COURS DE THEORIE DES GRAPHES ET APPLICATIONS

Prof Adama COULIBALY [1]

1er juin 2020

1. Université Félix Houphouët-Boigny, UFR de Mathématiques et Informatique ; 22 BP 582 Abidjan 22

Table des matières

Chapitre 1

Généralités sur les graphes

1.1 Graphes orientés

1.1.1 Définitions

Définition 1.1.1 *Un graphe orienté G est défini par la donnée d'un ensemble fini X dont les éléments sont appelés des sommets ou noeuds et d'un ensemble U dont les éléments sont des couples d'éléments de X appelés arcs. On le note $G = (X, U)$.*

Définition 1.1.2 *Etant donné u graphe orienté $G = (X, U)$. Le nombre n de sommets de G est appelé l'ordre de G et m celui des arcs de G est appelé la taille de G.*

Définition 1.1.3 *Etant donné $G = (X, U)$ un graphe orienté, si $u = (x, y)$ est un arc, on dit que les sommets x et y sont les extémités de l'arc u. Plus précisément, x est l'origine et y l'extrémité de u. On dit aussi que x est l'extrémité initiale et y l'extrémité finale ou terminale de u.*

Définition 1.1.4 *Dans un graphe orienté, un arc dont les deux extrémités coïncident est appelé une boucle.*

Définition 1.1.5 *Etant donné un graphe orienté, deux arcs u et v sont parallèles si leurs extrémités initiales ainsi que leurs extrémités terminales coïncident.*

Dans tout ce qui suit le terme "graphe" concerne uniquement les graphes sans arcs parallèles

Graphiquement, les sommets sont représentés par des points ou des cercles et un arc $u = (x, y)$ est représenté par une flèche joignant les sommets x et y, la flèche étant pointée sur y. On parle de représentation sagittale du graphe.

Définition 1.1.6 *Etant donné $G = (X, U)$ un graphe orienté. On dit que deux sommets x et y sont adjacents ou voisins s'ils sont reliés par un arc.*

Définition 1.1.7 *Un sommet qui n'est adjacent à aucun sommet est dit isolé.*

Un graphe dont les sommets sont isolés est dit discret.

Définition 1.1.8 *Etant donné $G = (X, U)$ un graphe orienté, si $u = (x, y)$ est un arc, on dit que l'arc u est incident à x et y. Plus précisement, il est dit incident intérieurement à y ou rentrant en y et il est incident extérieurement à x ou sortant de x.*

On note xU l'ensemble des arcs sortant de x et Ux l'ensemble des arcs rentrant en x.

Exemple 1.1.1

Soit $G = (X, U)$ le graphe orienté défini par : $X = \{x, y, z\}$ et $U = \{a, b, c, d, e\}$ où $a = (x, z)$, $b = (z, x)$, $c = (x, y)$, $d = (z, y)$ et $e = (y, y)$.

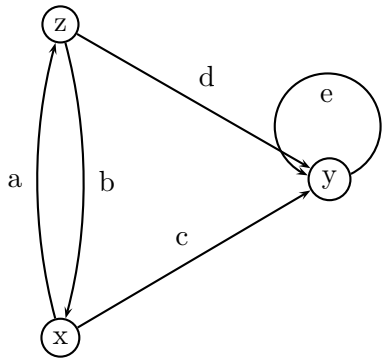


FIGURE 1.1 – Représentation graphique ou sagittale

Définition 1.1.9 *Soient $G = (X, U)$ un graphe orienté et x, y deux sommets de G.*

On dit que x est un précédent ou un prédécesseur de y s'il existe un arc de G de la forme $u = (x, y)$. Dans ce cas, y est dit suivant ou successeur de x.

Pour un sommet x de G, on note $S(x)$ l'ensemble des suivants et $P(x)$ l'ensemble des précédents de x.

Définition 1.1.10 *Deux arcs sont adjacents s'ils ont en commun une extrémité.*

1.1.2 Demi-degré et degré d'un sommet

Le demi-degré extérieur ou *demi-degré sortant* de sommet x, noté $d^+_G(x)$ ou plus simplement $d^+(x)$ est le nombre d'arcs ayant x pour extrémité initiale.
Le demi-degré intérieur ou *demi-degré entrant* de sommet x, noté $d^-_G(x)$ ou plus simplement $d^-(x)$ est le nombre d'arcs ayant x pour extrémité finale.
Le degré du sommet x, noté $d_G(x)$ ou plus simplement $d(x)$ est : $d(x) = d^+(x) + d^-(x)$.

On a le résultat suivant dont la démonstration est évidente.

Proposition 1.1.1 *Etant donné $G = (X, U)$ un graphe orienté, on a toujours :*

$$\sum_{x \in X} d^+_G(x) = \sum_{x \in X} d^-_G(x) = \text{ taille du graphe.}$$

Preuve : Chaque arc u a exactement une extrémité initiale et une extrémité terminale. Chaque membre de l'égalité ci-dessus est égal au nombre d'arcs du graphe. □

Proposition 1.1.2 *La somme des degrés des sommets de G est égale à deux fois la taille de G.*

Preuve : Il suffit de remarquer que dans la somme des degrés chaque arc est compté deux fois : une fois dans le demi-degré extérieur et une autre fois dans le demi-degré intérieur. □

Corollaire 1.1.1 *Dans un graphe orienté, le nombre de sommets de degrés impairs est pair.*

1.1.3 Quelques propriétés des graphes

Soit $G = (X, U)$ un graphe orienté. On dit que G est :
- *réflexif* si $\forall\, x \in X, (x, x) \in U$. Il y a une boucle sur chaque sommet.
- *anti-réflexif* si $\forall\, x \in X, (x, x) \notin U$. C'est un graphe sans boucle.
- *symétrique* si $\forall\, x \in X,\ \forall\, y \in X : (x, y) \in U \Rightarrow (y, x) \in U$.
L'existence d'un arc implique l'existence de l'arc opposé.
- *anti-symétrique* si $\forall\, x \in X,\ \forall\, y \in X : (x, y) \in U$ et $(y, x) \in U \Rightarrow x = y$.
L'existence d'un arc interdit l'existence de l'arc opposé, sauf dans le cas d'une boucle.
- *transitif* si $\forall\, x, y, z \in X : (x, y) \in U$ et $(y, z) \in U \Rightarrow (x, z) \in U$.
- *biparti* si l'ensemble des sommets X peut être partitionné en deux sous ensembles X_1 et X_2 de tel sorte que, tout arc a une extrémité dans X_1 et l'autre dans X_2.

1.1.4 Graphes complémentaires

Si $G = (X, U)$ est strict on appelle *graphe complémentaire* de G le graphe $\overline{G} = (X, \bar{U})$ ayant le même ensemble de sommets que G et ayant pour arcs les arcs complémentaires à U. C'est-à-dire :

$$\begin{cases} (x, y) \in U \Rightarrow (x, y) \notin \bar{U} \\ (x, y) \notin U \Rightarrow (x, y) \in \bar{U} \end{cases}$$

Exemple 1.1.2

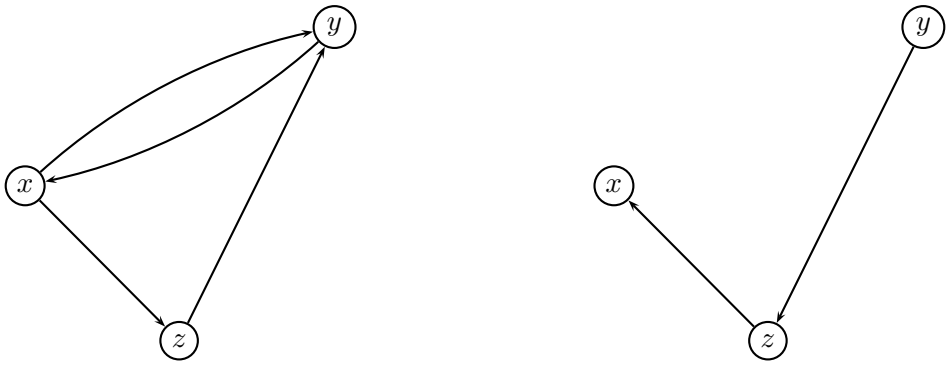


FIGURE 1.2 – Graphes complémentaires

1.1.5 Graphes homomorphes et isomorphes

Définition 1.1.11 *Deux graphes orientés $G = (X, U)$ et $G' = (X', U')$ sont dits homomorphes s'il existe une fonction $f : X \to X'$ telle que :*

$$(x, y) \in U \Rightarrow (f(x), f(y)) \in U'.$$

Définition 1.1.12 *Deux graphes orientés $G = (X, U)$ et $G' = (X', U')$ sont dits isomorphes s'il existe deux bijections $B_s : X \to X'$ et $B_a : U \to U'$ telles que deux arcs qui se correspondent dans la bijection B_a aient pour extrémités initiales et terminales respectivement des sommets qui se correspondent dans la bijection B_s. C'est-à-dire si :*

$$u = (x, y) \text{ et } u' = (x', y') \text{ avec } u' = B_a(u), \text{ alors } B_s(x) = x' \text{ et } B_s(y) = y'.$$

L'isomorphisme des graphes est une relation d'équivalence. Une classe d'équivalence de cette relation est appelée un graphe non étiqueté. Graphiquement on représente le graphe en omettant d'attribuer des noms aux sommets.

Exemple 1.1.3

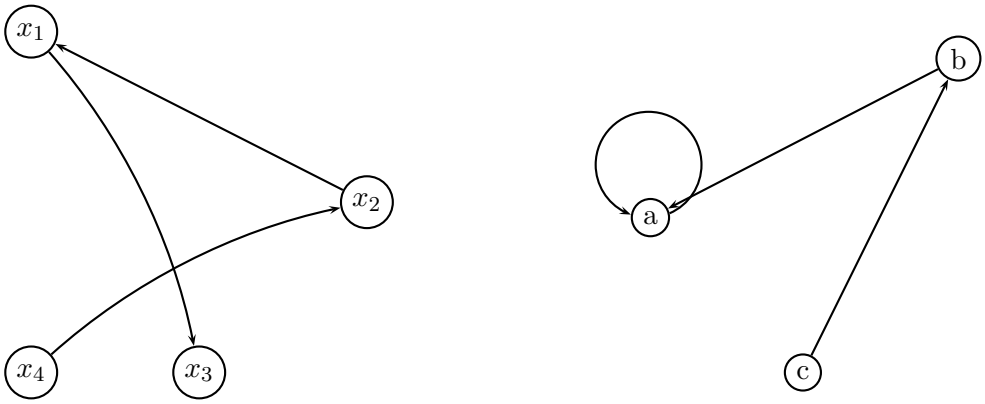


FIGURE 1.3 – Graphes homomorphes

Exemple 1.1.4

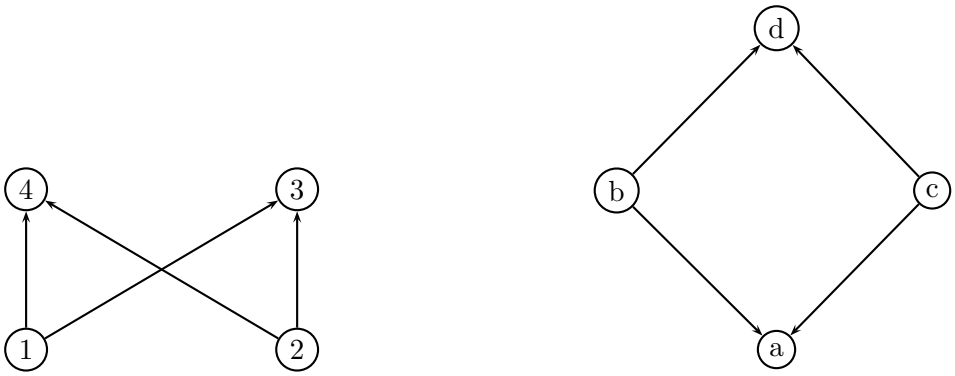


FIGURE 1.4 – Graphes isomorphes

1.1.6 Sous-graphes et graphes partiels

Soient $G = (X, U)$ un graphe orienté, Y une partie de X et V une partie de U.

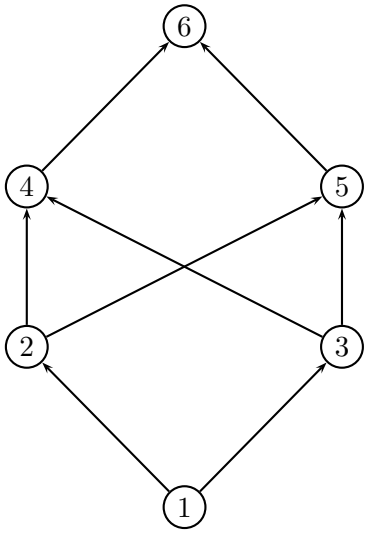
On appelle *sous-graphe* engendré par Y, le graphe noté G_Y dont les sommets sont les éléments de Y et dont les arcs sont les arcs de G ayant leurs deux extrémités dans Y. On a : $G_Y = (Y, U \cap (Y \times Y))$.

On appelle *graphe partiel* engendré par V, le graphe noté $G[V]$ ayant le même ensemble X de sommets que G et dont les arcs sont ceux de V. On a : $G[V] = (X, V)$.

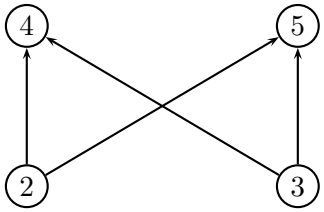
Le *sous-graphe partiel* engendré par Y et V noté $G_Y[V]$ est le graphe partiel de G_Y engendré par V. On a : $G_Y[V] = (Y, V \cap (Y \times Y))$.

Exemple 1.1.5

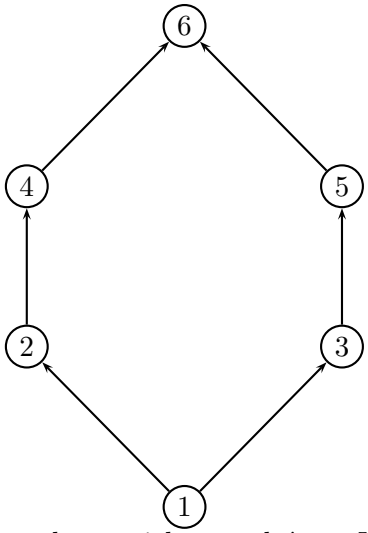
On considère le graphe $G = (X, U)$ ci-dessous.



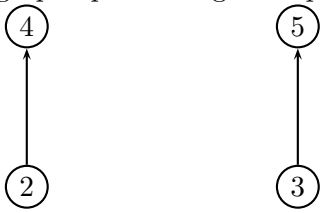
Le sous-graphe engendré par $Y = \{2, 3, 4, 5\}$ est



Le graphe partiel engendré par $V = U - \{(2, 5), (3, 4)\}$ est :



Le sous-graphe partiel engendré par Y et V est :



1.1.7 Cheminement dans un graphe

Chemin, circuit

Définition 1.1.13 *Etant donné un graphe orienté $G = (X, U)$ et x, y deux sommets de G, on appelle chemin de x à y toute suite ordonnée d'arcs $\mu = (u_1, \cdots, u_q)$ telle que :*

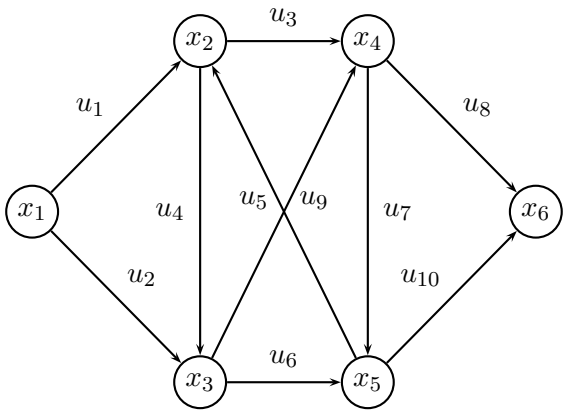
i) l'origine de u_1 est x,

ii) pour tout $i \in \{1, \cdots, q-1\}$, l'origine de u_{i+1} est l'extrémité de u_i,

iii) l'extrémité de u_q est y.

Les sommets x et y sont alors dits extrémités du chemin μ qui est de longueur q au sens des arcs.

Exemple 1.1.6



Dans le graphe ci-dessus, $\mu = (u_1, u_3, u_7, u_9, u_4, u_5, u_8)$ est un chemin de x_1 à x_6.

Un *chemin élémentaire* est un chemin tel qu'en le parcourant, on ne rencontre pas deux fois le même sommet. Dans un chemin élémentaire tous les sommets sont de degré 2 au plus.

Remarque 1.1.1 *Si le graphe n'a pas d'arcs parallèles comme c'est le cas de ce cours, un chemin peut être défini par la suite des sommets qu'il traverse.*

Un *chemin simple* est chemin dont la sequence d'arcs ne contient pas plusieurs fois un même élément. C'est-à-dire en le parcourant, on ne rencontre pas plusieurs fois un même arc.

Proposition 1.1.3 *Tout chemin élémentaire est simple.*

Proposition 1.1.4 *De tout chemin on peut extraire un chemin élémentaire.*

Définition 1.1.14 *Etant donné un graphe orienté $G = (X, U)$, un sommet y est dit descendant de x, s'il existe un chemin de x à y. dans ce cas , x est un ascendant de y.*

Un *circuit* est un chemin dont les extrémités coïncident.

Un *circuit élémentaire* est un chemin élémentaire dont les deux extrémités coïncident.

Un *circuit simple* est un chemin simple dont les deux extrémités coïncident.

On montre que

Proposition 1.1.5 *Tout circuit simple est union disjointe de circuits élémentaires.*

Définition 1.1.15 *Etant donné un graphe orienté $G = (X, U)$,*

- un chemin est hamiltonien s'il passe une fois et une seule par chaque sommet du graphe. Ou encore un chemin élémentaire passant par tous les sommets du graphe.

- Un circuit hamiltonien est un chemin hamiltonien dont les deux extrémités coïncident.

On parle de chemin pré-hamiltonien s'il passe au moins une fois par chaque sommet du graphe.

Définition 1.1.16 *Etant donné un graphe orienté $G = (X, U)$,*

- un chemin est eulérien s'il passe une fois et une seule par chaque arc du graphe. Ou encore un chemin simple passant par tous les arcs du graphe.

- Un circuit eulérien est un chemin eulérien dont les deux extrémités coïncident.

On parle de chemin pré-eulérien s'il passe au moins une fois par chaque arc du graphe.

Chaîne, cycle

Une *chaîne* de longueur k est une suite $\mu = a_1, a_2, \dots a_k$ de k arcs telle que deux arcs consécutifs sont adjacents. On notera $\mu = x_0 a_1 x_1 \dots x_{k-1} a_k x_k$, si l'on veut préciser les sommets rencontrés. Les sommets x_0 et x_k sont appelés les *extrémités de la chaîne* μ. On dit que la chaîne μ rélie les sommets x_0 et x_k.

Une chaîne peut être "vue" comme une suite de sommets telle que deux sommets consécutifs soient liés par un arc, certains arcs peuvent être parcourus dans le sens de l'orientation (sens +), les autres arcs dans le sens opposé (sens -).

Une *chaîne élémentaire* est une chaîne telle qu'en la parcourant, on ne rencontre pas deux fois le même sommet.

Une *chaîne simple* est une chaîne dont la sequence d'arcs ne contient pas plusieurs fois un même élément.

Proposition 1.1.6 *Toute chaîne élémentaire est simple.*

Proposition 1.1.7 *De toute chaîne on peut extraire un chaîne élémentaire.*

Un *cycle* est une chaîne dont les extrémités coïncident.

Un *cycle élémentaire* (resp. *simple*) est une chaîne élémentaire (resp. simple) dont les deux extrémités coïncident.

Proposition 1.1.8 *Tout cycle simple est union disjointe de cycles élémentaires.*

Définition 1.1.17 *Etant donné un graphe orienté $G = (X, U)$,*

- une chaîne est hamiltonienne si elle passe une fois et une seule par chaque sommet du graphe. Ou encore une chaîne élémentaire passant par tous les sommets du graphe.

- Un cycle hamiltonien est une chaîne hamiltonienne dont les deux extrémités coïncident.

Le graphe G est dit hamiltonien s'il possède un cycle hamiltonien.

On parle de chaîne pré-hamiltonienne si elle passe au moins une fois par chaque sommet du graphe.

Définition 1.1.18 *Etant donné un graphe orienté $G = (X, U)$,*

- une chaîne est eulérienne si elle passe une fois et une seule par chaque arc du graphe. Ou encore une chaîne simple passant par tous les arcs du graphe.

- Un cycle eulérien est un cycle simple passant par tous les arcs du graphe.

Le graphe G est dit eulérien s'il possède un cycle eulérien

1.1.8 Connexité et forte connexité

Graphe connexe

Soit $G = (X, U)$ un graphe orienté. On dit que G est *connexe* si, pour tout couple de sommets (x, y), il existe une chaîne reliant x et y.

On associe à cette notion de connexité une relation d'équivalence $\mathcal{R}$ définie par :

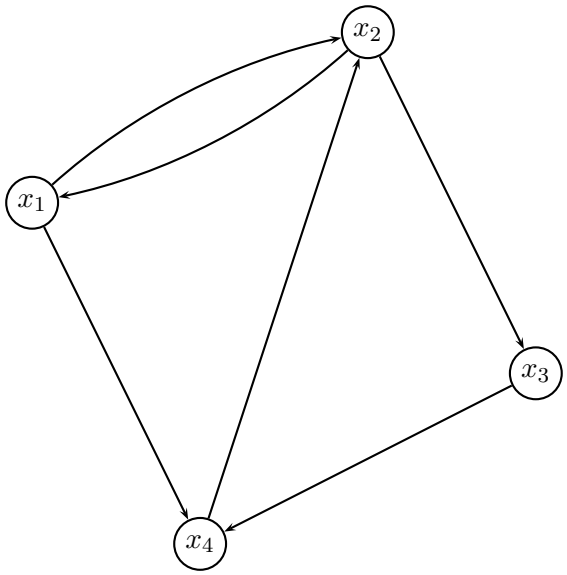
$$x \mathcal{R} y \Leftrightarrow \begin{cases} \text{soit} \quad x = y, \\ \text{soit il existe une chaîne reliant} \quad x \text{ et } y. \end{cases}$$

Les classes d'équivalence induites sur X par cette relation sont appelées classes connexes de G : elles forment une partition de X en : $X_1, \dots, X_p$. Le nombre p de classes distinctes est

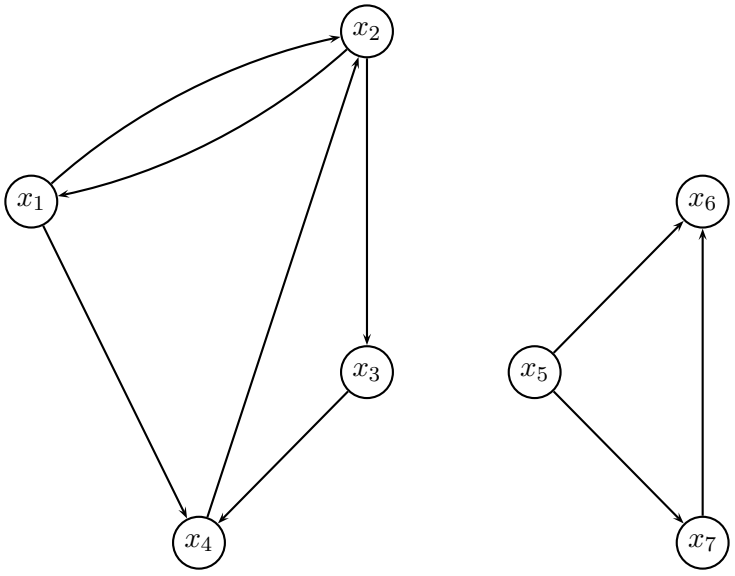
appelé le *nombre de connexité* du graphe. Les sous-graphes induits par les classes connexes sont appelés composantes connexes de G.

Il s'ensuit alors que G est connexe si et seulement si son nombre de connexité est $p = 1$.

