MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABDELHAMID BEN BADIS DE MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET DE L'INFORMATIQUE
DÉPARTEMENT DE MATHÉMATIQUES ET INFORMATIQUE
FILIÈRE : MATHÉMATIQUES


MÉMOIRE DE FIN D'ÉTUDES

**Pour l'Obtention du Diplôme de Master en Mathématiques délivré par**

**Université de Mostaganem**

**Spécialité "Modélisation, Contrôle et Optimisation"**

*présenté par :*

**Hayat MILOUDI**

# Sur Les Problème Du Flot Dans Un Réseau

*soutenu le 23 juin 2021devant le jury composé de :*

| | | | |
|---|---|---|---|
| **Président :** | Maghnia HAMOU MAAMAR | Grade | UMAB |
| **Examinateur :** | Prénoms NOM | Grade | UMAB |
| **Encadreur :** | Hocine ABLAOUI | Grade | UMAB |

Année Universitaire : 2020 / 2021

MASTER

# Remarciements

Notre premier remerciement va à Allah soubhanou Wa taala.
En terminant notre mémoire de fin d'étude, je suis est agréable d'adresser nos vifs remerciements à tous ceux qui m'aidé de près ou de loin à élaborer cet ouvrage.
Je tiens à remercier vivement mon encadreur. Mr. Ablaoui Hocine , trouve ici le témoignage de ma profonde reconnaissance. Ses encouragements, et surtout ses critiques, Sa sensibilisation, ont largement contribué à l'accomplissement de mes travaux. Je le remercie infiniment de m'avoir toujours poussé vers l'avant.
Je vous remercie pour votre aide précieuse et votre temps madame Hamou Maamar Maghnia.
Je remercie aussi les personnes qui m'ont aidé Hamou Mohamed, Djaafer Abdelhak.
Mes remerciements également à l'ensemble de nos enseignants de la faculté des sciences exactes et informatique pour tous le savoir qu'ils nous ont donné toute au long de notre formation à l'université de Mostaganem.
Ala fin je donne un grand remerciement tous nos collègues d'étude, particulièrement notre promotion.
Merci à tous à toutes.

# Dédicase

**A mes très chers parents**

A celle que sa prière m'a ouvert toute les portes. C'est ma chère maman mes yeux, mon appui.
Sans oublier mon père chéri et la lumière de mes yeux Que dieu leur fasse miséricorde et le fasse habiter en paix.

**A mes très chères sœurs, frères**

Aucune dédicace ne serait exprimée assez profondément ce que je ressens envers vous.
Je vous dirai tout simplement, un grand merci, je vous aime. À mes frères d'amour, Et mes adorables sœurs

**A mes très chers amies**

En témoignage de l'amitié sincère qui nous a liées et des bons moments passés ensemble. Je vous dédie ce travail en vous souhaitant un avenir radieux et plein de bonnes promesses.
Particulièrement :
Imen ,Kheira, Hakima, Fatiha, Lahcen, Abdelhak
En souvenir de nos éclats de rire, des bons moments et des nuits blanches.

**MILOUDY HAYAT.**

# Résumé

Dans ce travaile, nous avons étudié et programmé l'algorithme de Ford et Fulkerson et ses variantes.

Nous avons sur tout proposé une méthode heuristique pour déterminer le flot maximum de $s$ à $p$ dans un réseau.

# Table des matières

# Liste des tableaux

# Table des figures

# Introduction

Un réseau de flot est un graphe orienté pondéré où chaque arc possède une capacité le long du quel peut circuler un flux. Le cumul des flots sur une arc ne peut pas excéder sa capacité. Pour qu'un flot soit réalisable, il faut que la somme des flots sont entrant au niveau de chaque nœud, autre que la source $s$ et le puits $p$,soit égale à la somme des flots sortants (loi de Kirchhoff). Un réseau peut être utilisé pour modéliser le trafic dans un réseau routier, la circulation de fluides dans des conduites d'eau, le gaz, d'électricité au d'autres fluides transitant à travers les nœuds d'un réseau.

Notre travail est composé d'une introduction et de cinque chapitres. Le premier chapitre est dédié aux rappels sur la théorie des graphes, le deuxième chapitre sera consacré flots et tensions,dans le troisième chapitre on introduit le problème du flot maximum d'un source $s$ vers un puits $p$, et nous présentons le fameux algorithme de Ford et Fulkerson [] et sa variante la procédure d'Edmonds et karp [2].Le quatrième chapitre traite du théorème de la capacité minimum et la présentation d'un heuristique de recherche des flot maximum de $s$ à $p$. et, c'est la que réside notre propre contribution. Au cinquième chapitre nous donnons quelques résultats numérique pour comparer l'efficacité de l'heuristique que nous avons conçu avec l'algorithme de Ford et Fulkerson et celui d'Edmonds et karp.

Nous terminons ce travail par dresser une court conclusion.

# Chapitre 1

# Préliminaire

La théorie des graphes est un outil très puissant pour modéliser des problèmes aussi divers que l'optimisation combinatoire, le problème d'économies de chimie, de physique et de bien d'autre domaines.

## 1 Définitions de base

soit X un ensemble
On note par :

- ◇ |X| : le cardinale de l'ensemble X
- ◇ $\mathscr{P}_2(x) = \{x, y / x \in X \quad , y \in X\}$
  On distingue deux classes de graphes [1] : les graphes orientés qui constituent la classe le plus générale et la classe des graphes non orientés qui modélisent un grand nombre de problèmes réels. Notons que tout graphe non orienté peut être considéré comme un graphe orienté.

**Définition 1.1** *un graphe* G *orienté est défini par la donnée :*

- *D'un ensemble fini* X *appelé ensemble de Sommets*
- *D'un ensemble fini* U *appelé ensemble d' Arcs*
- *d'une application :*

$$I : U \rightarrow X$$

*pour tout arc u de* U *ou lui associe son extrémité initiale* $I(u)$

- *d'une application :*

$$T : U \rightarrow X$$

*Pour tout arc u de* U *ou lui associe son extrémité terminale* $T(u)$.
*On en conclut qu'un graphe* G *est un quadruplé* $G = (X, U, I, T)$.
*Par abus d'écriture, et s'il n'y a pas de confusions possibles, on écrira :* $G = (X, U)$ *et les applications* I *et* T *sont considérées de manière implicite.*
*On appelle ordre d'un graphe* $G = (X, U)$*, le nombra* $n = |X|$.
*dans tout ce qui suit, on supposera (sauf indication) que :*
$G = (X, U)$, $n = |X|$ *et* $m = |U|$.
$X = \{x_1, x_2, ..., x_n\}$
$U = \{u_1, u_2, ..., u_m\}$ *Soit u un arc de* U *dont les extrémités,respectivement, initiale et terminale sont x et y alors :*

- *u est noté par :* $u = (x, y)$ *( ou* $u = (y, x)$*).*

- *les sommet x et y sont dits adjacents*
- *si x et y sont confondus alors u est dit boucle.*

**Représentation graphique d'un graphe orienté**
On représente les sommets par des ronds et les arcs par des flêches

**Exemple 1.1**

$$x \in \mathrm{X}, y \in \mathrm{X}, u \in \mathrm{U}$$
$$\mathrm{I}(u) = x, \mathrm{T}(u) = y$$

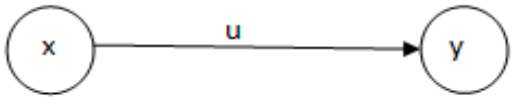



FIGURE 1.1 – graphe oriente

**Définition 1.2** *Soit x un sommet d'un graphe orienté :*

- $\mathrm{U}^-(x) = \{y \in \mathrm{U}, (y,x) \in \mathrm{U}\}$ *: ensemble des prédécesseurs de x*
- $\mathrm{U}^+(x) = \{y \in \mathrm{U}, (x,y) \in \mathrm{U}\}$ *: ensemble des successeurs de x*
- $\mathrm{U}(x) = \mathrm{U}^-(x) \cap \mathrm{U}^-(x)$ *: ensemble des voisins (ou sommets adjacents) de x*

