# Chapitre 5

# Problèmes de flot

## 5.1 Introduction et définitions

Les problèmes dits de transport remontent au début des années cinquante. Il s'agissait alors d'acheminer une certaine denrée, on parle de *commodité* (anglais commodity), entre des sites de production et des sites de consommation, via un «réseau» formé de liens de communication connectant les sites entre eux (piple-lines, routes, lignes téléphoniques, réseau télécom) ; de plus chaque lien supporte un débit maximal limité que l'on appelle capacité du lien.

En général, la plupart des sites ne produisent ni ne consomment, ils servent simplement a l'interconnection des liens. On associe aussi souvent aux sites de production et de consommation des réels correspondant respectivement à leur production et leur consommation maximales.

**L'objectif principal est de maximiser le trafic écoulé.**

Dans un premier temps, nous considérons que le réseau est muni de liens directionnels, i.e. qui peuvent être utilisés dans un seul sens. Le modèle mathématique est celui du *réseau de flot* suivant :

**Définition 5.1** (Réseau de flot)**.** Un *réseau de flot* est un quadruplet $(G, pr_{max}, co_{max}, c)$ tel que :
- $G$ est un graphe orienté dont les sommets représentent les sites et les arc les liens ;
- $pr_{max}$ est une fonction de $V(G)$ dans $\mathbb{R}^+ \cup +\infty$ ; $pr_{max}(v)$ correspond à la production maximale possible en $v$. Si $v$ n'est pas un site de production $pr_{max}(v) = 0$.
- $co_{max}$ est une fonction de $V(G)$ dans $\mathbb{R}^+ \cup +\infty$ ; $co_{max}(v)$ correspond à la consommation maximale possible en $v$. Si $v$ n'est pas un site de consommation alors $co_{max}(v) = 0$.
- $c$ est une fonction de $E(G)$ dans $\mathbb{R}^+ \cup +\infty$ ; $c(e)$ correspond à la capacité du lien $e$.

**Définition 5.2** (Flot)**.** Le *flot* est un triplet $F = (pr, co, f)$ où
- $pr$ est une fonction de production (acheminée) telle que pour tout sommet $v$, $0 \leq pr(v) \leq pr_{max}(v)$ ;
- $co$ est une fonction de consommation (reçue) telle que pour tout sommet $v$, $0 \leq co(v) \leq co_{max}(v)$ ;
- $f$ est une fonction définie sur $E$ appelée *fonction de flot* qui vérifie les contraintes suivantes :

$$\begin{array}{rll} \text{Positivité :} & \forall e \in E\ , & f(e) \geq 0 \\ \text{Contrainte de capacité :} & \forall e \in E, & f(e) \leq c(e) \\ \text{Conservation du flot :} & \forall v \in V, & \displaystyle\sum_{(u,v)\in E} f((u,v)) + pr(v) = \sum_{(v,u)\in E} f((v,u)) + co(v) \end{array}$$

En sommant les $|V|$ équations de conservation de flot, on obtient :

$$\sum_{v\in V} pr(v) = \sum_{v\in V} co(v)$$

Autrement dit, la somme des productions acheminées est égale à la somme des consommations reçues. Cette valeur correspond à la quantité de trafic écoulé, c'est la *valeur du flot* notée $v(F)$.

Ainsi notre problème est le suivant :

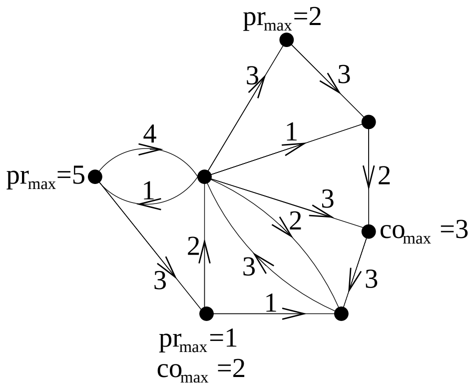



FIG. 5.1 – Exemple de problème de transport et un flot de valeur 4

> **Problème 5.3** (flot maximal). Etant donné un réseau de flot, trouver un flot de valeur maximale.

## 5.2 Réduction à la forme élémentaire

Avant d'étudier ce problème, nous allons voir que nous pouvons nous ramener à un problème sous forme élémentaire pour lequel un seul sommet produit avec production maximale infinie et un seul sommet consomme avec consommation maximale infinie.

**Définition 5.4** (Réseau élémentaire). Un réseau de flot $(G, \infty_s, \infty_p, c)$ est dit *élémentaire* si :
- $s$ et $p$ deux sommets distincts tels que $s$ est une source (i. e. $d^-(s) = 0$) et $p$ est un puits (i.e. $d^+(p) = 0$) ;
- $\infty_s(s) = +\infty$ et $\infty_s(v) = 0$ si $v \neq s$ ;
- $\infty_p(p) = +\infty$ et $\infty_p(v) = 0$ si $v \neq p$.

Par commodité, on dénote aussi $(G, \infty_s, \infty_p, c)$ par $(G, s, p, c)$.

**Définition 5.5** (Réseau élémentaire associé). Si $N = (G, pr_{max}, co_{max}, c)$ est un réseau de flot, son réseau de flot élémentaire *associé* est le réseau élémentaire suivant de $N = (\bar{G}, s, p, \bar{c})$ obtenu à partir de $G$ de la façon suivante :
- on ajoute une source $s$ et un puits $p$ ;
- on relie $s$ à chaque site de production non nulle $v$ avec un arc $(s, v)$ de capacité $pr_{max}(v)$ ;
- on relie chaque site de consommation $v$ à $p$ avec un arc $(v, p)$ de capacité $co_{max}(v)$ ;

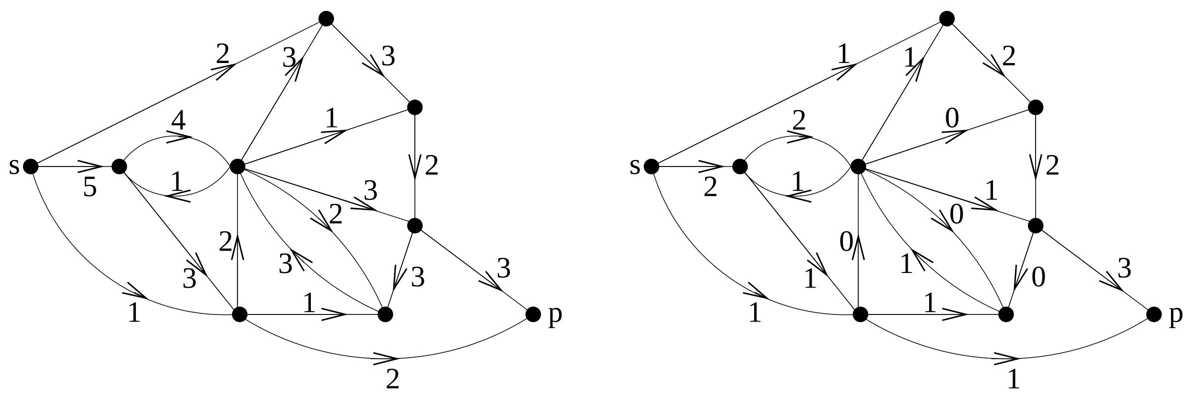



FIG. 5.2 – Problème élémentaire et flot associé au problème et flot de la Figure 5.1

Il est facile de voir que tout flot $F = (pr, co, f)$ du problème élémentaire est en bijection avec le flot du problème originel $F' = (pr', co', f')$ défini comme suit : pour tout site de production $u$, $pr'(u) = f((s,u))$ ; pour tout site de consommation $v$, $co'(v) = f((v,p))$ et pour tout arc $(x,y)$, $f'(x,y) = f(x,y)$.