Exemple 1.1.7



Graphe connexe

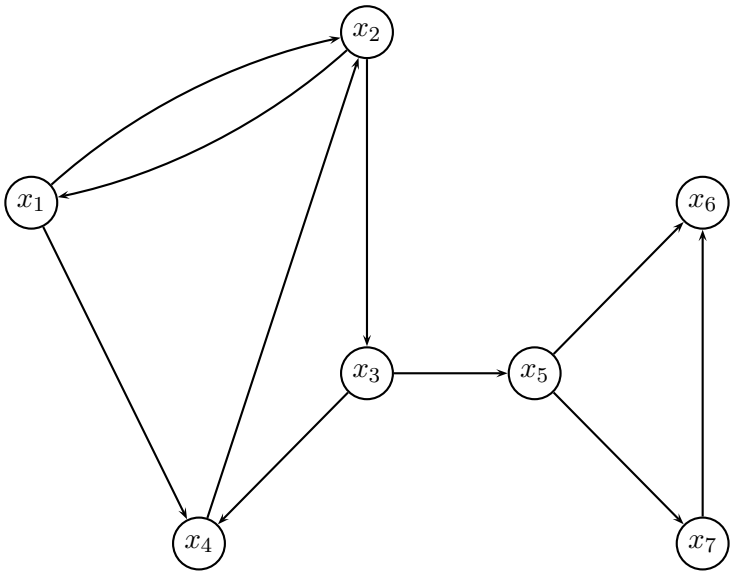


Graphe non connexe

Définition 1.1.19 *Un point d'articulation (resp. ensemble d'articulation) d'un graphe est un sommet (resp. un ensemble de sommets) dont la suppression augmente le nombre de composantes connexes.*

Un isthme est un arc dont la suppression a le même effet.

Exemple 1.1.8



Dans le graphe ci-dessus, les sommets x_3 et x_5 sont des points d'articulation. L'ensemble $\{x_3, x_5\}$ est un ensemble d'articulation et l'arc (x_3, x_5) est un isthme.

On peut alors remarquer que :

Remarque 1.1.2 *Lorsque G est connexe, Y est un ensemble d'articulation si et seulement si $G_{X\setminus Y}$ n'est plus connexe et u un arc de G est un isthme si et seulement si $G[U \setminus \{u\}]$ n'est plus connexe.*

Graphe fortement connexe

Soit $G = (X, U)$ un graphe orienté. On dit que G est *fortement connexe* si, pour tout couple de sommets (x, y), il existe un chemin allant de x à y. On associe à cette notion de forte connexité une relation d'équivalence $\mathcal{T}$ définie par :

$$x\mathcal{T}y \Leftrightarrow \begin{cases} \text{soit } \ x = y, \\ \text{soit il existe un chemin allant de } \ x \text{ à } y \text{ et un chemin allant de } y \text{ à } x. \end{cases}$$

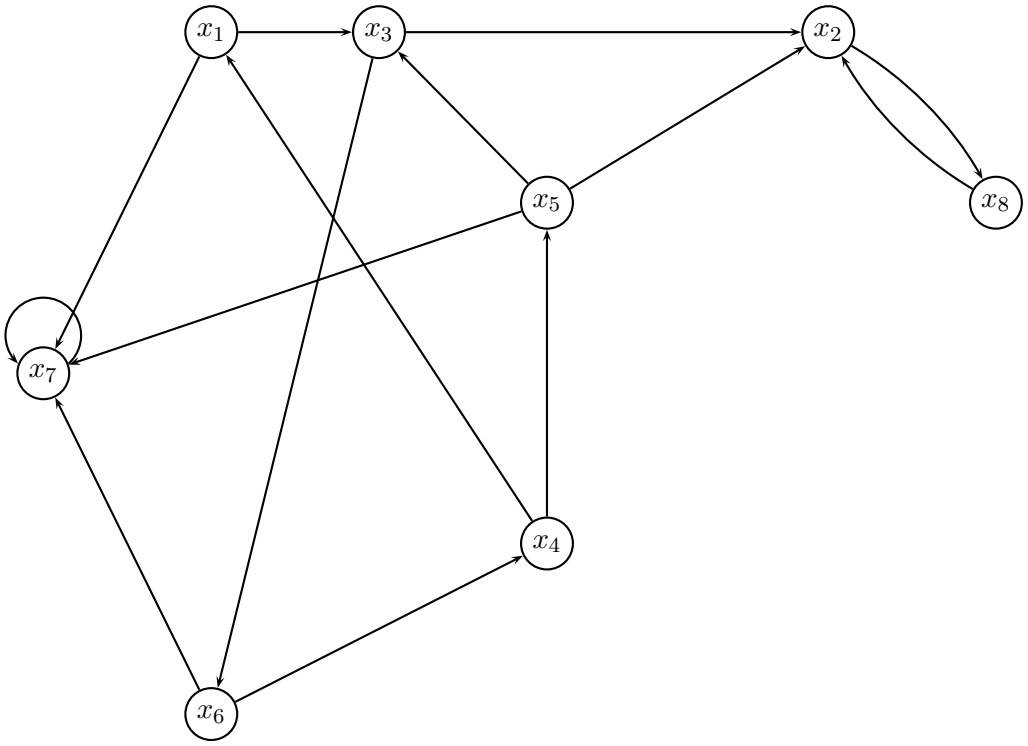
Les classes d'équivalence induites sur X par cette relation forment une partition de X en : $X_1, \ldots, X_q$. Le nombre q de classes distinctes est appelé le *nombre de forte connexité* du graphe.

Les sous graphes $G_1, \ldots, G_q$ engendrés par les sous ensembles $X_1, \ldots, X_q$ sont appelés les *composantes fortement connexes* du graphe G. Chaque composante fortement connexe est un graphe fortement connexe.

Définition 1.1.20 *G est fortement connexe s'il n'a qu'une seule composante fortement connexe.*

Exemple 1.1.9

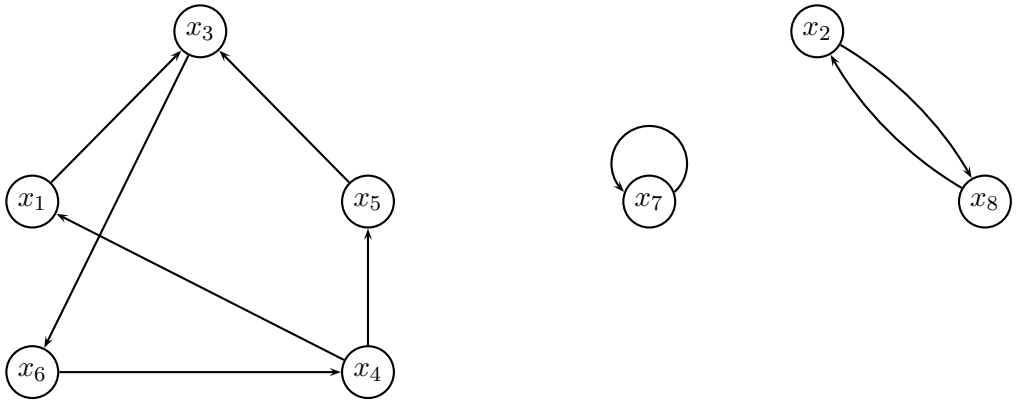
Considérons le graphe ci-dessous



Les classes fortement connexes sont

$$\{x_1, x_3, x_4, x_5, x_6\}, \quad \{x_2, x_8\}, \quad \{x_7\}.$$

Les composantes fortement connexes sont :



1.2 Graphes non orientés

Dans l'étude de certaines propriétés des graphes, il arrive que l'orientation des arcs, c'est-à-dire la distinction entre extrémité initiale et extrémité terminale ne joue aucun rôle. On s'intéresse simplement à l'existence ou non d'un arc entre deux sommets sans en préciser l'ordre. On parle alors de graphe non orienté ou simplement de *graphe*.

1.2.1 Définitions

Définition 1.2.1 *Un graphe non orienté G est défini par la donnée d'un ensemble fini X dont les éléments sont appelés des sommets ou noeuds et d'un ensemble E dont les éléments sont des sous-ensembles à 1 ou 2 éléments de X appelés arêtes. On le note $G = (X, E)$. Le nombre n de sommets de G est appelé l'ordre de G et le nombre m des arêtes de G est appelée la taille de G.*

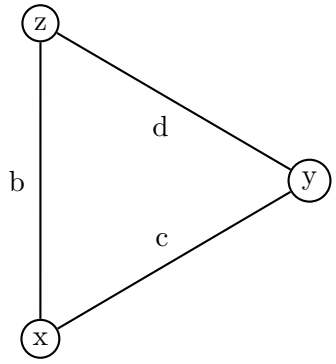
Les arêtes de la forme $u = \{x\} \in E$ sont appelées boucles.

Définition 1.2.2 *Etant donné un graphe non orienté, deux arêtes u et v sont parallèles si elles ont les mêmes extrémités.*

Dans tout ce qui suit le terme "graphe non orienté" ne concerne que les graphes sans arêtes parallèles

Exemple 1.2.1

Soit $G = (X, E)$ le graphe défini par : $S = \{x, y, z\}$ et $E = \{a, b, c, d\}$ où $b = \{z, x\}$, $c = \{x, y\}$ et $d = \{z, y\}$.



A un graphe orienté, on associe un graphe non orienté en "laissant tomber l'orientation". De même à un graphe non orienté, on associe un graphe orienté, soit en introduisant pour toute arête les deux arcs qui lui correspondent, soit en choisissant un seul des deux arcs.

Définition 1.2.3 *Etant donnée une arête $u = \{x, y\}$ d'un graphe non orienté $G = (X, E)$, on dit que les sommets x et y sont les extrémités de u. On dit également que les sommets x et y sont adjacents et que l'arête u est incidente à x et y. On appelle voisinage de x, l'ensemble des sommets adjacents à x. On le note $\mathcal{V}(x)$.*

Un sommet qui n'est adjacent à aucun autre sommet est dit isolé.

Définition 1.2.4 *Un graphe non orienté G est dit simple s'il est sans boucle.*

Définition 1.2.5 *Un graphe non orienté G est dit discret si sa taille est nulle. C'est-à-dire tous ses sommets sont isolés.*

Définition 1.2.6 *Un graphe non orienté simple G est dit complet si deux sommets quelconques sont adjacents.*

On note K_n un graphe non orienté simple d'odre n en hommage à Kuratovski.

Définition 1.2.7 *Deux arêtes sont adjacentes si elles sont incidentes à au moins une extrémité commune.*
Le degré du sommet x, noté $d_G(x)$ ou plus simplement $d(x)$ est le nombre d'arêtes ayant x pour extréminté.

Les résultats de la proposition 1.1.2 et du corollaire 1.1.1 restent valables.

Définition 1.2.8 *Un graphe est dit régulier si tous ses sommets ont même degré.*

Proposition 1.2.1 *Soit $G = (X, E)$ un graphe non orienté simple d'ordre n et de taille m. Alors on a $m \leq \frac{n(n-1)}{2}$.*

Proposition 1.2.2 *Soit $G = (X, E)$ un graphe non orienté simple d'ordre n et de taille m. Alors G est complet si et seulement si $m = \frac{n(n-1)}{2}$.*

Proposition 1.2.3 *Soit $G = (X, E)$ un graphe non orienté simple d'ordre n et de taille m. Alors il existe au moins deux sommets ayant même degré.*

Remarque 1.2.1 *Les notions de demi-degré, de chemin, circuit et de forte connexité n'ont pas d'intérêt dans le cas des graphes non orienté.*

1.2.2 Chaîne, cycle

Soit G un graphe non orienté. Une *chaîne* de longueur k dans G est une suite $\mu = e_1, e_2, \ldots, e_k$ de k arêtes telle que deux arêtes consécutives sont adjacentes. On notera $\mu = x_1 e_1 x_2 \ldots x_k e_k x_{k+1}$, si l'on veut préciser les sommets rencontrés. Les sommets x_1 et x_k sont appelés les *extrémités de la chaîne* μ. On dit que la chaîne μ rélie les sommets x_1 et x_k.

Une *chaîne élémentaire* est une chaîne telle qu'en la parcourant, on ne rencontre pas deux fois le même sommet.

Une chaîne est *simple* si la séquence d'arêtes qui la constitue ne comporte pas plusieurs fois le même élément.

La proposition 1.1.7 est valable pour les chaînes.

Un *cycle* est une chaîne dont les extrémités coïncident.

Un *cycle élémentaire* est un cycle minimal pour l'inclusion i.e ne contenant strictement aucun autre cycle.

Définition 1.2.9 *Etant donné un graphe non orienté $G = (X, E)$,*

- une chaîne est hamiltonienne si elle passe une fois et une seule par chaque sommet du graphe. Ou encore une chaîne élémentaire passant par tous les sommets du graphe.

- Un cycle hamiltonien est une chaîne hamiltonienne dont les deux extrémités coïncident.

Le graphe G est dit hamiltonien s'il possède un cycle hamiltonien.

On parle de chaîne pré-hamiltonienne si elle passe au moins une fois par chaque sommet du graphe.

Définition 1.2.10 *Etant donné un graphe non orienté $G = (X, E)$,*

- une chaîne est eulérienne si elle passe une fois et une seule par chaque arc du graphe. Ou encore une chaîne simple passant par tous les arcs du graphe.

- Un cycle eulérien est un cycle simple passant par tous les arcs du graphe.

Le graphe G est dit eulérien s'il possède un cycle eulérien

On a le résultat suivant démontré par L. Euler.

Théorème 1.2.1 *Etant donné un graphe non orienté G sans sommet isolé, G a une chaîne eulerienne si et seulement si G a 0 ou 2 sommets de degré impair.*

L. Euler a établi ce théorème afin de résoudre le problème ds 7 ponts de Königsberg et du coup initia la théorie des graphes.

NB : Les notions d'isomorphie, de connexité de sous-graphe et graphes partiel s'étendent de manière directe au cas des graphes non orienté.

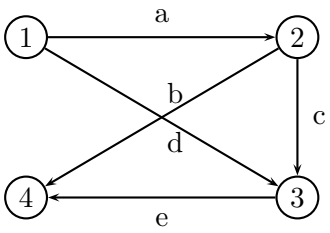
1.3 Autres représentations des graphes

1.3.1 Cas des graphes orientés

Définition 1.3.1 (Matrice d'incidence (sommets-arcs)) *Soit $G = (X, U)$ un graphe orienté tel que $X = \{x_1, \cdots, x_n\}$ et $U = \{u_1, \cdots, u_m\}$. La matrice d'incidence sommets-arcs de G est la $n \times m$ matrice $A = (a_{ij})$ à coefficients entiers : 0, -1 et 1 telle que*

$$a_{ij} = \begin{cases} 1 & \text{si } x_i \text{ est origine de l'arc } u_j \\ -1 & \text{si } x_i \text{ est extrémité de l'arc } u_j \\ 0 & \text{dans les autres cas} \end{cases}$$

Exemple 1.3.1



On obtient la matrice d'incidence :

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & -1 & 0 & 0 & -1 \end{pmatrix}$$

Définition 1.3.2 (Matrice d'adjacence (sommets-sommets)) *Soit $G = (X, U)$ un graphe orienté tel que $X = \{x_1, \cdots, x_n\}$. La matrice d'adjacence sommets-sommets ou matrice booléenne de G est la matrice carrée d'ordre n $A = (a_{ij})$ à coefficients entiers : 0 et 1 telle que pour tout x_i et x_j dans X on a :*

$$a_{ij} = 1 \text{ si et seulement si } (x_i, x_j) \in U.$$

La matrice d'adjacence de l'exemple précédent est

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

On montre facilement que :

Proposition 1.3.1 *Si M est la matrice d'adjacence d'un graphe G, le coefficient d'indice ij de la matrice M^k est le nombre de chemins de longueur k reliant x_i à x_j.*

Définition 1.3.3 *Soit G un graphe orienté. On appelle dictionnaire des précédents (resp. suivants), le tableau à simple entrée qui à tout sommet x énumère les précédents $P(x)$ (resp. les suivants $S(x)$) de x.*

Sommets	Précédents
x	$P(x)$

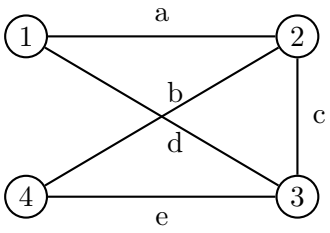
Sommets	Suivants
x	$S(x)$

1.3.2 Cas des graphes non orientés

Définition 1.3.4 (Matrice d'incidence (sommets-arêtes)) *Soit $G = (X, U)$ un graphe non orienté tel que $X = \{x_1, \cdots, x_n\}$ et $E = \{e_1, \cdots, e_m\}$. La matrice d'incidence sommets-arêtes de G est la $n \times m$ matrice $A = (a_{ij})$ à coefficients entiers : 0 et 1 telle que*

$$a_{ij} = \begin{cases} 1 & \text{si } x_i \text{ est une extrémité l'arête } u_j \\ 0 & \text{dans les autres cas} \end{cases}$$

Exemple 1.3.2



On obtient la matrice d'incidence :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Définition 1.3.5 (Matrice d'adjacence (sommets-sommets)) *La matrice d'adjacence sommets-sommets ou matrice booléenne d'un graphe non orienté $G = (X, E)$ avec $X = \{x_1, \cdots, x_n\}$ est la matrice $A = (a_{ij})$ à coefficients entiers : 0 et 1 telle que pour tout x_i et x_j dans X on a :*

$$a_{ij} = 1 \text{ si et seulement si } \{x_i, x_j\} \in E.$$

La matrice d'adjacence de l'exemple précédent est

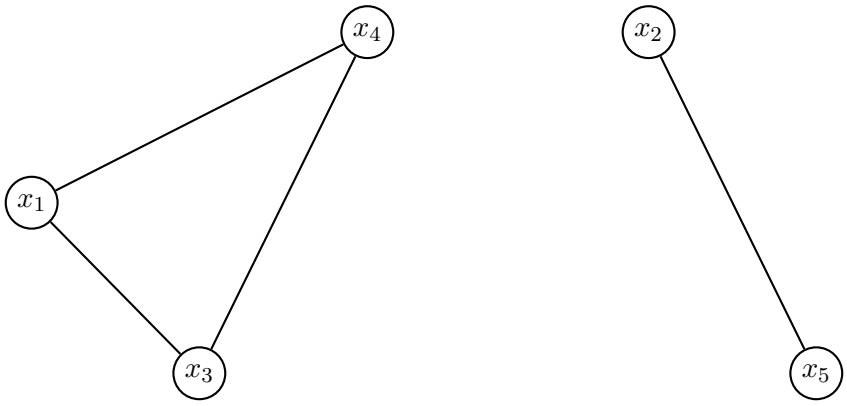
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

1.4 Quelques notions particulières

1.4.1 Clique, Stable et Coloration des sommets

Définition 1.4.1 *Soit G un graphe non orienté sans boucle. Une clique de G est un sous-ensemble C de sommets tel que deux sommets quelconques de C sont reliés par une arête.*

Exemple 1.4.1



Dans le graphe ci-dessus, l'ensemble $C = \{x_1, x_3, x_4\}$ est une clique.

Définition 1.4.2 *Soit G un graphe non orienté sans boucle. Un stable de G est un sous-ensemble S de sommets tel que deux sommets quelconques de S sont non adjacents.*

Le nombre de stabilité de G est le cardinal du plus grand stable de G. On le note $\alpha(G)$.

Colorier les sommets d'un graphe consiste à associer une couleur à chaque de façon que deux sommets adjacents n'aient pas la même couleur. Plus formellement

Définition 1.4.3 *Soit G un graphe non orienté sans boucle. Une coloration de G est une application $\gamma : X \to C$ de l'ensemble des sommets de G dans un ensemble fini C de couleurs, telle que si une arête e est incidente à x et y alors $\gamma(x) \neq \gamma(y)$.*

Remarque 1.4.1 *Une coloration est donc une partition du graphe en stables.*

Une coloration avec k couleurs, on dit aussi une k-coloration est une partition de l'ensemble des sommets en k stables.

Définition 1.4.4 *Un graphe est k chromatique s'il peut être colorié avec k couleurs.*

Le nombre chromatique d'un graphe G est le nombre minimum de couleurs distcintes nécessaires à la coloration des sommets de G. On le note $\gamma(G)$.

Les propriétés suivantes sont faciles à prouver :

Proposition 1.4.1 *- Le nombre chromatique du graphe K_n est n ;*

- Le nombre chromatique d'un graphe d'ordre n est inférieur ou égal à n ;

- Si un graphe d'ordre n a un sous-graphe complet d'ordre k, alors son nombre chromatique est compris entre k et n.

On considère les notations suivantes :

Notations : On note par $\lceil x \rceil$ le plus petit entier supérieur ou égal à x et par $\lfloor x \rfloor$ le plus grand entier inférieur ou égal à x.

On montre que

Proposition 1.4.2 *Soit $G = (X, E)$ un graphe non orienté sans boucle d'ordre n. On a :*

$$\left\lceil \frac{n}{\alpha(G)} \right\rceil \leq \gamma(G) \leq \Delta + 1.$$

où $\Delta = \max\{d_G(x) : x \in X\}$.

Preuve :

Notons $\{X_1, X_2, \cdots, X_k\}$ une partition de X en k stables avec $k = \gamma(G)$.

Alors :

$$n = \sum_{i=1}^{k} |X_i| \leq \sum_{i=1}^{k} \alpha(G) = k\alpha(G) = \gamma(G)\alpha(G).$$

n, $\gamma(G)$, α étant des entiers naturels, l'inégalité de qauche est établie.

Contruisons une partition de X en sous-ensembles stables de X.

- On considère un sommet x_1 arbitraire de X et X_1 une plus grande partie stable contenant x_1.

- S'il existe un sommet x_2 de X qui n'est pas dans X_1, on construit X_2 une plus grande partie stable de X contenant x_2 disjointe avec X_1 ;

- S'il existe un sommet x_3 de X qui n'est pas dans $X_1 \bigcup X_2$, on construit X_3 une plus grande partie stable de X contenant x_3 telle que X_1, X_2 et X_3 soient 2 à 2 disjoints.

Comme X est fini, ce procédé se terminera et on obtient une partition $\{X_1, X_2, \cdots, X_k\}$ de X.

En choisissant une couleur par élément de la partition, on aura nécessairement $\gamma(G) \leq k$.

Considérons à présent un sommet x de la partie X_k. Le caractère maximal des parties construites assure que ce sommet est adjacent à au moins un sommet de chaque partie X_i, $1 \leq i \leq k-1$. On en déduit alors que $d_G(x) \geq k-1$, d'où $\Delta \geq d_G(x) \geq k-1 \geq \gamma(G) - 1$. Ce qui établit la deuxième inégalité. □

On montr aussi que :

Proposition 1.4.3 *Si G est un graphe non orienté sans boucle d'ordre n, on a*

$$\alpha(G) + \gamma(G) \leq n + 1.$$

Preuve :

Considérons S une partie stable de l'ensemble des sommets de G de cardinal $\alpha(G)$.

Une coloration possible des sommets consiste à colorier les sommets de S d'une même couleur. On en déduit que

$$\gamma(G) \leq 1 + (n - \alpha(G)).$$

Ce qui est équivalent à :

$$\alpha(G) + \gamma(G) \leq n + 1.$$

D'où la proposition □

1.4.2 Ferméture transitive d'un graphe

Définition 1.4.5 *Soit $G = (X, U)$ un graphe orienté. On appelle ferméture transitive de G le graphe $\widehat{G} = (X, \widehat{U})$ de même ensemble de sommet que G et tel que $(x, y) \in \widehat{U}$ s'il existe dans G un chemin de x à y.*

On dit que la ferméture de G est le plus petit graphe transitif contenant G, c'est-à-dire ayant G comme graphe partiel et possédant le plus petit nombre d'arcs.

Schématiquement, on construit $\widehat{U}$ en ajoutant à U un arc (x, z) qui n'appartient pas à U si $(x, y) \in U$ et $(y, z) \in U$ pour au moins un sommet $y \in X$ et ainsi de suite jusqu'à l'obtention d'un graphe transitif.

