Graphes et réseaux

Réseaux

Réseaux routiers

Réseaux

Réseaux de transports en commun

Réseaux

Réseaux de gaz

Réseaux

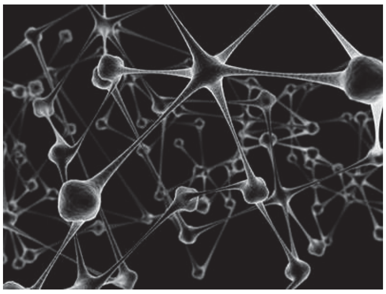
Réseaux d’eau

Réseaux

Réseaux d’ordinateurs

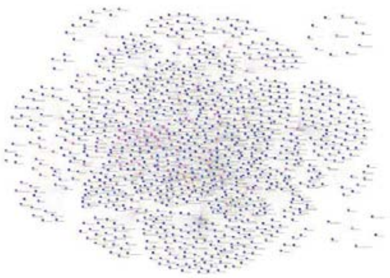
Réseaux

Réseaux de neurones



Réseaux

Réseaux sociaux

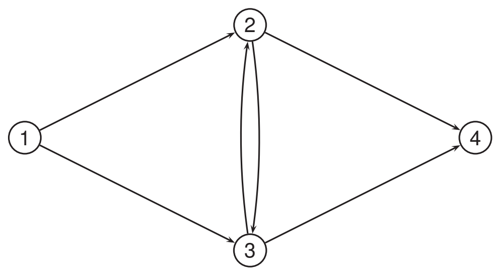


Réseaux

Formalisation mathématique du concept de réseaux

Graphe

Un graphe orienté $G = (N, A)$ est constitué d'un ensemble N de noeuds *et d'un ensemble A de paires de noeuds distincts, appelées* arcs.



Réseaux

Arête

Une arête est un arc dont on ignore l'orientation.

Chaîne

Une suite consécutive d'arêtes est appelée une chaîne.

Chemin

Une suite consécutive d'arcs est appelée un chemin.

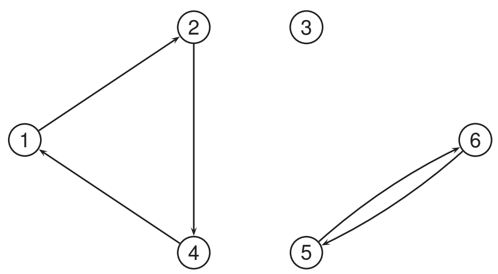
Graphe connexe

Un graphe $G = (N, A)$ est connexe si, quelque soit $i, j \in N$, il existe une chaîne de i à j.

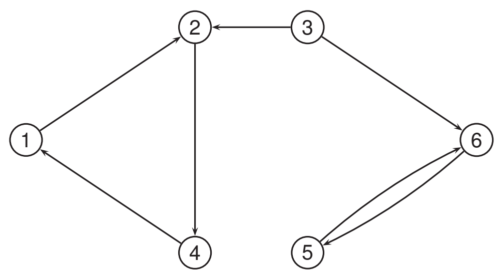
Graphe fortement connexe

Un graphe $G = (N, A)$ est fortement connexe si, quelque soit $i, j \in N$, il existe un chemin de i à j.

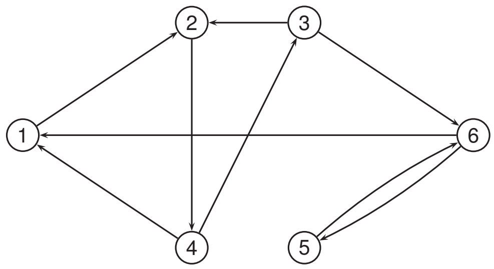
Graphe non connexe



Graphe connexe mais pas fortement

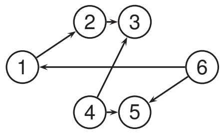


Graphe fortement connexe

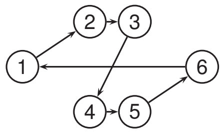


Réseaux

Cycle
Une chaîne dont les deux sommets extrémités sont identiques est un cycle.

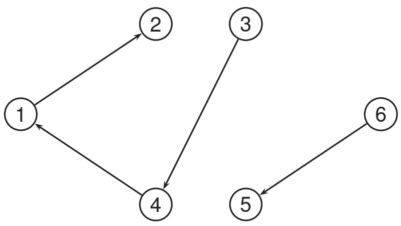


Circuit
Un chemin dont les deux sommets extrémités ne *sont pas* *identiques est un circuit.*



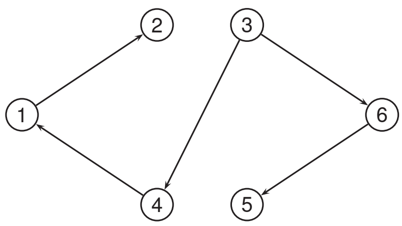
Réseaux

Forêt
Un graphe sans cycle est appelé une forêt.