Clairement, ces deux flots ont même valeur $v(F) = v(F') = pr(s) = co(p)$. Ainsi,
**Résoudre le problème du flot maximal dans un réseau de flot est équivalent à résoudre le problème du flot maximal dans le réseau de flot élémentaire associé.**

**Dans la suite, nous ne considérerons plus que des réseaux de flots élémentaires.**

Remarquons que dans un réseau de flot (élémentaire) $N = (G, s, p, c)$, la contrainte de conservation du flot s'écrit :

$$\forall v \in V \setminus \{s,p\}, \sum_{(u,v)\in E} f((u,v)) = \sum_{(v,u)\in E} f((v,u))$$

De plus, un flot $F = (pr, co, f)$ d'un réseau (élémentaire) est entièrement défini par sa fonction de flot $f$ car d'après la conservation du flot en $s$ et $p$, on a

$$\begin{aligned} v(F) &= pr(s) = \sum_{u\in V,(s,u)\in E} f((s,u)) \\ &= co(p) = \sum_{u\in V,(u,p)\in E} f((u,p)) \end{aligned}$$

Ainsi, par commodité, nous identifierons souvent un flot $F$ avec sa fonction $f$ et nous noterons $v(f)$ la valeur du flot.

Afin d'alléger les notations, pour un flot $f$ ou une capacité $c$, nous noterons $f((u,v))$ par $f(u,v)$ et $c((u,v))$ par $c(u,v)$.

## 5.3 Coupe et majoration du flot maximal

Nous allons montrer que ce qui limite la valeur du flot maximal dans un réseau c'est l'existence d'un ou plusieurs «goulots d'entranglement» par où doit transiter le flot (e.g. la commodité). Informellement, pour voyager de la source au puits, le flot doit traverser la frontière d'un ensemble de sommet, et c'est la taille (la capacité globale des arêtes impliquées) qui va limiter la valeur du flot. On appelle une telle frontière une *coupe*.

**Définition 5.6** (Coupe)**.** Dans un réseau de flot $N = (G, s, p, c)$, on appelle *coupe* une bipartition $C = (V_s, V_p)$ des sommets de $G$ tel que $s \in V_s$ et $p \in V_p$. La *capacité de la coupe* $C$, notée $\delta(C)$, la somme des capacités des arcs sortant de $V_s$ (i.e. des arcs $(u,v)$ avec $u \in V_s$ et $v \in V_p$).

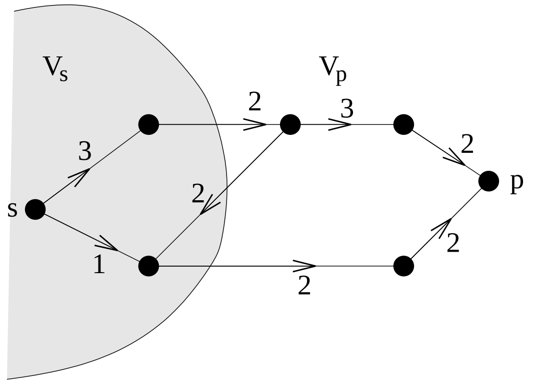



FIG. 5.3 – Un réseau et une coupe de capacité 4.

**Définition 5.7** (flot entrant, flot sortant). Soit $f$ un flot et $C = (V_s, V_p)$ une coupe, on note $out(f, C)$ la somme des flots sortant de $V_s$, et $in(f, C)$ la somme des flots entrant dans $V_s$ :

$$out(f, C) = \sum_{(u,v)\in E,\ u\in V_s, v\in V_p} f(u, v) \tag{5.1}$$

$$in(f, C) = \sum_{(u,v)\in E,\ u\in V_p, v\in V_s} f(u, v) \tag{5.2}$$

Notons que la loi de conservation implique que

$$\forall C \text{ une coupe }, v(f) = out(f, C) - in(f, C) \tag{5.3}$$

En particulier la valeur du flot est toujours inférieure à $out(f, C)$. Or, clairement, $out(f, C) \leq \delta(C)$, donc $v(f) \leq \delta(C)$.

Une coupe peut être considérée comme un passage par où le flot doit passer, sa capacité limite donc la valeur du flot.

Posons $v_{max} = max\{v(f), f \text{ un flot }\}$ et $\delta_{min} = min\{\delta(C),\ C \text{ une coupe}\}$. D'après ce qui précède :

$$v_{max} \leq \delta_{min}$$

Le théorème suivant, dû à Ford et Fulkerson, affirme qu'il y a en fait égalité.

**Théorème 5.8** (Ford & Fulkerson). *La valeur maximum d'un flot est égale à la valeur minimum de la capacité d'une coupe, i.e.*

$$v_{max} = \delta_{min}$$

On dit souvent en abrégé que *flot max égale coupe min*, c'est un exemple de «théorème min-max».

Bien qu'il existe une preuve non constructive de ce théorème, notre preuve sera la preuve originale qui repose sur un algorithme calculant un flot et une coupe de valeur et capacité identiques.

## 5.4 Réseau auxiliaire et poussée

La plupart des algorithmes de calcul de flot maximum sont basés sur l'idée suivante : partant d'un flot déjà existant (au départ il peut être nul) on va augmenter le flot allant de la source au puits en poussant littéralement la commodité là où c'est possible.

Les algorithmes se distinguent seulement par la méthode utilisée afin de déterminer par où et comment pousser du flot.

On définit pour cela le *réseau auxiliaire*, ce graphe dépend du flot $f$ déja établi, nous le notons $N(f)$.

**Définition 5.9** (réseau auxiliaire). Étant donné un réseau de flot $N = (G, s, p, c)$ et un flot $f$, nous allons construire *le réseau auxiliaire* associé $N(f) = (G(f), s, p, c_f)$. Pour cela, pour tout couple de sommets $(u, v)$, on pose

$$c_f(u, v) = c(u, v) - f(u, v) + f(v, u)$$

avec $c(u, v)$, $f(u, v)$ et $f(v, u)$ égales à 0 si elle ne sont pas définies. Notons que $c_f(u, v) \geq 0$ puisque $c(u, v) - f(u, v) \geq 0$. Le réseau $G(f)$ est alors défini comme suit :

$$\begin{aligned} V(G(f)) &= V(G) \\ E(G(f)) &= \{(u, v) \mid c_f(u, v) > 0\} \end{aligned}$$

Notons que $c_f(u, v) + c_f(v, u) = c(u, v) + c(v, u)$. Intuitivement, $c_f(u, v)$ est la somme de la capacité restante sur l'arc $(u, v)$ soit $c(u, v) - f(u, v)$ à laquelle est ajoutée la capacité fictive $f(v, u)$, qui permet de «retirer» le flot circulant sur l'arc $(v, u)$ ce qui correspond à pousser fictivement du flot le long de l'arc $(u, v)$. On verra les exemples des figures 5.4 et 5.5.

Notons que le réseau auxiliaire n'a aucun arc de capacité nulle. Ceci est important car cela garantit qu'en trouvant un chemin $P$ dans $G(f)$ la capacité minimale d'un de ses arcs est strictement positive.