Exemple 1.4.2

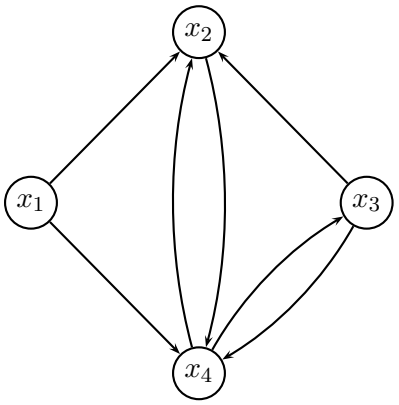


FIGURE 1.5 – Graphe de départ

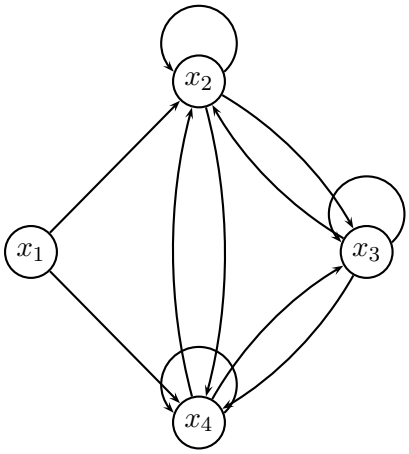


FIGURE 1.6 – Ferméture transitive du graphe

On appelle somme booléenne l'opération notée $\oplus$ telle que

1. $0 \oplus 0 = 0$
2. $1 \oplus 0 = 1$
3. $0 \oplus 1 = 1$
4. $1 \oplus 1 = 1$

et le produit booléen est l'opération notée $\otimes$ telle que

1. $0 \otimes 0 = 0$
2. $1 \otimes 0 = 0$
3. $0 \otimes 1 = 0$
4. $1 \otimes 1 = 1$.

La somme et le produit booléens de deux matrices s'obtiennent en substituant $+$ et $\times$ aux opérateurs $\oplus$ et $\otimes$ dans la somme et le produit matriciels usuels.

On montre que :

Proposition 1.4.4 *Si $G = (X; U)$ un graphe orienté d'ordre n avec $X = \{x_1, x_2, \ldots, x_n\}$ et M sa matrice d'adjacence,*

$$\widetilde{M} = \underbrace{M \otimes M \otimes \cdots \otimes M}_{p \text{ facteurs}} \equiv M^{[p]}$$

la $p^{ième}$ puissance booléenne (p un entier naturel non nul) de la matrice d'adjacence M avec $\widetilde{M} = (\widetilde{m}_{ij})$, alors on a :

$$\widetilde{m}_{ij} = 1 \text{ si et seulement si il existe dans } G \text{ un chemin de cardinalité } p \text{ entre } x_i \text{ et } x_j.$$

Ce qui veut dire que la $p^{\text{ième}}$ puissance booléenne (p un entier naturel non nul) de la matrice booléeenne M de G est la matrice d'adjacence d'un graphe de même ensemble de sommets que G où un arc entre x et y signifie qu'il existe dans G un chemin allant de x à y de longueur p au sens des arcs.

En pratique, pour obtenir la matrice $M^{[p]}$, on calcule M^p puis on remplace tous les coefficients non nuls par des 1.

Remarque 1.4.2 *Dans un graphe à n sommets de matrice d'adjacence M, si $M^{[n]}$ n'est pas nulle, alors il existe un chemin de longueur n ; il passe par $n+1$ sommets, donc le graphe contient au moins un circuit.*

Proposition 1.4.5 *Soit M la matrice booléenne d'un graphe orienté d'ordre n. Alors la matrice booléeene de la ferméture de G est la matrice*

$$\widehat{M} = M \oplus M^{[2]} \oplus M^{[3]} \oplus \cdots \oplus M^{[n]}.$$

Proposition 1.4.6 *Le graphe G est sans circuit si et seulement si les éléments diagonaux de la matrice d'adjacence de sa ferméture transitive sont tous nuls.*

1.4.3 Ordonnacement par niveaux des sommets

Dans un graphe orienté sans circuit, on peut ordonner les sommets par niveaux. On définit :

Définition 1.4.6 *Soit G un graphe orienté sans circuit. On appelle niveau d'un sommet x, la longueur (au sens des arcs) du plus long chemin ayant pour extrémité terminale x.*

Pour déterminer les niveaux des sommets de G un graphe orienté sans circuit, on peut utiliser soit le dictionnaire des précédents soit la matrice booléenne.

Algorithme utilisant le dictionnaire des précédents

1. On note N_0 l'ensemble des sommets de G qui n'ont pas de précédents, c'est-à-dire de niveau 0.
2. Barrer dans le dictionnaire des précédents les éléments de N_0 partout où ils se trouvent. Les sommets n'ayant plus de précédents sont de niveau 1. Soit N_1 cet ensemble.
3. On réitère cette procédure en augmentant de 1, la valeur des niveaux jusqu'à ce que tous les sommets soient barrés.

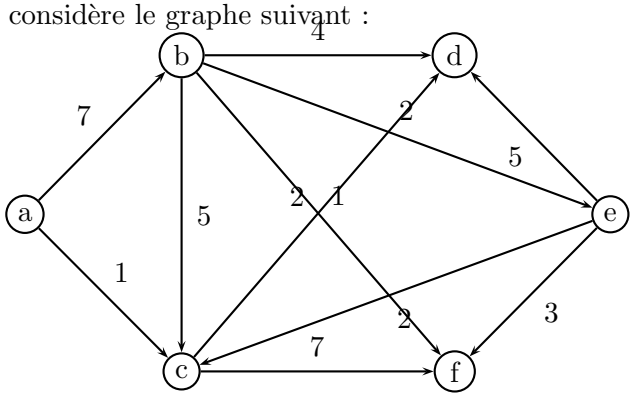
Algorithme utilisant la matrice booléenne

1. L'ensemble N_0 des sommets de niveau 0 est l'ensemble des sommets dont les colonnes sont nulles.
2. Barrer dans la matrice booléenne les colonnes et les lignes des éléments de N_0. Les sommets dont les colonnes sont nulles dans la matrice ainsi obtenue sont de niveau 1. Soit N_1 cet ensemble.
3. On réitère cette procédure en augmentant de 1, la valeur des niveaux jusqu'à ce que toutes les colonnes soient barrées.

Le graphe ordonnancé par niveaux du graphe est une représentation graphique plus simple (avec moins d'intersections possibles des arcs). Dans cette représentation les sommets d'un même niveau sur une verticale, les niveaux étant placés de gauche à droite par ordre croissant.

Exemple 1.4.3

On considère le graphe suivant :



On a les niveaux suivants :

$$N_0 = \{a\},\ N_1 = \{b\},\ N_2 = \{e\},\ N_3 = \{c\},\ N_4 = \{d, f\}.$$

Chapitre 2

Arbres et arborescences

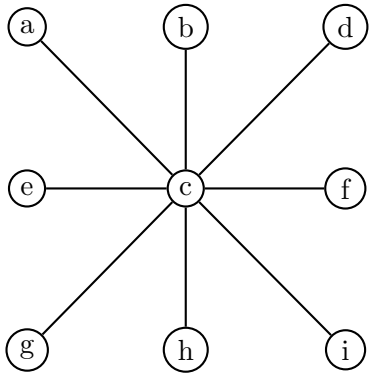
La notion d'arbre est fondamentale en recherche opérationnelle. Elle est utilisée dans de treès nombreux domaines notamment l'informatique, les sciences sociales, l'optimisation combinatoire et la théorie des réseaux électriques.

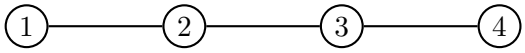
2.1 Arbres

2.1.1 Définitions et propriétés des arbres

Un *arbre* est un graphe simple connexe sans cycles. Un graphe simple sans cycles qui n'est pas nécessairement connexe est appelé une *forêt* (chaque composante connexe est un arbre). Une chaîne élémentaire est un arbre.

Exemples d'arbre





Proposition 2.1.1 *Un arbre tel que $n \geq 2$ possède au moins deux sommets pendants (ie de degré 1).*

Preuve : Il suffit de considérer une chaîne $\mu = \{s_0, e_1, s_1, \ldots, s_{k-1}, e_k, s_k\}$ de longueur la plus grande possible (maximale) où $k \geq 1$ car $n \geq 2$. On montre par l'absurde que s_0 et s_k sont des sommets pendants. Par exemple supposons que s_0 n'est pas pendant alors il existerait une arête $f \neq e_1$ incidente à s_0 et soit y l'autre sommet incident à f. Si $y \in \{s_0, s_1, \ldots, s_k\}$ alors l'arbre contiendrait un cycle ce qui est absurde ; sinon la chaîne $\sigma = \{y, f, s_0, e_1, s_1, \ldots, s_{k-1}, e_k, s_k\}$ contiendrait μ ce qui est également absurde. □

Proposition 2.1.2 *Si G est un arbre alors $m = n - 1$.*

Preuve : On fait une preuve par récurrence sur n. Il est facile d'établir le résultat pour $n = 1$ car un arbre ne peut contenir une boucle. On suppose que $n > 1$. D'après la proposition 2.1.1, il existe un sommet pendant s. Il est facile de voir que le graphe $G' = G - \{s\}$ est un arbre et donc par hypothèse de récurrence on a : $m_{G'} = n_{G'} - 1$. Comme $m_{G'} = m_G - 1$ et $n_{G'} = n_G - 1$ on obtient $m_G = n_G - 1$. □

Proposition 2.1.3 *Dans un arbre, deux sommets quelconques sont reliés par une chaîne élémentaire unique.*

Preuve : Soient x et y deux sommets d'un arbre. Il existe une chaîne élémentaire qui relie x et y car le graphe est connexe et supposons qu'il existe une autre chaîne distincte de la précédente qui les relie. En concaténant les deux chaînes on obtient un cycle ce qui contredit le fait que le graphe est acyclique.□

Proposition 2.1.4 *Une arête $e \in E$ est un isthme de $G = (S, E)$ si et seulement si e n'appartient à aucun cycle de G*

Preuve : Soit $G = (S, E)$ un graphe connexe (cela suffit). Si $e = \{s, t\} \in E$ n'est pas un isthme alors il existe dans $G - \{e\}$ une chaîne μ reliant s et t les extrémités de e et donc en concaténant μ et e on obtient un cycle contenant e. Donc si e n'appartient à aucun cycle de G alors e est un isthme. Inversement supposons que e appartient à un cycle $c = (s_0, e, s_1, e_2, s_2, \ldots, s_{k-1}, e_k, s_0)$ et x et y deux sommets de $G - \{e\}$. Comme G est connexe il existe une chaîne $\sigma = \{x = t_0, f_1, t_1, f_2, \ldots, t_{k-1}, f_k, t_k = y\}$ qui relie x et y. Si σ est $G - \{e\}$ alors $G - \{e\}$ est connexe et e n'est pas un isthme. Sinon $e = f_i$ avec $i = 1, \ldots, k$ et posons $s_0 = t_{i-1}$ et $s_1 = t_i$. Il suffit de considérer la chaîne obtenue par la concaténation des trois chaînes suivantes : $\{x = t_0, f_1, \ldots, t_{i-1}\}$, $\{s_1, e_2, \ldots, e_k, s_0\}$ et $\{t_i, f_{i+1}, \ldots, f_k, t_k = y\}$. Elle est dans $G - \{e\}$ et donc $G - \{e\}$ est connexe. Donc si e est un isthme alors e n'appartient à aucun cycle de G. □

Corollaire 2.1.1 *Toute arête d'un arbre est un isthme.*

Théorème 2.1.1 *Soit G un graphe. Les assertions suivantes sont équivalentes :*

1. *G est un arbre.*
2. *G est acyclique et $m = n - 1$.*
3. *G est connexe et $m = n - 1$.*
4. *G est acyclique et si on ajoute une arête, on crée un cycle et un seul.*
5. *G est connexe et toute arête est un isthme.*
6. *Deux sommets quelconques de G sont reliés par une et une seule chaîne élémentaire.*

Preuve : (En exercice)

Proposition 2.1.5 *Une forêt d'ordre n et ayant p arbres possède $n - p$ arêtes.*

2.1.2 Problème de l'arbre couvrant

A) Position du problème

Soit $G = (S, E)$ un graphe simple. Un *arbre de G* ou encore *arbre couvrant de G* est un graphe partiel connexe et sans cycle de G. Une *forêt de G* ou encore *forêt couvrante de G* est un graphe partiel sans cycle de G (non nécessairement connexe).

Proposition 2.1.6 *Tout graphe connexe a un arbre couvrant.*

Preuve : Il suffit de supprimer toutes les arêtes qui ne sont pas des isthmes. □

Corollaire 2.1.2 *Si G est connexe alors $m \geq n-1$ et on a l'égalité si et seulement si G est un arbre.*

Preuve : G admet un arbre T couvrant, donc $m \geq m_T = n_T - 1 = n - 1$. comme T est un graphe partiel de G le cas de l'égalité n'est possible que lorsque $G = T$. □

Proposition 2.1.7 *Un graphe partiel d'un graphe connexe G est un arbre couvrant de G si et seulement s'il est connexe et minimal par rapport à la suppression d'arêtes.*

Preuve : Soit H un graphe partiel d'un graphe connexe G.
Si H est un arbre couvrant de G alors H est connexe et toute arête e de H y est un isthme donc $H - \{e\}$ n'est pas connexe. Réciproquement supposons que H est connexe et minimal par rapport à la suppression d'arêtes. Donc la suppression d'une arête quelconque e de H augmente le nombre de composante connexe de H et l'arête e est un isthme. Donc e n'appartient à aucun cycle et par conséquence H est acyclique et donc H est un arbre couvrant. □

Proposition 2.1.8 *Un graphe partiel d'un graphe connexe G est un arbre couvrant de G si et seulement s'il est acyclique et maximal par rapport à l'ajout d'arêtes.*

Preuve : Soit H un graphe partiel d'un graphe connexe G.
Si H est un arbre couvrant alors H est acyclique et soit e une arête quelconque de G n'appartenant pas à H. Il existe une chaîne $\mu = \{x, e_1, x_1, \ldots, x_{k-1}, e_k, y\}$ reliant les extrémités x et y de e. En concaténant μ et e on obtient le cycle $C = (x, e_1, x_1, \ldots, x_{k-1}, e_k, y, e, x)$; ce qui établit que la condition est nécessaire. Pour montrer qu'elle est suffisante supposons que H est acyclique et maximal par rapport à l'ajout d'arêtes. soient x et y deux sommets quelconques de H, il existe dans G une chaîne unique $\mu = \{x_0 = x, e_1, x_1, \ldots, x_{k-1}, e_k, x_k = y\}$. Si toutes les arêtes e_i avec $i = 1, 2, \ldots, k$ sont dans H alors H est connexe. Sinon on construit à partir de μ une chaîne dans H réliant x et y de la façon suivante : Pour toute arête e_i qui n'est pas dans H, il existe dans H une chaîne unique $\sigma_i = \{x_{i-1}, e_{i_1}, x_{i_1}, \ldots, e_{i_k}, x_i\}$ qui rélie x_{i-1} et x_i. On remplace x_{i-1}, e_i, x_i dans μ par σ_i. □

Définition 2.1.1 (Graphe valué) *Soit $G = (S, E)$ un graphe (orienté ou non). On associe à chaque arc (ou arête) $e \in E$ un réel $l(e)$ appelé poids ou longueur de e. L'application l ainsi définie :*

$$\begin{array}{rccc} l: & E & \to & \mathbb{R} \\ & e & \mapsto & l(e) \end{array}$$

est appelée une valuation du graphe G, on note $G = (S, E, l)$ et on dit que le graphe est valué. On l'appelle aussi réseau.

Définition 2.1.2 *Soit $G = (S; E)$ un graphe simple valué et $E' \subset E$. On appelle poids du graphe partiel engendré par E', le nombre : $c(E') = \sum_{e \in E'} c(e)$.*

Soit $G = (S, E, l)$ un graphe valué. Le problème de l'arbre couvrant minimum (resp. maximum) de G est la détermination d'un arbre couvrant H de G de poids le plus petit (resp. grand) possible. On présente ici deux algorithmes : celui de Kruskal et celui de Prim.

B) Méthode de Kruskal

La méthode de Kruskal construit un arbre couvrant de poids minimal en ajoutant à chaque itération une arête de plus petit poids n'ajoutant pas de cycle.

Algorithme de Kruskal

Initialisation $\mathcal{F} = E$ et $E_H = \emptyset$

Tant que $|E_H| < n - 1$ Faire

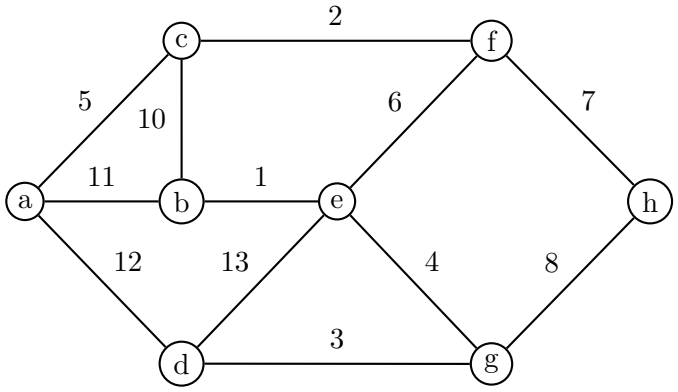
Sélectionner $e \in \mathcal{F}$ tel que $l(e)$ soit minimum.
$\mathcal{F} = \mathcal{F} - \{e\}$

Si $G[E_H \cup \{e\}]$ est acyclique alors $E_H = E_H \cup \{e\}$

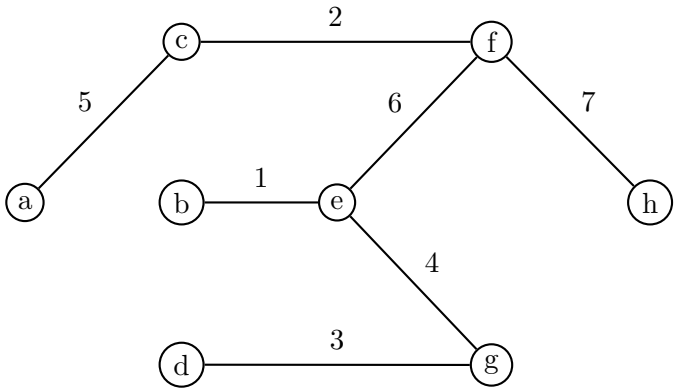
Fin Tant que

Exemple

On considère le graphe connexe suivant :



En appliquant Kruskal on obtient l'arbre couvrant minimum suivant :



Le poids de l'arbre est $1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$.
Une variante de l'algorithme de Kruskal est la suivante.

Variante de l'Algorithme de Kruskal

Initialisation $\mathcal{F} = E$ et $E_H = E$

Tant que $|E_H| > n - 1$ Faire

Sélectionner $e \in \mathcal{F}$ tel que $l(e)$ soit maximum.
$\mathcal{F} = \mathcal{F} - \{e\}$

Si $G[E_H - \{e\}]$ est connexe alors $E_H = E_H - \{e\}$

Fin Tant que

C) Méthode de Prim

Soit $G = (X, E, l)$ un graphe connexe valué. La méthode consiste à construire de proche en proche un arbre couvrant minimum $H = (X, U')$. Pour ce faire on fixe un sommet quelconque s_0. Dans cet algorithme T est l'ensemble des sommets visités et $\omega(T)$ est le cocycle de T.

On définit :

- $\omega^+(S)$ est l'ensemble des arcs ayant leur extrémité initiale dans S et leur extrémité terminale dans $X - S$.
- $\omega^-(S)$ est l'ensemble des arcs ayant leur extrémité initiale dans $X - S$ et leur extrémité terminale dans S.
- $\omega(S) = \omega^+(S) \cup \omega^-(S)$ est appelé un *cocycle* du graphe.

Algorithme

Initialisation $T = \{s_0\}$, $\theta = \omega(s_0)$ et $U' = \emptyset$;
Tant que $X - T \neq \emptyset$ Faire
Selectionner l'arc $u \in \theta$ tel que $l(u)$ soit minimum.
Soit s_i l'extrémité de u qui n'est pas dans T ;
$U' = U' \cup \{u\}$, $T = T \cup \{s_i\}$ et $\theta = \omega(T) = (\theta \cup \omega(s_i)) - (\theta \cap \omega(s_i))$
Fin Tant que

Une solution du problème de l'arbre couvrant de poids maximal peut s'obtenir en utilisant les mêmes algorithmes après avoir au préalable multiplié les poids par -1.

On peut aussi modifier très légèrement ces algorithmes pour obtenir un arbre couvrant de poids maximal. Il suffit de remplacer "minimum" par "maximum" dans les algorithmes ci-dessus. Mais dans la variante de Kruskal, il faut remplacer "maximum" par "minimum".

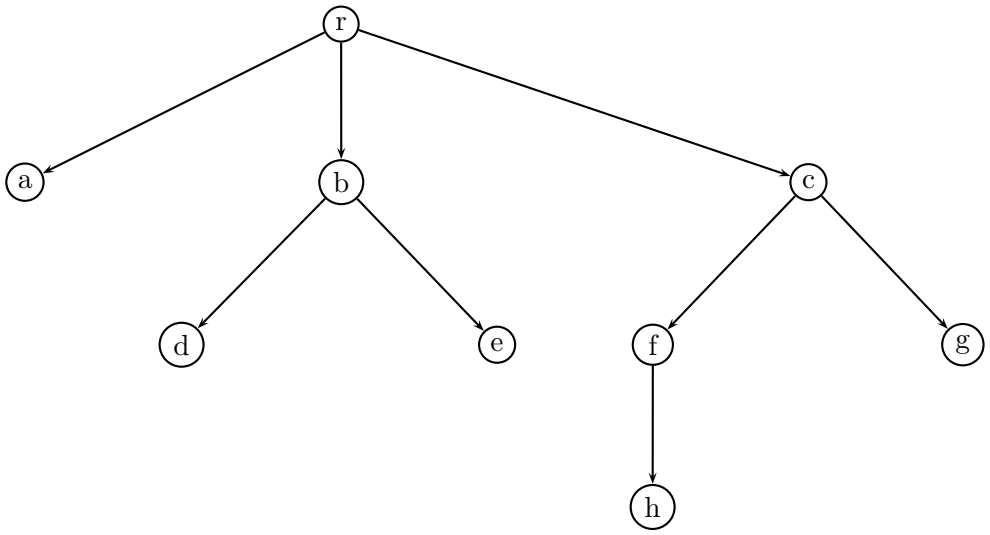
2.2 Arborescences

2.2.1 Définitions

Soient $G = (X, U)$ un graphe orienté et $r \in X$. On dit que r est une *racine* de G si tout sommet de G peut être atteint par un chemin d'origne r. Une *arborescence* est un graphe orienté à une seule racine et sans cycle.

On dit que G possède la *condition des demi-degrés intérieurs* lorsque $d^-(x) = 1$ pour tout sommet x de G sauf pour un sommet qu'on note r, pour lequel $d^-(r) = 0$.

Exemple



Les sommets puits a, d, e, h, g sont appelés *feuilles*. d et e sont *frères* et *fils* de b. les *descendants* de c sont f, g, h.

2.2.2 Caractérisation des arborescences

Proposition 2.2.1 *Soit $G = (X, U)$ un graphe orienté. Les conditions suivantes sont équivalentes*

1. *G est une arborescence.*
2. *G a une racine et est acyclique.*
3. *G a une racine et $m = n - 1$.*
4. *Il existe un sommet $r \in X$ tel que pour tout sommet $x \in X$, il existe un chemin unique allant de r à x.*
5. *G est connexe et possède la condition des demi-degrés intérieurs.*
6. *G est acyclique et possède la condition des demi-degrés intérieurs.*
7. *G est sans circuits et possède la condition des demi-degrés intérieurs.*

Preuve : Exercice. □

Chapitre 3

Chemins optimaux

3.1 Position du problème de cheminement

3.1.1 Plus court chemin et distance

Définition 3.1.1 *Soient $G = (S, A, l)$ un graphe valué et $\mu(s_0, s_k)$ un chemin allant de s_0 à s_k. On appelle longueur du chemin ou bien poids du chemin $\mu(s_0, s_k)$ et on note $l(\mu(s_0, s_k))$ la somme des poids des arcs de $\mu(s_0, s_k)$.*

Le problème du plus court chemin entre deux sommets x et y de $G = (S, A, l)$ est de déterminer un chemin μ entre x et y qui soit de longueur minimale.

De façon analogue le problème du plus long chemin entre x et y de $G = (S, A, l)$ est de déterminer un chemin μ entre x et y qui soit de longueur maximale.