**Définition 1.3** *degre d'un sommet*
*soit* $\mathrm{G} = (\mathrm{X}, \mathrm{U}, \mathrm{I}, \mathrm{T})$ *un graphe et soit* $x \in \mathrm{X}$
*On defini :*

— *le demi-degré extérieur de x dans* G *, que l'on note* $d^+_{\mathrm{G}}(x)$ *, par :*

$$d^+_{\mathrm{G}}(x) = |\{u, u \in \mathrm{U}/\mathrm{I}(u) = x\}|$$

— *le demi-degré intérieur de x dans* G *, que l'on note* $d^-_{\mathrm{G}}(x)$*, par :*

$$d^-_{\mathrm{G}}(x) = |\{e, e \in \mathrm{U}/\mathrm{T}(e) = x\}|$$

— *et enfin le degré de x dans* G *, que l'on note* $d_{\mathrm{G}}(x)$*, par :*

$$d_{\mathrm{G}}(x) = d^+_{\mathrm{G}}(x) + d^-_{\mathrm{G}}(x)$$

**Exemple 1.2** *soit* $\mathrm{G} = (\mathrm{X}, \mathrm{U}, \mathrm{I}, \mathrm{T})$ *un graphe,* $\mathrm{X} = \{1,2,3,4\}$ *,* $\mathrm{U} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ *et les applications* I *et* T *sont donnés par le tableau ci-dessous :*

| $e$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ |
|---|---|---|---|---|---|---|---|
| $\mathrm{I}(e)$ | 1 | 1 | 3 | 2 | 2 | 1 | 1 |
| $\mathrm{T}(e)$ | 2 | 4 | 3 | 1 | 2 | 3 | 2 |

*Notre graphe peut être représenté comme suit :*

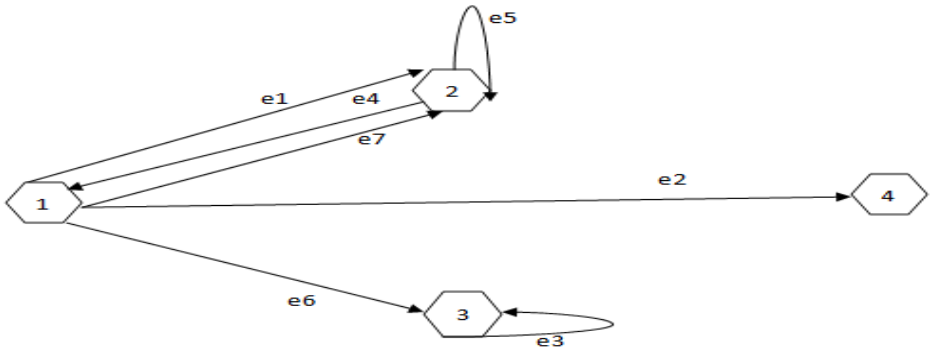



FIGURE 1.2

*les demi-degrés exterieurs et interieurs et les degrés de chaque sommet sont donnés par le tableau suivant :*

| $x$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $d^+_G(x)$ | 3 | 2 | 1 | 0 |
| $d^-_G$ | 1 | 3 | 2 | 1 |
| $d_G(x)$ | 4 | 5 | 3 | 1 |

**Définition 1.4** *Un graphe* G *non orienté est defini par la donnée :*

- *d'un ensemble* X*, fini,appelé aussi ensemeble de sommets*
- *d'un ensemble* U*, fini, appelé ensemble d'arêtes*
- *d'une application e :*

$$u : \mathrm{U} \longrightarrow \mathscr{P}_2(\mathrm{X})$$

*pour toute arête u de* U*, on lui associe ses extrémités* $e(u)$ *, qu' on note tout simplement* $(x, y) = e(u)$

**Exemple 1.3** *soit* $\mathrm{G} = (\mathrm{X}, \mathrm{U})$ *avec* $\mathrm{X} = \{1, 2, 3\}$ *et* $\mathrm{U} = \{e_1, e_2, e_3, e_4, e_5\}$

| $e$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|---|
| $x_i$ | 1 | 1 | 1 | 3 | 2 |
| $x_j$ | 2 | 2 | 3 | 3 | 3 |

*Notre graphe peut être représenté comme suit :*

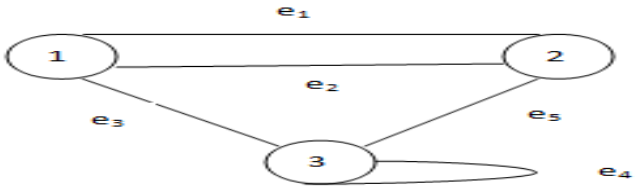



FIGURE 1.3 – représentation graphique G

**Définition 1.5** *Deux sommets sont adjacents s'ils sont liés par une arête ( ou un arc )*

**Définition 1.6** *Soient* $G = (X, U)$ *un graphe orienté et* $u = (x, y)$ *un arc de* $U$
*x et y sont dits Adjacents*
*x et y sont deux sommets Adjacents*
*L'arc u est adjacent aux sommets x et y*
*x et y sont Adjacents á u*
*deux arcs u et v sont dits Adjacents s'ils sont adjacents á* 1 *un sommet commun*

**Définition 1.7** *Soient* $G = (X, U)$ *un graphe,* $X_1$ *un sous-ensemble de sommets de* $X$ *et* $U_1$ *un sous-ensemble d'arcs ( ou arêtes ).*

*Le sous-graphe construit sur* $X_1$ *est le graphe* $G_1 = (X_1, E_1)$ *avec*
$E_1 = \{u \in U /$ *les deux extremites de u sont dans* $X_1$ *}*

*Le graphe partiel construit sur* $E_1$ *est le graphe* $G_2 = (X, E_1)$

*Le sous-graphe partiel construit sur* $X_1$ *et* $E_1$ *est le graphe* $G_3 = (X_1, E_2)$ *avec*
$E_2 = \{u \in E_1 /$ *les deux extremites de u sont dans* $X_1\}$

**Définition 1.8** *Un chemin* C, *allant d'un sommet a à b, est une séquence d'arcs* $(u_1, u_2, ..., u_p)$ *vérifiant :*

*1-* $a = I(u_1)$

*2-* $b = T(u_p)$

*3-* $\forall i = 2, 3, ..., p : I(u_i) = T(u_{i-1})$

**Exemple 1.4** *soit* $G = (X, U)$ *un graphe avec* $X = \{1, 2, 3, 4\}$ *et* $U = \{u_1, u_2, u_3, u_4, u_5\}$

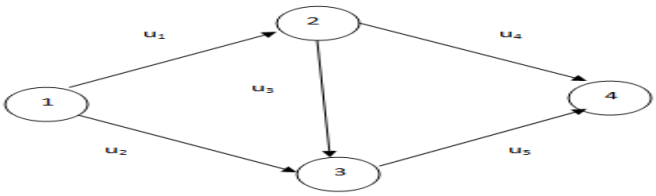



FIGURE 1.4

- *la chemin reliant le sommet* 1 *au sommet* 4 *est* $C = \{u_1, u_4\}$ *et les sommets qui se trouvent sur le chemin sont* $Y = \{1, 2, 5\}$
- *la chemin reliant le sommet* 1 *au sommet* 4 *est* $C = \{u_1, u_3, u_5\}$ *et les sommets appartenant au chemin sont* $Y = \{1, 2, 3, 4\}$

**Définition 1.9** *Une chaine reliant les sommets a et b est une séquence d'arêtes ( ou arcs ),* $(u_1, u_2, ..., u_p)$, *vérifiant :*

*1-* $a$ *est une extrémité de* $u_1$

*2-* $b$ *est adjacent à* $u_p$

*3-* $\forall i = 1, 2, ..., p$ $u_i$ *et* $u_{i-1}$ *sont adjacents.*

**Exemple 1.5** *soit* $G = (X, U)$ *un graphe avec* $X = \{1, 2, 3, 4, 5\}$ *et* $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$

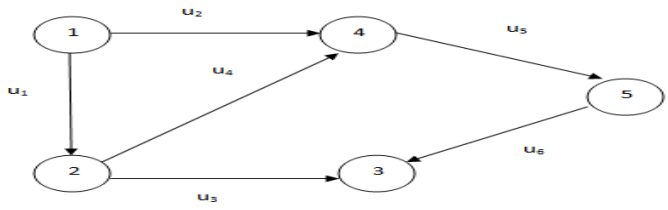



FIGURE 1.5

- *la chaine reliant les sommets* 1 *et* 5 *est* $C = \{u_2, u_5\}$.
- *la chaine reliant les sommets* 1 *et* 5 *est* $C = \{u_1, u_3, u_6\}$.