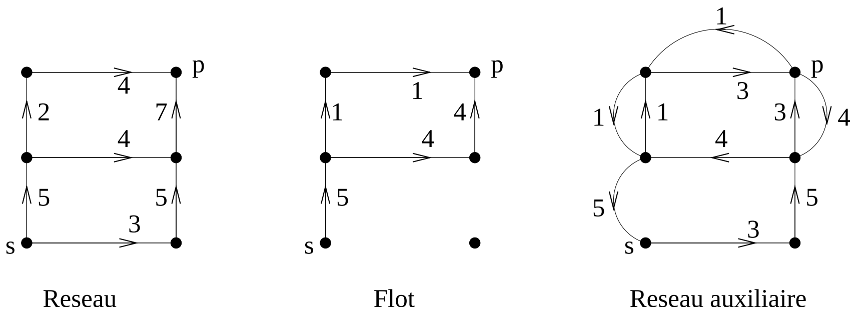



FIG. 5.4 – Un réseau, un flot et le réseau auxiliaire associé

Soit $P$ est un chemin de $s$ vers $p$ dans $N(f)$, où la capacité minimale d'un arc de $P$ est $\varepsilon > 0$, le flot $f'$ obtenu en ***poussant $\varepsilon$ unités de flot*** le long de $P$ est défini comme suit :
Pout tout arc $(u,v) \in E(P)$, on va pousser le flot sur $(u,v)$ c'est à dire "augmenter" le flot de $\varepsilon$ sur $(u,v)$. Deux cas peuvent se produire :

- Si la capacité de $(u,v)$ est suffisante pour faire passer $\varepsilon$ unités de flots i.e. $f(u,v)+\varepsilon \leq c(u,v)$ alors $f'(u,v) = f(u,v)+\varepsilon$ et $f'(v,u) = f(v,u)$.
- Si la capacité de $(u,v)$ est insuffisante pour faire passer $\varepsilon$ unités de flots i.e. $f(u,v)+\varepsilon > c(u,v)$, alors d'après la définition du réseau auxiliaire, cela implique que $f(v,u) \geq f(u,v)+\varepsilon - c(u,v) > 0$. On a donc acheminé $f(u,v)+\varepsilon-c(u,v)$ unités de flots en trop sur $(v,u)$ qu'il faut retirer. On pose donc $f'(u,v) = c(u,v)$ et $f'(v,u) = f(v,u) - (f(u,v)+\varepsilon) + c(u,v)$.

Si $(u,v)$ et $(v,u)$ ne sont pas dans $P$ alors le flot reste inchangé sur ces arcs, $f'(u,v) = f(u,v)$ et $f'(v,u) = f(v,u)$.

On vérifie facilement que :

**Lemme 5.10.** *Si $f$ est un flot de valeur $v(f)$, $f'$ est un flot de valeur $v(f)+\varepsilon$.*

*Démonstration.* Laissée au lecteur. Voir Exercice 5.5. □

Par exemple, à partir du flot dans le réseau dessiné Figure 5.4, en prenant le $(s,p)$-chemin représenté Figure 5.5 de capacité minimale 1, en effectuant une poussée, on obtient le nouveau flot et le nouveau réseau auxiliaire de cette même figure.

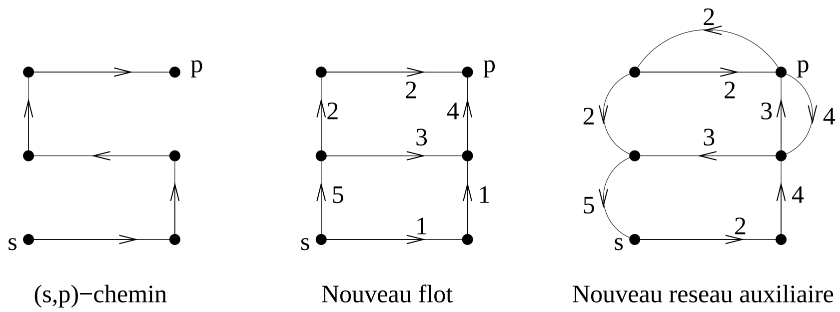



FIG. 5.5 – Une poussée suivant un $(s,p)$-chemin du réseau auxiliaire de la Figure 5.4.

Nous disposons donc d'un morceau d'algorithme :

1. Prendre un flot quelconque $f$ ;
2. Calculer le réseau auxiliaire $N(f)$ ;
3. Chercher un chemin de $s$ vers $p$ dans $G(f)$ ;
4. S'il existe un tel chemin $P$, pousser du flot le long de $P$, mettre $f$ à jour.

Notons que s'il n'existe pas de chemin entre la source et le puits dans le graphe auxiliaire notre stratégie n'aboutit à rien ; mais dans ce cas nous allons voir que le flot courant est maximal !

**Proposition 5.11.** *Si $G(f)$ ne contient pas de chemin de la source vers le puits, le flot $f$ est de valeur maximum.*

*Démonstration.* Soit $V_s$, l'ensemble de tous les sommets que l'on peut atteindre depuis $s$ dans $G(f)$ par un chemin. Nous savons que $V_s$ ne contient pas le puits, puisqu'il n'existe pas de $(s,p)$-chemin. Donc $V_p = V \setminus V_s$ contient $p$.

Soit $(u,v)$ un arc de $G$ avec $u \in V_s$ et $v \in V_p$. Par définition de $V_s$, l'arc $(u,v)$ n'est pas dans $G(f)$. D'après la définition de $G(f)$, ceci signifie que $c_f(u,v) = 0$. Donc $f$ est tel que $f(u,v) = c(u,v)$ et $f(v,u) = 0$.

Soit $C$ la coupe $(V_s, V_p)$. On a alors $out(f,C) = \delta(C)$ et $in(f,C) = 0$. En termes naturels, il n'y a pas de flot entrant dans $V_s$, et tous les arcs sortant de $V_s$ sont saturés.

Or $v(f) = out(f,C) - in(f,C)$ donc

$$v(f) = \delta(C)$$

Le flot courant est donc de valeur égale à la capacité de la coupe $C$ ; il est donc maximum. □

Notons que cette preuve permet aussi d'exhiber une coupe de capacité égale au flot de valeur maximum, il suffit en effet de prendre l'ensemble des sommets atteignables depuis la source et son complémentaire dans le réseau auxiliaire lorsque qu'il n'existe plus de chemin reliant la source au puits dans $G(f)$.

Cela nous permet donc de montrer le Théorème 5.8.
**Preuve du Théorème 5.8** : Soit $f$ un flot de valeur maximale dans un réseau de flot $(G,s,p,c)$ et $N(f)$ le réseau de flot auxiliaire. Dans $G(f)$, il n'y a pas de $(s,p)$-chemin sinon en effectuant une poussée suivant ce chemin par le Lemme 5.10, on obtient un flot de plus grande valeur que $f$ ce qui est une contradiction. Ainsi suivant la preuve de Proposition 5.11, la coupe $C = (V_s, V_p)$ où $V_s$ est l'ensemble des sommets $v$ tels qu'il existe un $(s,v)$-chemin dans $G(f)$ est de capacité $v(f) = v_{max}$. Ainsi $\delta_{min} \leq \delta(C) \leq v_{max}$. □

## 5.5 Algorithme initial (Ford Fulkerson 1956)

Nous disposons maintenant d'un véritable algorithme :

**Algorithme 5.1** (Ford et Fulkerson)**.**
1. Débuter par le flot nul $f = 0$ ;
2. Calculer le réseau auxiliaire $N(f)$ ;
3. Chercher un chemin de $s$ vers $p$ dans $G(f)$ ;
4. S'il existe un tel chemin $P$, pousser du flot le long de $P$, mettre $f$ à jour et aller en 2 ;
5. Sinon terminer et retourner $f$ comme solution du problème.

Il n'y a donc plus qu'à le programmer, en essayant de l'implémenter de façon efficace ; mais attention :

1) Certes, si cet algorithme se termine il fournit toujours la bonne solution ; mais se termine-t-il toujours ?
2) Et si c'est le cas, quel est son temps d'éxecution ?