Réseaux

Arbre
Un graphe connexe sans cycle est appelé un arbre.



Réseaux

- On supposera qu'il existe au plus un seul arc reliant deux noeuds dans une même direction.
- Dans ce cours, un graphe est un graphe *orienté*.

Réseau
Un réseau est un graphe, pour lequel des valeurs numériques ont été associées aux noeuds et/ou aux arcs.

- Longueur d'une route.
- Capacité d'un tuyau.
- Nombre de personnes habitants en un endroit.
- Flot de véhicules empruntant une autoroute.
- etc.

Flots

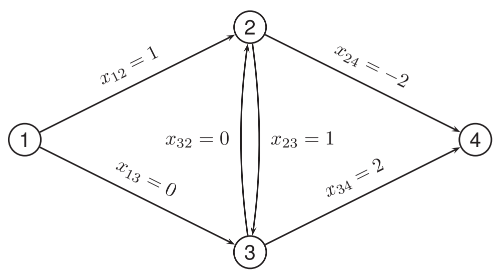
- Notations:
  - Noeuds : i et j.
  - Arc : (i, j).
  - Flot sur l'arc (i, j) : x_{ij}.
  - Si $x_{ij} < 0$, le flot va à contre sens.
- Vecteur de flots:

$$\{x_{ij} \text{ tel que } (i, j) \in A\}.$$

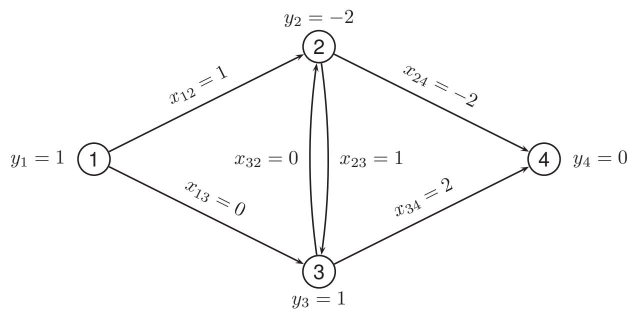
- Divergence : bilan des flots en un noeud i

$$y_i = \sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji}, \quad \forall i \in N.$$

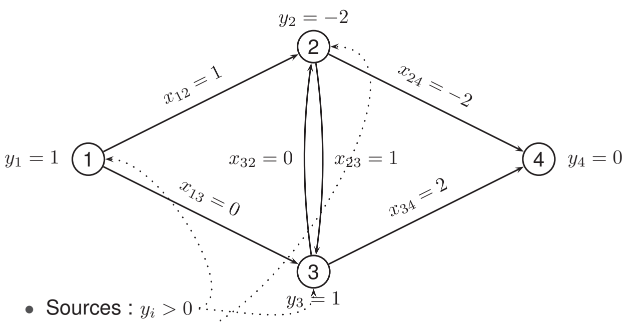
Flots



Flots



Flots

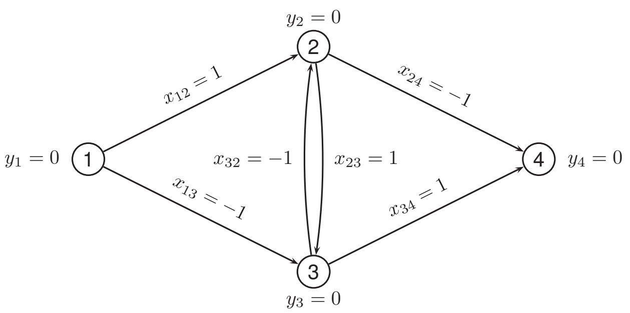


- Sources : $y_i > 0$
- Puits : $y_i < 0$

Flots

- On a toujours
$$\sum_{i \in N} y_i = 0.$$
- Si $y_i = 0, \forall i \in N$, on dit que le vecteur de flots est une circulation.

Flots : exemple de circulation



Problème de transbordement

- Une entreprise doit transporter ses produits de ses usines (lieux de production) vers ses clients.
- Elle désire minimiser ses coûts.
- Elle doit se plier aux contraintes de capacité du système de transport.
- Elle peut éventuellement transborder les marchandises en tout noeud du réseau.

Problème de transbordement

- Trouver un vecteur de flots :
  - qui minimise une fonction de coût (linéaire),
  - qui produise un vecteur de divergence donné,
  - qui vérifie les contraintes de capacité.

Problème de transbordement

Données

- a_{ij} : coût unitaire de transport sur l'arc (i, j),
- b_{ij} : flot minimum sur l'arc (i, j) (souvent 0),
- c_{ij} : capacité de l'arc (i, j),
- s_i : divergences désirées.
  - Si $s_i > 0$ alors s_i est l'offre en i, c.-à-d. ce qui est produit par l'usine située en i.
  - Si $s_i < 0$ alors s_i est la demande en i, c.-à-d. ce qui est commandé par le client situé en i.