Remarquer que si la valuation l est constante égale à 1 alors la notion définie ici coïncide avec la notion de longueur d'un chemin comme nombre de ses arcs. On parle de longueur au sens des arcs.

La recherche d'un plus long chemin est analogue à la recherche d'un plus court chemin. En effet, rechercher un plus long chemin entre deux sommets x et y dans $G = (S, A, l)$ est équivalent à la recherche du plus court chemin entre x et y dans $G = (S, A, -l)$. On peut donc sans perdre de généralités se restreindre au problème du plus court chemin.

Le problème de plus court chemin a de nombreuses applications pratiques car la longueur $l(u)$ d'un arc u peut s'interpréter comme un coût de transport sur l'arc, comme les dépenses de construction de l'arc, ou comme le temps nécessaire pour parcourir l'arc u.

Les algorithmes de résolutions du problème de plus court chemin sont différents selon les propriétés du graphe et le problème considéré.

Parmi les propriétés du graphe, on distingue les cas suivants :

1. on a $l(u) \geq 0$ pour tout $u \in A$,
2. on a $l(u) = 1$ pour tout $u \in A$,
3. $l(u)$ est quelconque,
4. le graphe G est sans circuit.

Les problèmes considérés sont :

1. Recherche d'un plus court chemin d'un sommet à un autre (*problème de 2 sommets*).
2. Recherche d'un plus court chemin d'un sommet à tous les autres (*problème avec un sommet origine unique*).
3. Recherche d'un plus court chemin entre tous les couples de sommets (*problème de tous les couples de sommets*).

On remarque que les problèmes 1) et 2) sont identiques. Dans la suite on ne s'intéressera qu'aux problèmes 2) et 3).

3.1.2 Condition d'optimalité

Définition 3.1.2 *Un circuit de longueur strictement négative est appelé circuit absorbant.*

Proposition 3.1.1 *Soient s et t deux sommets de G. Pour qu'il existe un plus court chemin allant de s à t il faut et il suffit que tout chemin allant de s à t ne contienne pas de circuit absorbant.*

Preuve : Soit $\mathcal{C}$ l'ensemble des chemins de s à t. Si $\mu \in \mathcal{C}$ contient un circuit ω alors on note μ' le chemin de s à t associé à μ mais n'empruntant pas le circuit ω et on a : $l(\mu) = l(\mu') + l(\omega)$.

- S'il existe un chemin $\mu \in \mathcal{C}$ contenant un circuit absorbant alors il n'existe pas de plus court chemin de s à t. Il suffit en effet de considérer le chemin obtenu à partir de μ en passant un assez grand nombre de fois par le circuit absorbant.
- Si tout chemin $\mu \in \mathcal{C}$ ne contient pas de circuit absorbant alors $l(\omega) \geq 0$, $l(\mu') \leq l(\mu)$ et on peut se restreindre aux chemins élémentaires. Comme ils forment un ensemble fini, il existera un plus court chemin. □

Soit $G = (S, A, l)$ un graphe orienté, valué et sans circuits absorbants. On appelle *distance* de s à t, deux sommets de G et on note $d(s, t)$ la plus petite longueur des chemins de s à t dans le graphe G.
Un *plus court chemin* de s à t est un chemin μ tel que $l(\mu) = d(s, t)$.

3.2 Quelques algorithmes

3.2.1 Cas où le graphe est non valué

On considère un graphe $G = (S, A)$ orienté non valué. On peut poser $G = (S, A, l)$ où l est constante égale à 1. On note r le sommet origine donc $d(r, r) = 0$. La distance est bien définie.

Calcul des distances

Le calcul des distances repose sur un algorithme de parcours en largeur du graphe G à partir de r. Soit F une file. On rappelle qu'une file fonctionne selon le principe du *premier entré, premier sorti*. Les opérations élémentaires sont.

- *enfiler(F,s)* : met s dans la file à la queue.
- *tête-file(F)* : retourne le sommet qui est en tête de F sans l'enlever.
- *défiler(F,s)* : enlève le sommet qui est en tête sans le retourner.
- *file-vide(F)* : retourne vrai ou faux selon que F est vide ou non.

Algorithme

a) Initialisation

Pour tout sommet $s \in S$
  Atteint(s) :=Faux
Fin Pour

b) Visite de r (sommet origine unique)

Atteint(r) :=Vrai
Enfiler(F,r)
$d(r, r) = 0$
$s := r$

c)

Tant que file-vide(F)=Faux

Pour tout successeur t de s qui n'est pas encore atteint
$d(r,t) = d(r,s) + 1$
Atteint(t) :=Vrai
Enfiler(F,t)

Fin Pour
Défiler(F,s) s :=tête-file(F)

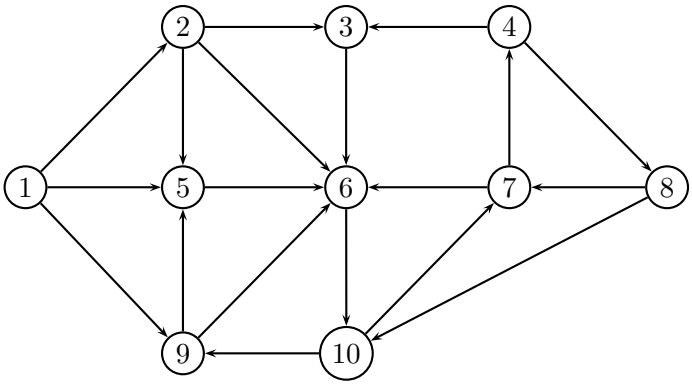
Fin Tant Que

Noter que Atteint( ) est un tableau indexé sur les sommets du graphe G.

Arborescence de plus courts chemins

L'ensemble de plus courts chemins déterminés définit dans le graphe une arborescence de plus courts chemins.

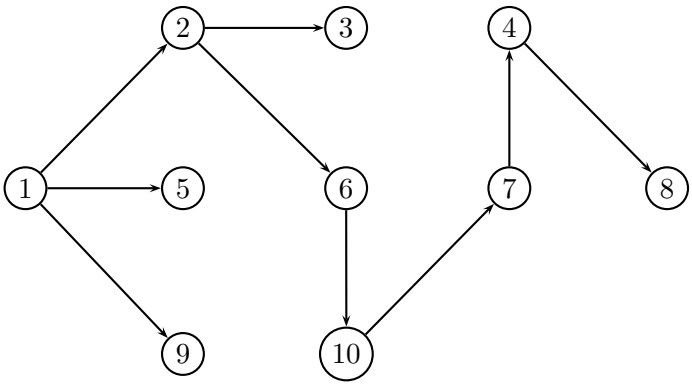
Exemple



L'algorithme permet de calculer les distances suivantes :

s	1	2	3	4	5	6	7	8	9	10
$d(r,s)$	0	1	2	5	1	2	4	6	1	3

Il existe plusieurs arborescences, en voici une.



3.2.2 Cas où les longueurs sont positives

Soit $G = (S, A, l)$ un graphe orienté strict où $S = \{s_1, s_2, \ldots, s_n\}$.

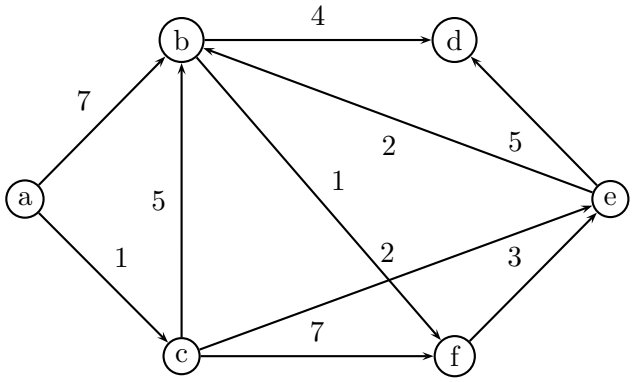
Algorithme de Dijkstra

1. **Initialisation**
   $S' = \{s_2, s_3, \ldots, s_n\}$
   $\lambda(s_1) = 0$
   $\lambda(s_i) = l(s_1, s_i)$ si s_i est un successeur de s_1 sinon $\lambda(s_i) = \infty$
2. **Sélection d'un sommet**
   Sélectionner $s_i \in S'$ tel que $\lambda(s_i) = \text{Min}_{s_k \in S'}(\lambda(s_k)$
   Faire $S' \leftarrow S' - \{s_i\}$
   Si S' est vide FIN
   Sinon aller en 3
3. **Calcul des valeurs de λ**
   Faire pour tout s_j à la fois dans $\Gamma(s_i)$ et dans S'
   $\lambda(s_j) \leftarrow \text{Min}(\lambda(s_j), \lambda(s_i) + l(s_i, s_j))$
   et retourner en 2

L'algorithme considère les sommets dans un ordre qui dépend des valeurs $\lambda(s_i)$ appelées labels. A la fin on a $\lambda(s_i) = d(s_1, s_i)$ pour tout $s_i \in S$.

Exemple d'application

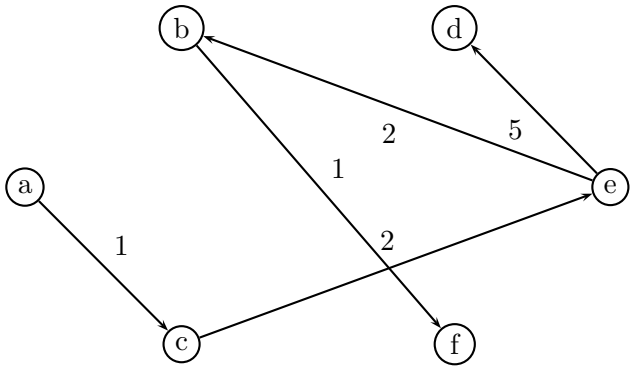
On considère le graphe suivant avec a comme sommet source :



On obtient alors :

Sommets	a	b	c	d	e	f
Initialisation	0	7	1*	∞	∞	∞
Itération 1	⋮	6	⋮	∞	3*	8
Itération 2	⋮	5*	⋮	8	⋮	8
Itération 3	⋮	⋮	⋮	8	⋮	6*
Itération 4	⋮	⋮	⋮	8*	⋮	⋮

On obtient une arborescence de parcours



3.2.3 Cas où les longueurs sont quelconques

Soit $G = (S, A, l)$ un graphe orienté strict où $S = \{s_1, s_2, \ldots, s_n\}$.

Algorithme de Bellman

1. **Initialisation**
   $\lambda^0(s_1) = 0$
   $\lambda^0(s_i) = \infty$ pour tout $s_i \neq s_1$
   $k = 1$
2. **A l'itération** k
   Faire $\lambda^k(s_1) = 0$
   et $\lambda^k(s_j) = \text{Min}(\lambda^{k-1}(s_j), \text{Min}_{s_i \in \Gamma^{-1}(s_j)}(\lambda^{k-1}(s_i) + l(s_i, s_j)))$ pour tout $s_j \neq s_1$
3. Si $\lambda^k(s_i) = \lambda^{k-1}(s_i)$ pour tout i alors FIN
   Si $k \leq n-1$ aller en 2 avec $k \leftarrow k+1$
   Si $k = n$ alors il existe un circuit absorbant

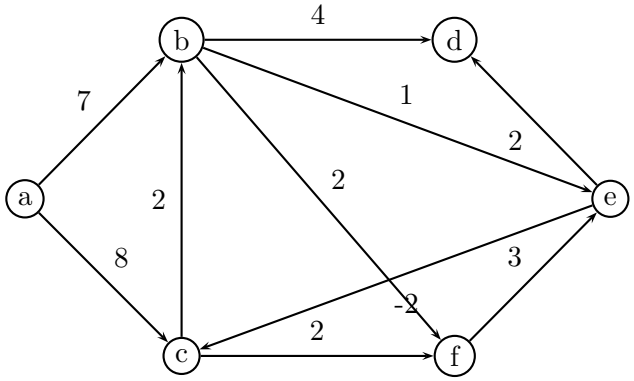
Remarque 3.2.1 *Dans l'algorithme de Bellman, on peut considérer à l'itération 3 la condition suivante :*

$$\lambda^k(s_j) = \text{Min}(\lambda^{k-1}(s_j), \text{Min}_{s_i \in \Gamma^{-1}(s_j)}(\lambda^p(s_i) + l(s_i, s_j)))$$

où p est inférieur à k est l'étape de la dernière mise à jour relative au sommet s_i.

Exemple d'application

On considère le graphe suivant avec a comme sommet source :



On obtient alors :

Sommets	a	b	c	d	e	f
Initialisation	0	∞	∞	∞	∞	∞
Itération 1	0	7	8	∞	∞	∞
Itération 2	0	7	8	11	8	9
Itération 3	0	7	6	10	8	9
Itération 4	0	7	6	10	8	8
Itération 5	0	7	6	10	8	8

On obtient une arborescence de parcours

3.2.4 Cas d'un graphe sans circuit

C) Algorithme de Ford

Soit $G = (S, A, l)$ un graphe orienté strict où $S = \{s_1, s_2, \ldots, s_n\}$. On suppose que $P(s_1) = \emptyset$.

1. **Initialisation** Poser $\lambda(s_1) = 0$ et $S' = \{s_1\}$
2. Rechercher un sommet $s_j \notin S'$ tel que $P(s_j) \subset S'$
3. Poser $\lambda(s_j) = \min[\lambda(s_i) + l(s_i, s_j) : s_i \in P(s_j)], S' := S' \cup \{s_j\}$
4. Si $|S'| = n$, FIN
5. Aller à 2

Exemple d'application

Appliquer l'algorithme de Ford aux graphes ci-dessus.

Chapitre 4

Problèmes d'ordonnancement

L'objet d'un problème d'ordonnancement est de faciliter la mise en œuvre et de guider l'exécution d'un ensemble complexe de tâches (programme de recherche ou de production, lancement d'un produit, construction d'un édifice ...). La technique d'analyse la plus connue, appelée méthode PERT (Program Evaluation and Review Technique) a été intriduite aux Etats-Unis en 1958 pour la conduite du programme de rechefche et de construction des fusées Polarix. Cette méthode tient une place dominante par sa simplicité, son efficacité et la variété d'extensions qui ont pu être développées.

En toute généralité, les problèmes d'ordonnancement se posent sous la forme suivnate. Etant donné un objectif qu'on se propose d'atteindre et dont la réalisation suppose l'exécution préalable de multiples tâcjes, soumises à de nombreuses contraintes, déterminer l'ordre et le calendrier d'exécution des diverses tâches.

Le critère d'optimalité peut porter sur le minimisation de la durée et/ou du coût de la réalisation du projet. Nous supposerons que le projet est décomposable en un nombre fini de tâches, caractérisées par leur durée d'exécution (généralement fixe, parfois aléatoire), éventuellement aussi par leur coût d'exécution, et soumises à des contraintes de postériorité stricte (une tâche ne peut commencer que si certaines autres tâches sont complètement terminées). Ce type de problème se nomme problème central de l'ordonnancement. Des cas plus complexes peuvent également être envisagés comme par exemple le cas des contraintes disjonctives, c'est-à-dire le cas où, en plus des contraintes de succession, on impose la réalisation non simultanée de certaines paires de tâches.

En pratique, le travail préliminaire à accomplir sera donc de dresser une liste des différentes opérations à mener que l'on décomposera plus ou moins finement selon la précision souhaitée. Généralement, les tâches sont définies pour que leurs durées d'exécution soient du même ordre de grandeur ; de plus, les contraintes de postériorité entre les tâches doivent pouvoir être établies avec précision. Ce travail important est souvent long et nécessite une collaboration étroite entre les différents acteurs du projet.

La représentation par un graphe d'un problème d'ordonnancement permet une bonne appréhension globale du problème. L'étude de ce graphe conduit à l'identification des tâches prioritaires et la détection des retards ou des dépassements de moyens afin de prendre les mesures correctives nécessaires.

La schématisation d'un problème d'ordonnancement sous forme d'un graphe peut se faire selon deux modes de repprésentation :

- la méthode MPM (Méthode des potentiels Metra) développée en France par Bernard Roy de la SEMA à l'occasion de la construction d'une centrale nucléaire.

- la méthode PERT (Program Evaluation and Review Technique) dévéloppée aux USA à l'époque où la marine américaine devait réaliser dans les meilleurs délais le système d'arme Polaris (missiles à longue portée embarqués dans des sous-marins et dotés d'une ogive nucléaire).

Soit un projet décomposé en n tâches élémentaires $1, 2, \ldots, n$. Pour chaque tâche i, il est

donné sa durée d_i et les tâches antérieures c'est-à-dire les tâches qui doivent être achévées pour que la tâche i puisse commencer.

Tâches	Durées	Tâches antérieures
$\vdots$	$\vdots$	$\vdots$
i	d_i	j, k
$\vdots$	$\vdots$	$\vdots$

En ne prenant en compte que les contraintes de succession, il s'agit de trouver un calendrier de déroulement des tâches du projet, c'est-à-dire déterminer les dates de démarrage de chaque tâche, qui minimisent le temps total de réalisation du projet et d'indiquer les tâches critiques.

4.1 Méthode MPM

La méthode MPM consiste à représenter ce problème d'ordonnancement de projet par un graphe dit graphe potentiel-tâches ou graphe MPM ou réseau MPM.

4.1.1 Principe de la représentation du réseau MPM

Le graphe potentiel-tâches associé au projet est un graphe orienté valué strict et sans circuits, $G = (S, A, l)$ avec $S = \{\alpha, 1, 2, \ldots, n, \omega\}$ où α désigne le début et ω la fin du projet, considérés comme des tâches fictives de durée nulle. $(i, j) \in A$ si la tâche i est antérieure à la tâche j. La longueur ou le poids de l'arc (i, j) est $l(i, j) = d_i$, c'est-à-dire la durée de la tâche i.

4.1.2 Construction du réseau MPM

On considère le tableau qui donne la liste des opérations préréquises c'est-à-dire le dictionnaire des précédents. On détermine les niveaux des sommets.

Pour construire le graphe, on place d'abord les niveaux (qui sont schématisés par des traits verticaux) par ordre croissant de gauche à droite. Les sommets sont placés par niveau. A un même niveau, ils sont situés de façon à limiter le nombre d'intersections entre les arcs.

Un arc reliera un sommet i à un sommet j, si $i \in P(j)$. Chaque sommet sera figuré par un rectangle du type :

T_i	T_i^*
i	

où i désigne le nom de la tâche, T_i la date de début au plus tôt de la tâche et T_i^* la date de début au plus tard de la tâche.

La tâche début des travaux de durée 0 sera reliée à tous les sommets sans précédents et la tâche de fin des travaux reliée à tous les sommets qui n'ont pas de suivants.

4.1.3 Détermination des dates

La date au plus tôt T_j d'une tâche j est la date la plus rapprochée (optimiste) à laquelle j peut commencer. C'est la longueur du plus long chemin conduisant α à j. On montre que :

$$\begin{cases} T_\alpha = 0 \\ T_j = \text{Max}_{i \in P(j)}(T_i + d_i) \quad \text{pour } j \in \{1, \ldots, n, \omega\} \end{cases}$$

et que T_ω est la durée minimum du projet.

Ces valeurs sont calculeés en prenant les sommets par niveau croissant.

La date au plus tard T_i^* d'une tâche i est la date la plus tardive (pessimiste) à laquelle i doit commencer pour que la durée minimale de réalisation du projet soit respectée : c'est-à-dire $T_\omega^* = T_\omega$. Les dates au plus tard se calculent par un compte à rebours à partir de T_ω^* et

$$T_i^* = \text{Min}_{j \in S(i)}(T_j^* - d_i) \quad \text{pour } i = n, n-1, \ldots, 2, 1, \alpha$$

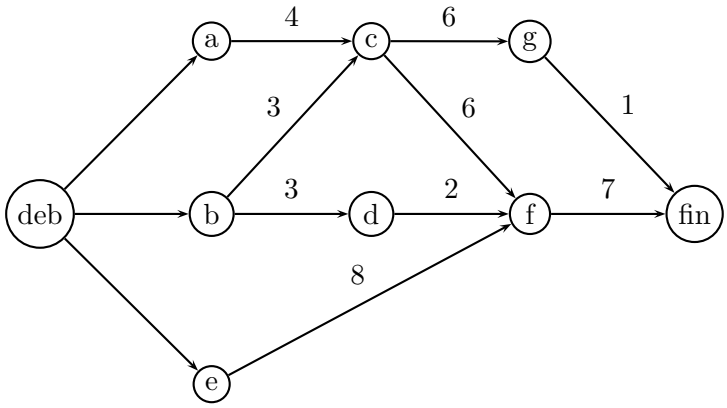
La date de fin au plus tard T_i' d'une tâche i est la date la plus pessimiste de fin d'une tâche et se calcule par la formule suivant :

$$T_i' = \text{Min}_{j \in S(i)}(T_j - d_i) \quad \text{pour } i = n, n-1, \ldots, 2, 1, \alpha$$

Exemple : On considère un projet qui comporte 7 tâches : a, b, c, d, e, f, g.

Tâches	a	b	c	d	e	f	g
Durées	4	3	6	2	8	7	1
Tâches antérieures			a,b	b		c,d,e	c

On obtient le graphe potentiel-tâches $G = (S, A, l)$ ci-dessous, où $S = \{deb, a, b, c, d, e, f, g, fin\}$



Tâches	deb	a	b	c	d	e	f	g	fin
Durées	0	4	3	6	2	8	7	1	0
T_i	0	0	0	4	3	0	10	10	17
T_i^*	0	0	1	4	8	2	10	16	17

4.1.4 Les marges

La marge totale est le délai $MT(i) = T_i^* - T_i$ pouvant être accordé à une tâche sans repousser la durée minimale du projet. On dit qu'une tâche i est *critique* si $MT(i) = 0$. La succession de tâches qui imposent la durée minimale du projet est appelée *chemin critique*.
La marge libre d'une tâche i est le délai $ML(i)$ pouvant être accordé au commencement de la tâche sans modifier la marge totale des tâches qui suivent.

$$ML(i) = \text{Min}_{j \in \Gamma(i)}(T_j - d_i) - T_i$$

ou encore

$$ML(i) = T_i' - T_i$$

. Contrairement à la marge totale, la marge libre peut être consommée sans aucune conséquence sur les tâches qui suivent dans le projet. La marge libre d'une tâche est toujours inférieure ou égale à sa marge totale.

4.2 Méthode PERT

La méthode PERT consiste à représenter le problème d'ordonnancement du projet par un graphe dit graphe potentiel-étapes ou graphe PERT ou encore réseau PERT.

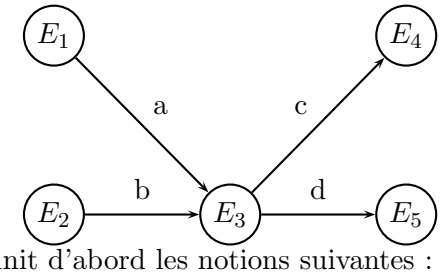
4.2.1 Principe de la représentation du graphe PERT

A chaque tâche correspond un arc du graphe dont la longueur est égale à la durée de la tâche.

Chaque sommet du graphe est un évènement (ou étape) signifiant :

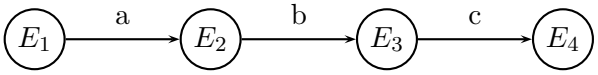
$$\begin{cases} \text{toutes les tâches qui arrivent sont terminées} \\ \text{toutes les tâches qui partent peuvent commencer} \end{cases}$$

Le schéma ci-dessous signifie par exemple : a et b doivent être terminées pour que c et d puissent commencer.

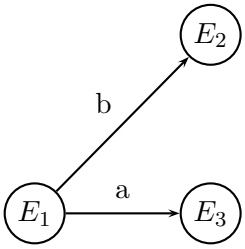


On définit d'abord les notions suivantes :

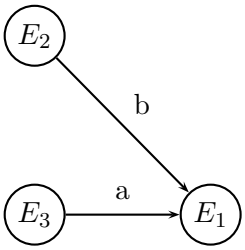
Les tâches successives se suivent dans le temps et sont représentées par un chemin. Par exemple les tâches a, b, c sont successives :



Les tâches simultanées ont le même début d'exécution. Par exemple les tâches a, b sont simultanées :



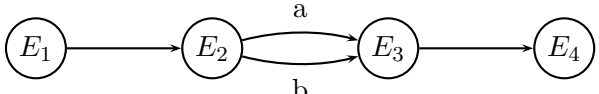
Les tâches convergentes aboutissent à la même étape. Par exemple a, b sont convergentes :



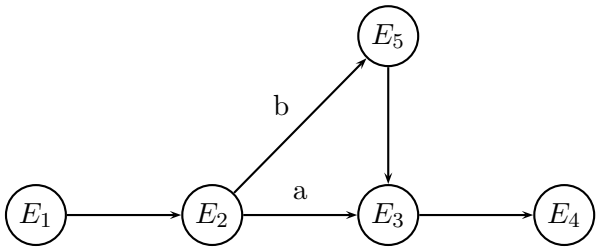
Les tâches parallèles sont à la fois simultanées et convergentes.