**Définition 1.10** *Soient* $G = (X, U)$ *un graphe et a un sommet de* X.

G *est dit fortement connexe ( ou f-connexe ) si pour tout couple de sommets x et y de* X *il existe un chemin de x à y.*

*L'ensemble des sommets, x, pour lesquels il existe un chemin de a à x et un chemin x à a est appelé composante f-connexe. un graphe qui n'est pas f-connexe est un graphe qui possède plusieurs composantes connexes.*

*De la même manière, on dira que* G *est connexe si pour touts les couples de sommets,x et y, il existe une chaine qui relie ces sommets.*

*L'ensemble des sommets, x, pour lesquels il existe une chaine reliant a et x est dit composante connexe.*

*Un graphe qui possède plusieurs composantes connexes est dit non connexe.*

**Définition 1.11** *Un cycle est une chaine constituée d'arêtes ( ou arcs ), toutes distinctes, reliant un sommet à lui même.*

**Définition 1.12** *Un circuit est une séquence d'arcs* $C = (u_1, u_2, ..., u_p)$, *tous distincts vérifiant :*

*1-* $\forall i = 2, 3, ..., p : I(u_i) = T(u_{i-1})$

*2-* $I(u_p) = T(u_1)$

**Définition 1.13** *Représentation d'un cycle par un vecteur*

*Soit* $G = (X, U)$ *un graphe et soit* C *un cycle de* G

*Choisissons un sens de parcours de* C *et soient :*

. $C^+ = \{u / u \in C / u$ *orienté dans le sens du parcours de* C$\}$

. $C^- = \{u / u \in C / u$ *orienté dans le sens contraire du sens de parcours de* C$\}$

. $C = C^+ \cup C^-$

*Un vecteur représentatif de* C *est donné par le vecteur ( de* $\mathbb{R}^m$ *)* $\gamma$ *suivant :*

$$\gamma_i = \begin{cases} 1 & si \quad u_i \in C^+ \\ -1 & si \quad u_i \in C^+ \\ 0 & sinon \end{cases}$$

**Exemple 1.6** *soit* $G = (X, U)$ *un graphe avec* $X = \{1, 2, 3, 4, 5, 6\}$ *et* $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$

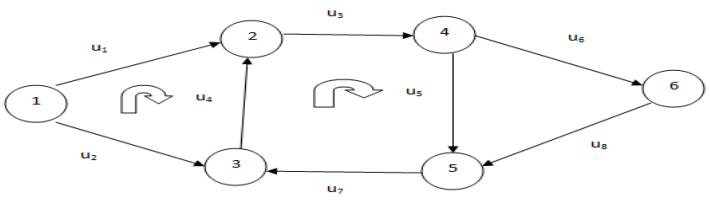



FIGURE 1.6

- $C_1 = \{u_1, u_4, u_2\}$ *est un cycle.*
- $C_2 = \{u_3, u_5, u_7, u_4\}$ *est un circuit.*
- *le vecteur représente* $C_1$ *est :* $\gamma_{C_1} = \begin{pmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}^t$

**Définition 1.14** *Soient* $G = (X, U)$ *un graphe et* $A \subset X$

*On appelle Cocycle associé à* A *le sous-ensemble* $\Omega(A)$ *suivant :*

$\Omega(A) = \{u, u \in U /$ *une extremite de u est dans* A *et l'autre n'est pas dans* A$\}$

*On définit les sous ensembles* $\Omega^+(A)$ *et* $\Omega^-(A)$ *par :*

- $\Omega^+(\mathrm{A}) = \{u, u \in \mathrm{U}/\mathrm{I}(u) \in \mathrm{A} \quad et \quad \mathrm{T}(u) \notin \mathrm{U}\}$
- $\Omega^-(\mathrm{A}) = \{u, u \in \mathrm{U}/\mathrm{T}(u) \in \mathrm{A} \quad et \quad \mathrm{I}(u) \notin \mathrm{U}\}$
- $\Omega(\mathrm{A}) = (\Omega^+(\mathrm{A}), \Omega^-(\mathrm{A}))$ *que l'on confond avec* $\Omega(\mathrm{A}) = \Omega^+(\mathrm{A}) \cup \Omega^-(\mathrm{A})$

*Comme pour les cycles , a tout Cocycle* $\Omega(\mathrm{A})$ *de* G, *On lui associe son vecteur représentatif* $\omega$ *par le vecteur de* $\mathbb{R}^m$ *suivant :*

$$\omega_i = \begin{cases} 1 & si \quad u_i \in \Omega^+(\mathrm{A}) \\ -1 & si \quad u_i \in \Omega^-(\mathrm{A}) \\ 0 & si \quad u_i \notin \Omega(\mathrm{A}) \end{cases}$$

**Remarque 1.1** *Si dans un cocycle on a* $\Omega^+(\mathrm{A})$ *ou* $\Omega^-(\mathrm{A})$ *qui est vide le cocycle est appelé cocircuit.*

**Exemple 1.7** *soient* $\mathrm{G} = (\mathrm{X}, \mathrm{U})$ *un graphe,* $\mathrm{X} = \{1,2,3,4,5,6\}, \mathrm{U} = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ *et* $\mathrm{A} \subset \mathrm{X}$ *avec* $\mathrm{A} = \{2,4\}$

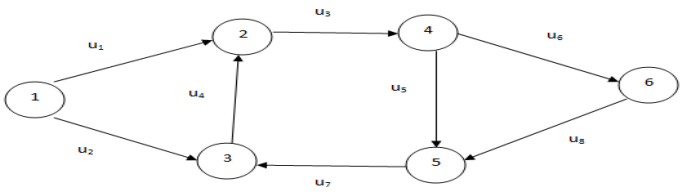



FIGURE 1.7

*Le coycle* $\Omega(\mathrm{A})$ *associe à* A *est* $\Omega(\mathrm{A}) = \{u_1, u_4, u_5, u_6\}; \Omega^+(\mathrm{A}) = \{u_5, u_6\}; \Omega^-(\mathrm{A}) = \{u_1, u_4\}$

**Théorème 1.1**

*Tout vecteur représentatif d'un cycle est orthogonal à tout vecteur représentatif d'un cocycle.*

# Chapitre 2

# Problème du flot et tension

En théorie des graphes, les flots et les tensions jouent un rôle important puisqu'ils modélisent beaucoup de problèmes d'optimisation combinatoires classiques telle que les problèmes de transport.

Nous donnons quelques définitions et propriétés élémentaires sur les flots et tensions.