La réponse à la question 1 est une sorte de ni oui ni non, puisque nous allons voir que l'algorithme se termine toujours si les capacités sont des nombres rationnels, mais qu'il peut durer un temps infini si les capacités sont réelles. Par ailleurs même quand les capacités sont des nombres entiers, le temps d'execution dépend linéairement de la valeur du flot maximum, qui peut être grande.

### Analyse de l'algorithme de Ford Fulkerson

**Proposition 5.12.** *L'algorithme de Ford & Fulkerson se termine après au plus $v(f)$ recherches de plus court chemin si les capacités sont entières, et $\mu v(f)$ recherches de plus court chemin (où $\mu$ est le PPCM des dénominateurs des capacités) si les capacités sont rationnelles.*

*Démonstration.* - Si les capacités sont des nombres entiers, nous allons effectuer des poussées d'au moins une unité de flot par itération (car dans ce cas $\varepsilon$ est un entier ; chaque itération nécessitant la recherche d'un chemin de $s$ vers $p$ dans le graphe $G(f)$ (Algorithme de Dijkstra de coût $O(\ |E|+|V|\log(|V|)\ )$ ). Soit un temps au pire de :

$$O(\ v(f)(|E|+|V|\log|V|)\ ).$$

- Si les capacités sont rationnelles, l'algorithme termine aussi puisque que les augmentations sont toujours au moins de $\frac{1}{\mu}$ où $\mu$ est le PPCM des dénominateurs des fractions utilisées : ce nombre $\mu$ peut être très grand mais il est fixé. En fait, on peut aussi résoudre le problème en multipliant les capacités par $\mu$ et en résolvant le problème entier obtenu : il est homothétique au premier.

□

**Remarque 5.13.** Si toutes les capacités sont entières alors l'algorithme de Ford-Fulkerson renvoie un flot maximal à valeur entière, puisque chaque poussé utilise un nombre entier d'unités de flot.

**Proposition 5.14.** *Si les capacités sont réelles l'algorithme peut ne pas se terminer.*

*Démonstration.* Si l'on utilise des capacités irrationnelles l'algorithme peut ne pas terminer : on peut construire un exemple compliqué où on effectue une série infinie de poussées de valeurs $\varepsilon_1, \varepsilon_2, \dots \varepsilon_n, \dots$, et où $\sum_{i=0,+\infty} \varepsilon_i < v_{max}$.) □

**Remarque 5.15.** Bien que la terminaison ne soit pas garantie dans le cas de capacités irrationelles, le théorème 5.8 ($v_{max} = \delta_{min}$), reste vrai. On prouve le cas réel par «passage à la limite du cas rationnel». Pratiquement, seul le cas rationnel est important, les ordinateurs travaillant en précision finie.

La borne $v(f)|E|$ sur le nombre de poussées, n'est pas satisfaisante, car elle dépend de la valeur du flot qui peut être très importante ; la figure 5.6 montre un exemple où $v_{max}$ poussées, si elles sont mal choisies, sont nécessaires afin d'atteindre un flot de valeur maximum.

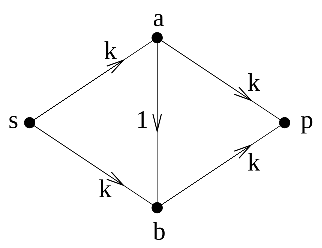



FIG. 5.6 – Exemple où $2k = v_{max}$ poussées peuvent être effectuées.

En effet, si on pousse alternativement sur $P_1 = (s, a, b, p)$ et $P_2 = (s, b, a, p)$, on poussera d'une unité de flot à chaque fois car la capacité de $(a, b)$ ou $(b, a)$ vaut 1. Ainsi $2k = v_{max}$ poussées ($k$ sur $P_1$ et $k$ sur $P_2$) seront donc nécessaires.

Pour améliorer l'algorithme précédent et assurer sa terminaison rapide dans tous les cas l'idée est d'effectuer les poussées dans un certain ordre.

## 5.6 Algorithme de poussée par les plus courts chemins (Edmonds et Karp)

Il s'est déroulé dix ans avant que la preuve de l'existence d'une bonne stratégie de poussée soit faite ; l'idée est pourtant plutôt simple, au lieu de choisir un chemin quelconque de la source vers le puits on choisit *un plus court chemin quelconque.*

**Algorithme 5.2** (Edmonds et Karp)**.**
1. Débuter par le flot nul $f = 0$ ;
2. Calculer le réseau auxiliaire $N(f)$ ;
3. Chercher un plus court chemin de $s$ vers $p$ dans $G(f)$ ;
4. S'il existe un tel chemin $P$, pousser du flot le long de $P$, mettre $f$ à jour et aller en 2 ;
5. Sinon terminer et retourner $f$ comme solution du problème.

Nous allons prouver que l'algorithme de poussée par les plus courts chemins effectue au plus $\frac{|E||V|}{2}$ poussées, soit une complexité d'au plus $\frac{|E||V|}{2}$ recherches de plus court chemin. Notons que à la différence de l'algorithme de Ford & Fulkerson cette borne est indépendante de la valeur des capacités.

La preuve est basée sur deux propriétés simples :

1) Au cours des itérations, la distance entre $s$ et $p$ dans les graphes auxiliaires successifs ne peut pas décroitre ;
2) Au cours des itérations, la distance entre $s$ et $p$ ne peut pas rester inchangée plus de $|E|$ fois consécutivement.

Supposons un flot $f_0$ fixé, et posons $G_0 = G(f_0)$ ; Nous notons
- $d_0$ la distance dans le graphe $G_0$.
- $E'$ l'ensemble des arcs de $G_0$ qui appartiennent à un plus court chemin de $s$ vers $p$ dans $G_0$ ;
- $E'^-$ l'ensemble des arcs inverses de $E'$ (i.e. les arcs obtenus en renversant ceux de $E'$) ;
- $G_1$ le graphe obtenu en ajoutant à $G_0$ les arcs de $E'^-$.

**Lemme 5.16.** *Le graphe $G_1$ a les propriétés suivantes :*

*(i) Un chemin de $s$ vers $p$ de longueur $d_0(s,p)$ n'utilise pas d'arc $E'^-$.*

*(ii) La distance de $s$ à $p$ dans $G_1$ est $d_0(s,p)$ ;*

*Démonstration.* (i) Soit $P$ un $(s,p)$-chemin $P$ dans $G_1$ contenant un ou plusieurs arcs de $E'^-$, et supposons que $(v,u)$ soit le dernier de ces arcs. Montrons que $P$ n'est pas un $(s,p)$-chemin de longueur minimale. Comme $(u,v)$ appartient à un plus court $(s,p)$-chemin, nous avons $d_0(u,p) = d_0(v,p) + 1$. Comme entre $u$ et $p$ le chemin $P$ reste dans $G_0$, sa longueur est $l + 1 + d_0(u,p)$ (où $l$ est la distance parcourue sur $P$ pour aller de $s$ à $v$). Ainsi $P$ est de longueur $l + 2 + d_0(v,p)$. Le chemin $P'$ obtenu en suivant $P$ jusqu'à $v$ puis en allant directement vers $p$ le long d'un plus court chemin, est de longueur $l + d_0(v,p) = l + d_0(v,p)$. Donc $P$ n'est pas de longueur minimale.

(ii) D'après (i), un plus court $(s,p)$-chemin est dans $G_0$ donc est de longueur $d_0(s,p)$. □

**Théorème 5.17.** *L'algorithme de poussée par les plus courts chemins effectue au plus $|E||V|$ poussées. Sa complexité est donc de $|E||V|$ recherches de plus court chemin soit $O(|E|^2|V| + |E||V|^2 \log |V|)$.*

*Démonstration.* Lors d'une poussée, on ajoute au réseau auxiliaire des arcs de $E'^-$. D'après le lemme 5.16, cet ajout ne peut pas réduire la distance entre $s$ et $p$. On peut donc décomposer l'algorithme en au plus $|V|$ étapes correspondant à l'ensemble des distances possibles entre $s$ et $p$. Une étape est alors constituée d'un ensemble des poussées avant lesquelles la distance entre $s$ et $p$ est identique.