La représentation de certaines relations d'antriorité nécessite dans la méthode PERT l'introduction de tâches fictives de durée 0.

Dans un premier temps, il y a la gestion des tâches parallèles qui conduit à un multigraphe comme l'indique le schéma ci-dessous :

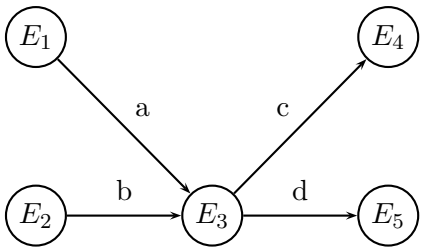


Pour éviter cela on introduit une tâche fictive entre E_5 et E_3. Ce qui donne la représentation suivante :

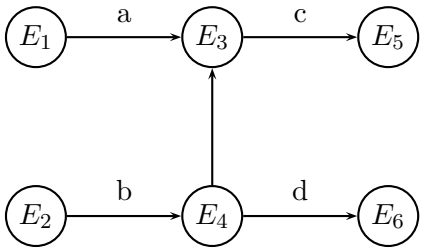


Il y a aussi certaines représentations qui peuvent impliquer des conditions qui ne sont pas prévues dans les données.

Par exemple la situation où a et b sont antérieures à c et b est antérieure à d ne peut être représentée par le schéma ci-dessous :



car cela impliquerait que a précède d, ce qui n'est pas une donnée du problème. On considère la représentation suivante :



4.2.2 Construction du graphe PERT

Chaque sommet sera figuré par un cercle divisé en deux parties. Dans la partie inférieure on indique le numéro de l'étape ou de l'évènement E_i (ce n'est pas une donnée initiale du problème mais résulte de la construction du graphe). La partie supérieure est divisée en deux ; à droite on note t_i qui désigne la date attendue de réalisation de l'évènement E_i et à gauche on note t_i^* qui désigne la date limite de réalisation de l'évènement E_i.

On considère tout d'abord un sommet initial E_0 d'où partent toutes les tâches dont la mise en route n'est soumise à aucune contrainte d'antériorité et à la fin un sommet final E_N auquel aboutissent toutes les tâches n'ayant pas de suivants.

En pratique on trace d'abord un graphe des niveaux en reliant les tâches successives entre deux niveaux consécutifs. Puis on trace le graphe PERT à partir de ce graphe de manière à ce que toutes les tâches aboutissent à une étape et que l'antériorité des tâches soit respectée.

4.2.3 Détermination des dates

Ordonnancement minimum ou au plus tôt

Définition 4.2.1 *On appelle date au plus tôt de réalisation de l'évènement E_j, est la longueur du plus long chemin dans le graphe reliant E_1 à E_j. C'est aussi la date attendue de réalisation de l'évènement E_j. On la note t_j.*

Définition 4.2.2 *On appelle ordonnancement minimum ou au plus tôt, l'ensemble de toutes les dates attendues t_i, $i = 1 \cdots N$.*

Pratiquement on utilise les formules suivantes pour calculer les t_j.

$$\begin{cases} t_0 = 0 \\ t_j = \text{Max}_{i \in P(j)}(t_i + d_{ij}) \quad \text{pour } j \in \{1, \ldots, N\} \end{cases}$$

où d_{ij} est la durée de la tâche (E_i, E_j).

Définition 4.2.3 *Pour toute tâche (E_i, E_j), la date de début au plus tôt est $T_{ij} = t_i$.*

Les dates atendues t_i étant connues, on détermine le (ou les) chemin(s) critique(s) constitués des arcs (E_i, E_j) pour lesquels $t_j - t_i = d_{ij}$.

Les évènements E_i situés sur le chemin critique sont dits évènements critiques.

Remarque 4.2.1 *Tout retard apporté sur la réalisation d'un évènement critique ralentit le programme entier de la même durée.*

Ordonnancement limite (ou au plus tard)

L'objectif étant de réaliser l'ensemble du programme en un temps minimum, on impose à l'évènement E_N, fin des travaux d'être réalisé à sa date attendue t_N. On définit alors :

Définition 4.2.4 *On appelle date limite (ou plus tard) de réalisation de l'évènement E_i, $t_i^* = t_N - \tau_{i,n}$ où $\tau_{i,n}$ est la longueur d'un plus long chemin dans le graphe reliant E_i à E_N.*

Définition 4.2.5 *On appelle ordonnancement limite ou au plus tard, l'ensemble de toutes les dates limites t_i^*, $i = 1 \cdots N$.*

Pratiquement on utilise les formules suivantes pour calculer les t_i^*.

$$\begin{cases} t_N^* = t_N \\ t_i^* = \min_{j \in S(i)}(t_j^* - d_{ij}) \quad \text{pour } j \in \{1, \ldots, N-1\} \end{cases}$$

Lorsque les dates limites de réalisation sont déterminées, les évènements critiques sont ceux tels que $t_i = t_i^*$.

Définition 4.2.6 *Pour toute tâche (E_i, E_j), la date de début au plus tard est $T_{ij}^* = t_j^* - d_{ij}$.*

Définition 4.2.7 *On appelle intervalle de flottement de l'évènement E_i, l'intervalle $[t_i, t_i^*]$.*

C'est l'intervalle dans lequel peut intervenir la réalisation de E_i sans compromettre la durée totale (minimale) d'exécution de l'ensemble du programme. Un tel intervalle est de longueur nulle pour un évènement critique.

4.2.4 Les marges

Définition 4.2.8 *La marge libre d'une tâche (E_i, E_j) est*

$$ML(i, j) = t_j - t_i - d_{ij}.$$

Elle représente le delai ou retard maximum que l'on peut apporter à sa mise en route sans perturber la date attendue de réalisation de l'évènement E_j, l'évènement E_i ayant eu lieu à sa date attendue.

Remarque 4.2.2 *La marge libre d'une tâche critique est nulle.*

Définition 4.2.9 *La marge totale d'une tâche* (E_i, E_j) *est*

$$MT(i,j) = t_j^* - t_i - d_{ij}.$$

Elle représente pour cette opération, le delai ou retard maximum que l'on peut apporter à sa mise en route lorsque l'évènement E_i a eu lieu à sa date attendue sans perturber la date de fin des travaux.

Remarque 4.2.3 *Pour les opérations critiques* (E_i, E_j)*, on a* $MT(i,j) = 0$*.*

On a toujours

$$0 \leq ML(i,j) \leq MT(i,j).$$

Chapitre 5

Réseaux de transport et Flots

5.1 Introduction

Ce chapitre présente des modèles et approches mathématiques sur les problèmes de flots.

Imaginer un réseau tel le réseau de distribution d'eau, ou des eaux usées ou de gaz naturel dans une grande ville, le réseau électrique, de télécom, routier, de métro, de chemin de fer ou le réseau des circuits intégrés d'ordinateur. Dans tous ces domaines, systèmes hydrauliques, systèmes de transport, systèmes électriques, systèmes électroniques, systèmes de communication, se trouvent des problèmes de flots de matériel- fluides, trains, courants, messages d'informations transmis d'un point à un autre, en traversant un lien d'une capacité limitée, qui représente seulement un élément d'un vaste ensemble.

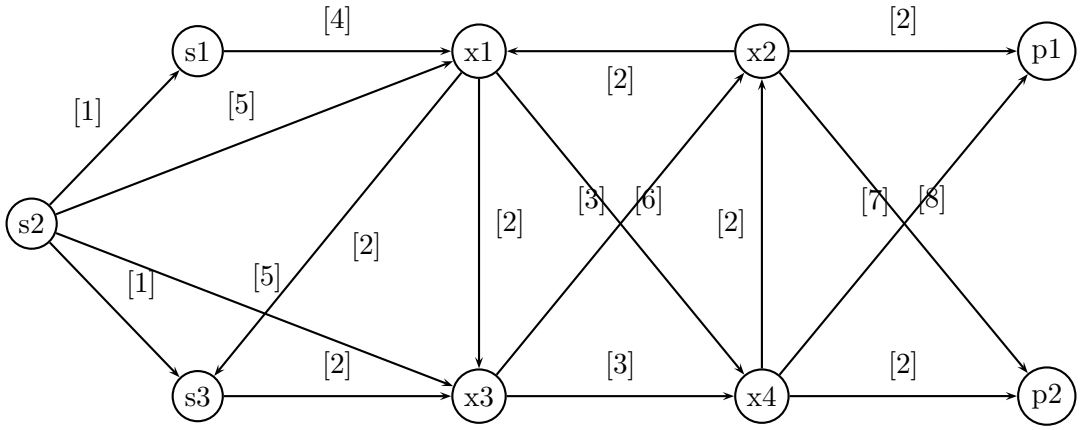
Le problème qui nous intéresse ici est de maximiser le flot dans un réseau de capacité limitée, ou bien de minimiser le coût total d'un flot dans un réseau quand chaque unité de flot dans un lien engendre une dépense.

5.2 Flot dans un réseau de transport

5.2.1 Réseau de transport

Un réseau de transport est un graphe $R = (X, U, S, P, c)$ valué positivement sans boucle ayant deux ensembles disjoints $S \subset X$, ensemble des *entrées* (ou *sources*) et $P \subset X$ l'ensemble des *sorties* (ou *puits*). Les valuations des arcs sont appelées capacités. On verra plus loin qu'on peut, sans perdre en généralité, se ramener à un réseau à une entrée et sortie uniques.
On rappelle que pour tout $Z \subset X$ on note $\omega^+(Z)$ (resp. $\omega^-(Z)$) l'ensemble des arcs sortant (entrant) de Z c'est-à-dire les arcs ayant l'extrémité initiale dans Z (resp. $X - Z$) et l'extrémité finale dans $X - Z$ (resp. Z).

Exemple



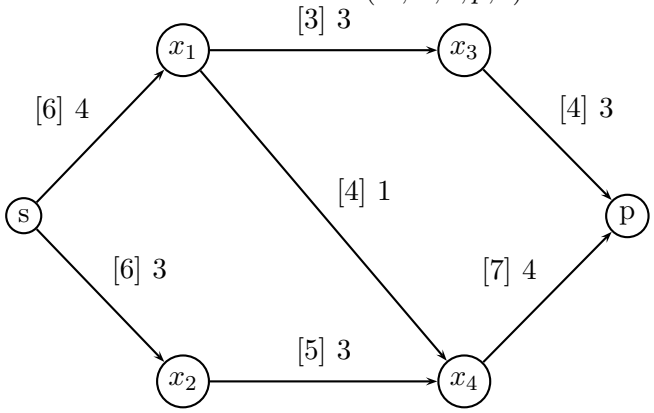
5.2.2 Flot

Un *flot* sur un réseau de transport $R = (X, U, S, P, c)$ est une application $f : U \to \mathbb{N}$ qui vérifie :

1. $0 \leq f(u) \leq c(u)$ pour tout $u \in U$
2. $\sum_{u \in \omega^+(\{x\})} f(u) = \sum_{u \in \omega^-(\{x\})} f(u)$ pour tout $x \in X \setminus (S \cup P)$ (on dit que le flot est conservatif en x).
3. $\sum_{u \in \omega^+(\{s\})} f(u) - \sum_{u \in \omega^-(\{s\})} f(u) \geq 0 \ \forall s \in S$
4. $\sum_{u \in \omega^-(\{p\})} f(u) - \sum_{u \in \omega^+(\{p\})} f(u) \geq 0 \ \forall p \in P$.

Exemple 5.2.1

On considère sur le réseau $R = (X, U, s, p, c)$ ci-dessous le flot suivant :



les valeurs entre les crchets sont les capacités des arcs et les valeurs à côté sont les flux sur les arcs.

Si on pose d'une manière générale pour tout $Y \subset X$,

$$d_f(Y) = \sum_{u \in \omega^+(Y)} f(u) - \sum_{u \in \omega^-(Y)} f(u),$$

c'est la différence entre la quantité de flot qui sort de Y et la quantité de flot qui entre en Y.

Pour $Y = \{x\}$ un singleton on notera :

$$d_f(\{x\}) = d_f(x).$$

On a alors la propriété suivante :

Proposition 5.2.1 *Si X_1 et X_2 sont deux sous-ensembles propres de X disjoints, on a*

$$d_f(X_1 \cup X_2) = d_f(X_1) + d_f(X_2).$$

Preuve :

On considère la notation suivante :

$$\sigma(V) = \sum_{u \in V} f(u) \quad \forall V \subset U.$$

On sait que $\omega^+(X_1)$, l'ensemble des arcs qui sortent de X_1 est égal à la réunion de l'ensemble $(\omega^+(X_1) \setminus \omega^-(X_2))$ des arcs qui sortent de X_1 et qui n'entrent pas dans X_2 et de l'ensemble $(\omega^+(X_1) \cap \omega^-(X_2))$ des arcs qui sortent de X_1 et qui entrent dans X_2. C'est-à-dire :

$$\omega^+(X_1) = (\omega^+(X_1) \setminus \omega^-(X_2)) \cup (\omega^+(X_1) \cap \omega^-(X_2)).$$

De même on a :

$$\omega^+(X_2) = (\omega^+(X_2) \setminus \omega^-(X_1)) \cup (\omega^+(X_2) \cap \omega^-(X_1)),$$

$$\omega^-(X_1) = (\omega^-(X_1) \setminus \omega^+(X_2)) \cup (\omega^-(X_1) \cap \omega^+(X_2)),$$

$$\omega^-(X_2) = (\omega^-(X_2) \setminus \omega^+(X_1)) \cup (\omega^-(X_2) \cap \omega^+(X_1)).$$

De façon analogue, l'ensemble $\omega^+(X_1 \cup X_2)$ des arcs qui sortent de $X_1 \cup X_2$ est égal à la réunion de l'ensemble $(\omega^+(X_1) \setminus \omega^-(X_2))$ des arcs qui sortent de X_1 et qui n'entrent pas dans X_2 et de l'ensemble des arcs qui sortent de X_2 et qui n'entrent pas dans X_1 : soit

$$\omega^+(X_1 \cup X_2) = (\omega^+(X_1) \setminus \omega^-(X_2)) \cup (\omega^+(X_2) \setminus \omega^-(X_1)).$$

De même on a :

$$\omega^-(X_1 \cup X_2) = (\omega^-(X_1) \setminus \omega^+(X_2)) \cup (\omega^-(X_2) \setminus \omega^+(X_1)).$$

Ainsi on obtient :

$$\begin{aligned} d_f(X_1) &= \sigma[(\omega^+(X_1) \setminus \omega^-(X_2)) \cup (\omega^+(X_1) \cap \omega^-(X_2))] \\ &\quad -\sigma[(\omega^-(X_1) \setminus \omega^+(X_2)) \cup (\omega^-(X_1) \cap \omega^+(X_2))] \\ &= [\sigma(\omega^+(X_1) \setminus \omega^-(X_2)) + \sigma(\omega^+(X_1) \cap \omega^-(X_2))] \\ &\quad -[\sigma(\omega^-(X_1) \setminus \omega^+(X_2)) + \sigma(\omega^-(X_1) \cap \omega^+(X_2))] \end{aligned}$$

car les ensembles $(\omega^+(X_1) \setminus \omega^-(X_2))$ et $(\omega^+(X_1) \cap \omega^-(X_2))$ sont disjoints ainsi que les ensembles $(\omega^-(X_1) \setminus \omega^+(X_2))$ et $(\omega^-(X_1) \cap \omega^+(X_2))$

de même on a :

$$\begin{aligned} d_f(X_2) &= \sigma[(\omega^+(X_2) \setminus \omega^-(X_1)) \cup (\omega^+(X_2) \cap \omega^-(X_1))] \\ &\quad -\sigma[(\omega^-(X_2) \setminus \omega^+(X_1)) \cup (\omega^-(X_2) \cap \omega^+(X_1))] \\ &= [\sigma(\omega^+(X_2) \setminus \omega^-(X_1)) + \sigma(\omega^+(X_2) \cap \omega^-(X_1))] \\ &\quad -[\sigma(\omega^-(X_2) \setminus \omega^+(X_1)) + \sigma(\omega^-(X_2) \cap \omega^+(X_1))] \end{aligned}$$

Donc

$$\begin{aligned} d_f(X_1) + d_f(X_2) &= \sigma(\omega^+(X_1) \setminus \omega^-(X_2)) + \sigma(\omega^+(X_1) \cap \omega^-(X_2)) \\ &\quad -\sigma(\omega^-(X_1) \setminus \omega^+(X_2)) - \sigma(\omega^-(X_1) \cap \omega^+(X_2)) \\ &\quad +\sigma(\omega^+(X_2) \setminus \omega^-(X_1)) + \sigma(\omega^+(X_2) \cap \omega^-(X_1)) \\ &\quad -\sigma(\omega^-(X_2) \setminus \omega^+(X_1)) - \sigma(\omega^-(X_2) \cap \omega^+(X_1)) \\ &= [\sigma(\omega^+(X_1) \setminus \omega^-(X_2)) + \sigma(\omega^+(X_2) \setminus \omega^-(X_1))] \\ &\quad - [\sigma(\omega^-(X_1) \setminus \omega^+(X_2)) + \sigma(\omega^-(X_2) \setminus \omega^+(X_1))] \\ &= \sigma\left[(\omega^+(X_1) \setminus \omega^-(X_2)) \cup (\omega^+(X_2) \setminus \omega^-(X_1))\right] \\ &\quad -\sigma\left[(\omega^-(X_1) \setminus \omega^+(X_2)) \cup (\omega^-(X_2) \setminus \omega^+(X_1))\right] \\ &= \sigma(\omega^+(X_1 \cup X_2)) - \sigma(\omega^-(X_1 \cup X_2)) \\ &= d_f(X_1 \cup X_2) \end{aligned}$$

D'où la proposition. □

Comme $\omega^+(X) = \omega^-(X) = \emptyset$, on a $d_f(X) = 0$.
On déduit aussi les conditions ci-dessous.

Corollaire 5.2.1 *Pour tout $Y \subset X$, on a $d_f(Y) = \sum_{y \in Y} d_f(y)$.*
$d_f(x) = 0$ pour tout $x \in X - (S \cup P)$.
$d_f(Y) = 0$ pour tout $Y \subset X$ avec $Y \cap (S \cup P) = \emptyset$.
$d_f(s) \geq 0$ pour tout $s \in S$.
$d_f(p) \leq 0$ pour tout $p \in P$.
On a : $d_f(S) \geq 0$ et $-d_f(P) \geq 0$.
On a : $d_f(S) = -d_f(P)$.
Si (X_1, X_2) est une partition de X telle que $S \subset X_1$ et $P \subset X_2$, on a $d_f(X_1) = d_f(S)$.

Définition 5.2.1 *On appelle valeur d'un flot f sur $R = (X, U, S, P, c)$, le nombre*

$$v(f) = d_f(S).$$

Proposition 5.2.2 *L'étude d'un flot dans un réseau de transport quelconque peut toujours se ramener à l'étude d'un flot dans un réseau de transport à entrée et sortie uniques.*

Preuve : Soit $R = (X, U, S, P, c)$ un réseau et posons $S = \{s_1, s_2, ..., s_r\}$ et $P = \{p_1, p_2, ..., p_t\}$. A ce réseau, on ajoute un sommet s avec les arcs (s, s_i) $(i = 1, 2, \ldots, r)$ de capacité ∞ et un sommet p avec les arcs (p_j, p) $(j = 1, 2, \ldots, t)$ de capacité ∞. On obtient ainsi un nouveau réseau R' à entrée et sortie unique et on pose $S' = \{s\}$ et $P' = \{p\}$. Il reste à montrer que tout flot f sur R correspond à un flot sur R' de même valeur et inversement.
Soit donc f un flot sur R. On définit f' comme suit : $f'/U = f$ et

$$f'((s, s_i)) = d_f(s_i) \quad \text{pour} \quad i = 1, 2, \ldots, r$$

et

$$f'((p_j, p)) = -d_f(p_j) \quad \text{pour} \quad j = 1, 2, \ldots, t$$

On vérifie (en exercice) facilement que f' est bien un flot sur R' et que $v(f) = v(f')$.
Inversement, étant donné un flot sur R' on en déduit directement un flot sur R de même valeur en considérant sa restriction sur les arcs de R. □

Dans toute la suite du chapitre, on suppose que les réseaux considérés sont à une entrée et sortie uniques notés respectivement s et p. Le réseau est alors noté $R = (X, U, s, p, c)$.

5.3 Flot maximum et coupe minimum

5.3.1 Définitions et propriétés

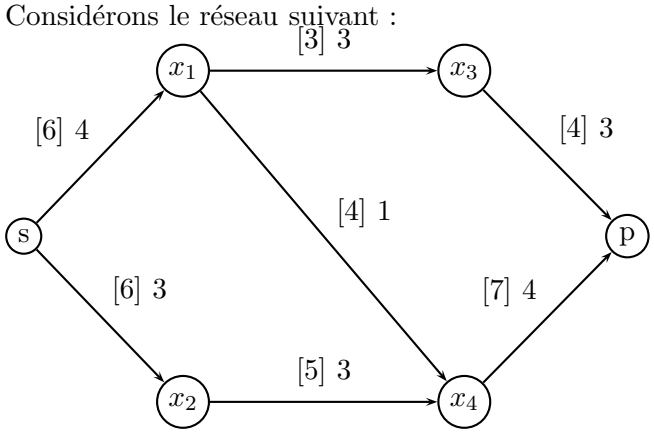
Soit $R = (X, U, s, p, c)$ un réseau. Une *coupe* de R est un ensemble de sommets $K \subset X$ tel que $s \in K$ et $p \notin K$. La *capacité* d'une coupe K est la somme des capacités des arcs sortant de K :

$$c(K) = \sum_{a \in \omega^+(K)} c(a).$$

Un *flot maximum* est un flot de valeur la plus grande possible. De même une *coupe minimum* est une coupe de capacité la plus petite possible.

Exemple 5.3.1

Considérons le réseau suivant :



L'ensemble $C = \{s, x_1, x_3\}$ est une coupe. Sa capacité est : $c(C) = 4 + 4 + 6 = 14$.

On montre que :

Proposition 5.3.1 *Pour tout flot f et toute coupe K sur un réseau R on a : $v(f) \leq c(K)$*

Preuve : On sait que : $s \in K$ et $p \notin K$. Donc $d_f(K) = d_f(s) = v(f)$. Et comme $f(u) \leq c(u)$ pour tout arc u, on a :

$$\begin{aligned} v(f) = d_f(s) = d_f(K) &= \textstyle\sum_{u \in \omega^+(K)} f(u) - \sum_{u \in \omega^-(K)} f(u) \\ &\leq \textstyle\sum_{u \in \omega^+(K)} f(u) \\ &\leq \textstyle\sum_{u \in \omega^+(K)} c(u) = c(K). \end{aligned}$$

D'où la proposition. □

Il est immédiat que :

Corollaire 5.3.1 *Pour tout flot f et toute coupe K sur un réseau R si $v(f) = c(K)$ alors f est un flot maximum et K est une coupe minimum.*

5.3.2 Chaîne augmentante ou améliorante

Soit $R = (X, U, s, p, c)$ un réseau. Un arc est *saturé* si le flux qui le traverse est égal à sa capacité. Un chemin ou une chaîne est *saturé(e)* s'il ou elle contient un arc saturé. Un chemin insaturé allant de s à p est dit *chemin augmentant ou améliorant*. Une *chaîne augmentante ou améliorante ou amplificatrice* est une chaîne de s à p dont les arcs parcourus dans le sens direct sont insaturés et ceux parcourus dans le sens inverse sont traversés par des flux strictement positifs.