Dans tout ce qui suit on suppose :

G = (X, U) un graphe, X = $\{x_1, x_2, ..., x_n\}$, U = $\{u_1, u_2, ..., u_m\}$ et $p$ le nombre de composantes connexes de G

## 1 Flot et tension

**Définition 2.1**

- *On appelle,* F*, le sous espace vectoriel des flots, le sous espace vectoriel de* $\mathbb{R}^m$ *construit sur l'ensemble des vecteurs représentatifs de cycles du graphe* G*.*
- *Et* T *le sous espace vectoriel des tensions, le sous espace vectoriel de* $\mathbb{R}^m$ *construit sur l?ensemble des vecteurs représentatifs de cocycles du graphe* G*.*

**Proposition 2.1**

- F = $\{f \in \mathbb{R}^m$ *telle que : f combinaison linéaire, à coefficients réels des vecteurs représentatifs de cycle du graphe* G$\}$
- T = $\{t \in \mathbb{R}^m$ *telle que : t combinaison linéaire, à coefficients réels des vecteurs représentatifs de cocycle du graphe* G$\}$

*Les dimensions de* F *et de* T *sont appelées, respectivement, le nombre cyclomatique et le nombre cocyclomatique et leurs vecteurs sont appelés, respectivement, flots et tensions.*

**Théorème 2.1**

*Les sous espaces* F *et* T *sont orthogonaux et on a :*

- $\mathbb{R}^m = F \oplus T$
- $dim(F) = m - n + p \quad ; \quad dim(T) = n - p$

**Exemple 2.1** *Soit* G *le graphe suivant :*

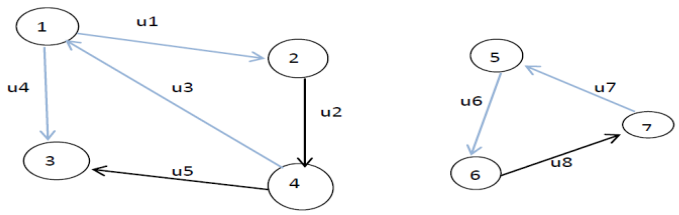



FIGURE 2.1

*Pour l'exemple çi-dessus, cherchons une base de* F *et de* T. *soit* F = (X, V) *un graphe partiel de* G *qui est une forêt ( les arcs de* V *sont représentés en bleu ). faisons la convention que chaque fois qu'on ajoute un arc, u, à la forêt, on choisira comme sens de parcours du cycle, contenant u,crée le même sens que u.*

- *Si on ajoute* $u_5$ *à* F *on crée le cycle* $\Gamma_{u_5} = (u_5, u_4, u_3)$ *et son vecteur représentatif* $\gamma_{u_5} = (0,0,-1,-1,1,0,0,0)^t$
- *Si on ajoute* $u_2$ *à* F *on crée le cycle* $\Gamma_{u_2} = (u_2, u_3, u_1)$ *et son vecteur représentatif* $\gamma_{u_2} = (1,1,1,0,0,0,0,0)^t$
- *Si on ajoute* $u_8$ *à* F *on crée le cycle* $\Gamma_{u_8} = (u_8, u_7, u_6)$ *et son vecteur représentatif* $\gamma_{u_8} = (0,0,0,0,0,1,1,1)^t$

*Donc* $\{\gamma_{u_5}, \gamma_{u_2}, \gamma_{u_8}\}$ *est une base de* F .
*Pour déterminer une base de* T, *on considère le graphe partiel* FC = (X, U − V) *(* U − V *est constitué d'arcs en noirs) et quand on ajoute un arc* $u \in$ U − V *on crée un cocyle,* $\Omega$, *de* G *dans le graphe* $(X, (U-V) \cup \{u\})$ *(résultat connu en théorie des graphes).*

*Par exemple si on rajoute* $u_1$ *à* FC *alors* $\Omega(\{2\}) = (u_1, u_2)$ *est un cocycle de* G *qui est inclus dans (* U − V *pour avoir le premier vecteur ,* $t_1 = (-1,1,0,0,0,0,0,0)^t$. *En procédant ainsi on trouve* $t_2 = (0,-1,1,0,1,0,0,0)^t$, $t_3 = (0,0,0,-1,-1,0,0,0)^t$, $t_4 = (0,0,0,0,0,-1,0,1)^t$ *et* $t_5 = (0,0,0,0,0,0,1,-1)^t$.

**Théorème 2.2**
*On s'intéresse aux cocycles de la forme* $\Omega(\{x\})$ *pour un sommet x quelconque de* X.
*Notons par* $\omega^x$ *le vecteur représentatif d'un cocycle* $\Omega(\{x\})$ *et soit* Y *un sous ensemble de* X, $\Omega$(Y) *le cocycle associé à* Y *et* $\omega$ *son vecteur représentatif alors :*

$$\omega = \sum_{x \in Y} \omega^x$$

**Théorème 2.3**
*Une condition nécessaire et suffisante pour qu'un vecteur* $f \in \mathbb{R}^m$ *soit un flot sur sur le graphe* G *est que :*

$$\sum_{u \in \Omega^+(\{x\})} f(u) - \sum_{u \in \Omega^-(\{x\})} f(u) = 0 \quad \forall x \in X \tag{2.1}$$

**Théorème 2.4** *Les conditions suivantes sont équivalentes et caractérisent une tension t :*

1− *Pour tout cycle élémentaire* $\Gamma$ *et pour tout sens de parcours sur ce cycle on a :*

$$\sum_{u \in \Gamma^+(\{x\})} t(u) - \sum_{u \in \Gamma^-(\{x\})} t(u) = 0 \tag{2.2}$$

2− *Il existe une fonction potentiel* $\Pi : X \longrightarrow \mathbb{R}$ *telle que pour tout arc* $u = (x, y)$ *on ait :*

$$t(u) = \Pi(y) - \Pi(x)$$

# Chapitre 3

# Problème du flot maximum

Le problème du flot maximum consiste à trouver la quantité maximum de flot qui peut circuler d'une source $s$ à un puits $p$ dans un réseau R.

Quelque fois le problème répond simplement à la question de trouver la valeur de ce flot.

Le problème du flot maximum peut être vu comme un cas particulier de plusieurs autres problèmes de flots dans les réseaux.

## 1 Position du problème

**Définition 3.1** *Soit un graphe* $G = (X, U)$ *dans lequel un arc* ***un arc spécial*** $u_r = (s, p)$ *est distingué appelé arc de Retour*
*s : source ( ou racine de* G *)*
*p : puits (ou Anti racine de* G *)*

*On suppose donné une application :*

$$c : U \to \mathbb{R}^+ \cup \{+\infty\}$$

*qui à chaque arc u, on fait correspondre sa capacité* $c(u)$ *. le problème du flot maximum de s à p dans le reséau* $R = (X, U, c)$ *consiste à trouver un vecteur* $f \in \mathbb{R}^n$

*(i)* $f$ *flot sur* $[X, U]$

*(ii) On ait*

$$0 \leq f(u) \leq c(u) \quad \forall u \in U \tag{3.1}$$

*(iii)* $f(u_r)$ *soit maximum sous les conditions précédentes*

**Remarque 3.1**

*si* $f$ *vérifie (i) et (ii) on dira que* $f$ *est réalisable*

**Remarque 3.2**

*Le problème du flot maximum de s à p dans réseau* R *peut se formuler comme un programme linéaire à variables bornées*

$$(F_m) \begin{cases} Af = 0 \\ f(u_r) = z(Max) \quad 0 \leq f \leq c \end{cases}$$

*ou* $A \in \mathbb{R}^{m \times n}$ *matrice d'incidence aux arcs du graphe* $(X, U)$ *( comme précédemment , on a* $n = |X|, m = |U|$*); la* $i^{eme}$ *contrainte* $A_i f = 0$ *exprime la conservation du flot au sommet* $x_i$ *.*

# 2 Algorithme de résolution

Soit G = (X, U) un graphe orienté, on définit une application C : U $\longrightarrow \mathbb{R}^+ \cup \{+\infty\}$
à tout arc $u$ , on lui associe sa capacité $c(u)$
On cherche un flot maximum $f$ de la source $s$ au le puits $p$ , sous contraintes du respect de capacité
Il y a plusieurs algorithmes de résolution du problème :

- Algorithme de Ford et Fulkerson dont le principe consiste, à partir d'un flot initial ( en générale nul ),de trouver un chemin ( ou une chaine ) le long duquel on peut augmenter la valeur du flot sur le réseau.
- Algorithme de d'Edmonds et Karp n'est qu'une version améliorée de l'algorithme de Ford et Fulkerson, il consiste à choisir le chemin ( ou la chaine ) augmentant d'une maniéré particulier. Cela consiste, au préalable, a résoudre un problème de recherche d'un plus court chemin ( ou chaine ) de $s$ à $p$ où les poids des arcs sont supposés égaux à l'unité.