Problème de transbordement

$$\min_x \sum_{(i,j)\in A} a_{ij}x_{ij}$$

sous contraintes

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = s_i \quad \forall i \in N \qquad \text{offre/demande}$$

$$b_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall (i,j) \in A \quad \text{capacités}$$

Problème de transbordement

- Il s'agit un problème d'optimisation linéaire.
- Il peut donc être résolu par l'algorithme du simplexe.
- Il généralise de nombreux problèmes dans les réseaux.

Le plus court chemin

- Le problème du plus court chemin consiste à déterminer le chemin de coût minimum reliant un noeud a à un noeud b.
- On peut le voir comme un problème de transbordement.
- Idée: on envoie une seule unité de flot de a à b.

Le plus court chemin

Données

- a_{ij} : longueur de l'arc (i, j),
- b_{ij} : 0,
- c_{ij} : 1,
- s_i :
  - Origine : $s_a = 1$.
  - Destination : $s_b = -1$.
  - Autres noeuds : $s_i = 0$, si $i \neq a$ et $i \neq b$.

Affectation

Je possède 4 chefs d'oeuvre que je désire vendre

Renoir

Van Gogh

Monet

Thaythay

Affectation

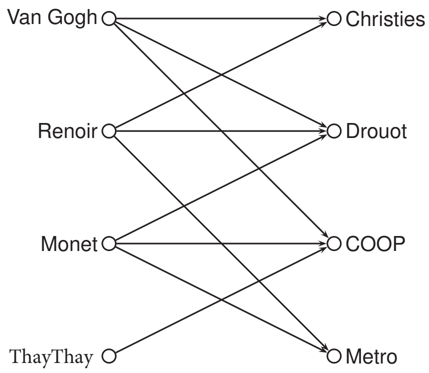
J'ai contacté quatre acheteurs qui ont fait des offres (en kCHF)

	Van Gogh	Renoir	Monet	ThayThay
Christie's	8000	11000	—	—
Drouot	9000	13000	12000	—
COOP	9000	—	11000	0.01
Metropolitan	—	14000	12000	—

Affectation

- Je désire vendre exactement une peinture à chaque acheteur.
- Quelle peinture dois je vendre à quel acheteur pour gagner un maximum ?
- On peut le voir comme un problème de transbordement.
- Représentation en réseau.

Affectation



Affectation

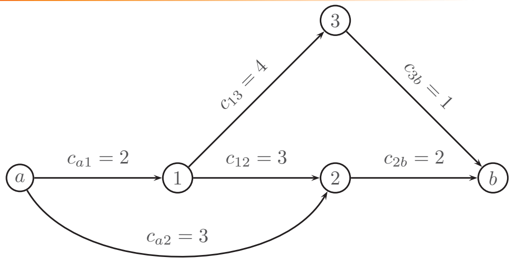
Données

- a_{ij} : valeur de l'offre, changée de signe,
- b_{ij} : 0,
- c_{ij} : 1.
- s_i :
  - Noeud i "oeuvre" : $s_i = 1$.
  - Noeud j "acheteur" : $s_j = -1$.

Flot maximal

- Une société pétrolière désire envoyer un maximum de pétrole via un réseau de pipelines entre un lieu a et un lieu b.
- Combien de litres par heure pourra-t-elle faire passer par le réseau ?
- Les capacités des pipelines (en kilolitres/heure) sont indiquées sur les arcs.

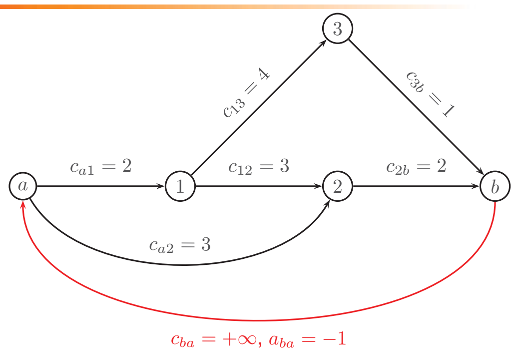
Flot maximal



Flot maximal

- On peut le voir comme un problème de transbordement.
- Il faut ajouter un arc artificiel.
- Idée : chaque unité de flot qui a réussi à passer à travers le réseau est ramenée artificiellement à a, en rapportant des bénéfices (coût négatif).

Flot maximal



Flot maximal

Données

- a_{ij} : $\begin{cases} 0 & \text{pour les arc réels} \\ -1 & \text{pour l'arc artificiel} \end{cases}$
- b_{ij} : 0,
- c_{ij} : capacités,
- $s_i = 0\ \forall i$: on désire une circulation.