Proposition 5.3.2 *S'il existe pour un flot f sur un réseau R une chaîne augmentante alors il existe un flot f' tel que $v(f) < v(f')$.*

Preuve : Soit $\mathcal{C} = \{u_1, \ldots, u_l\}$ une chaîne augmentante de s à p. On note $\mathcal{C}^+$ l'ensemble des arcs dits directs de $\mathcal{C}$ parcourus dans le sens de s à p et $\mathcal{C}^-$ celui des arcs dits arcs inverses de $\mathcal{C}$ parcourus dans le sens inverse.

On pose :

$$\tau(\mu) = \mathrm{Min}_{i=1,2,\ldots,k}\, \tau(u_i)$$

où $\tau(u_i) = c(u_i) - f(u_i)$ si $u_i \in \mu^+$ et $\tau(u_i) = f(u_i) > 0$ lorsque $u_i \in \mu^-$.

On définit l'application f' sur U par :

$$f'(u) = \begin{cases} f(u) + \tau(\mu) & \text{si } u \in \mu^+ \\ f(u) - \tau(\mu) & \text{si } u \in \mu^- \\ f(u) & \text{si } u \notin \mu \end{cases}$$

Pour tout arc u, on $f'(u) \geq 0$ car soit $f'(u) = f(u) \geq 0$ soit $f'(u) = f(u) + \delta > 0$, soit enfin $f'(u) = f(u) - \delta$. Ce dernier cas ne peut se produire que pour les arcs inverses de $\mathcal{C}$, et dans ce cas on a $f(u) \geq \delta_2 \geq \delta$. Donc on a $f'(u) = f(u) - \delta \geq 0$.

De même on a pour chaque arc u, $f'(u) \leq c(u)$ car soit $f'(u) = f(u) \leq c(u)$, soit $f'(u) = f(u) - \delta \leq f(u) \leq c(u)$, soit $f'(u) = f(u) + \delta$. Cette dernière éventualité est relative au cas où l'arc u est direct dans la chaîne $\mathcal{C}$. Or dans ce cas on a $\delta \leq \delta_1 \leq c(u) - f(u)$. Ce qui implique que $f'(u) = f(u) + \delta \leq c(u)$.

Il reste à montrer que f' est conservatif c'est-à-dire que pour tout sommet $x \in X \setminus \{s, p\}$ on doit avoir

$$\mathrm{d}'(x) = \sum_{u \in \omega^+(x)} f'(u) - \sum_{u \in \omega^-(x)} f'(u) = 0.$$

• Si $x \notin \mathcal{C}$, il ne passe par x aucun des arcs de $\mathcal{C}$. Par conséquent

$$\omega^+(x) \cap \{u_1, u_2, \cdots, u_l\} = \emptyset,\ \omega^-(x) \cap \{u_1, u_2, \cdots, u_l\} = \emptyset$$

et pour chaque arc u de $\omega^+(x)$ ou de $\omega^-(x)$, on a $f'(x) = f(u)$ et donc $\mathrm{d}'(x) = \mathrm{d}(x) = 0$.

• Si $x \in \mathcal{C} \setminus \{s, p\}$ on a quatre possibilités.

a) x est extrémité de u_i et est origine de u_{i+1}

Dans ce cas $f'(u_i) = f(u_i) + \delta$, $u_i \in \omega^-(x)$ et $f'(u_{i+1}) = f(u_{i+1}) + \delta$, $u_{i+1} \in \omega^+(x)$. Les autres arcs de $\omega^+(x)$ et de $\omega^-(x)$ n'appartiennent pas à $\{u_1, u_2, \cdots, u_l\}$ et sur chacun, on a $f'(u_i) = f(u_i)$. Par conséquent

$$\sum_{u \in \omega^+(x)} f'(u) = \sum_{u \in \omega^+(x)} f(u) + \delta$$

et

$$\sum_{u \in \omega^-(x)} f'(u) = \sum_{u \in \omega^+(x)} f(u) + \delta$$

$\mathrm{d}'(x) = \mathrm{d}(x) = 0$.

b) x est extrémité de u_i et de u_{i+1}

On a $f'(u_i) = f(u_i) + \delta$, $u_i \in \omega^-(x)$ et $f'(u_{i+1}) = f(u_{i+1}) - \delta$, $u_{i+1} \in \omega^-(x)$. Les autres arcs de $\omega^+(x)$ et de $\omega^-(x)$ n'appartiennent pas à $\{u_1, u_2, \cdots, u_l\}$ et sur chacun, on a $f'(u_i) = f(u_i)$. Par conséquent

$$\sum_{u \in \omega^+(x)} f'(u) = \sum_{u \in \omega^+(x)} f(u)$$

et

$$\sum_{u \in \omega^-(x)} f'(u) = \sum_{u \in \omega^+(x)} f(u) + \delta - \delta = \sum_{u \in \omega^+(x)} f(u)$$

Alors $\mathrm{d}'(x) = \mathrm{d}(x) = 0$.

c) x est origine de u_i et de u_{i+1}

On a $dans cecas f'(u_i) = f(u_i) - \delta$, $u_i \in \omega^+(x)$ et $f'(u_{i+1}) = f(u_{i+1}) + \delta$, $u_{i+1} \in \omega^+(x)$. Et on trouve $\mathrm{d}'(x) = \mathrm{d}(x) = 0$.

d) x est origine de u_i et extrémité de u_{i+1}

On a $dans cecas f'(u_i) = f(u_i) - \delta$, $u_i \in \omega^+(x)$ et $f'(u_{i+1}) = f(u_{i+1}) - \delta$, $u_{i+1} \in \omega^-(x)$. Et on trouve également $\mathrm{d}'(x) = \mathrm{d}(x) = 0$. On a donc f' qui est conservatif.

Montrons à présent que $v(f') = v(f) + \delta$.

On sait que $v(f') = \mathrm{d}'(s)$.

• Si s est origine u_1, on a $f'(u_1) = f(u_1) + \delta$ $u_1 \in \omega^+(s)$, les autres arcs de $\omega^+(s)$ et de $\omega^-(s)$ sont différents de $u_2,\ u_3, \cdots, u_l$ et sur chacun d'eux $f'(u) = f(u)$. Par conséquent $\mathrm{d}'(s) = \mathrm{d}(s) + \delta$ et donc $v(f') = \mathrm{d}'(s) = \mathrm{d}(s) + \delta = v(f) + \delta$.

• Si s est extrémité de u_1, on a $f'(u_1) = f(u_1) - \delta$, $u_1 \in \omega^-(s)$, les autres arcs de $\omega^+(s)$ et de $\omega^-(s)$ sont différents de $u_2,\ u_3, \cdots, u_l$. Ici encore on trouve $\mathrm{d}'(s) = \mathrm{d}(s) + \delta$ et donc $v(f') = \mathrm{d}'(s) = \mathrm{d}(s) + \delta = v(f) + \delta$.

Donc f' est bien un flot sur R. On a $v(f') = v(f) + \tau(\mu)$ ($\tau(\mu)$ est la capacité résiduelle minimale de la chaîne μ) et comme $\tau(\mu) > 0$, $v(f') > v(f)$. □

Proposition 5.3.3 *S'il n'existe pas pour le flot f sur R une chaîne augmentante alors il existe une coupe K telle que $c(K) = v(f)$.*

Preuve : Considérons l'ensemble K défini comme suit :

1. $s \in K$
2. *Si $x \in X$ et si $u = (x, y) \in U$ est un arc, alors $y \in K$ si et seulement si $f(u) < c(u)$*
3. *Si $x \in X$ et si $u = (y, x) \in U$ est un arc, alors $y \in K$ si et seulement si $f(u) > 0$.*

On remarque que :

- $s \in K$,

- si u est un arc sortant d'un sommet de K et si $f(u) < c(u)$ alors l'extémité terminale de u appartient à K.

- si u est un arc entrant en un sommet de K et si le flux circulant sur u, $f(u)$ n'est pas nul, alors l'extrémité initiale de u appartient à K.

Par construction, pour tout $x \in K$ ($x \neq s$) il existe au moins une chaîne μ reliant s à x avec tous ses sommets qui sont dans K et pour une telle chaîne, la capacité résiduelle minimale est strictement positive.

Si p appartenait à K, alors il existerait une chaîne μ de s à p de capacité résiduelle minimale strcitement positive, et donc améliorante de s à p. Ce qui est en contradiction avec l'hypothèse de départ. Donc $p \notin K$. Par suite K est une coupe de R.

Pour cette coupe K, on a :

- pour tout arc $u \in \omega^+(K)$, $f(u) = c(u)$;

- pour tout arc $u \in \omega^-(K)$, $f(u) = 0$.

On peut alors écrire :

$$\begin{aligned} c(K) &= \sum_{u \in \omega^+(K)} c(u) \\ &= \sum_{u \in \omega^+(K)} f(u) \\ &= \sum_{u \in \omega^+(K)} f(u) - \sum_{u \in \omega^-(K)} f(u) \\ &= d_f(K) = d_f(s) = v(f). \end{aligned}$$

Ce qui termine la preuve. □

Les preuves des propositions suivantes qu'on laisse en exercice, sont des conséquences directes des propositions précédentes.

Proposition 5.3.4 *Un flot sur un réseau est maximum si et seulement s'il n'existe pas de chaîne augmentante.*

Proposition 5.3.5 *Dans un réseau la valeur d'un flot maximum est égale à la capacité d'une coupe minimum.*

5.3.3 Prcocédure de détection d'une chaine améliorante

Etant donné f un flot sur R, pour déterminer une chaîne améliorante pour f on peut utiliser l'algorithme suivant :

1) Marquer s

2) Marquer s tout sommet suivant x de s tel que pour l'arc $u = (s, x)$ on a $f(u) < c(u)$

2) x étant un sommet déjà marqué, marquer "x"

- tout sommet suivant y de x non encore marqué tel que pour l'arc $u = (x, y)$ on a $f(u) < c(u)$.

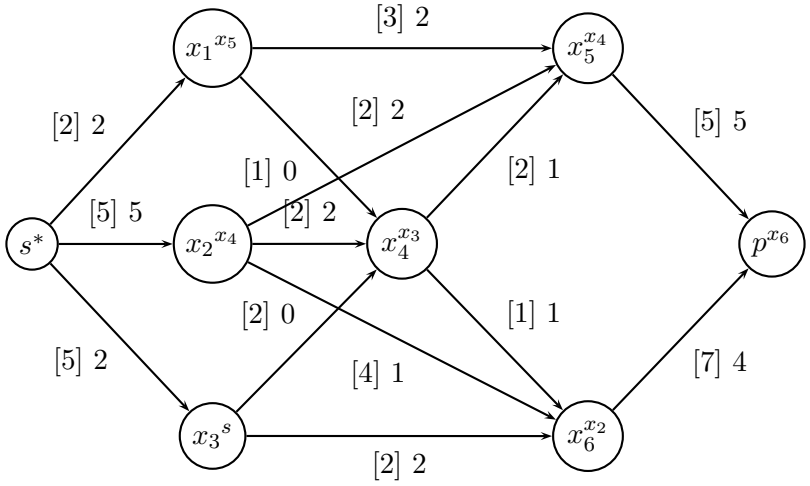
- tout sommet précédent y de x non encore marqué tel que pour l'arc $u = (y, x)$ on a $f(u) > 0$.

Si par cette procédure, p ne peut pas être marqué, alors il n'existe pas de chaine améliorante reliant s à p.

Si au contraire, la procédure permet de marquer p, on obtient à partir de p, en utilisant la marque de certains sommets marqués une chaine améliorante reliant s à p. Cette chaine se construit à l'envers de la façon suivante.

- Relier p au sommet correspondant à la marque de p.
- Recommencer à partir du sommet obtenu et ainsi de suite jusqu'à s.

Exemple 5.3.2



Le sommet p est marqué. La chaîne $\mathcal{C} = (s, x_3, x_4, x_2, x_6, p)$ est une chaîne améliorante. Donc le flot ci-dessus n'est pas de valeur maximale.

5.3.4 Procédure d'amélioration d'un flot

Etant donnée une chaîne améliorante μ pour un flot f sur R, la procédure suivante est dite procédure d'amélioration du flot f.

On pose :

$$\tau(\mu) = \text{Min}_{i=1,2,\ldots,k}\, \tau(u_i)$$

où $\tau(u_i) = c(u_i) - f(u_i)$ si $u_i \in \mu^+$ et $\tau(u_i) = f(u_i) > 0$ lorsque $u_i \in \mu^-$. On considère le flot f' défini par :

$$f'(u) = \begin{cases} f(u) + \tau(\mu) & \text{si } u \in \mu^+ \\ f(u) - \tau(\mu) & \text{si } u \in \mu^- \\ f(u) & \text{si } u \notin \mu \end{cases}$$

On sait que $v(f') = v(f) + \tau(\mu)$ avec $\tau(\mu) > 0$. Donc le flot f' améliore la valeur de f sur le réseau.

5.3.5 Algorithme de recherche d'un flot complet

Soit $R = (X, U, s, p, c)$ un réseau. Un flot sur R est dit *complet* s'il n'existe pas de chemin non saturé allant de s à p.

Un flot complet n'est pas nécessairement maximal mais il fournit une excellente solution de depart pour appliquer un algorithme itératif de détermination d'un flot maximum.

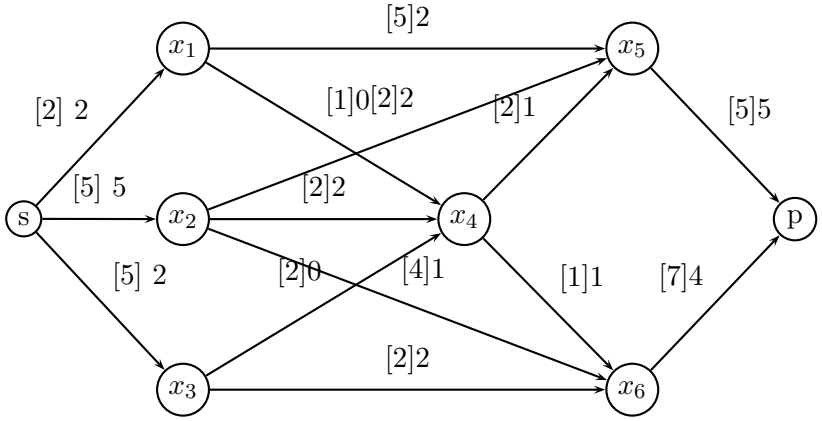
Algorithme

On part d'un flot f (par exemple le flot nul) et on l'améliore pas à pas par une procédure de marquage.

1. *Marquer s*
2. *Soit x un sommet marqué et non encore examiné ;*
   Marquer y par (+x) si y est un successeur non marqué de x avec $f(x, y) < c(x, y)$
3. *Si p est marqué (on a un chemin augmentant), aller en (5)*
4. *Si tous les sommets marqués sont examinés alors le flot est complet ;*
   sinon aller en (2)
5. *Améliorer le flot*
   Effacer les marques
   Aller en (1)

Exemple 5.3.3

Le flot ci-dessous est complet.



5.3.6 Algorithme de recherche d'un flot maximum (Ford-Fulkerson)

On part, par exemple d'un flot complet. On l'améliore tant qu'il existe des chaînes augmentantes.

Algorithme

1. *Marquer l'entrée s*
2. *Soit x un sommet marqué non examiné*

   Etudier tous les successeurs y de x :
   Marquer y par $(+x)$ s'il est non marqué et si $f(x,y) < c(x,y)$.

   Etudier tous les prédécesseurs z de x :
   Marquer z par $(-x)$ s'il est non marqué et si $f(z,x) > 0$
3. *Si p est marqué (on a une chaîne augmentante), aller en (4)*
   S'il reste des sommets marqués non examinés aller (2)
   sinon le flot est maximum, FIN.
4. *Améliorer le flot à l'aide de la chaîne améliorante ayant permis de marquer p.*
   Effacer les marques et aller en (1)

5.3.7 Réseau canonique

Pour un flot f sur un réseau $R = (X, U, s, p, c)$, on est amener à calculer sa valeur $v(f)$ en considérant la formule

$$v(f) = d_f(s) = \sum_{u \in \omega^+(s)} f(u) - \sum_{u \in \omega^-(s)} f(u).$$

Pour éviter ces calculs, on adjoint au réseau initial R, un nouvel arc u_r d'origine p et d'extrémité s de capacité $+\infty$ dit arc retour. Le nouveau réseau ainsi construit est appelé réseau canonique. On le note R_c.

Un flot sur le réseau canonique vérifie ;

$$d_f(x) = 0 \quad \forall\, x \in X.$$

On obtient alors $d_f(s) = f(u_r)$, c'est-à-dire $v(f) = f(u_r)$.

Dans les différents algorithmes de détermination de flot complet et de modification de flux, on considère l'arc retour comme un arc direct de la chaîne améliorante et donc le flux y est augmenté.

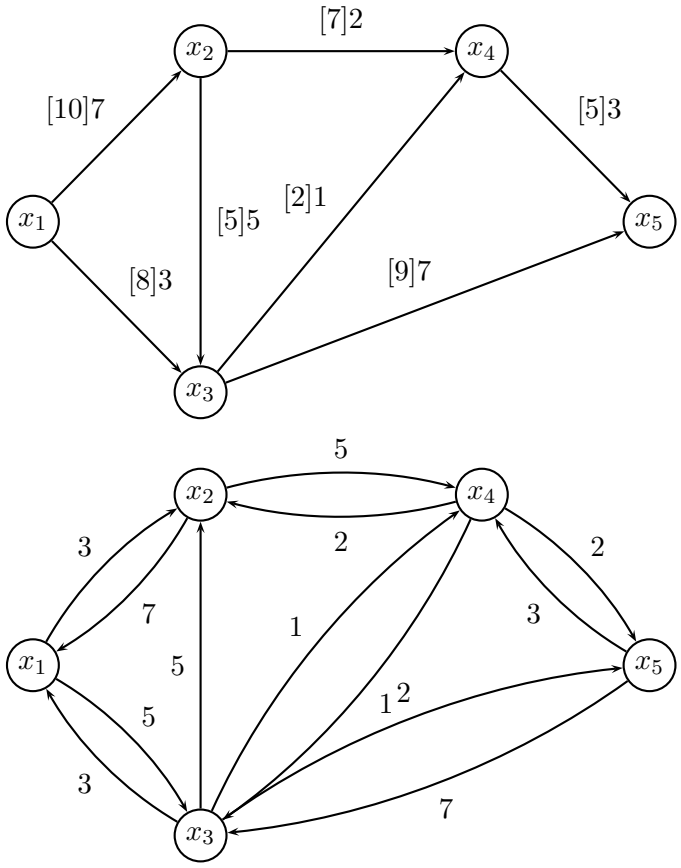
5.3.8 Algorithme utilisant les graphes d'écart

Définition 5.3.1 *On considère le réseau $R = (X, U, s, p, c)$ et f un flot sur R. Si $G = (X, U)$, on appelle graphe d'écart associé au couple (G, f), le graphe $G^e(f) = (X, U^e(f))$ dont les arcs sont munis de capacités tel que :*

Pour tout arc $u = (x, y) \in U$ correspondent au plus deux arcs u^+ d'origine x et d'extrémité y de capacité $c(u) - f(u)$ si $c(u) - f(u) > 0$ et u^- d'origine y et d'extrémité x de capacité $f(u)$ si $f(u) > 0$.

Les arcs u^+ et u^- sont dits duaux l'un de l'autre.

Remarque 5.3.1 *Dans un graphe d'écart, on ne représente pas les arcs de capacité nulle.*



Théorème 5.3.1 *Il existe une bijection entre les chaînes non saturées de G et les chemins allant de s à p dans le graphe d'écart associé.*

Preuve :

Soit $\mathcal{C}$ une chaîne améliorante de s à p dans G. On considère $\mathcal{C}^+$ l'ensemble des arcs directs et $\mathcal{C}^-$ l'ensemble des acrs inverses de la chaîne $\mathcal{C}$.

Si $u \in \mathcal{C}^+$, on considère l'arc u^+ de $U^e(f))$

Si $u \in \mathcal{C}^-$, on considère l'arc u^- de $U^e(f))$

En concatenant les arcs u^+ et u^- de $U^e(f))$ correspondant aux arcs de $\mathcal{C}$, on obtient un chemin μ de s à p dans $G^e(f)$.

Réciproquement, soit μ un chemin de s à p dans $G^e(f)$. On sait que par ocnstruction, à chaque arc de $G^e(f)$ correspond un arc et un seul dans G. En concatnenat les arcs de G associés à ceux de μ, on obtient une chaîne $\mathcal{C}$ allant de s à p qui est par définition une chaîne améliorante. □

On montre que

Proposition 5.3.6 *Soit μ un chemin allant de s à p dans un graphe d'écart relativement à un flot f et*

$$\delta = \min[c_f^e(u) : u \in \mu].$$

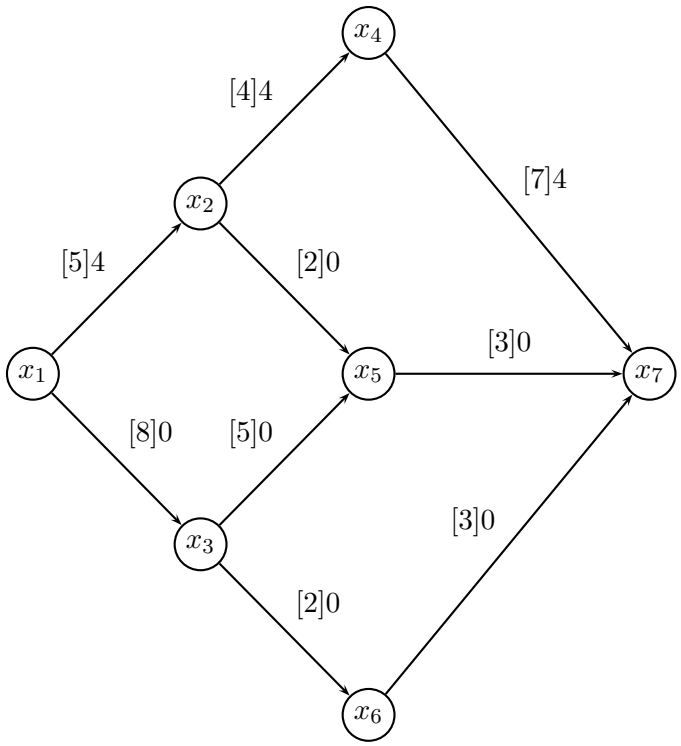
Alors l'application f' définie sur U par :

$$f'(u) = \begin{cases} f(u) + \delta & si \;\; u^+ \in \mu \\ f(u) - \delta & si \;\; u^- \in \mu \\ f(u) & ailleurs \end{cases}$$

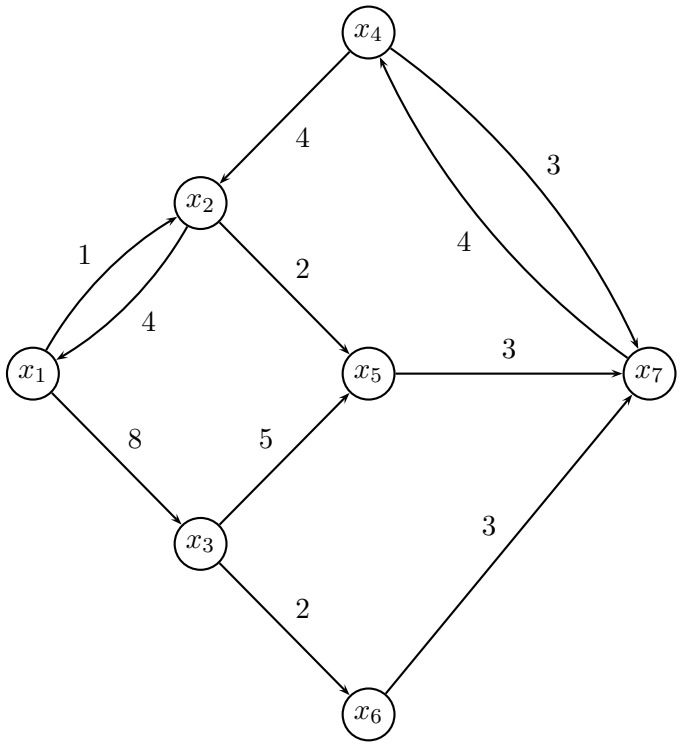
est un flot sur R et on a $v(f') = v(f) + \delta$.

