Optimisation Combinatoire

Chien-Chen Huang

3 octobre 2024

Table des matières

1 Max-Flow (min-cut)

1.1 Ford-Fulkerson

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.

1.2 Push-Relabel

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 :

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

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 :

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

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 $\mathcal{O}\left(V^2E\right)$. ■

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

Pour essayer d'améliorer l'algorithme on propose la version suivante : Il est clair que l'algorithme reste correcte, 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 $\mathcal{O}(V^{1+o(1)}\log(E))$

1.3 Edmonds-Karp

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. ■

Démonstration 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. ■

2 Couplage Maximal

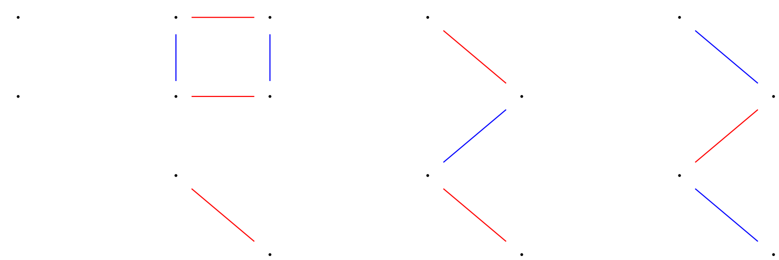
2.1 Théorème de Kőnig

Définition 2.1 Etant donné un graphe $G = (V, E)$, $M \subseteq E$ est un couplage si et seulement si $\delta_M(v) \leq 1$ pour tout noeud v.
Un chemin P est dit M-alternant s'il alterne entre une arête de M et une arêtre de $E \setminus M$.
Un chemin M-alternant P est dit M-augmentant s'il commence et termine par un noeud non couvert par M.

Théorème 2.1 Un couplage M est maximal si et seulement si il n'y a pas de chemin M-augmentant.

Démonstration. $\Leftarrow$ S'il y a un chemin M-augmentant

$\Rightarrow$ Supposons qu'il existe un M^* avec $|M^*| > |M|$ est M n'a pas de chemin augmentant. $M^* \otimes M$ est un sous-graphe de G avec degré dans $\{0, 1, 2\}$. On a plusieurs possiblités pour ce graphe (M^* est en rouge, M en bleu).



Toutefois, le quatrième type de graphe n'est pas possible puisque s'il y a plus de rouges que de bleus, on a un chemin augmentant pour M. Alors en suivant le

■

Théorème 2.2 On peut trouver un couplage maximal en $\mathcal{O}(mn)$.

Démonstration. ■

Théorème 2.3 — de Kőnig Dans un graphe biparti $G = (A \sqcup B, E)$:

$$\max_{couplage} |M| = \min_{\text{VERTEX COVER}} |C|$$

Avant de prouver le théorème, une observation :

Proposition 2.1 — Dualité Faible. Dans un graphe G (non nécessairement biparti), on a :

$$|M| \leq |C|$$

Démonstration. Puisque les arêtes de M ne couvrent qu'au plus une fois chaque sommet, en particulier :

$$\sum_{v \in C} 1 \geq \sum_{(u,v) \in M} (1 + 1) \geq \sum_{e \in M} 1$$

■

Démonstration du Théorème de Kőnig. Soit M un couplage maximal de G. On pose U l'ensemble des sommets non atteints par M dans A et Z l'ensemble des sommets connecté par un chemin M alternant aux sommets de U. On pose $S = Z \cap X$ et $T = Z \cap Y$. Alors, chaque sommet de T est atteint par M et T est l'ensemble des voisins de S. Posons $K = (X \setminus S) \cup T$. Chaque arête de G a une de ses extrémités dans K. Donc K est une couverture des sommets de G et $|M| = |K|$ d'où le résultat. ■

2.2 Dualité et Programmation Linéaire

Définition 2.2 La programmation linéaire s'intéresse à la maximisiation de ${}^{t}wX$ en vérifiant $X \leq 0$ et $AX \leq b$ (problème de packing) ou à la minimisation de $b^T Y$ en vérifiant ${}^{t}Ay \geq w$ avec $y \geq 0$ (problème de covering).

Quelques exemples :

■ **Exemple 2.1 — Couplage Maximal comme Programmation Linéaire.** On cherche à maximiser $\sum_e X_e$ en ayant $\forall v \in A \cup B \sum_{e \in \delta(v)} X_e \leq 1$ et $\forall e \in E, X_e \geq 0$. On a donc ici A la matrice $(A_{v \in V, e \in E})$ d'incidence du graphe et b et w le vecteur Attila. On essaye de mettre autant d'arêtes que possible (packing).

Ce problème peut donc aussi s'écrire comme la minimisation de $\sum_{v \in A \cup B} Y_v$ en ayant $Y_u + Y_v \geq 1 \forall e = (u, v) \in E$, $Y_u \geq 0 \forall u \in A \cup B$. On essaye de mettre aussi peu d'arêtes que possible (covering). ■

Proposition 2.2 La dualité faible de la programmation linéaire est : $\forall x, y$ qui vérifie les contraintes, on a :

$$ {}^{t}wX \leq {}^{t}bY $$

Démonstration.

$$ {}^{t}wx \leq \left({}^{t}YA\right) X = {}^{t}Y\,(AX) \leq {}^{t}Yb = {}^{t}bY $$

■

Ceci redémontre la dualité faible pour les graphes.

Théorème 2.4 — Dualité Forte Il existe X^*, Y^* tels que ${}^{t}wX^* = {}^{t}bY^*$.