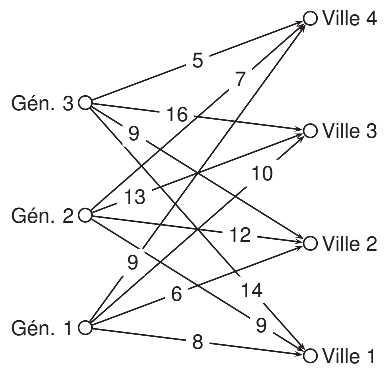
Problème de transport

- Une société électrique possède trois générateurs pour fournir 4 villes en électricité.
- Les générateurs produisent resp. 35, 50 et 40 GWh.
- Les villes consomment resp. 45, 20, 30 et 30 GWh.
- Les coûts de transport d'un GWh d'un générateur à une ville sont repris dans le tableau suivant.

	Ville 1	Ville 2	Ville 3	Ville 4
Gén. 1	8	6	10	9
Gén. 2	9	12	13	7
Gén. 3	14	9	16	5

- Comment approvisionner les villes à moindre coût ?

Problème de transport



Problème de transport

Données

- a_{ij} : prix entre générateur i et ville j,
- b_{ij} : 0,
- c_{ij} : $+\infty$,
- s_i: offre et demande

$$s_i = \begin{cases} \text{capacité de production} & \text{si } i = \text{générateur} \\ -\text{demande} & \text{si } i = \text{ville} \end{cases}$$

Résumé

- Le problème de transbordement généralise beaucoup de problèmes dans les réseaux.
- Il s'agit d'un problème d'optimisation linéaire.
- L'algorithme du simplexe peut être utilisé.
- Dans de nombreux cas, il est possible d'exploiter mieux la structure du problème afin d'obtenir un algorithme plus efficace.
- Exemple : problème du plus court chemin.

Algorithme du plus court chemin

Le plus court chemin

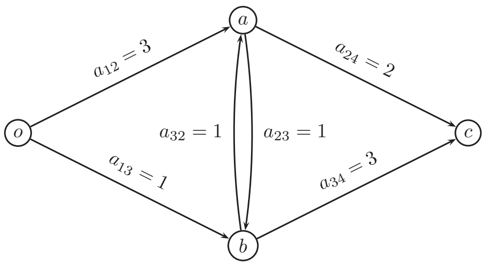
- Le problème du plus court chemin consiste à déterminer le chemin de coût minimum reliant un nœud a à un nœud b.
- On peut le voir comme un problème de transbordement.
- Cependant, il est plus efficace d'utiliser des algorithmes spécialisés.

Le plus court chemin

Problème :

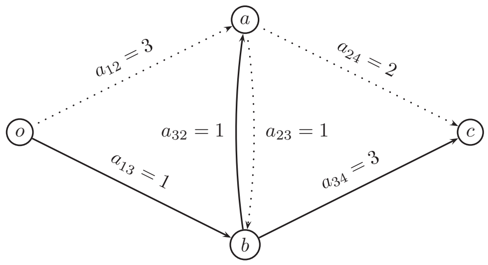
- Soit un réseau $G = (N, A)$.
- Un coût a_{ij} est associé à chaque arc $(i, j) \in A$:
  - distance,
  - temps de trajet,
  - etc.
- Soit un nœud appelé *origine*. Par convention, ce sera le nœud o.
- Nous cherchons le chemin de coût minimum reliant le nœud o à n'importe quel autre nœud du réseau.

Le plus court chemin



Le plus court chemin

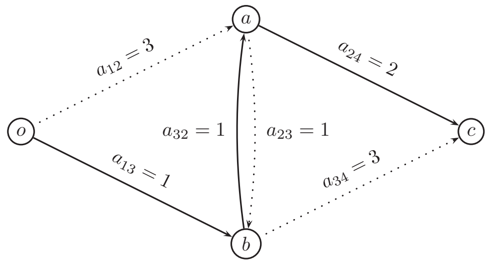
- La solution est un arbre.



Note : chaque nœud dans l'arbre a exactement un prédécesseur.

Le plus court chemin

- La solution n'est pas nécessairement unique.



Idée générale de l'algorithme

- Parcours systématique du réseau à partir de l'origine.
- A chaque nœud visité, une étiquette est associée.
- Cette étiquette est potentiellement mise à jour à chaque visite du nœud.

Conditions d'optimalité

- Soient $d_i \in \mathbb{R}$, $i \in N$ tels que
$$d_j \leq d_i + a_{ij} \ \ \forall (i,j) \in A.$$
- Soit P un chemin entre l'origine o et un nœud ℓ.
- Si
$$d_j = d_i + a_{ij} \ \ \forall (i,j) \in P,$$
alors P est un plus court chemin entre o et ℓ.