# 3 L'algorithme de Ford et Fulkerson

## 3.1 Principe de l'algorithme

L'algorithme de Ford et Fulkerson qui est de complexité en $O(mf(u_r))$ [4] ( ou $f(u_r)$ est la valeur du flot ) est constitué de deux procédures :

- la procédure de marquage, qui consiste à trouver un chemin de $s$ à $p$ ou une chaine reliant $s$ à $p$, ou la valeur du flot peut être augmentée, de $\delta$, le long du cycle, $\Gamma$, constitué du chemin ( ou chaine ) auquel l'arc de retour $u_r$ est adjoint.
  Notons par $\Gamma^+$ l'ensemble des arcs de $\Gamma$ qui sont orientés dans le même sens de parcours que ce dernier ( c'est à dire de $s$ à $p$ ), et $\Gamma^-$ l'ensemble des arcs de $\Gamma$ qui sont orientés dans le sens contraire du sens de parcourt de $\Gamma$.

$$\begin{aligned} \delta &= \min(\delta_1, \delta_2) \\ \text{où} \quad \delta_1 &= \min_{u \in \Gamma^-}(f(u)) \\ \text{et} \quad \delta_2 &= \min_{u \in \Gamma^+}(c(u) - f(u)) \end{aligned}$$

- la procédure de changement du flot qui consiste, une fois la valeur de $\delta$ et le chemin ( ou la chaine ) augmentant(e) sont déterminés par la procédure de marquage, à changer le flot le long du dit chemin ( chaine ) et à augmenter la valeur du flot de $\delta$.

$$f(u) \longleftarrow \begin{cases} f(u) + \delta & \text{si} \quad u \in \Gamma^+ \\ f(u) - \delta & \text{si} \quad u \in \Gamma^- \end{cases}$$

çi-dessous, nous donnons de maniéré formulle l'algorithme de Ford et Fulkerson [3] :

**Algorithme :** Marquage
**Donner :** réseau R = (X, U, $c$) ; arc $u_r$ ; flot $f$
**résultats :** scalaire $\delta$ , arborescence A ; ensemble des sommet Y
/ * $f$ est supposé réalisable sur R * /
$\delta := \delta(s) := c(u_r) - f(u_r)$ ; Y := $\{s\}$
**tant que** $p \notin$ Y et $\delta > 0$ **faire**
**si** il existe $u = (xy)$ avec $x \in$ Y, $y \notin$ Y et $f(u) < c(u)$
Y := Y $\cup \{y\}$ ; A($y$) := $u$ ; $\delta(y) := \min[\delta(x); c(u) - f(u)]$
**sinon si** il existe $u = (yx)$ avec $x \in$ Y, $y \notin$ Y et $f(u) > 0$ **alors**
Y := Y $\cup \{y\}$ ; A($y$) := $u$ ; $\delta(y) := \min[\delta(x); f(u)]$
**sinon** $\delta := 0$ /* aucun nouvel arc ne peut être marque */
**fin si**
**fin si**
**fin tant que**
**si** $p \in$ Y **alors** $\delta := \delta(p)$
**fin algorithme**

**Algorithme :** Changement Flot
**Donner :** réseau R = (X, U, $c$) ; arc $u_r$ ; scalaire $\delta$ , arborescence A
**Donnée modifiées :** flot $f$
/* cette procédure n'est appelée que si $\delta > 0$ , i.e. si $p$ a été marqué */
$x := p$ ; $f(u_r) := f(u_r) + \delta$ ;
**tant que** $x \neq s$ **faire** $u :=$ A($x$)
**si** $x =$ T($u$) **alors** $f(u) := f(u) + \delta$ ; $x :=$ I($u$)
**sinon** $f(u) := f(u) - \delta$ ; $x :=$ T($u$)
**fin si**
**fin tant que**
**fin algorithme**

**Algorithme :** Flot Max
**Donner :** réseau R = (X, U, $c$) ; arc $u_r$
**résultats :** flot $f$ ; ensemble se sommets Y
/* $p$ et $s$ sont les extrémités de l'arc $u_r$ */
**pour tout** $u \in$ U **faire** $f(u) := 0$ **fin pour tout**
**itérer**
MARQUAGE(R; $u_r$; $f$; $\delta$; A; Y)
**arrêt** $\delta := 0$
CHANGEMENT FLOT(R; $u_r$; $f$; $\delta$; A)
**fin itérer**
**fin algorithme**

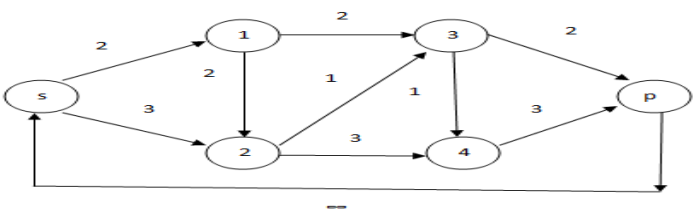



FIGURE 3.1 – exemple sur algorithme de Ford et Fulkerson

**Exemple 3.1** ***itération 1 :***

| algorithme de marquage | | | changement de flot |
|---|---|---|---|
| Y | A | δ | flot |
| $\{s\}$ | $(p,s)$ | $\infty$ | 1 |
| $\{s,1\}$ | $(s,1)$ | 2 | 1 |
| $\{s,1,2\}$ | $(1,2)$ | 2 | 1 |
| $\{s,1,2,3\}$ | $(2,3)$ | 1 | 1 |
| $\{s,1,2,3,4\}$ | $(3,4)$ | 1 | 1 |
| $\{s,1,2,3,4,p\}$ | $(4,p)$ | 1 | 1 |

***itération 2 :***

| algorithme de marquage | | | changement de flot |
|---|---|---|---|
| Y | A | δ | flot |
| $\{s\}$ | $(p,s)$ | $\infty$ | 2 |
| $\{s,1\}$ | $(s,1)$ | 1 | 2 |
| $\{s,1,2\}$ | $(1,2)$ | 1 | 2 |
| $\{s,1,2,4\}$ | $(2,4)$ | 1 | 1 |
| $\{s,1,2,4,p\}$ | $(4,p)$ | 1 | 2 |

***itération 3 :***

| algorithme de marquage | | | changement de flot |
|---|---|---|---|
| Y | A | δ | flot |
| $\{s\}$ | $(p,s)$ | $\infty$ | 3 |
| $\{s,2\}$ | $(s,2)$ | 3 | 1 |
| $\{s,2,4\}$ | $(2,4)$ | 2 | 2 |
| $\{s,1,2,4,p\}$ | $(4,p)$ | 1 | 3 |

***itération 4 :***

| algorithme de marquage | | | changement de flot |
|---|---|---|---|
| Y | A | δ | flot |
| $\{s\}$ | $(p,s)$ | $\infty$ | 4 |
| $\{s,2\}$ | $(s,2)$ | 2 | 2 |
| $\{s,2,4\}$ | $(2,4)$ | 1 | 3 |
| $\{s,2,4,3\}$ | $(3,4)$ | 1 | 0 |
| $\{s,2,4,3,p\}$ | $(3,p)$ | 1 | 1 |