La dualité permet de faire les liens entre Max-Flow et Min-Cut ainsi qu'entre Max-Matching et Min-Vertex-Cover.

2.3 Couplage Pondéré

Définition 2.3 Soit un graphe $G = (A \cup B, E = A \times B)$ avec $|A| = |B|$ et une fonction de poids entière sur les arêtes, on dit que $\pi : A \cup B \to \mathbb{N}$ est une couverture si $\forall e \in E, \pi(a) + \pi(b) \geq w(e)$.

Théorème 2.5 — Egrevary On a :

$$ \max_{M \text{ couplage parfait}} w(M) = \min_{\pi \text{ couverture entière}} \sum_{v \in A \cup B} \pi(v) $$

Proposition 2.3 — Dualité Faible. On a : $w(M) \leq \sum \pi(v)$.

Démonstration. De même que précédemment en remplaçant par l'inégalité d'une couverture. ■

Démonstration du Théorème d'Egevary. Soit π une couverture minimale. Posons $G_\pi \subseteq G$ où une arête $e = (a, b)$ est dans G_π si et seulement si $\pi(a) + \pi(b) = w(e)$. C'est le sous-graphe des arêtes qu'on ne peut pas modifier. Soit M le couplage maximal dans G_π. Si $|M| = |A| = |B|$ alors on a fini puisqu'on a toujours égalité dans la preuve de la dualité faible. Sinon, si M n'est pas parfait, on définit π' comme suit :

$$\begin{cases} \forall a \in A_1 & \pi'(a) = \pi(a) - 1 \\ \forall b \in B_1 & \pi'(b) = \pi(b) + 1 \\ \forall v \notin A_1 \cup B_1 & \pi'(v) = \pi(v) \end{cases}$$

où A_1 est l'ensemble des sommets atteignables depuis un sommet non atteint pas M passant par un chemin augmentant et B_1 est l'ensemble des sommets couplés à des sommets de A_1. On définit également B_2 l'ensemble des sommets de B atteignables depuis un sommet non atteint par M en passant par un chemin M-augmentant, A_2 l'ensemble des sommets couplés à des sommets de B_2 et A_3, B_3 le reste de A et B. Puisque les arêtes qui ne sont pas dans G_ε vérifient $\pi(a) + \pi(b) \geq w\left((a, b)\right) + 1$, on a toujours $\pi'(a) + \pi'(b) \geq w(e)$. De plus, on a bien $\sum \pi' < \sum \pi$ et π' couvre toutes les arêtes, mais π' peut être négative. Si $\pi'(a_0) = -1$, alors on définit π'' comme :

$$\begin{cases} \pi''(a) = \pi'(a) + 1 & \forall a \in A \\ \pi''(b) = \pi'(b) - 1 & \forall b \in B \end{cases}$$

Ceci garantit que tous les $\pi''(a)$ sont non négatifs et que $\sum \pi'' = \sum \pi'$. Par ailleurs les $\pi''(b)$ sont aussi non négatifs. En effet, si on arrive à celà, on a une arête d'un poids $0 \leq w(e) \leq -1 + 0$. Donc π'' est une couverture plus petite que π. ■

Ceci nous donne par ailleurs un algorithme pour calculer une couverture minimale/un couplage maximal. Toutefois, celui-ci effectue au plus $|A| \max_e w(e) = \frac{nw}{2}$ itérations et est donc pseudo polynomial. Par ailleurs, cet algorithme ne termine pas nécessairement sur les réels.

La version ci-dessous, appelée méthode hongroise, termine quant à elle sur les réels et est plus efficace :

Algorithme 5 Méthode Hongroise (Kuhn 1957)

```
M ← ∅, π(a) ← max_{e∈δ(a)} w(e) et π(b) ← 0
while M n'est pas parfait do
    Construire G_π.
    Trouver le couplage maximal M dans G_π (par augmentation) et définir A_1, A_2, A_3, B_1, B_2, B_3
    Poser Δ ← min_{a∈A_1, b∈B_2∪B_3} π(a) + π(b) − w((a, b))
    π(a) ← π(a) − Δ et π(b) ← π(b) + Δ
    δ ← min_{a∈A} π(a).
    if δ < 0 then
        π(a) ← π(a) − δ et π(b) ← π(b) + δ
    end if
end while
```

Théorème 2.6 Cette méthode termine et est demande un temps $\mathcal{O}(n^2nn)$.

Démonstration.

- La boucle prend un temps $\mathcal{O}(m)$ pour s'effectuer.
- On peut l'effectuer au plus n fois (lorsqu'$|A_1|$ augmente) et n fois lorsque $|M|$ augmente. Puisque ces deux effets peuvent se combiner, on a bien le résultat.

■

2.4 Relaxation et Optimalité

Proposition 2.4 — Relaxation. Étant donné un problème de programmation linéaire dual, X^* et Y^* sont tous les deux optimaux si et seulement si :

$$X_i^* \Longrightarrow \left({}^{\mathsf{t}}AY^*\right)_i = W_i$$
$$Y_j^* \Longrightarrow (AX^*)_j = b_j$$

Démonstration. On a :

$$^{\mathsf{t}}WX \leq {}^{\mathsf{t}}YAX \leq {}^{\mathsf{t}}Yb$$

Il y a égalité si et seulement si les contraintes sont serrées. D'où le résultat. ■

On peut appliquer ceci aux problèmes de couplage maximal : Si une arête est utilisée ($X_e > 0$), alors $Y_u + Y_v = 1$ et donc elle est la seule à atteindre l'une de ses extrémités.

On peut appliquer de même cela à Max-Flow Min-Cut