

Arbres et arborescences

Recherche d'un arbre couvrant de poids minimal

Algorithme de Prim

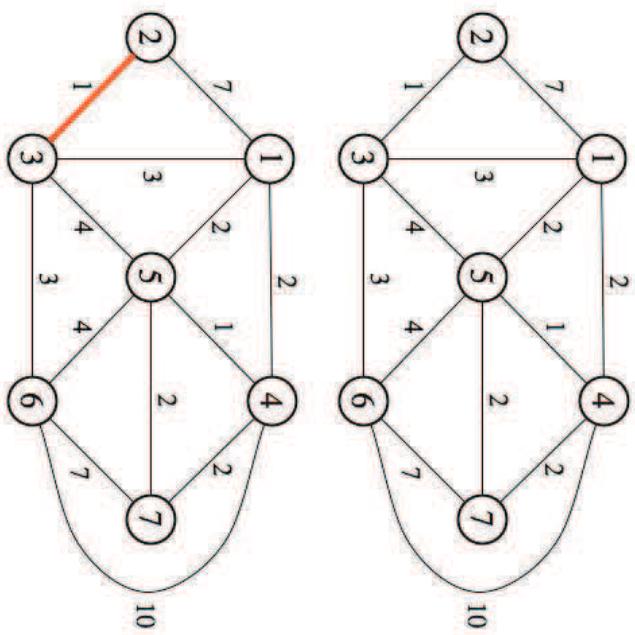
- On part d'un arbre initial réduit à un seul sommet
- A chaque itération, on greffe l'arête de poids le plus faible à l'arbre
- On arrête lorsque tous les sommets font partie de l'arbre

Algorithme de Kruskal

- On part du graphe tout entier et on le parcourt
- A chaque itération, on contracte l'arête de poids de faible
- On arrête lorsque le graphe se réduit à un sommet unique

Arbres et arborescences

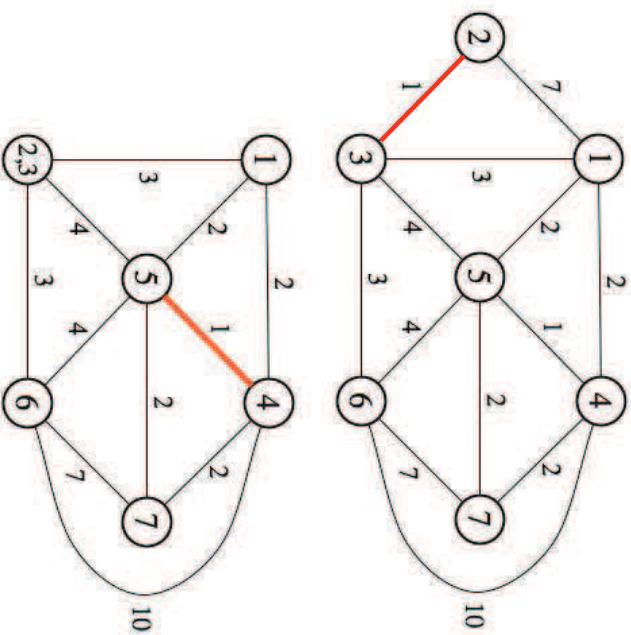
Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



3

Arbres et arborescences

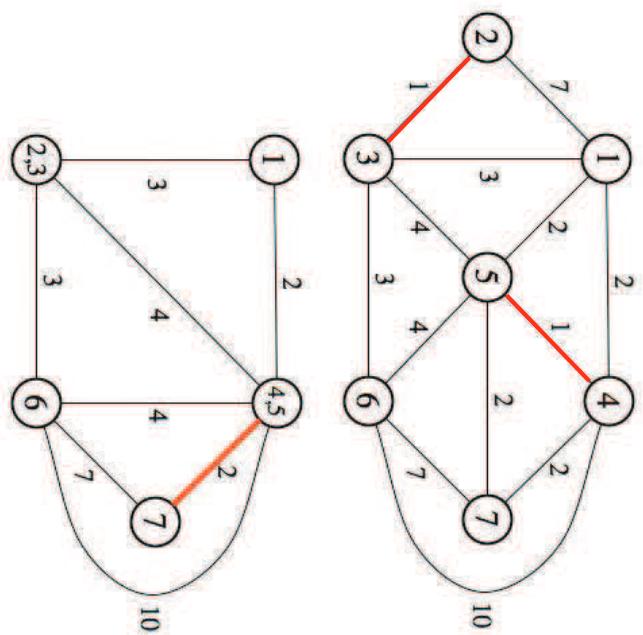
Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