***itération 5 :***

| algorithme de marquage | | | changement de flot |
|---|---|---|---|
| Y | A | δ | flot |
| $\{s\}$ | $(p,s)$ | $\infty$ | 5 |
| $\{s,2\}$ | $(s,2)$ | 1 | 3 |
| $\{s,2,1\}$ | $(1,2)$ | 1 | 1 |
| $\{s,2,1,3\}$ | $(1,3)$ | 1 | 1 |
| $\{s,2,1,3,4\}$ | $(3,4)$ | 1 | 1 |
| $\{s,2,1,3,4,p\}$ | $(3,p)$ | 1 | 2 |

***itération 6 :***

$f\,max = 5$ *temps d'éxcution est :* $0.0857\,s$

# 4 L'algorithme d'Edmonds-krap

## 4.1 Principe de l'algorithme

La procédure d'Edmonds et karp est en O($mn^2$) [4]. Elle différé de celle de Ford et Fulkerson dans le choix du chemin ( chaine ) augmentant(e). Avant d'appliquer l'algorithme par une procédure de recherche en largeur, On déterminé un plus court chemin ( ou chaine ) de $s$ à $p$, où les poids des arcs sont supposés égaux à l'unité. On détermine la valeur de δ et on applique la procédure de changement de flot.

L'algorithme d'Edmond-Karp se résume à[2] :

- initialiser le flot $f$ avec la valeur 0 sur chaque arc .
- tant qu'il existe un chemin augmentant de $s$ à $p$, augmenter le flot de δ jusqu'à l'obtention d'un flot complet.
- tant qu'il existe une chaine augmentant de $s$ à $p$, augmenter le flot de δ jusqu'à l'obtention d'un flot maximal
- retourner $f$

Pour trouver un plus court chemin, nous utilisons l'algorithme de marquage ( modifié )

**Exemple 3.2** *Soit le graphe* $G = (X, U, s, p, c), X = \{A, B, C, D, E\}, U = \{u_i\}\, i = \overline{1,11}, u_r$

| U | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_r$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I($u$) | $s$ | $s$ | A | B | B | B | C | C | D | D | E | $p$ |
| T($u$) | A | C | B | $s$ | C | D | D | E | A | $p$ | $p$ | $s$ |
| C($u$) | 3 | 3 | 4 | 3 | 1 | 2 | 2 | 6 | 1 | 1 | 9 | ∞ |

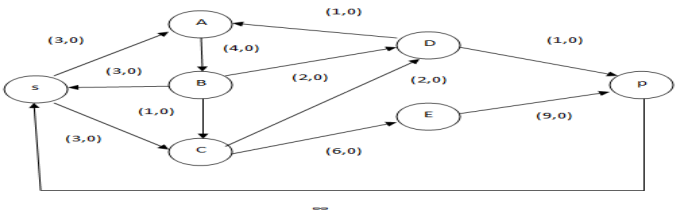



FIGURE 3.2

***itération 1 :***

$$
\begin{aligned}
S &= \{s, C, E, p\} \\
C &= \{u_2, u_8, u_{10}\} \\
\delta &= \min_{u \in U}\{c(u_2) - f(u_2), c(u_8) - f(u_8), c(u_{11}) - f(u_{11})\} \\
&= \min_{u \in U}\{3-0, 6-0, 9-0\} \\
&= \min_{u \in U}\{3, 6, 9\} \\
&= 3
\end{aligned}
$$

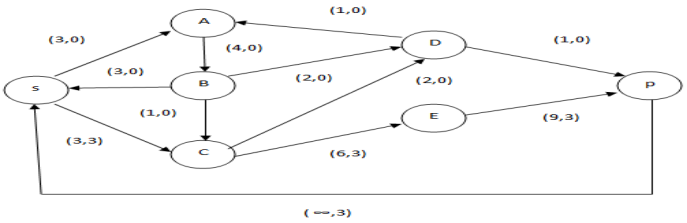



FIGURE 3.3 – ( longueur 4 )

***itération 2 :***

$$
\begin{aligned}
&S = \{s, A, D, D, p\}\\
&C = \{u_1, u_3, u_6, u_{10}\}\\
&\delta = \min_{u\in U}\{c(u_1) - f(u_1), c(u_3) - f(u_3), c(u_6) - f(u_6), c(u_{10}) - f(u_{10})\}\\
&\quad = \min_{u\in U}\{3-0, 4-0, 2-0, 1-0\}\\
&\quad = \min_{u\in U}\{3, 4, 2, 1\}\\
&\quad = 1
\end{aligned}
$$

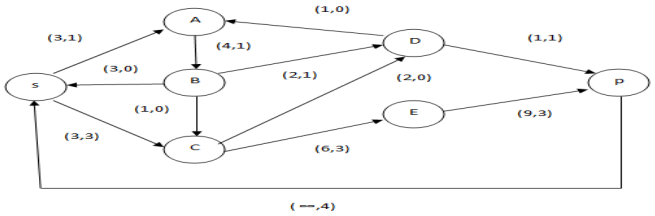



FIGURE 3.4 – ( longueur 5 )

***itération 3 :***

$$
\begin{aligned}
&S = \{s, A, B, D, C, E, p\}\\
&C = \{u_1, u_3, u_6, u_5, u_8, u_{11}\}\\
&\delta = \min_{u\in U}\{c(u_1) - f(u_1), c(u_3) - f(u_3), c(u_6) - f(u_6), c(u_5) - f(u_5), c(u_8) - f(u_8), c(u_{11}) - f(u_{11})\}\\
&\quad = \min_{u\in U}\{3-1, 4-1, 2-1, 1-0, 6-3, 9-3\}\\
&\quad = \min_{u\in U}\{2, 3, 1, 1, 3, 6\}\\
&\quad = 1
\end{aligned}
$$

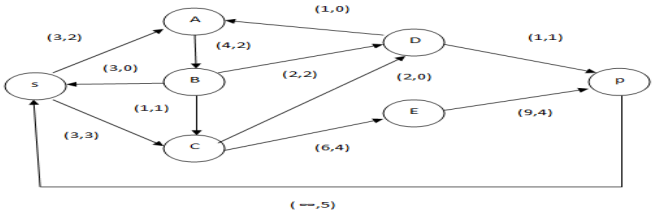



FIGURE 3.5 – ( longueur 7 )

**Remarque 3.3** *Si les capacités $c(u)$, des arcs sont entiers et si le flot de départ est nul alors, toutes les améliorations consisteront à ajouter des entiers strictement positifs. On en conclut que :*

- *La suite des valeurs des flots obtenus par l'algorithme d'Edmonds-krap est une suite strictement croissante majorée donc convergente.*
- *D'où l'existence d'un flot maximum.*

# Chapitre 4

# Le théorème de la coupe minimum et heuristique de recherche du flot maximum

Dans ce chapitre nous donnons une condition nécessaire et suffisante d'existence d'un flot maximum dans un réseau. Le résultat est donné sous forme de théorème de la coupe minimum.

En utilisant ce théorème ( qu'on citera çi-dessous ) nous avons pu déduire une méthode approchée qui nous donne une très bonne borne supérieure du problème de flot maximum.