Considérons maintenant différentes poussées laissant la distance $d(s,p)$ inchangée. Lors des poussées, le réseau auxiliaire $G_0$ varie, puisque certains arcs disparaissent tandis que d'autres sont ajoutés. Cependant, comme la distance entre $s$ et $p$ reste inchangée, les chemins le long desquels les poussées sont effectuées n'utilisent que les arcs de $G_0$ (puisque l'utilisation d'arcs de $E^-$ induit une longueur de $d_0(s,p) + 2$ au moins (cf lemme 5.16). Comme, à chaque poussée au moins un arc de $G_0$ disparaît (On pousse le maximum possible sur le chemin choisi, donc un arc du chemin disparaît.), on effectue donc au plus $|E|$ telles poussées.

Lorsque la distance de $s$ à $p$ augmente, on applique à nouveau l'argument (en définissant un nouveau graphe $G_0$). Comme $d(s,p) \leq |V|$, on a au plus $|V|$ étapes d'au plus $|E|$ poussées. □

**Remarque 5.18.** On peut se demander si l'analyse effectuée est pertinente, autrement dit, existe-t-il un réseau de flot pour lequel il faille effectuer environ $|V||E|$ poussées. La figure 5.7 répond positivement puisque l'algorithme d'Edmonds & Karp utilise $\Theta(|E||V|)$ poussées sur le réseau décrit.

FIG. 5.7 – Un réseau pour lequel $\Theta(|V||E|)$ poussées sont nécessaires avant d'obtenir un flot maximal

## 5.7 Algorithme utilisant un facteur d'échelle

Le problème principal dans l'algorithme de Ford & Fulkerson est la disparité de la valeur des capacités. Dans l'exemple de la Figure 5.6, les poussées peuvent être effectuées le long d'un arc de capacité 1 alors qu'il existe un chemin reliant la source au puits qui est de capacité $k$. L'idée du facteur d'échelle est d'essayer de travailler avec des capacités de tailles comparables, de les saturer et d'introduire alors des capacités de valeur moindre. L'idée et de commencer par considérer les capacités élevées, de pousser du flot dans le réseau constitué uniquement de celles-ci, en essayant de les saturer au mieux ; ensuite on remplace le réseau par le réseau auxiliaire courant auquel on adjoint les capacités plus petites qui n'étaient pas considérées.

Soit $m$ le plus petit entier tel que $c(e) < 2^m$ pour tout arc $e$.

**Algorithme 5.3** (Algorithme échelonné)**.**
0. Calculer le plus petit entier $m$ tel que $c(e) < 2^m$ pour tout arc $e$.
   Poser $c = \sum_{j=0}^{m-1} 2^j c_j$ avec $c_j(e) \in \{0, 1\}$.
1. $i := m - 1$ ; $c' := 0$ et $f := 0$ ;
2. Si $i < 0$ terminer, sinon $c' := c' + 2^i c_i$.
3. Augmenter au maximum le flot $f$ dans $(G, s, p, c')$ ; $i := i - 1$ ; aller Etape 2.

Cet algorithme calcule clairement un flot maximal puisqu'à la fin toutes les capacités sont prises en compte.

Examinons maintenant la complexité de cet algorithme. Pour cela, nous avons besoin de la proposition suivante dont la preuve (aisée) est laissée au lecteur.

**Proposition 5.19.** *Soit $N = (G, s, p, c)$ un réseau de flot et $\alpha$ un réel positif. Soit $e$ un arc de $G$ et $N'$ le réseau de flot $(G, s, p, c')$ avec $c'(e) = c + \alpha$ et $c'(f) = c(f)$ si $f \neq e$. Alors $v_{max}(N') \leq v_{max}(N) + \alpha$.*

**Lemme 5.20.** *Si le flot maximum est de valeur au plus $2^n - 1$, l'algorithme 5.3 effectue au plus $n.|E|$ poussées de flot le long d'un chemin.*

*Démonstration.* Soit $T_i$ le nombre de poussées effectuées lors de l'étape 3 avec $i$, $c'_i = \sum_{j=i}^{m-1} 2^j c_j$ la capacité à cette étape 3.($i$), $g_i$ le flot ajouté lors de cette étape et $f_{i+1}$ le flot total avant cette étape. ($f_i = f_{i+1} + g_i$). Comme toutes les capacités non nulles et le flot sont multiples de $2^i$, les chemins dans le réseau auxiliaire auront capacité minimale multiple de $2^i$. Or après l'etape $3(i+1)$, on ne pouvait pas augmenter le flot donc dans le graphe auxiliaire il n'y a pas de chemin de poids au moins $2^{i+1}$ donc un chemin dans le graphe auxiliaire à l'étape 3.($i$) sera forcément de capacité minimale $2^i$. Ainsi chaque poussée sera d'exactement $2^i$ unités de flot à l'étape 3.($i$) et $v(g_i) \leq |E|2^i$ d'après la Proposition 5.19.

Si $i > n$, i. e. $2^i > v(f)$, on effectuera aucune poussée. Si $i \leq n$, trouver $g_i$ revient à trouver un flot maximal $g'_i$ dans le réseau à capacités entières $(G, s, p, (c'_i - f_{i+1})/2^i)$ avec $v(g'_i) \leq v(g_i)/2^i \leq |E|$. D'après l'analyse de l'algorithme de Ford-Fulkerson, ceci se fait en au plus $|E|$ poussées.

Au total, on a donc bien effectué au plus $n.|E|$ poussées. □

## 5.8 Flots dans les graphes non orientés

Le problème que nous avons étudié jusqu'à présent était modélisé par un graphe orienté mais il y a des problèmes dans lesquels le graphe sous-jacent est non orienté. C'est le cas lorsque les liens sont bidirectionnels. Dans ce cas, chaque lien dispose d'une capacité maximale, mais le flot peut circuler au choix dans un sens et dans l'autre pourvu que la somme des deux trafics ne dépasse pas la capacité du lien.

Un *réseau de flot non orienté* $N = (G, pr_{max}, co_{max}, c)$ est alors défini de manière analogue au cas orienté. La définition de flot cependant change un petit peu car deux valeurs doivent être associées à chaque arête $uv$, $f(u, v)$ correspondant au trafic de $u$ vers $v$ et $f(v, u)$ correspondant au trafic de $v$ vers $u$.

**Définition 5.21** (Flot non orienté)**.** Le *flot* est un triplet $F = (pr, co, f)$ où

- $pr$ est une fonction de production (acheminée) telle que pour tout sommet $v$, $pr(v) \leq pr_{max}(v)$ ;
- $co$ est une fonction de consommation (reçue) telle que pour tout sommet $v$, $co(v) \leq co_{max}(v)$ ;
- $f$ est une fonction définie sur les couples (ordonnés) $(u, v)$ de sommets reliés appelée *fonction de flot* les contraintes suivantes :

$$\begin{array}{rll} \text{Positivité :} & \forall e \in E\ , & f(e) \geq 0 \\ \text{Contrainte de capacité :} & \forall uv \in E, & f((u,v)) + f((v,u)) \leq c(uv) \\ \text{Conservation du flot :} & \forall v \in V, & \displaystyle\sum_{(u,v)\in E} f((u,v)) + pr(v) = \sum_{(v,u)\in E} f((v,u)) + co(v) \end{array}$$

De même que dans le cas orienté, la *valeur du flot* $F$ est

$$v(F) = \sum_{v\in V} pr(v) = \sum_{v\in V} co(v)$$

Soit $N = (G, pr_{max}, co_{max}, c)$ un réseau de flot non orienté. Le réseau de flot (orienté) *associé* à $N$ est $\vec{N} = (\vec{G}, pr_{max}, co_{max}, \vec{c})$ obtenu en remplaçant chaque arête $uv$ par un arc $(u, v)$ et un arc $(v, u)$ tous deux de capacités $c(uv)$. Formellement, ($\vec{G} = (V(G), \displaystyle\bigcup_{uv\in E(G)} \{(u, v), (v, u)\}$) et $\vec{c}((u, v)) = \vec{c}((v, u)) = c(uv)$.