Conditions d’optimalité

Preuve:

- P est composé d’arcs

$$(o, i_1), (i_1, i_2), \dots, (i_k, \ell)$$

- Longueur de P:

$$L(P) = a_{oi_1} + a_{i_1 i_2} + \cdots + a_{i_k \ell}$$

- Comme $a_{ij} = d_j - d_i$,

$$L(P) = (d_{i_1} - d_o) + (d_{i_2} - d_{i_1}) + \cdots + (d_\ell - d_{i_k}) = d_\ell - d_o.$$

Conditions d’optimalité

- Soit Q un chemin quelconque entre o et ℓ.
- Q est composé d’arcs

$$(o, j_1), (j_1, j_2), \dots, (j_n, \ell)$$

- Longueur de Q:

$$L(Q) = a_{oj_1} + a_{j_1 j_2} + \cdots + a_{j_n \ell}$$

- Comme $a_{ij} \geq d_j - d_i$,

$$L(Q) \geq (d_{j_1} - d_o) + (d_{j_2} - d_{j_1}) + \cdots + (d_\ell - d_{j_m}) = d_\ell - d_o = L(P).$$

- La longueur de P est donc plus courte que la longueur de Q.
- Comme Q est arbitraire, P est le plus court chemin entre o et ℓ.

Algorithme

Idée :

- On démarre avec un vecteur d’étiquettes $(d_i)_{i \in N}$.
- On sélectionne un arc (i, j) qui viole les conditions d’optimalité, c.-à-d. tel que

$$d_j > d_i + a_{ij}.$$

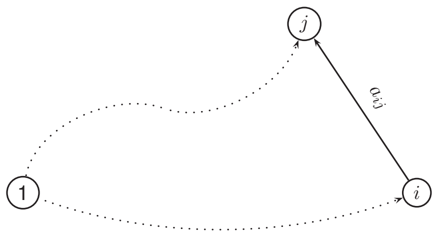
- On met à jour l’étiquette de j:

$$d_j = d_i + a_{ij}.$$

- Et ainsi de suite jusqu’à ce que tous les arcs vérifient la condition.

Interprétation

- d_i : longueur d’un chemin entre le nœud o et le nœud i.
- Si $d_j > d_i + a_{ij}$, chemin $o \to i \to j$ plus court que le chemin $o \to j$.



Exploration du graphe

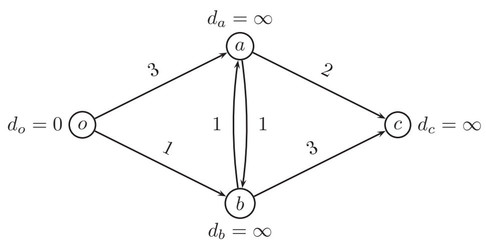
- Travailler nœud par nœud.
- Pour un nœud donné, traiter tous les arcs sortants.
- Dès qu'un nœud est atteint, on l'ajoute à la liste.
- Dès qu'un nœud est traité, on le supprime de la liste.
- On arrête lorsque la liste est vide.
- Notons V la liste des nœuds à traiter.

Algorithme

Initialisation
- Liste de nœud : $V = \{o\}$.
- Étiquettes : $d_o = 0$, $d_i = +\infty$, $\forall i \neq o$.

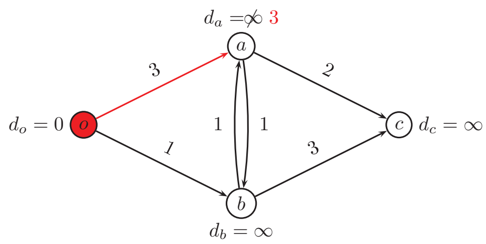
Itérations Tant que $V \neq \emptyset$,
- Choisir i dans V.
- $V = V \setminus \{i\}$.
- Pour chaque arc $(i, j) \in A$
  - Si $d_j > d_i + a_{ij}$,
    - $d_j = d_i + a_{ij}$.
    - $V = V \cup \{j\}$.

Exemple



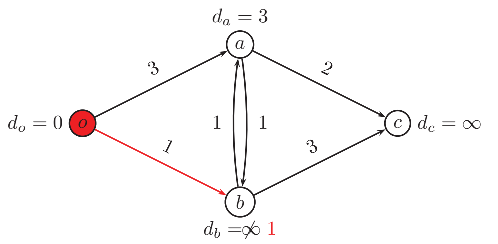
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o

Exemple



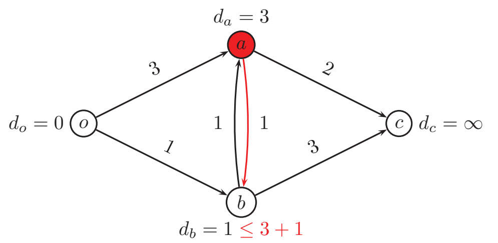
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a\}$	0	—	3	o	∞	—	∞	—	

Exemple



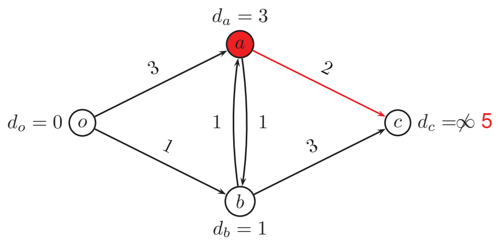
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a

