $G(f)$. Pour que l'arc soit réinséré dans $G(f)$ pour un autre push saturant, v doit d'abord être réétiqueté. Ensuite, après un push sur (v, u), u doit être réétiqueté. Au cours du processus, $d(u)$ augmente d'au moins 2 Il y a donc $\mathcal{O}(V)$ push saturants sur (u, v) et donc $\mathcal{O}(VE)$ push saturants au total.
- Le `Push Non-Saturant` qu'on effectue un nombre $\mathcal{O}(V^2E)$ de fois. En effet, borner le nombre de push non-saturants peut se faire à partir d'un argument de potentiel. On utilise la fonction de potentiel $\Phi = \sum_{v \text{ actif}} d(v)$. Il est clair que $\Phi = 0$ à l'initialisation et reste toujours positive durant l'exécution. Par ailleurs, un push non-saturant diminue Φ d'au moins 1. De plus, le relabel et le push augmentent Φ d'au plus 1 et d'au plus $(2V-1)$ respectivement. On a donc : $\Phi \leq (2V-1)(V-2) + (2V-1)(2VE)$. On a donc : $\Phi \leq \mathcal{O}\left(V^2E\right)$.

L'algorithme prend donc un temps $\mathcal{O}\left(V^2E\right)$. ■

Pour essayer d'améliorer l'algorithme on propose la version suivante : Il est clair que l'algorithme reste correcte,

Algorithme 3 Push-Relabel +

Boucle : On choisit le sommet actif v avec la plus haute étiquette, on effectue deux actions :

`Push` S'il existe $(u, v) \in G(f)$ admissible, on pousse $\min(\text{exces}(u), u(e))$ selon l'arête (u, v).

`Relabel` On pose $d(u) = \min_{v|(u,v)\in G(f)} d(v) + 1$

toutefois, on change la complexité pour de V^2E à V^3. On peut même encore améliorer la complexité pour obtenir $\mathcal{O}(V^2\sqrt{E})$, et même $\mathcal{O}(V^{1+o(1)}\log(E))$

Algorithme 4 Edmonds-Karp

Pour cet algorithme, on applique Ford-Fulkerson en choisissant le plus court des chemins de s à t.

Théorème 1.5 — Complexité d'Edmonds-Karp L'algorithme de Edmonds-Karp prend un temps $\mathcal{O}(VE^2)$.

Dans la suite, on note $f_0, f_1, \cdots$ les flots obtenus de sorte que f_{i+1} est obtenu du plus court chemin P_i dans $G(f_i)$.

Lemme 1.3 $\forall i$:

- $|P_i| \leq |P_{i+1}|$
- Si P_i et P_{i+1} utilisent deux arcs opposés (i.e. (u,v) et (v,u)), alors $|P_i| + 2 \leq |P_{i+1}|$.

Démonstration. On pose $H = P_i \cup P_{i+1}$ où les arcs opposés sont supprimés. On ajoute alors 2 arcs supplémentaires de t à s. Comme alors H est eulérien, il existe deux chemins disjoints q_1, q_2 de s à t dans H. Notons que toutes les arêtes de H (sauf les arêtes de t à s) sont dans $G(f_i)$. On a de plus $|P_i| \leq q_1, q_2$ d'où

$$2\,|P_i| \leq |q_1| + |q_2| \leq |H| \leq |P_i| + |P_{i+1}| - 2$$

■

Lemme 1.4 Soit $l < k$ tel que P_l et P_k utilisent des arcs opposés. Alors, $|P_l| + 2 \leq |P_k|$.

Démonstration. La preuve précédente peut être aisément adaptée : On peut supposer que pour $l < i < k$, P_i n'a pas d'arcs opposés avec P_k, sinon le résultat se déduit par récurrence par le lemme précédent. On pose $H = P_l \cup P_k$ où les arcs opposés sont supprimés. On ajoute alors 2 arcs supplémentaires de t à s. Comme alors H est eulérien, il existe deux chemins disjoints q_1, q_2 de s à t dans H. Notons que toutes les arêtes de H (sauf les arêtes de t à s) sont dans $G(f_l)$. On a de plus $|P_l| \leq q_1, q_2$ d'où

$$2\,|P_l| \leq |q_1| + |q_2| \leq |H| \leq |P_l| + |P_k| - 2$$

■

Lemme 1.5 Un arc dans $G(f)$ peut être une arête bottleneck (c'est à dire une arête avec le moins de capacité) au plus $\mathcal{O}(n)$ fois.

Démonstration. Pour qu'une arête e soit bottleneck une nouvelle fois, il faut que la longueur du nouveau plus court chemin ait augmenté au moins de 2. ■

de la Complexité d'Edmonds-Karp. Chaque arête peut être utilisée au plus $\mathcal{O}(n)$, il y a donc au plus nm itérations, et les itérations se font en temps $\mathcal{O}(m)$, ce qui est le résultat. ■