Clairement un flot de fonction $f$ dans $N$ et un flot de fonction $\vec{f}$ dans $\vec{N}$ satisfont les mêmes contraintes mis à part celle de capacité. Celle-ci dans $N$ est

$$\forall uv \in E(G), f((u, v)) + f((v, u)) \leq c(uv)$$

ce qui est plus contraignant que celle dans $\vec{N}$ qui peut s'écrire :

$$\forall uv \in E(G), \vec{f}((u, v)) \leq c(uv) \text{ et } \vec{f}((v, u)) \leq c(uv)$$

Ainsi tout flot dans $N$ est un flot de $\vec{N}$. Donc la valeur maximale d'un flot dans $N$ est inférieure ou égale à la valeur maximale d'un flot dans $\vec{N}$ ; autrement dit $v_{max}(N) \leq v_{max}(\vec{N})$. Nous allons montrer qu'en fait il y a égalité entre ces deux valeurs.

**Définition 5.22** (flot simple)**.** Un flot est *simple* si pour toute paire de sommets $\{u, v\}$, on a $f(u, v) = 0$ ou $f(v, u) = 0$. A tout flot $f$ correspond un flot simple $\tilde{f}$ appelé *simplifié* défini par : si $f(u, v) \geq f(v, u) > 0$ alors $\tilde{f}(u, v) = f(u, v) - f(v, u)$ et $\tilde{f}(v, u) = 0$. Il est facile de voir que $v(f) = v(\tilde{f})$ puisque $f(v, s) = 0$ pour tout sommet $v$ car $d^-(s) = 0$ et donc $f(s, v) = \tilde{f}(s, v)$.

**Proposition 5.23.** *Soit $N$ un réseau de flot non-orienté et $\vec{N}$ son réseau de flot orienté associé.*

$$v_{max}(N) = v_{max}(\vec{N})$$

*Démonstration.* Soit $\vec{F} = (pr, co, \vec{f})$ un flot dans $\vec{N}$. Soit $F = (pr, co, f)$ le flot simplifié de $\vec{F}$. Alors $v(F) = v(\vec{F})$. Comme $F$ est simple, pour toute arête $uv \in E(G)$, on a $f(u, v) = 0$ ou $f(v, u) = 0$. Par ailleurs, la contrainte de capacité dans $\vec{N}$, nous donne $f(u, v) \leq c(uv)$ ou $f(v, u) \leq c(uv)$. Ainsi pour tout $uv \in E(G)$, $f((u, v)) + f((v, u)) \leq c(uv)$. Donc $F$ est un flot de $N$.

Ainsi à tout flot de $\vec{N}$ correspond un flot de $N$ de même valeur. □

Cette proposition nous permet de nous ramener au cas orienté. Pour trouver un flot maximal dans un réseau non orienté, il suffit de résoudre le problème dans le réseau orienté associé et de prendre le simplifié du résultat.

## 5.9 Applications du problème de flot maximal

### 5.9.1 Arête-connexité dans les graphes

Étant donné un graphe (resp. un digraphe), on se demande combien d'arêtes peuvent tomber en panne tout en conservant un réseau en état de marche c'est à dire connexe (resp. fortement connexe).

**Définition 5.24.** Soit $G = (V, E)$ un graphe ou un digraphe.

Soit $u$ et $v$ deux sommets de $G$. L'*arête-connexité* entre $u$ et $v$, notée $\kappa'(u, v)$, est le nombre minimal d'arêtes qu'il faut supprimer afin qu'il n'y ait plus d' $(u, v)$-chemin. Notons que dans le cas, non-orienté, on a $\kappa'(u, v) = \kappa'(v, u)$ alors que dans le cas orienté, ces deux valeurs peuvent être différentes.

L'*arête-connexité* de $G$ est $\kappa' = \min\{\kappa'(u, v) \mid u, v \in V\}$. C' est le nombre minimum de liens à supprimer afin qu'il n'existe pas de $(u, v)$-chemins pour un couple de sommets $(u, v)$. C'est donc le nombre minimum d'arêtes à supprimer afin que le graphe soit non-connexe (resp. non fortement connexe) si $G$ est non-orienté (resp. orienté).

**Théorème 5.25** (Menger). *Soit $u$ et $v$ deux sommets d'un graphe ou digraphe $G$. L'arête-connexité $\kappa'(u, v)$ est égale au nombre maximum d' $(u, v)$-chemins deux à deux arête-disjoints.*

*Démonstration.* La connectivité $\kappa'(u, v)$ n'est autre que la valeur de la capacité d'une coupe dans le réseau de flot $N$ obtenu en affectant à chaque arc une capacité 1 et en choisissant pour source $u$ et pour puits $v$. En effet, si $C = (V_u, V_v)$ est une coupe alors en ôtant les $\delta(C)$ arêtes de $V_u$ vers $V_v$, il n'y a plus de $(u, v)$-chemin. Donc $\kappa'(u, v) \leq \delta_{min}(N)$. Réciproquement, soit $E'$ un ensemble d'arêtes tels qu'il n'existe pas de $(u, v)$-chemin dans $G' = (V, E \setminus E')$. Soit $V_u$, l'ensemble des sommets $w$ de $G'$ tel qu'il existe un $(u, w)$-chemin dans $G'$ et $V_v = V \setminus V_u$. Par définition de $V_u$, toute arête $(x, y)$ avec $x \in V_u$ et $y \in V_v$ est dans $E'$. Donc $\delta((V_u, V_v)) \leq |E'|$. Ainsi $\delta_{min}(N) \leq \kappa'(u, v)$.

D'après le Théorème 5.8, il existe dans ce réseau un flot de valeur $\kappa'(u, v)$. Comme les capacités des arêtes sont unitaires, le flot de valeur $\kappa'(u, v)$ se décompose en un ensemble de $\kappa'(u, v)$ $(u, v)$-chemins deux-à-deux arêtes-disjoints. □

**Corollaire 5.26** (Menger). *Dans un graphe d'arête-connexité $\kappa'$, il existe $\kappa'$ chemins arêtes-disjoints allant d'un sommet à un autre.*

Ce théorème de Menger est plus ancien que le théorème du flot maximum ; il peut être considéré comme un cas particulier de celui-ci, en fait il est équivalent à celui-ci. La preuve de Menger utilisait aussi une méthode de poussée.

### 5.9.2 Couplage maximum dans les graphes bipartis

Les théorèmes de couplage dans les graphes bipartis sont des applications directes du Théorème 5.8. Nous donnons maintenant une preuve du Théorème 4.3 (*"Soit $G = (A, B)$ biparti, il existe un couplage de taille $k$ si et seulement si* $\forall S \subset A, |A| - |S| + |N(S)| \geq k$*"*) en utilisant les flots.

**Preuve du Théorème 4.3** Soit $N = (H, s, p, c)$ défini comme suit : $H$ est le digraphe obtenu à partir de $G$ en orientant toutes les arêtes de $G$ de $A$ vers $B$, en ajoutant une source $s$ et un puits $p$, les arcs $(s, a)$ $a \in A$, $(b, p)$ pour $b \in B$ ; la capacité $c$ vaut 1 sur tous les arcs.