4

Arbres et arborescences

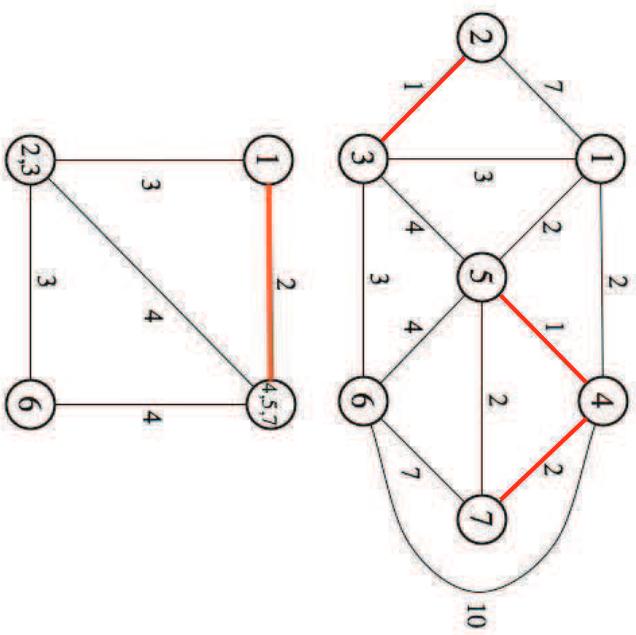
Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



5

Arbres et arborescences

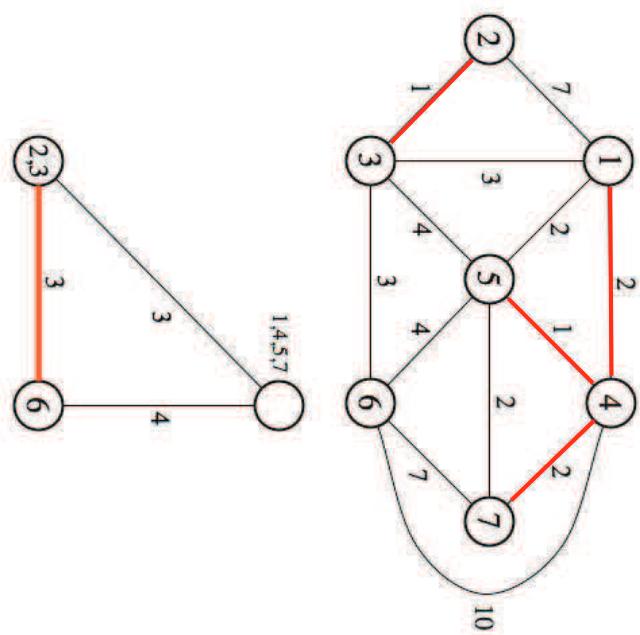
Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