Exemple



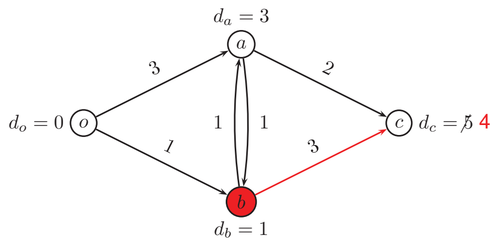
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a, b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b\}$	0	—	3	o	1	o	∞	—	

Exemple



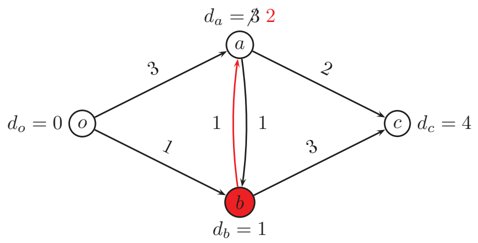
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	1
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b

Exemple



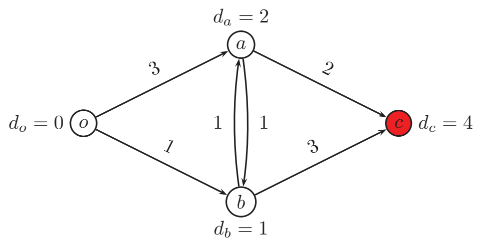
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b, c\}$	0	—	3	o	1	o	5	a	b
3	$\{c\}$	0	—	3	o	1	o	4	b	

Exemple



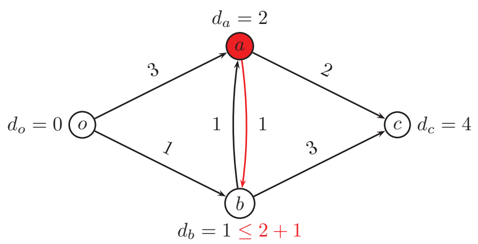
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c

Exemple



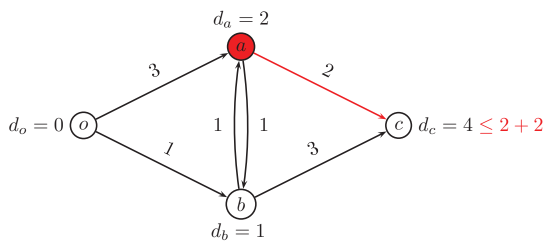
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c
4	$\{a\}$	0	—	2	b	1	o	4	b	a

Exemple



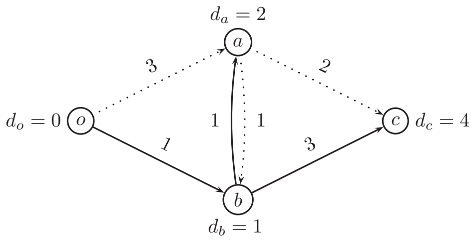
Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c
4	$\{a\}$	0	—	2	b	1	o	4	b	a
5	$\{\}$	0	—	2	b	1	o	4	b	

Exemple



Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c
4	$\{a\}$	0	—	2	b	1	o	4	b	a
5	$\{\}$	0	—	2	b	1	o	4	b	

Exemple



Iter	V	d_o		d_a		d_b		d_c		Traiter
0	$\{o\}$	0	—	∞	—	∞	—	∞	—	o
1	$\{a,b\}$	0	—	3	o	1	o	∞	—	a
2	$\{b,c\}$	0	—	3	o	1	o	5	a	b
3	$\{c,a\}$	0	—	2	b	1	o	4	b	c
4	$\{a\}$	0	—	2	b	1	o	4	b	a
5	$\{\}$	0	—	2	b	1	o	4	b	

Propriétés à la fin de chaque itération

- Si $d_i < \infty$, alors d_i est la longueur d'un chemin reliant o à i.
- Si $i \notin V$, alors
  - soit $d_i = \infty$ (le nœud n'a pas encore été atteint),
  - soit $d_j \leq d_i + a_{ij}$, $\forall j$ tel que $(i, j) \in A$ (les arcs sortant ont été traités).

Propriétés si l'algorithme se termine

- Pour tout nœud j tel que $d_j < \infty$,
  - $d_o = 0$;
  - d_j est la longueur du plus court chemin entre 1 et j;
  - Équation de Bellman :

$$d_j = \min_{(i,j) \in A} d_i + a_{ij} \text{ si } j \neq o.$$

- $d_j = \infty$ si et seulement s'il n'y a pas de chemin reliant 1 et j.
- Dans ce cas, le graphe n'est pas connexe.
- L'algorithme se termine si et seulement s'il n'y a aucun chemin commençant en o et contenant un circuit à coût négatif.

Algorithme de Dijkstra

- Algorithme "générique" ne précise pas comment choisir le nœud suivant à traiter.
- Dijkstra : le nœud i à traiter est celui correspondant à la plus petite étiquette.