## 1 Théorème de coupe minimum

**Définition 4.1**

*Soit* $\mathrm{R} = (\mathrm{X}, \mathrm{U}, c)$ *un réseau et soient s une source et p un puits de* R. *Notons par* $u_r = (p, s)$ *l'arc de retour.*

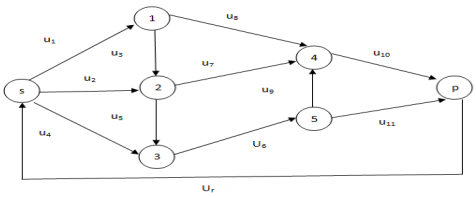
$\mathscr{C}$ *est dit coupe séparant p de s s'il existe* $\mathrm{Y} \subset \mathrm{X}$ *avec* $s \in \mathrm{Y}$, $p \notin \mathrm{Y}$ *et* $\mathscr{C} = \Omega^+(\mathrm{Y})$.

*ou :*

$$\mathscr{C} = \{u \in \mathrm{U} / \mathrm{I}(u) \in \mathrm{Y}, \mathrm{T}(u) \notin \mathrm{Y}\} = \Omega^+(\mathrm{Y})$$

**Exemple 4.1**

$$\mathscr{C}_1 = \{u_8, u_3, u_2, u_4\}$$
$$\mathrm{Y}_1 = \{s, 1\}$$
$$\mathscr{C}_2 = \{u_{10}, u_{11}\}$$
$$\mathrm{Y}_1 = \{s, 1, 2, 3, 4, 5\}$$




Une coupe $\mathscr{C}$ séparant $p$ de $s$ intersecte tout chemin C de $s$ á $p$ dans R .
Un ensemble d'arcs minimal pour cette propriété est une coupe.

**Définition 4.2** *On définit la Capacité de* $\mathscr{C}$ *par :*

$$c(\mathscr{C}) = \sum_{u \in \mathscr{C}} c(u)$$

**Exemple 4.2** *en utilisant ( l'exemple 1 )*

| U | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_r$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C($u$) | 2 | 1 | 1 | 1 | 3 | 2 | 2 | 1 | 1 | 5 | 1 | $\infty$ |

$$c(\mathscr{C}_1) = 1+1+1+1 = 4; c(\mathscr{C}_2) = 5+1 = 6$$

**Théorème 4.1**

*Pour tout flot réalisable $f$ sur* R = (X, U, $c$) *et pour tout coupe $\mathscr{C}$ séparant $p$ de $s$ ( avec $c(u_r = c(p,s) = \infty$) ) on a :*

$$f(u_r) \leq c(\mathscr{C})$$

soit $f$ un flot sur R = (X, U, $c$) et soit $\mathscr{C}$ une coupe séparant $p$ de $s$
On suppose $\mathscr{C} = \Omega^+(Y)$, $Y \cup X$ avec $s \in Y$ et $p \notin Y$
$\sum_{u\in\Omega^-(Y)} f(u) - \sum_{u\in\Omega^+(Y)} f(u) = 0$ ( loi de kirchhoff )
On remarque que $u_r \in \Omega^-(Y)$

$$\Rightarrow f(u_r) \leq \sum_{u\in\Omega^-(Y)} f(u) = \sum_{u\in\Omega^+(Y)} f(u) \leq \sum_{u\in\Omega^+(Y)} c(u)$$

$$\forall u \Rightarrow 0 \leq f(u) \leq c(u)$$

$$\Rightarrow f(u_r) \leq c(\mathscr{C})$$

## 1.1 Justification

Si les capacité sont des multiples entiers d'un nombre $\lambda$ et s'il existe une coupe séparant $p$ de $s$ de capacité finie. L'algorithme de Ford et Fulkerson donne " en un nombre fini d'itérations " une solution optimale du problème du flot maximum de $s$ à $p$ sur R. Toutes les composantes du flot obtenu sont des multiples entiers de $\lambda$ .

On a en fait montré que s'il existe une coupe de capacité finie et si les capacités des arcs sont entières ( ou rationnelles, on prend alors $\lambda = 1/D - D$ étant le plus petit commun dénominateur des valeurs des capacités) alors l'algorithme donne " en un nombre fini d'itérations " une solution entière ( ou rationnelle) mais on peut construire des réseaux dont les capacité des arcs sont irrationnelles et tel que l'application " maladroit " de l'algorithme de Ford et Fulkerson conduise à un nombre infini d'itérations ( alors qu'il existe une coupe de capacité finie). De plus $f(u_r)$ peut converge vers une valeur strictement inférieure à celle de la capacité d'un coupe de capacité minimum. On notera que cette question a un intérêt essentiellement académique car d'une part les rationnels étant denses dans le corps des réels on peut toujours supposer qu'avec la précision souhaitée les données sont rationnelles[3].

En pratique le problème ne se pose pas puisque tout irrationnel est approché par un rationnel ( en vertu de la densité de $\mathbb{Q}$ dans $\mathbb{R}$ ).

Les calculs se font sur machine et cette dernier " travaille " uniquement. Lorsque l'algorithme de Ford et Fulkerson s'arrête avec un flot $f$ sans que l'on ait pu marquer $p$ ( dans l'hypothèse ou $c(u_r) = \infty$ ) , $f$ est solution optimale du problème du flot maximum de $s$ á $p$ dans R
Soient Y l'ensemble des sommets marqués et $p \in Y$, on a les implications suivantes

- si $u \in \Omega^+(Y) \Rightarrow f(u) = c(u)$, sinon le processus de marquage direct aurait permis d'inclure $y = T(u)$ dans Y

- si $u \in \Omega^{-}(\mathrm{Y}) \Rightarrow f(u) = 0$, sinon le processus de marquage inverse aurait permis d'inclure $y = \mathrm{I}(u)$ dans Y

**Remarque 4.1**

*Pour résoudre le problème du flot maximum de s à p dans un réseau* R*, il suffit de déterminer toutes les coupes séparant p de s, de calculer leurs capacités et d'en choisir celle qui est de capacité minimum. Cette capacité représente la valeur du flot (* $\max f(u_r)$ *).*

*Le nombre de coupes étant égale à* $2^{n-2}$*. Pour n assez grand,* $2^{n-1}$ *devient astronomique d'où l'impossibilité d'appliquer cette méthode naïve.*

*Au lieu d'appliquer cette méthode dans son intégralité nous avons pensé à une application partielle où à défaut d'obtenir une solution optimale. On obtient une bonne borne supérieure de problème.*

# 2 Heuristique de recherche d'une borne supérieure

Le principe de la méthode consiste à générer une série de coupes séparant $p$ de $s$. on retiendra la coupe qui soit de capacité minimum. Soit de manière formelle l'algorithme :

**données :** R = (X, U, $c$)
**sorties :** flot max " $fmax$ "
**pour :** ( tout $k$ variant de 1 à $n-1$) **faire**
  $fmin \longleftarrow 0$
  **pour** ($j$ = 1 jusqu'à $m$ ) **faire**
    **si** ( $a(j,1) \leq k$ et $a(j,2) > k$ )
      $fmin \longleftarrow fmin + a(j,3)$
    **fin si**
  **fin pour**
  **si** ($fmax > fmin$)
    $fmax \longleftarrow fmin$
  **fin si**
 **fin pour**
**fin algorithme**

# Chapitre 5

# Résultats numériques

Nous avons programmé, en langage Matlab, l'algorithme de Ford et Fulkerson, celui d'Edmonds et Karp ainsi que la procédure approchée que nous proposons.

Nous avons travaillé sur un micro-ordinateur de marque " acer " doté d'un processeur Intel (R) Core (TM) i3 et 4 giga de RAM.

Bien que la complexité des procédures soit connue (cité plus haut), nous avons surtout fait une étude comparative des trois méthodes pour pouvoir estimer la qualité des solutions fournies par la procédure approchée que nous proposons.