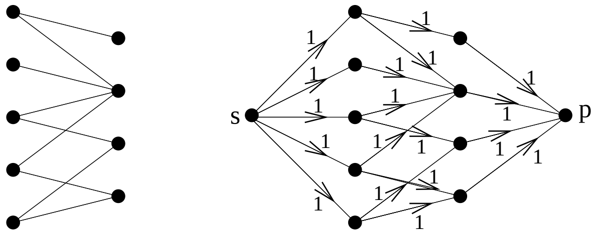



Fig. 5.8 – Graphe biparti et réseau de flot associé

Montrons que la taille maximale d'un couplage de $G$, $\mu$, est égale à la valeur maximale d'un flot dans $N$, $v_{max}$. Soit un couplage $M$ de taille $\mu$ dans $G$ alors le flot $f_M$ valant 1 sur les arcs $(s,a)$, $(a,b)$ et $(b,p)$ si $ab \in M$ et 0 ailleurs est clairement de valeur $\mu$. Donc $\mu \leq v_{max}$. A l'inverse, d'après la Remarque 5.13, il existe un flot maximal $f$ à valeur entières. Donc pour tout arc $e$ $f(e) \in \{0,1\}$. Il est facile de voir que l'ensemble d'arêtes $M_f = \{ab \mid f(a,b) = 1\}$ est un couplage de taille $v(f) = v_{max}$. Donc $\mu \geq v_{max}$.

Soit $C = (V_s, V_p)$ une coupe minimum. Posons $A_s = A \cap V_s$ et $B_s = B \cap V_s$. S'il existe un sommet $b \in (B \cap N(A_s)) \setminus B_s$. Alors posant $C' = (V_s \cup \{b\}, V_p \setminus \{b\})$, on a $\delta(C') \leq \delta(C) - 1 + 1 = \delta(C)$. Ainsi quitte à rajouter les sommets de $(B \cap N(A_s)) \setminus B_s$ dans $V_s$, on peut supposer que la coupe minimum que nous considérons est telle que $N(A_s) \subseteq B_s$.

Considérons maintenant la capacité de $C$.

$$\begin{aligned}\delta_{min} = \delta(C) &= |\{(s,a) \mid a \in A \setminus A_s\}| + |\{(b,p) \mid b \in B_s\}| + |\{(a,b)\} \mid a \in A_s, b \in B \setminus B_s\}| \\ &= |A| - |A_s| + |B_s|\end{aligned}$$

Comme $N(A_s) \subseteq B_s$, il vient $\delta_{min} \geq |A| - |A_s| + |N(A_s)|$. De plus, la coupe $(A_s \cup N(A_s), V(H) \setminus (A_s \cup N(A_s)))$ est de capacité $|A| - |A_s| + |N(A_s|$. Donc $\delta_{min} = |A| - |A_s| + |N(A_s)|$. Nous concluons que

$$\delta_{min} = \min\{S \subset A \mid |A| - |S| + N(S)\}$$

Ainsi, d'après le Théorème 5.8, $\mu = \min\{S \subset A \mid |A| - |S| + N(S)\}$. □

### 5.9.3 Clôture de bénéfice maximal

Dans ce problème, on dispose d'un certain nombre de tâches $t \in T$ auxquelles on associe un bénéfice $b(t)$. Le bénéfice peut être négatif (ce qui correspond à une perte). On note $T^+$ (resp. $T^-$) l'ensemble des tâches de bénéfice positif (resp. négatif).

Par ailleurs, il existe des contraintes de *clôture*, c'est à dire que le choix d'une tâche peut impliquer le choix d'une ou plusieurs autres tâches. On représente cette relation de clôture (e.g. d'implication) par un digraphe dont les sommets sont les tâches, et où il existe un arc $(t_1, t_2)$ si et seulement si le choix de $t_1$ implique le choix de $t_2$.

L'objectif est de trouver un ensemble de tâches *cohérentes*, de bénéfice maximal, c'est à dire un sous-ensemble $A$ de $T$ tel que :

- Aucun arc ne sorte de $A$ $((A, \overline{A}) = \emptyset)$, (cohérence) ;
- $\sum_{t \in A} b(t)$ soit maximum.

Montrons comment ce problème se ramène à un problème de coupe de capacité minimum, et donc à un problème de flot maximum. Soit $N = (G, s, p, c)$ le réseau de flot suivant :

- $V(G) = T \cup \{s, p\}$ ;
- pour toute tâche $t$ de bénéfice positif, on relie la source $s$ à sommet $t$ avec un arc $(s,t)$ de capacité $c(s,t) = b(t)$ ;
- pour toute tâche $t$ de bénéfice negatif, on relie le sommet $t$ au puits $p$ avec un arc $(t,p)$ de capacité $c(t,p) = -b(t)$ ;
- si une tâche $t_1$ implique une tâche $t_2$, on ajoute l'arc $(t_1, t_2)$ de capacité infinie au réseau.

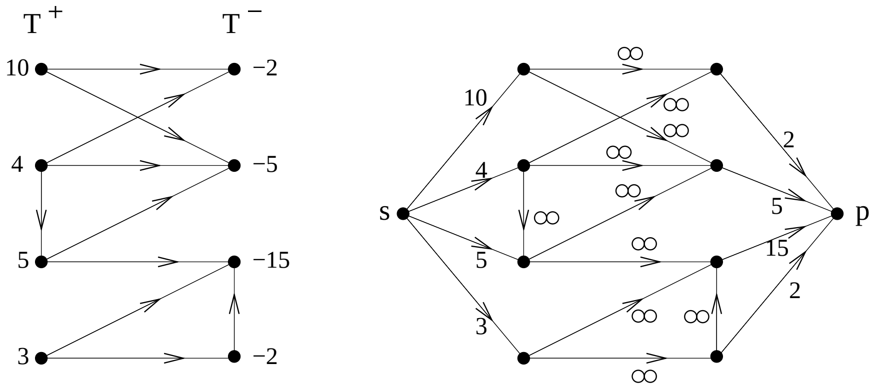



FIG. 5.9 – Un problème de clôture et le réseau de flot associé

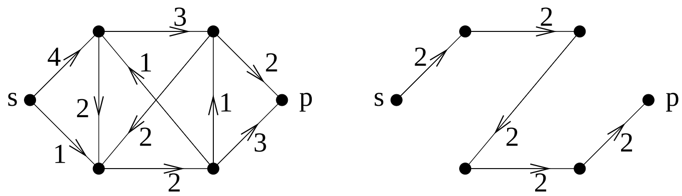
Considérons maintenant une coupe $C = (V_s, V_p)$ de capacité finie du réseau de flot. Nous avons $S = \{s\} \cup A$ avec $A \subset T$. De plus, la coupe étant de capacité finie, $A$ est un ensemble de tâches cohérentes. Soit $A^+ = A \cap T^+$ et $A^- = A \cap T^-$. La capacité de $C$ est alors :

$$\begin{aligned}
\delta(C) &= \sum_{t \in T^+ \setminus A^+} c(s,t) + \sum_{t \in A^-} c(t,p) \\
&= \sum_{t \in T^+ \setminus A^+} b(t) - \sum_{t \in A^-} b(t) \\
&= \sum_{t \in T^+} b(t) - \sum_{t \in A^+} b(t) - \sum_{t \in A^-} b(t) \\
&= \sum_{t \in T^+} b(t) - \sum_{t \in A} b(t)
\end{aligned}$$

Comme $\sum_{t \in T^+} b(t)$, la somme de tous les bénéfices positifs, est une constante, minimiser la capacité de la coupe $S = \{s\} \cup A$ équivaut à maximiser le bénéfice de $A$.