Algorithme

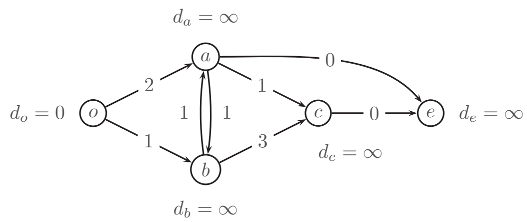
Initialisation

- Liste de nœuds: $V = \{o\}$.
- Étiquettes : $d_o = 0$, $d_i = +\infty$, $\forall i \neq o$.

Itérations Tant que $V \neq \emptyset$,

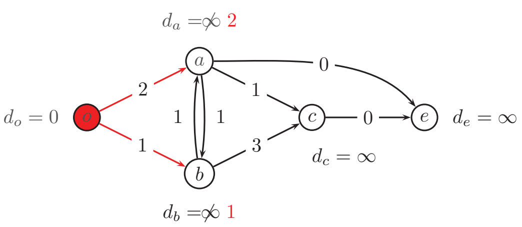
- Soit $i \in V$ tel que $d_i \leq d_j$, $\forall j \in V$.
- $V = V \setminus \{i\}$.
- Pour chaque arc $(i, j) \in A$
  - Si $d_j > d_i + a_{ij}$,
    - $d_j = d_i + a_{ij}$.
    - $V = V \cup \{j\}$.

Exemple



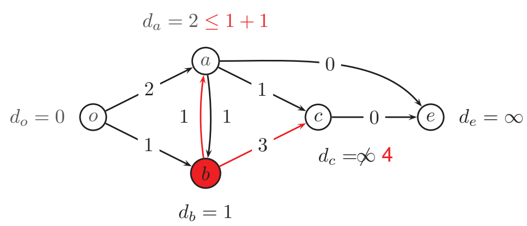
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o

Exemple



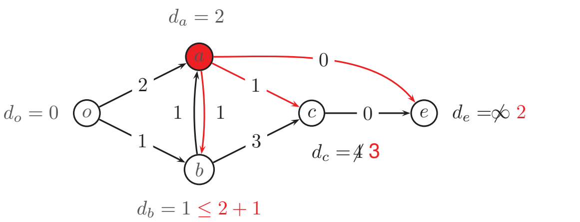
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b

Exemple



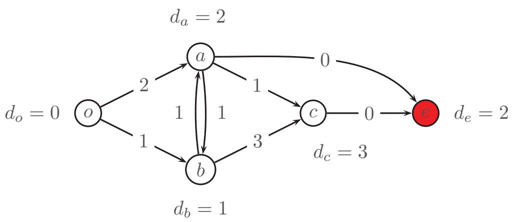
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a

Exemple



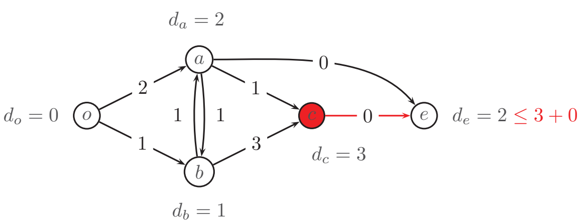
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c,e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e

Exemple



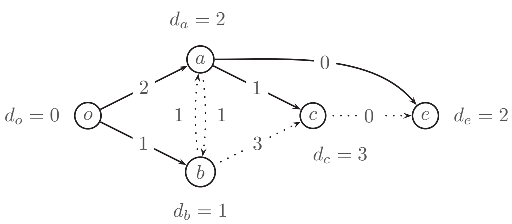
Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c,e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	{ c }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c

Exemple



Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c,e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	{ c }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c
5	{ }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	

Exemple



Iter	V	o	a	b	c	e	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	{ c,e }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	{ c }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c
5	{ }	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	

Exemple

Iter	V	o	a	b	c	e	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{a,b\}$	0 (-)	2 (o)	1 (o)	∞ (-)	∞ (-)	b
2	$\{a,c\}$	0 (-)	2 (o)	1 (o)	4 (b)	∞ (-)	a
3	$\{c,e\}$	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	e
4	$\{c\}$	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	c
5	$\{\}$	0 (-)	2 (o)	1 (o)	3 (a)	2 (a)	

Note : Chaque nœud n'a été traité qu'une seule fois.

Algorithme de Dijkstra

- Soit l'ensemble

$$W = \{i | d_i < \infty \text{ et } i \notin V\}.$$

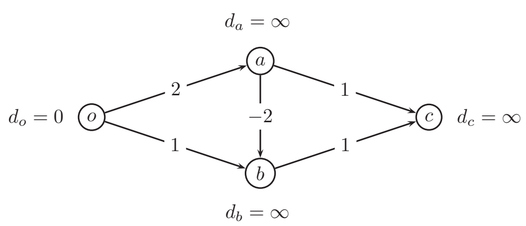
- Si les coûts sur les arcs sont non négatifs, alors à chaque itération
  - aucun nœud dans W au début de l'itération n'entre dans V lors de l'itération,
  - à la fin de l'itération, $d_i \leq d_j$ si $i \in W$ et $j \notin W$.