Exemple

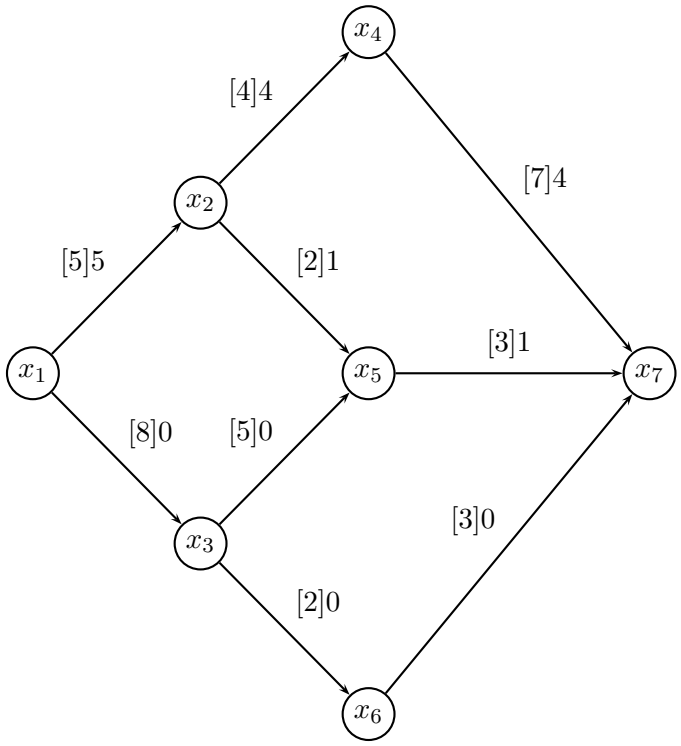
Considérons le réseau $R = (X, U, x_1, x_7, c)$ et le flot ci-dessous.



La valeur du flot est $v(f) = 4$
Le graphe d'écart associé est



Pour le chemin $\mu = (x_1, x_2, x_5, x_7)$, on a $\delta = 1$. Le flot f' correspondant est le suivant.



On a $v(f') = 5$.

Théorème 5.3.2 *Une condition nécessaire et suffisante pour qu'un flot soit de valeur maximale est qu'il n'existe pas de chemin allant de s à p dans le graphe d'écart associé au couple (G, f).*

Soit f un flot sur $R = (X, U, s, p, c)$ et μ un chemin dans le graphe d'écart relatif à f.
Soit

$$\delta = \min[c_f^e(u) : u \in \mu].$$

Soit f' le flot défini par :

$$f'(u) = \begin{cases} f(u) + \delta & \text{si } u^+ \in \mu \\ f(u) - \delta & \text{si } u^- \in \mu \\ f(u) & \text{ailleurs} \end{cases}$$

Le graphe d'écart associé à f' est obtenu à aprtir de celui relatif à f moyennant les modifications suivantes.
Pour tout arc $u = (x, y) \in \mu$,
poser : $c_{f'}^e(x, y) = c_f^e(x, y) - \delta$
Supprimer (x, y) si $c_{f'}^e(x, y) = 0$
Créer (y, x) s'il n'existait pas et poser $c_{f'}^e(y, x) = c_f^e(y, x) + \delta$
Pour les autres arcs u, garder $c_{f'}^e(u) = c_f^e(u)$.
On a l'algorithme suivant :

Algorithme

On suppose qu'un flot f circule sur le réseau.

On construit le graphe d'écart $G^* = G^e(f)$ associé au couple (G, f).

tant qu'il existe dans G^* un chemin μ de s à p faire :

Début

Soit δ la plus petite des capacité des arcs de μ

Pour chacun des arcs a de μ faire

Début

$c^*(a) = c^*(a) - \delta$

Soit a' *l'arc dual de* a

$c^*(a') = c^*(a) + \delta$

Fin

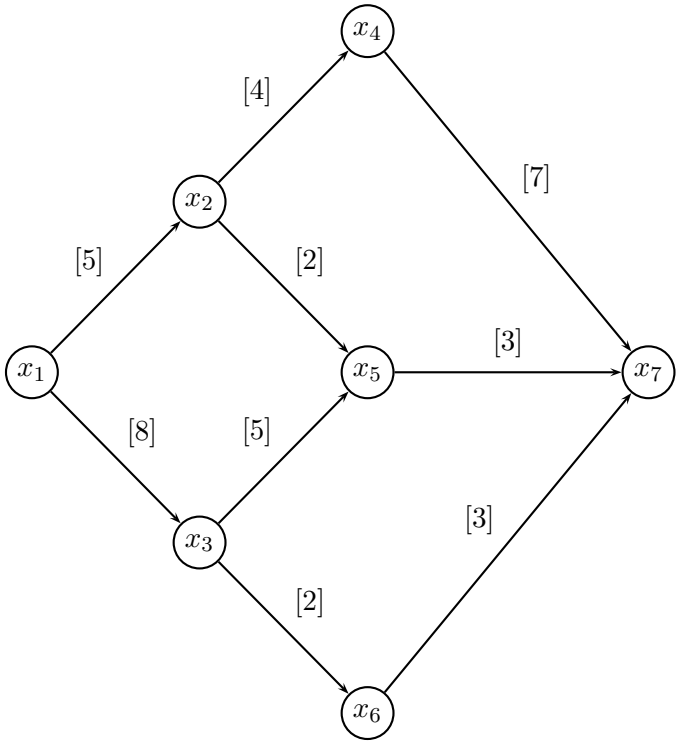
Fin

Enfin, on revient au réseau R.

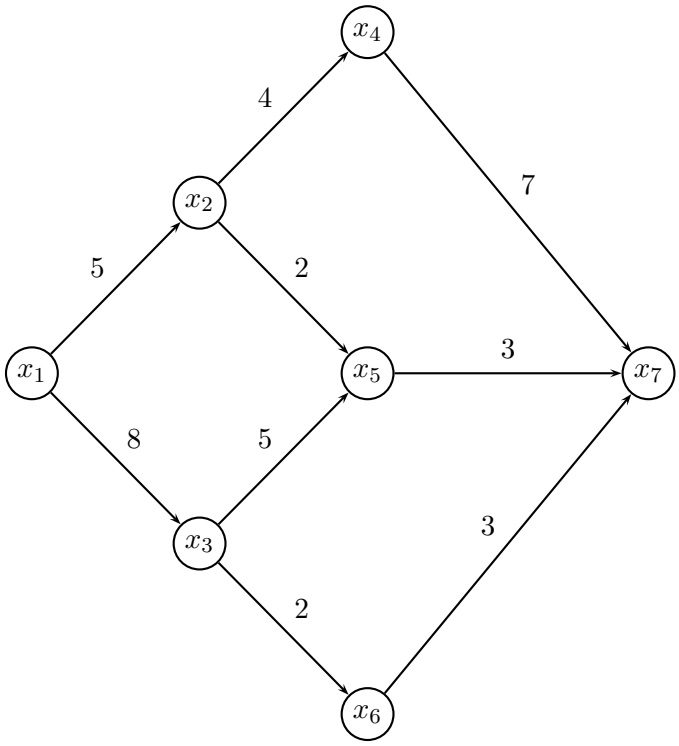
Et pour chacun des arcs u *de* G, *on fait* $f(u) = c^*(u^-)$ *avec* $u^-(y, x)$ *si* $u = (x, y)$.

Exemple

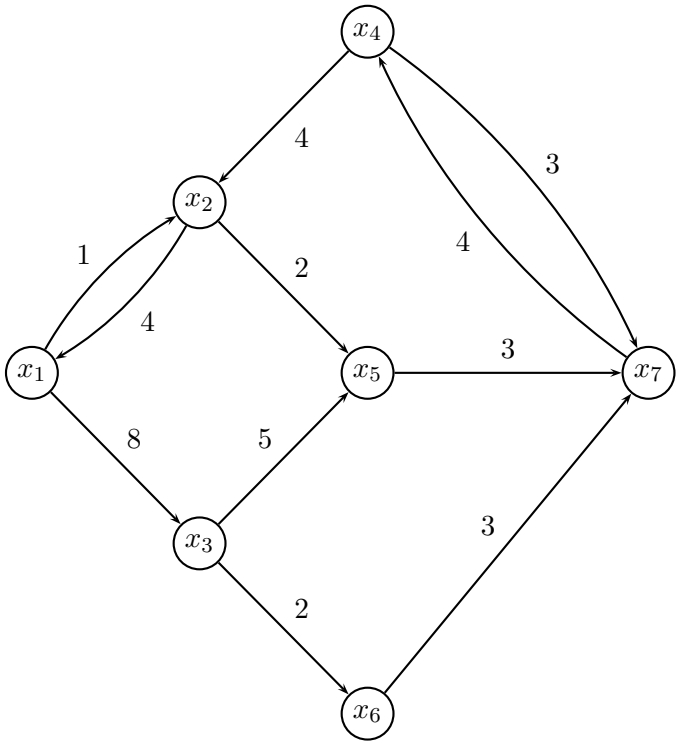
On considère le réseau ci-dessous $R = (X, U, x_1, x_7, c)$.



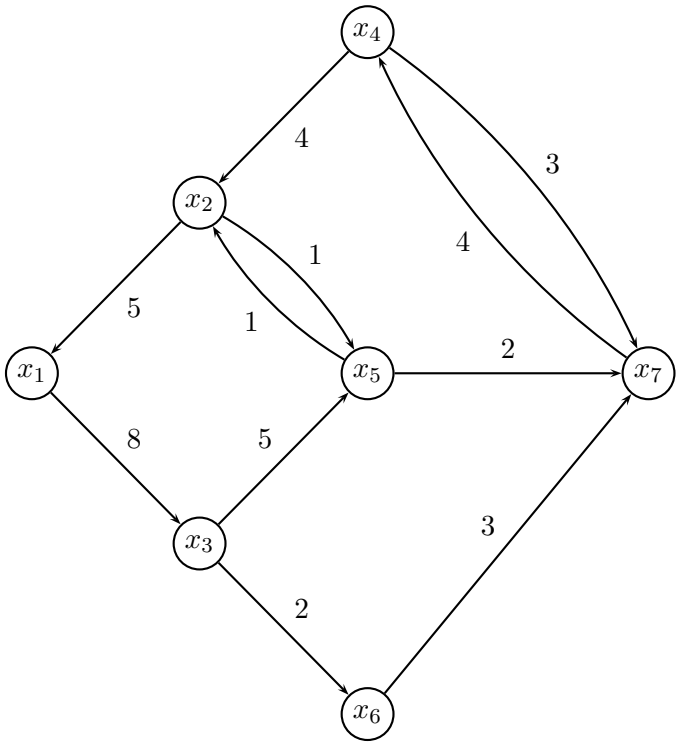
On part du flot nul $f_0 = 0$ et le graphe d'écart associé est :



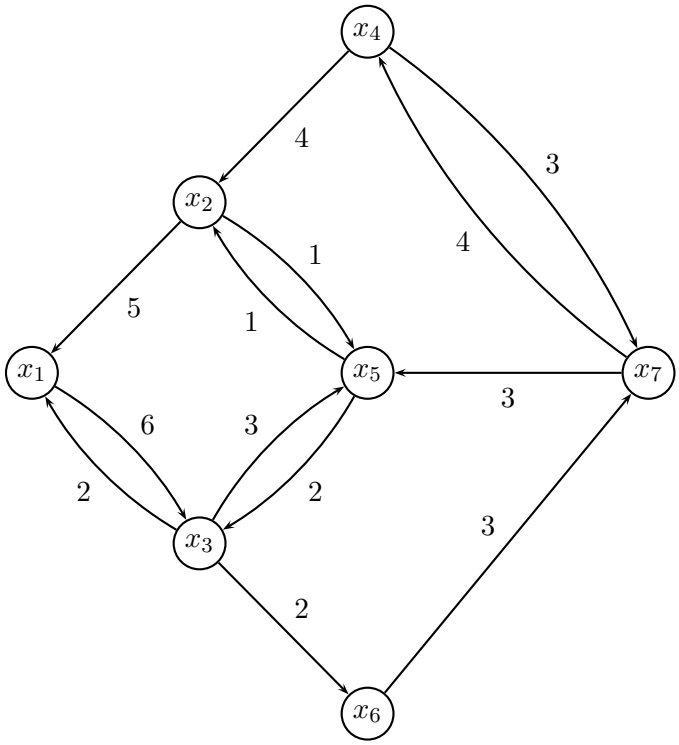
Il y a un chemin de x_1 à x_7.
Soit le chemin $\mu_0 = (x_1, x_2, x_4, x_7)$, on a $\delta_0 = 4$ et le nouveau graphe d'écart associé est :



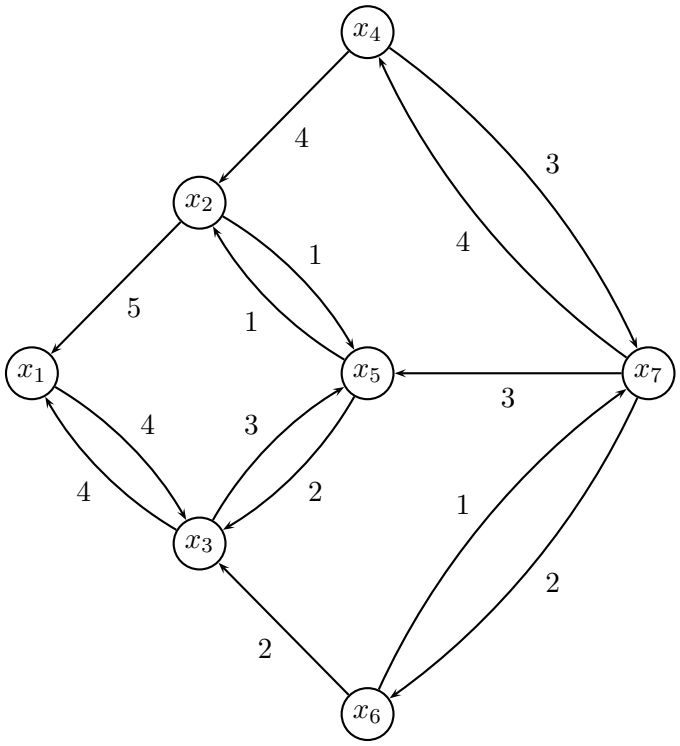
Il y a un chemin de x_1 à x_7.
Soit le chemin $\mu_1 = (x_1, x_2, x_5, x_7)$, on a $\delta_1 = 1$ et le nouveau graphe d'écart associé est :



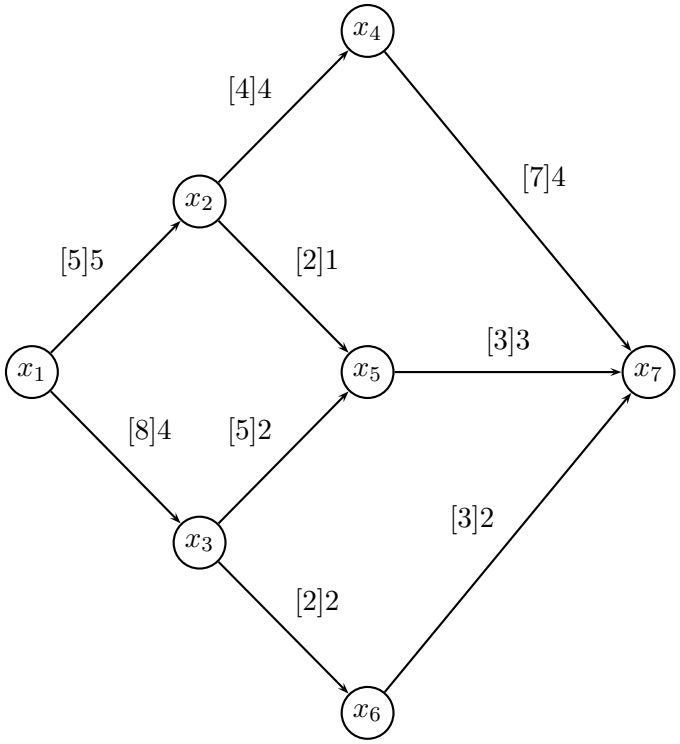
Il y a un chemin de x_1 à x_7.
Soit le chemin $\mu_2 = (x_1, x_3, x_5, x_7)$, on a $\delta_2 = 2$ et le nouveau graphe d'écart associé est :



Il y a un chemin de x_1 à x_7.
Soit le chemin $\mu_3 = (x_1, x_3, x_6, x_7)$, on a $\delta_3 = 2$ et le nouveau graphe d'écart associé est :



Il n y a plus de chemin de x_1 à x_7, le flot ci-dessous est donc de valeur maximale.



la valeur du flot est $v(f) = 9$ et la coupe de capacité minimale est $K = \{x_1, x_2, x_3, x_5\}$ et la capacité est : $c(K) = 9$.

5.3.9 Flots canalisés ou bornés

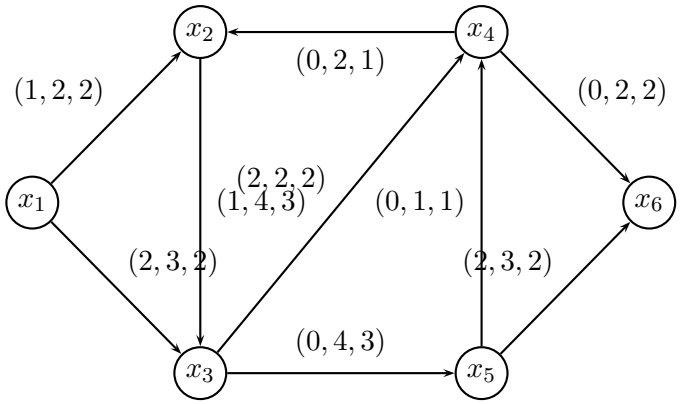
Ici on suppose que dans le réseau $R = (X, U, s, p, c)$, outre les capacités, chaque arc u est doté d'une autre grandeur $b(u) \geq 0$ dite borne de l'arc.

On note alors $R_b = (X, U, s, p, b, c)$

Définition 5.3.2 *Soit le réseau $R_b = (X, U, s, p, b, c)$. Un flot canalisé f est un flot qui doit satisfaire outre les contraintes de Kirchoff et de capacités, des contraintes dites de bornes :*

$$\forall\, u \in U \quad b(u) \leq f(u).$$

Exemple



Définition 5.3.3 *On considère le réseau $R_b = (X, U, s, p, b, c)$ et f un flot sur R_b. Si $G = (X, U)$, on appelle graphe d'écart associé au couple (G, f), le graphe $G^e(f) = (X, U^e(f))$ dont les arcs sont munis de capacités telles que :*

Pour tout arc $u = (x, y) \in U$ correspondent au plus deux arcs u^+ d'origine x et d'extrémité y de capacité $c(u) - f(u)$ si $c(u) - f(u) > 0$ et u^- d'origine y et d'extrémité x de capacité $f(u) - b(u)$ si $f(u) > b(u)$.

On rappelle que dans un graphe d'écart, on ne représente pas les arcs de capacité nulle.

Pour appliquer l'algorithme de Ford-Fulkerson, il est nécessaire d'avoir un flot canalisé initial.

5.4 Flot de valeur maximale de coût minimum

5.4.1 Formulation du problème

Soit $R = (X, U, s, p, c)$ un réseau tel que pour tout arc $u \in U$, on associe $k(u) \geq 0$ qui représente le coût unitaire de mouvement de matière le long de cet arc. Alors le coût total d'un flot f sur R est :

$$k(f) = \sum_{u \in U} k(u) f(u).$$

Notons par ν^* la valeur maximale d'un flot sur le réseau R.

Si on considère l'ensemble $\mathcal{F}_\nu$ des flots de même valeur ν, avec $0 \leq \nu \leq \nu^*$, un flot f^* de cet ensemble est dit de coût minimum, s'il minimise le coût total $k(f)$ des flots $f \in \mathcal{F}_\nu$:

$$k(f^*) = \begin{array}{l} \min k(f) \\ f \in \mathcal{F}_\nu \end{array}.$$

Si $\nu = \nu^*$, on parle de flot de valeur maximale et de coût minimum sur le réseau R.

5.4.2 Flot de valeur donnée et de coût minimum

Pour déterminer un flot de valeur maximale et de coût minimum sur le réseau R, il suffit de savoir déterminer un flot de valeur donnée et de coût minimum.

L'algorithme ci-dessous permet de déterminer un flot de valeur ν donnée et de coût minimum sur le réseau R.

Algorithme de Roy

1. *On part du flot nul* $f = 0$, $v(f) = 0$.
2. *Construire le graphe d'écart* $G^e(f) = (X, U^e(f))$ *associé au couple* (G, f).
3. *S'il n'existe pas dans* $G^e(f)$ *un chemin* μ *de* s *à* p *stop : le flot obtenu est de valeur maximale strictement inférieure à* ν *et de coût minimum :*
4. *Sinon, on définit les longueurs suivantes sur le graphe d'écart*

$$l(u) = \begin{cases} k(u) & si \quad u \in U^e(f) \cap U \\ -k(u) & si \quad u \in U^e(f) \setminus U \end{cases}$$

5. *Déterminer dans* $G^e(f)$ *un chemin* μ *de* s *à* p *de longueur minimale.*
6. *Déterminer* δ *la capacité résiduelle minimale sur* μ *et on pose*

$$v(f) = v(f) + \delta.$$

7. *Si* $v(f) \geq \nu$, *le flot suivant est de valeur* ν *et de coût minimum.*

$$f(u) = \begin{cases} f(u) + \nu - v(f) & si \quad u \in \mu \cap U \\ f(u) - \nu + v(f) & si \quad u \in \mu \setminus U \end{cases}$$

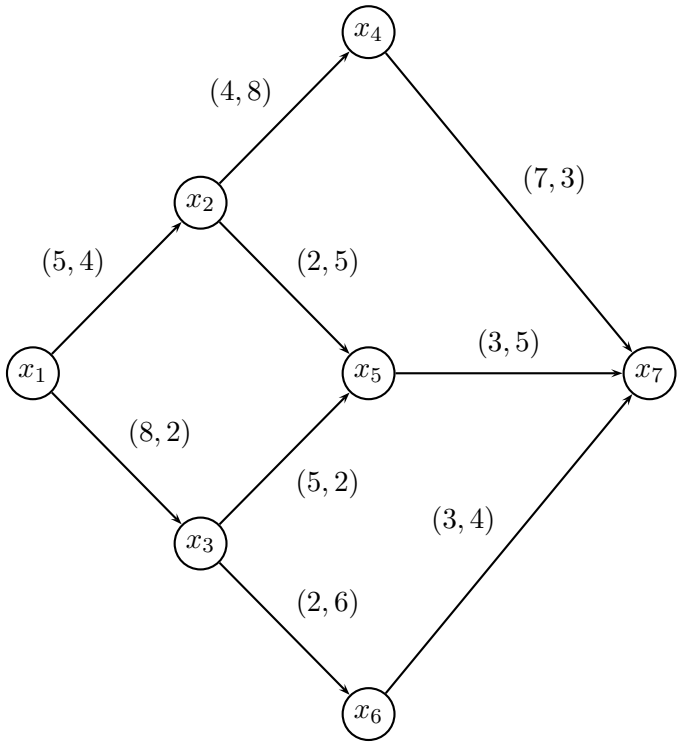
8. *sinon on considère le nouveau flot* f *suivant*

$$f(u) = \begin{cases} f(u) + \delta & si \quad u \in \mu \cap U \\ f(u) - \delta & si \quad u \in \mu \setminus U \end{cases}$$

9. *Aller à l'étape (2)*

Exemple

Soit à déterminer un flot de valeur ν de coût minimal dans le réseau $R = (X, U, x_1, x_7, c)$ ci-dessous où les couples de nombres associés aux arcs représentent, respectivement, les capacités et les coûts unitaires.



Notons $G = (X, U)$ le graphe correspondant.

Etape 1

1) Graphe d'écart valué

Considérons comme flot initial f le flot nul de valeur $v(f) = 0$.