Pour cela nous avons utilisé les instances de OR-Library (donnés par Beasley) et comparé les trois procédures sur les quarante exemples . les résultats sont résumés sur les neuf tableaux ci-dessous :

| | G=(X,U), n = 100 /* sommet */ et m=200 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed1 | 7 | 0.1467 | 4 | 7 | 0.1081 | 4 | 7 | $3.6677e^{-004}$ | / |
| pmed2 | 7 | 0.3993 | 4 | 7 | 0.3212 | 3 | 7 | $3.6818e^{-004}$ | / |
| pmed3 | 39 | 1.0479 | 13 | 39 | 0.5048 | 6 | 39 | $3.6861e^{-004}$ | / |
| pmed4 | 44 | 1.7200 | 19 | 44 | 0.6087 | 11 | 44 | $3.6006e^{-004}$ | / |
| pmed5 | 71 | 1.7758 | 19 | 71 | 1.9379 | 24 | 71 | $3.7716e^{-004}$ | / |

TABLEAU 5.1 – tableau 1

| | G=(X,U), n = 200 /* sommet */ et m=800 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du Ford et Fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed6 | 68 | 26.9931 | 53 | 68 | 1.0962 | 2 | 68 | 0.0031 | / |
| pmed7 | 38 | 17.4618 | 32 | 38 | 1.1878 | 4 | 38 | 0.0033 | / |
| pmed8 | 68 | 28.1400 | 34 | 68 | 2.5168 | 14 | 68 | 0.0034 | / |
| pmed9 | 56 | 19.8141 | 32 | 56 | 9.1277 | 19 | 56 | 0.0032 | / |
| pmed10 | 35 | 13.8592 | 25 | 35 | 4.0740 | 10 | 35 | 0.0034 | / |

TABLEAU 5.2 – tableau 2

| | G=(X,U), n = 300 /* sommet */ et m=1800 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed11 | 59 | 115.0521 | 57 | 59 | 11.9712 | 14 | 59 | 0.0103 | / |
| pmed12 | 72 | 129.8681 | 64 | 72 | 5.5133 | 12 | 72 | 0.0100 | / |
| pmed13 | 13 | 18.5653 | 10 | 13 | 3.2841 | 3 | 13 | 0.0102 | / |
| pmed14 | 13 | 23.3977 | 12 | 13 | 2.9359 | 2 | 13 | 0.0101 | / |
| pmed15 | 98 | 183.9160 | 94 | 98 | 44.6711 | 48 | 98 | 0.0103 | / |

TABLEAU 5.3 – tableau 3

| | G=(X,U), n = 400 /* sommet */ et m=3200 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed16 | 30 | 141.4080 | 29 | 30 | 8.1600 | 5 | 30 | 0.0257 | / |
| pmed17 | 84 | 459.2211 | 85 | 84 | 72.8216 | 41 | 84 | 0.0250 | / |
| pmed18 | 40 | 190.6697 | 39 | 40 | 7.7483 | 5 | 40 | 0.0250 | / |
| pmed19 | 28 | 140.1627 | 28 | 28 | 23.5819 | 14 | 28 | 0.0249 | / |
| pmed20 | 25 | 124.9166 | 25 | 25 | 6.7045 | 2 | 27 | 0.0255 | / |

TABLEAU 5.4 – tableau 4

| | G=(X,U), n = 500 /* sommet */ et m=5000 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed21 | 53 | 579.5491 | 54 | 53 | 34.5804 | 13 | 53 | 0.0510 | / |
| pmed22 | 5 | 64.2992 | 6 | 5 | 14.1309 | 2 | 5 | 0.0510 | / |
| pmed23 | 6 | 73.8917 | 7 | 6 | 14.8597 | 2 | 6 | 0.0522 | / |
| pmed24 | 30 | 323.5305 | 31 | 30 | 20.6007 | 7 | 30 | 0.0512 | / |
| pmed25 | 39 | 402.4948 | 40 | 39 | 51.4829 | 14 | 39 | 0.0511 | / |

TABLEAU 5.5 – tableau 5

| | G=(X,U), n = 600 /* sommet */ et m=7200 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed26 | 100 | $1.9881e+003$ | 101 | 100 | 52.4320 | 20 | 100 | 0.0916 | / |
| pmed27 | 33 | 651.6462 | 33 | 33 | 29.1758 | 6 | 33 | 0.0969 | / |
| pmed28 | 88 | $1.7683e+003$ | 89 | 88 | 66.0113 | 23 | 88 | 0.0956 | / |
| pmed29 | 49 | 979.8778 | 49 | 49 | 73.4059 | 18 | 49 | 0.0932 | / |
| pmed30 | 81 | $1.6490e+003$ | 82 | 81 | 392.1677 | 49 | 81 | 0.0921 | / |

TABLEAU 5.6 – tableau 6

| | G=(X,U), n = 700 /* sommet */ et m=9800 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed31 | 3 | 153.5898 | 4 | 3 | 46.9864 | 3 | 3 | 0.1512 | / |
| pmed32 | 72 | $2.7418e^{+003}$ | 73 | 72 | 99.7571 | 18 | 72 | 0.1499 | / |
| pmed33 | 40 | $1.5139e^{+003}$ | 41 | 40 | 76.6884 | 12 | 40 | 0.1505 | / |
| pmed34 | 45 | $1.7294e^{+003}$ | 64 | 45 | 54.1197 | 7 | 45 | 0.1522 | / |

TABLEAU 5.7 – tableau 7

| | G=(X,U), n = 800 /* sommet */ et m=12800 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed35 | 70 | $5.2075e^{+003}$ | 71 | 70 | 92.9955 | 9 | 70 | 0.2288 | / |
| pmed36 | 50 | $3.7658e^{+003}$ | 51 | 50 | 108.1145 | 11 | 50 | 0.2307 | / |
| pmed37 | 8 | 669.5679 | 9 | 8 | 84.1607 | 4 | 8 | 0.2329 | / |

TABLEAU 5.8 – tableau 8

| | G=(X,U), n = 900 /* sommet */ et m=16200 /* arc */ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | méthode du ford et fulkerson | | | méthode d'Edmonds-krap | | | méthode coupe minimum | | |
| | flot max | temps | N iter | flot max | temps | N iter | flot max | temps | N iter |
| pmed38 | 40 | $4.7459e^{+003}$ | 41 | 40 | 232.0030 | 12 | 40 | 0.3300 | / |
| pmed39 | 87 | $9.9870e^{+003}$ | 88 | 87 | 683.6027 | 34 | 87 | 0.3345 | / |
| pmed40 | 7 | 927.1102 | 8 | 7 | 109.1357 | 2 | 7 | 0.3273 | / |

TABLEAU 5.9 – tableau 9

La première constatation que l'on peut faire, à la lumière des ces résultats, est l'extrême efficacité de la procédure que nous proposons : nous obtenons 100% solutions exactes. Et les temps CPU de notre procédure sont $10^3$ fois meilleurs.

Les résultats numériques confirment aussi l'amélioration sensible de la méthode d'Edmonds et Karp par rapport à celle de Ford et Fulkerson.

# 1 conclusion

Le problème de la recherche du flot maximum dans un réseau est un problème d'optimisation combination par excellence puisqu'il possède énormément d'applications pratiques et englobe en son sein plusieurs autres problèmes d'optimisation tels que ceux du transport.

Nous avons programmé l'algorithme de Ford et Fulkerson non sans beaucoup de difficultés (ce qui a pris la majeure partie de notre temps) et celui d'Edmonds et Karp.

Nous avons proposé une méthode approchée que nous pensons facilement améliorer

# Bibliographie


[1] A. Bretto, A. Faisant, and F. Hennecart. *Éléments de théorie des graphes.* Springer, 2012. 5

[2] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2) :248–264, 1972. 4, 17

[3] F. Lestringant. L'orientalisme dévoilé : Musset, lecteur de hugo. *Revue d'histoire littéraire de la France*, 102(4) :563–578, 2002. 15, 21

[4] C. MacRury. *Approximation Algorithms for Network Flow and Minimum Cut Problems.* McGill University (Canada), 2019. 14, 17