W : ensemble des étiquettes permanentes.

Notes

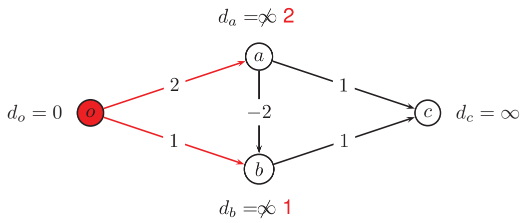
- Si l'on désire calculer le plus court chemin de o à b, on peut arrêter l'algorithme de Dijkstra dès que le nœud b est dans W.
- Si au moins un arc a un coût négatif, rien ne garantit le caractère permanent des étiquettes.

Exemple : coût négatif



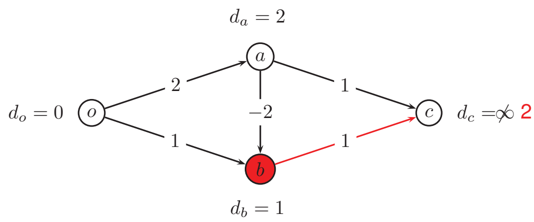
Iter	V	o	a	b	c	Traiter
0	$\{o\}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o

Exemple : coût négatif



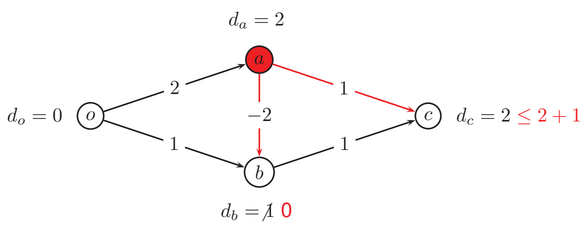
Iter	V	o	a	b	c	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	b

Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	2 (b)	a

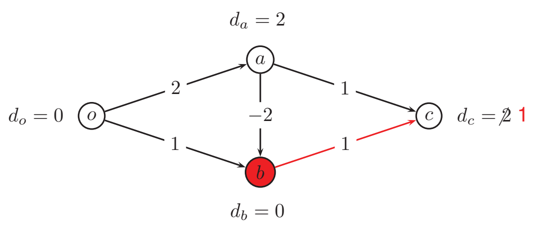
Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	2 (b)	a
3	{ b,c }	0 (-)	2 (o)	0 (a)	2 (b)	b

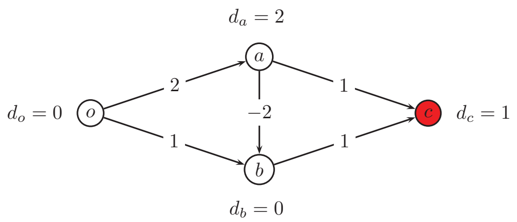
!!!

Exemple : coût négatif



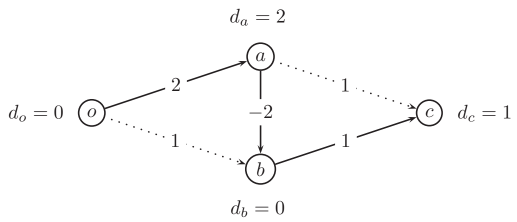
Iter	V	o	a	b	c	Traiter
0	{ o }	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	{ a,b }	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	{ a,c }	0 (-)	2 (o)	1 (o)	2 (b)	a
3	{ b,c }	0 (-)	2 (o)	0 (a)	2 (b)	b
4	{ c }	0 (-)	2 (o)	0 (a)	1 (b)	c

Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	$\{ o \}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{ a,b \}$	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	$\{ a,c \}$	0 (-)	2 (o)	1 (o)	2 (b)	a
3	$\{ b,c \}$	0 (-)	2 (o)	0 (a)	2 (b)	b
4	$\{ c \}$	0 (-)	2 (o)	0 (a)	1 (b)	c
5	$\{ \}$	0 (-)	2 (o)	0 (a)	1 (b)	

Exemple : coût négatif



Iter	V	o	a	b	c	Traiter
0	$\{ o \}$	0 (-)	∞ (-)	∞ (-)	∞ (-)	o
1	$\{ a,b \}$	0 (-)	2 (o)	1 (o)	∞ (-)	b
2	$\{ a,c \}$	0 (-)	2 (o)	1 (o)	2 (b)	a
3	$\{ b,c \}$	0 (-)	2 (o)	0 (a)	2 (b)	b
4	$\{ c \}$	0 (-)	2 (o)	0 (a)	1 (b)	c
5	$\{ \}$	0 (-)	2 (o)	0 (a)	1 (b)	

Dijkstra et coût négatif

- L'algorithme converge.
- Mais le concept d'étiquettes permanentes n'est plus pertinent.
- Toute implémentation basée sur cette propriété ne peut fonctionner qu'avec des coûts positifs.