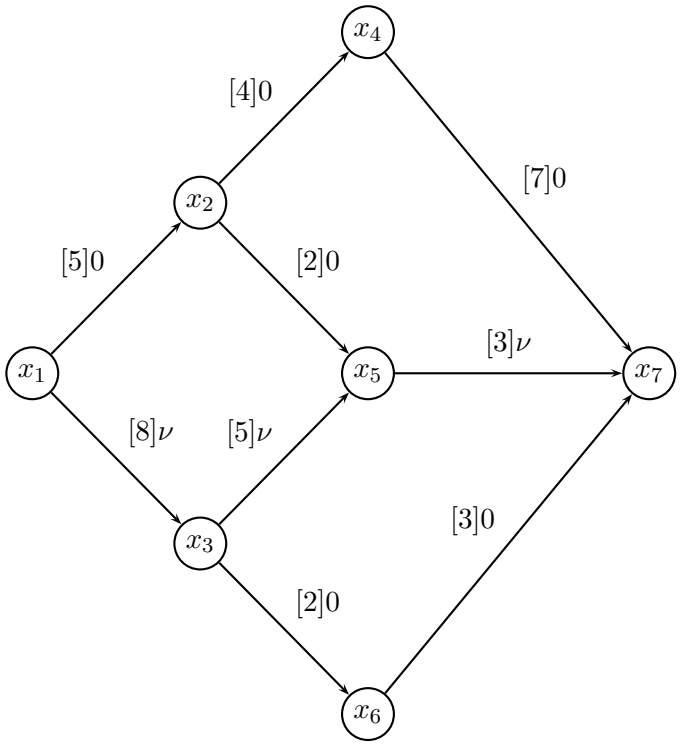
Alors le graphe d'écart associé $G^e(f)$ est identique au graphe G les cpacités sont égales aux capacités du graphe G et les longueurs sont égales aux coûts unitaires.

2) Amélioration du flot de coût minimal

Pour déterminer un chemin μ de longueur minimale on peut utiliser l'algorithme de Dijkstra puisque les longeurs sont positives. Par application de cet algorithme, on obtient le tableau suivant :

x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	4	2^*	∞	∞	∞	∞
$\vdots$	4^*	$\vdots$	∞	4	8	∞
$\vdots$	$\vdots$	$\vdots$	12	4^*	8	∞
$\vdots$	$\vdots$	$\vdots$	12	$\vdots$	8^*	9
$\vdots$	$\vdots$	$\vdots$	12	$\vdots$	$\vdots$	9^*
$\vdots$	$\vdots$	$\vdots$	12^*	$\vdots$	$\vdots$	$\vdots$

Un chemin μ de longueur minimale (de coût minimal) joignant x_1 à x_7 est donc $\mu_1 = (x_1, x_3, x_5, x_7)$. La capacité minimale est $\delta_1 = 3$, le flot peut être amélioré de $\delta_1 = 3$. Un flot de valeur ν, où $0 < \nu \leq 3$ de coût minimal est donc :

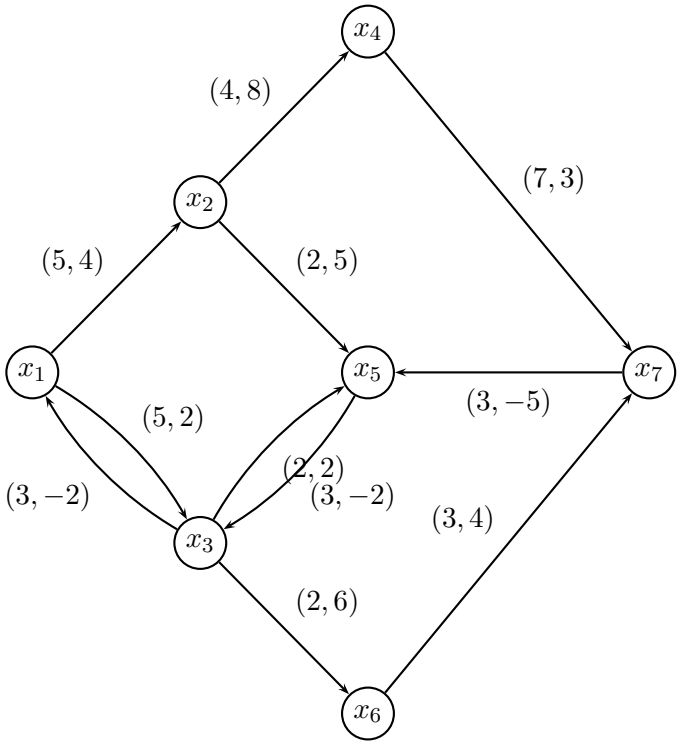


Si la valeur de flot choisie ν est supérieure à 3, le flot ci-dessus avec $\nu = 3$ sert de flot initial pour l'étape 2.

Etape 2

1) Graphe d'écart valué

Le graphe d'écart valué associé au flot obtenu à l'étape 1 pour $\nu = 3$ est donné ci-dessous.

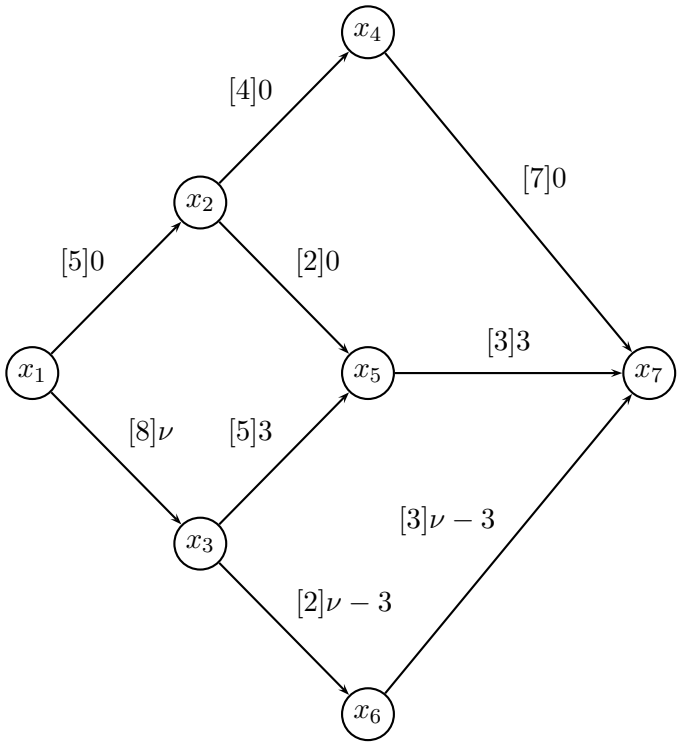


2) Amélioration du flot de coût minimal

Pour déterminer un chemin μ de longueur minimale on peut utiliser l'algorithme de Bellman celui de Dijkstra ne peut être utilisé car il existe des longueurs négatives. Par application de cet algorithme, on obtient le tableau suivant :

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
$k=0$	0	∞	∞	∞	∞	∞	∞
$k=1$	0	4	2	∞	∞	∞	∞
$k=2$	0	4	2	12	4	8	∞
$k=3$	0	4	2	12	4	8	12
$k=4$	0	4	2	12	4	8	12

Un chemin de longueur minimale (de coût minimal) joignant x_1 à x_7 est donc $\mu_2 = (x_1, x_3, x_6, x_7)$. La capacité minimale est $\delta_2 = 2$, le flot peut être amélioré de $\delta_2 = 2$. Un flot de valeur ν, où $3 < \nu \leq 5$ de coût minimal est donc :

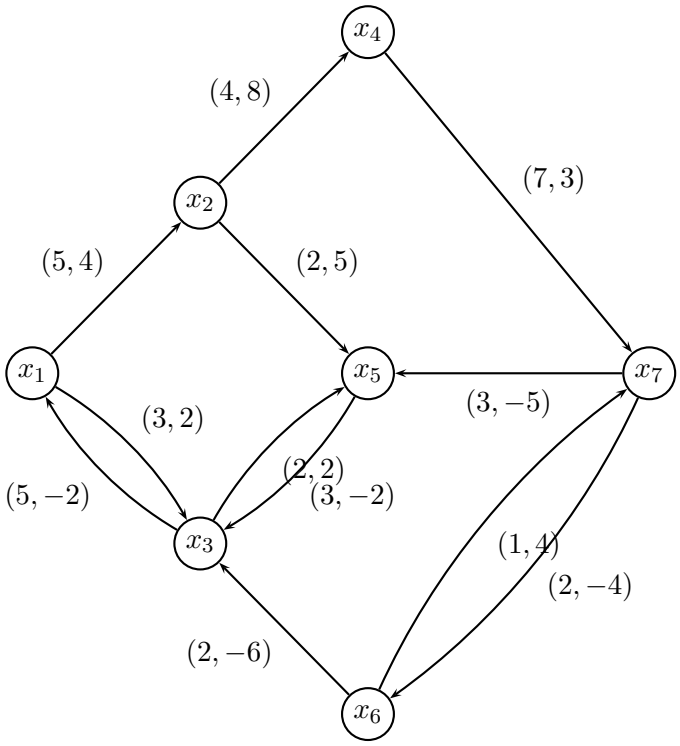


Si la valeur de flot choisie ν est supérieure à 5, le flot ci-dessus avec $\nu = 5$ sert de flot initial pour l'étape 3.

Etape 3

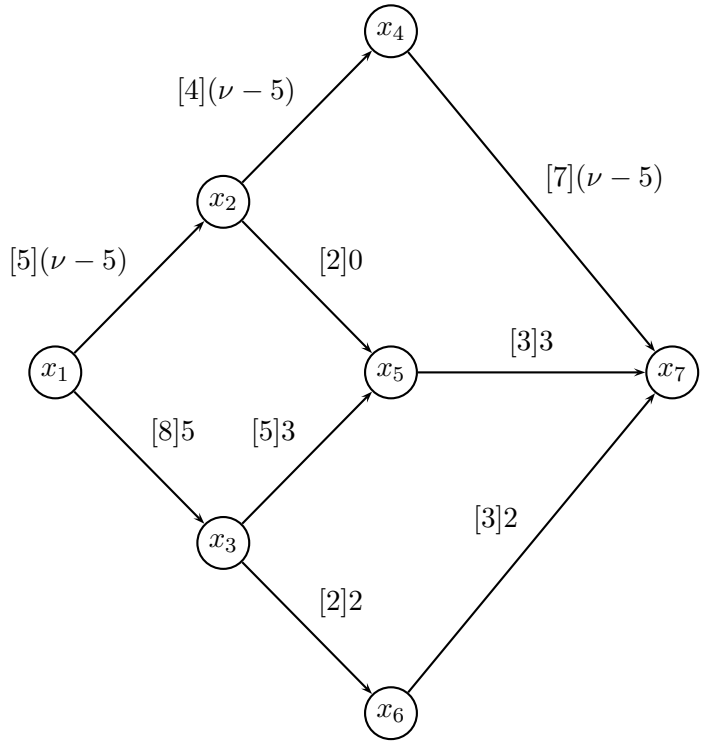
1) Graphe d'écart valué

Le graphe d'écart valué associé au flot obtenu à l'étape 2 pour $\nu = 5$ est donné ci-dessous.



2) Amélioration du flot de coût minimal

Il n'existe dans $G^e(f)$ qu'un seul chemin allant de x_1 à x_7 : $\mu_3 = (x_1, x_2, x_4, x_7)$ sa capacité résiduelle minimale est $\delta_3 = 4$, le flot peut être amélioré de 4. Un flot de valeur ν, où $5 < \nu \leq 9$ de coût minimal est donc :

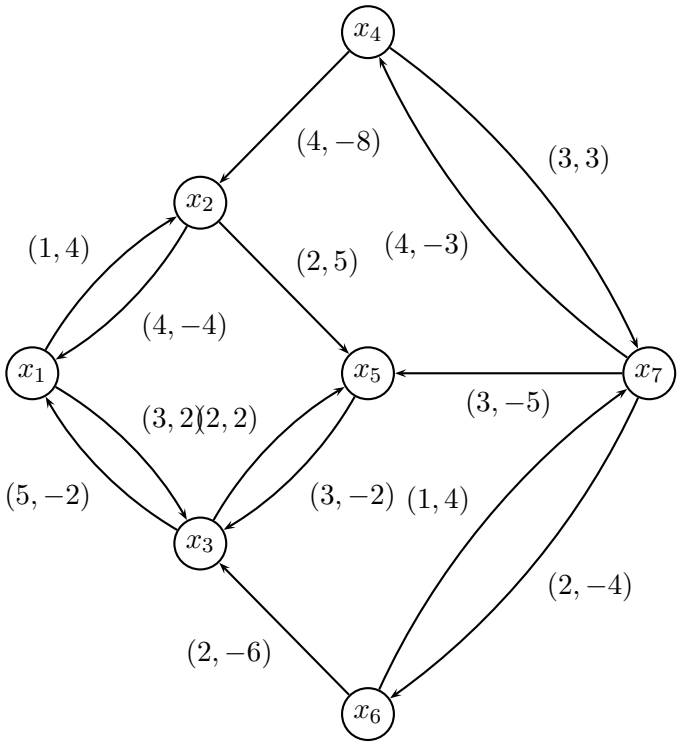


Si la valeur de flot choisie ν est supérieure à 9, le flot ci-dessus avec $\nu = 9$ sert de flot initial pour l'étape 4.

Etape 4

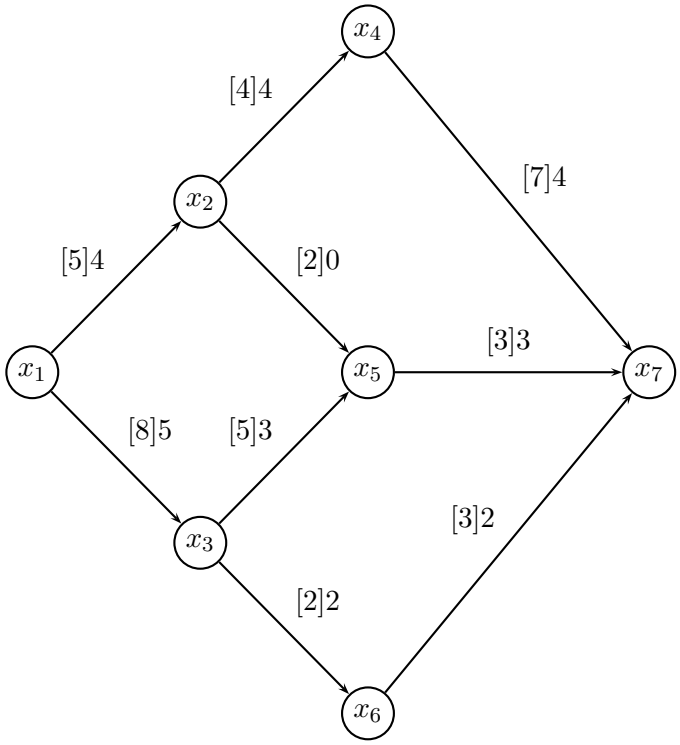
1) Graphe d'écart valué

Le graphe d'écart valué associé au flot obtenu à l'étape 2 pour $\nu = 5$ est donné ci-dessous.



2) Flot de valeur maximale de coût minimal

Il n'existe dans $G^e(f)$ de chemin allant de x_1 à x_7, le flot obtenu à l'étape 3 pour $\nu = 9$ est de valeur maximale et de coût minimal ce flot est donc :



et le coût total correspondant est 111.

5.5 Extensions diverses

5.5.1 Flot dans un graphe non orienté

Nous avons supposé jusqu'ici que le graphe $G = (X, U)$ était un graphe orienté. Il est cependant facile d'utiliser l'algorithme de Ford-Fulkerson pour rechercher un flot maximum dans un graphe où certains arcs sont nonn orienté (le flot pouvant passer dans les deux sens). Il suffit pour cela de se ramener au cas orienté en construisant à partir du graphe de départ, un graphe orienté où tout arc $u = (x, y)$ non orienté de capacité $c(u)$, est remplacé par deux arcs orienté

$$\begin{array}{ll} u' = (x, y) & \text{de capacité } c(u), \\ u'' = (y, x) & \text{de capacité } c(u) \end{array}$$

5.5.2 Flot dans un graphe avec capacités aux sommets

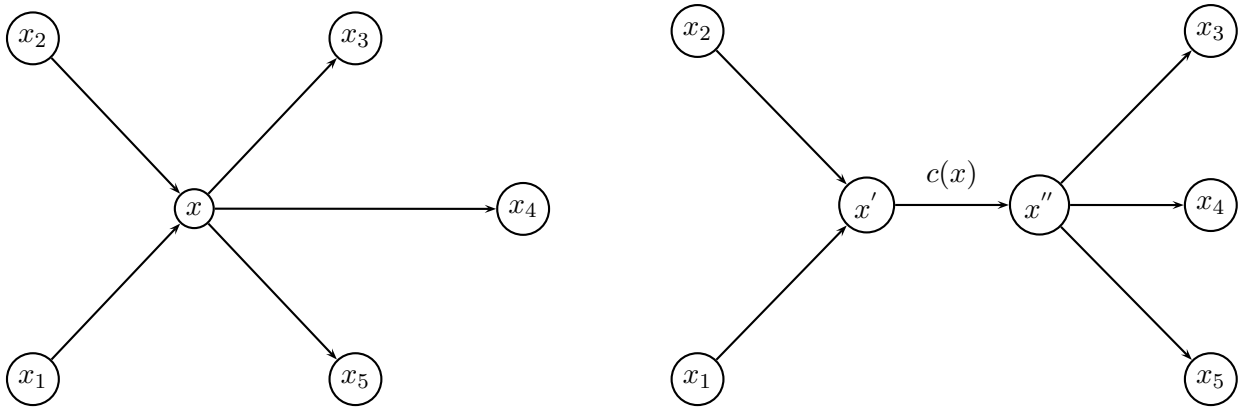
On peut introduire des capacités aux sommets $c(x)$, $x \in X$ afin de borner supérieurement la quantité

$$\sum_{x \in \omega^-(x)} f(u) = \sum_{x \in \omega^+(x)} f(u).$$

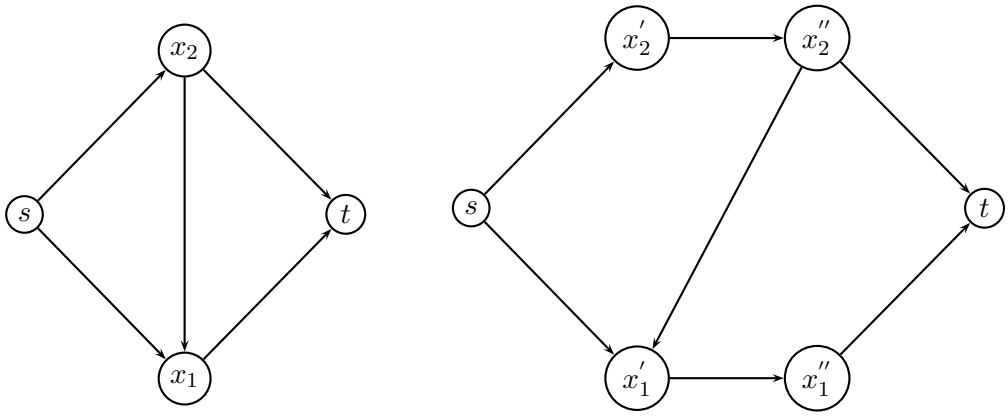
Autrement dit, le flux transitant par le sommet x.

Pour tenir compte de cette contrainte, il suffit de dédoubler le sommet x en deux sommets x' (sommet d'arrivé) et x'' (sommet de depart). Les sommets x' et x'' sont reliés par un arc (x', x'') de capacité $c(x)$.

Les arcs $u \in \omega^-(x)$ dont x constitue l'extrémité terminale sont alors rattachés à x' et les arcs $u \in \omega^+(x)$ sont rattachés à x''. On n a plus qu'à chercher un flot maximum sur le nouveau graphe ainsi formé.



Exemple



5.6 Problèmes pouvant se modéliser comme problème de flots

Nombre de problèmes, ne relevant pas à première vue de la théorie des flots, peuvent cependant s'exprimer comme des problèmes de flots de valeur maximale ou comme des problèmes de flots birnés (éventuellement de coût minimal).

5.6.1 Problème d 'affectation

Soient $I = \{1, \cdots, m\}$ un ensemble de postulants et $J\{1, \cdots, n\}$ un ensemble d'emplois. Chaque postulant $i \in I$, est ou non apte à occuper chaque emploi $j \in J$.

Il s'agit d'affecter le maximum d'individus i à des emplois j qu'ils peuvent occuper, étant entendu que chaque individu peut occuper au plus un emploi, et qu'un emploi ne peut être attribué qu'à un individu au plus.

On construit un réseau $R = (X, U, s, p, c)$, comme suit : soient s et t deux sommets fictifs, on prend $X = \{s, t\} \cup I \cup J$.

L'ensemble des arcs U est tel que :

• pour tout couple $(i, j) \in I \times J$, on associe un arc (i, j) de capacité $c(i, j) = 1$, si i est apte à occuper l'emploi j.

• pour tout i de I, on associe un arc (s, i) de capacité $c(s, i) = 1$;

• pour tout j de J, on associe un arc (j, t) de capacité $c(s, i) = 1$;

Le problème se ramène alors à trouver un flot f de valeur maximale sur le réseau construit.

Sur chaque arc (i, j), on aura soit $f(i, j) = 1$ si i est affecté à j, soit $f(i, j) = 0$. et la valeur maximale $v(f)$ donnera le nombre total d'individus affecté.

En considérant chaque j comme une entreprise susceptible d'embaucher au plus $c(j)$ candidats, on pourra résoudre un problème d'affectation généralisé, en prenant $c(j, t) = c(j)$ pour capacité de l'arc (j, t).

En considérant chaque j comme une entreprise susceptible d'embaucher au moins $b(j$ et au plus $c(j)$ employés, on se ramenera à la détermination d'un flot borné, en prenant pour tout u du graphe, $b(u) = 0$, sauf pour les arcs (j, t) où on prendra $b(j, t) = b(j)$.

On pourra également résoudre le problème d'affectation avec coûts, en prenant les coûts unitaires $l(s, i) = 0$, pour tout i, $l(j, t) = 0$, pour tout j, et $l(i, j)$ égale au coût d'affectation de i à j, pour tout arc (i, j).

On obtiendra de la sorte une affectation maximale de coût minimal.

5.6.2 Problème de transport

Pour un produit donné, on a un ensemble $I = \{1, \cdots, m\}$ de fournisseurs et un ensemble $J\{1, \cdots, n\}$ de demandeurs. Chaque fournisseur $i \in I$ dispose d'un nombre a_i d'unités du produit, et chaque demandeur $j \in J$ souhaite en recevoir b_j unités. On cherche les quantités à transporter de chez chaque fournisseur vers chaque consommateur de façon à satisfaire au maximum la demande.

Ce problème peut se modéliser au moyen d'un réseau $R = (X, U, s, p, c)$ comme suit : soient s et t deux sommets fictifs, on prend

• $X = \{s, t\} \cup I \cup J$, $U = (I \times J) \cup \{(s, i) : i \in I\} \cup \{(j, t) : j \in J\}$.

• pour chaque arc (s, i), on prend pour capacité $c(s, i) = a_i$;

• pour chaque arc (j, t), on prend pour capacité $c(j, t) = b_j$;

• pour chaque arc (i, j), on prend pour capacité $c(i, j) = +\infty$;

Le problème se ramène alors à trouver un flot f de valeur maximale sur le réseau construit.

Si cahaque fournisseur i doit adresser au moins a'_i unités de produit et chaque destinataire j en recevoir au moins b'_j unités, on recourra à des flots bornés.

Si on considère que le coût de transport d'une unité de produit de i à j est de c_{ij}, le problème se ramenera à un problème de flot de valeur maximale et de coût minimal en prenant les coûts unitaires $l(s, i) = 0$, pour tout i, $l(j, t) = 0$, pour tout j, et $l(i, j) = c_{ij}$.

Ainsi le problème de transport peut-être abordé comme un problème de flot , éventuellement borné, avec coûts.

Bibliographie

[1] C. Berge, *Graphes*, Gauthier-Villars, 1983

[2] C. Berge, *Hypergraphs : Combinatorics of Finite sets*, Vol. 45 of North-Holland Mathematical library, Elsevier Science Publ. B.V., Amsterdam, 1989.

[3] M. Gondran et M. Minoux, *Graphes et Algorithmes*, Eyrolles, 1993.

[4] J.C. Fournier, *Théorie des Graphes et Applications*, Hermes-Lavoisier.

[5] Roseaux, *Exercices résolus de Recherche Opérationnelle tome 1 : Graphe.*