## 5.10 Exercices

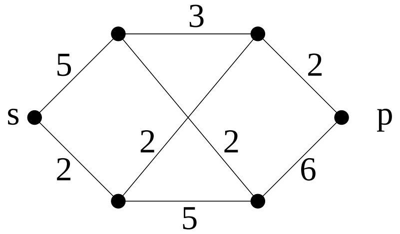
**Exercice 5.1.** Soient $N$ le réseau de flot et $f_0$ le flot dans $N$ dessiné ci-dessous.




reseau flot initial

1) Partir du flot $f_0$ pour trouver un flot maximum. On détaillera les étapes de l'algorithme.
2) Donner une coupe minimum de ce réseau.

**Exercice 5.2.** Trouver le flot maximum et la coupe minimum dans le réseau de flot ci-dessous. (On détaillera les étapes de l'algorithme de «poussée».)



**Exercice 5.3.** Modéliser le problème suivant à l'aide d'un problème de flot : on dispose de 3 sites de production $A, B, C$ et de 5 sites de consommation $1, 2, 3, 4, 5$ ;

la production des sites, et leur consommation sont données dans le tableau suivant :

| $A$ | $B$ | $C$ |
|---|---|---|
| 5 | 4 | 7 |

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 3 | 4 | 5 | 2 | 1 |

Enfin chaque site de production ne peut servir que certains sites de consommation :

| $A$ | $B$ | $C$ |
|---|---|---|
| 13 | 24 | 345 |

Donner une solution au problème ou expliquer pourquoi elle n'existe pas.

**Exercice 5.4.** Montrer la propriété, pour toute coupe $C = (V_s, V_p)$, $v(f) = out(f, C) - in(f, C)$. En quoi la propriété $s \in V_s, p \in V_p$ est importante ?

**Exercice 5.5.** Prouver le lemme 5.10. On verifiera que $f'$ vérifie les contraintes de capacité, de positivité et la loi de conservation.

**Exercice 5.6.** Le *support* d'un flot est l'ensemble des arcs où la fonction de flot est strictement positive. Montrer qu'il existe un toujours un flot maximum dont le support ne contient pas de circuit.

**Exercice 5.7.** Modifier l'algorithme de calcul de flot maximal afin d'obtenir un algorithme calculant la coupe minimale dans un réseau de flot.

**Exercice 5.8.** Au lieu d'effectuer une poussée le long d'un chemin quelconque on veut utiliser le chemin offrant la capacité résiduelle maximale. Donner un algorithme recherchant le chemin dont la capacité résiduelle est maximale. Montrer que, si lors d'une étape on pousse $x$ unités de flot, on poussera moins de $x$ unités de flot pour toutes les étapes suivantes.

**Exercice 5.9.** Démontrer le théorème 5.8 en utilisant le théorème 5.25 de Menger.

**Exercice 5.10.** On considère un réseau de flot $N = (G, s, p, c)$ et l'on associe à chaque sommet $v \in V(G)$ un entier $w(v)$ ; on souhaite alors calculer un flot maximum dans $N$ qui satisfasse la condition supplémentaire suivante $\forall v \in V$ la somme des flots entrant en $v$ est inférieure à $w(v)$ (i.e. $\displaystyle\sum_{u \in V, uv \in E} f(uv) \leq w(v)$). Montrer comment le faire en calculant un flot maximum sur un réseau obtenu en modifiant légerement $N$.

**Exercice 5.11.** Soit $G$ un graphe. Soient $u$ et $v$ deux sommets. La *connexité* entre $u$ et $v$, notée $\kappa(u, v)$, et le nombre minimal de sommets que l'on doit retirer du graphe afin qu'il soit dans des composantes connexes différentes. La *connexité* de $G$, notée $\kappa(G)$, est la cardinalité minimum d'un ensemble $S \subset V$ tel que $G - S$ n'est pas connexe. Un graphe est *$k$-connexe* si $\kappa(G) \geq k$.

1) Montrer que $\kappa(G) = \min\{\kappa(u, v) \mid u, v \in V(G)\}$.

2) Montrer que $\kappa(u, v)$ est égal au nombre maximum de chemins deux à deux sommet-disjoints reliant $u$ et $v$ . (On pourra utiliser l'exercice 5.10.)

3) En déduire le théorème suivant du à Menger : *Un graphe est $k$-connexe si et seulement si pour tout couple de sommets $(u, v)$, il existe $k$ $(u, v)$-chemins sommet-disjoints.*

4) Donner un exemple de graphe régulier de degré $k$ et de connexité 1.

5) Montrer que un graphe est régulier de degré $4k$, alors il contient un sous-graphe $k$-connexe. (On pourra montrer le résultat par induction sur le nombre de sommets de $G$).

**Exercice 5.12.** On dispose de 4 tâches à donner à 4 machines. Chaque tache peut être effectuée par chaque machine en un temps donné par le tableau ci-dessous.

| Tâches | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| Machine A | 3 | 4 | 6 | 5 |
| Machine B | 6 | 2 | 4 | 6 |
| Machine C | 4 | 4 | 5 | 6 |
| Machine D | 4 | 3 | 6 | 4 |

Chaque machine ne reçoit qu'une tâche. On souhaite minimiser le temps total de calcul (somme de tous les temps de calcul). Formuler le problème comme un problème de couplage ou de flot. Montrer que si une solution n'est pas optimale, alors on peut pousser du flot le long d'un cycle du graphe résiduel et diminuer le prix de la solution.

**Exercice 5.13.** Plusieurs entreprises envoient des représentants à une conférence ; la $i$ème entreprise envoie $m_i$ représentants. Lors de la conférence, plusieurs groupes de travail sont menés simultanément ; le $j$ème groupe peut recevoir au plus $n_j$ participants. Les organisateurs veulent répartir les participants dans les groupes de manière à ce que deux participants d'une même enteprise ne soient pas dans le même groupe. (Les groupes n'ont pas besoin d'être remplis.)

a) Montrer comment utiliser un réseau de flot pour tester si les contraintes peuvent être satisfaites.

b) S'il y a $p$ compagnies et $q$ groupes indexés de telle manière que $m_1 \geq \cdots m_p$ et $n_1 \leq \cdots \leq n_q$. Montrer qu'il existe une répartition des participants en groupes satisfiant les contraintes si et seulement si, pour tout $0 \leq k \leq p$ et $0 \leq l \leq q$, on a $k(q-l) + \sum_{j=1}^{l} n_j \geq \sum_{i=1}^{k} m_i$.

**Exercice 5.14.** Le problème du routage optique avec $k$ couleurs est le suivant : on souhaite effectuer un ensemble $R \subset \{s\} \times V$ de requêtes de communication point à point en mode connecté dans un graphe orienté $G$. Pour cela, on doit donc affecter à chaque requête un chemin du graphe ; de plus, on doit affecter à chaque requête une des $k$ couleurs disponibles de façon à ce que deux requêtes partageant un arc aient des couleurs différentes.

1) Modéliser le problème du routage optique avec 1 couleur comme un problème de flot entier sur le graphe $G$.

2) Modéliser le problème du routage optique avec $k$ couleurs comme un problème de flot entier sur un graphe $G$ obtenu à partir de $k$ copies de $G$.

3) En déduire une borne supérieure de la complexité du problème.

**Exercice 5.15** (flot insécable)**.** On considère un problème de flot avec une source et plusieurs puits $p_1, p_2, \ldots p_n$. Le puits $p_i$ demande $d_i$ unités de flot. On ajoute la contrainte suivante : le trafic de $s$ à $p_i$ doit être routé le long d'un chemin unique.

Montrer que ce problème est NP-complet.