6

Arbres et arborescences

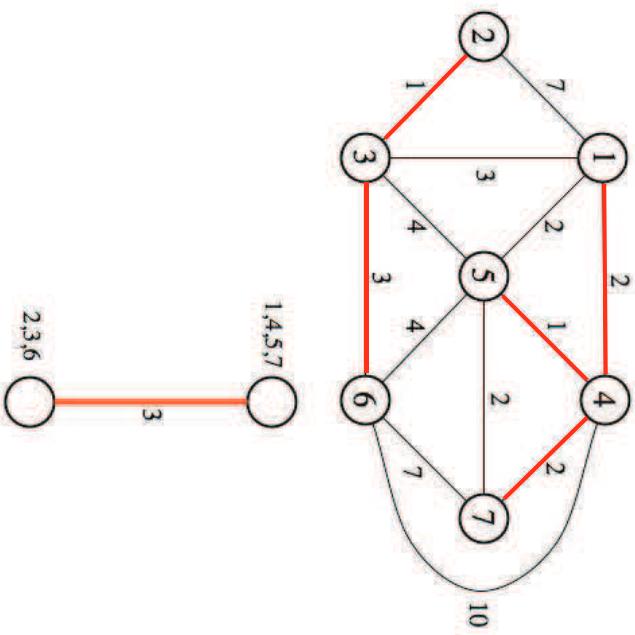
Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



7

Arbres et arborescences

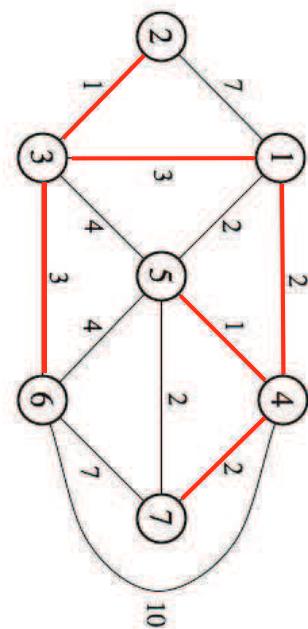
Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



8

Arbres et arborescences

Algorithme de Kruskal
Recherche d'un arbre couvrant de poids minimal



9

Réseaux, réseaux de transport et flots

Définition 22

- Un graphe fortement connexe, sans boucle et ayant plus d'un sommet, est appelé un **réseau**.
- On appelle **nœud** d'un réseau un sommet qui a plus de deux arcs incidents. Les autres sommets sont appelés **antinœuds**.
- On appelle **branche** tout chemin pour lequel seuls les premiers et derniers sommets sont des nœuds.
- Dans un graphe orienté G , un **flot** est l'affectation d'une valeur réelle à chaque arc de G , représentant une quantité transportée sur cet arc, de telle sorte que, en chaque sommet, la somme des flots entrants soit égale à la somme des flots sortants (loi de Kirchhoff : conservation des flux en chaque sommet).

Problème classique : recherche d'un flot maximal.

On se donne une capacité maximale sur chaque arc. Le problème du flot maximal consiste à déterminer un flot dont la valeur en un certain lieu est maximale. On peut, de plus, se donner un coût de transport d'une unité de flot sur chaque arc et chercher le flot maximal de coût minimal.

Réseaux, réseaux de transport et flots

Définition 23

- On appelle **réseau de transport** un graphe orienté antisymétrique valué $G=(X, A, C)$, sans boucle et dans lequel il existe :
 - un sommet x_1 sans prédecesseur nommé **entrée** ou **source** du réseau,
 - un sommet x_n sans successeur nommé **sortie** ou **puits** du réseau,
- et tel qu'au moins un chemin unisse x_1 à x_n dans G .
- La fonction de pondération C est supposée positive et l'on nomme **capacité** de l'arc a le nombre $C(a)$.

Définition 24

- Une fonction $\varphi(a)$ définie sur A et à valeurs réelles est un **flot** pour le réseau de transport si :
 - il est positif : $\varphi(a) > 0, \forall a \in A$
 - il vérifie la loi des nœuds de Kirchhoff $\forall x \neq x_1$ et $x \neq x_2$
- Il ne dépasse pas la capacité des arcs : $\varphi(a) \leq C(a), \forall a \in A$.

11

Réseaux, réseaux de transport et flots

Si x n'est ni x_1 ni x_n , la quantité entrante en x doit être égale à la quantité sortante que nous désignons par :

$$\varphi_x = \sum_{a \in A_x^-} \varphi(a) = \sum_{a \in A_x^+} \varphi(a)$$

Si φ est un flot sur un réseau de transport G , alors on a $\varphi_{x1} = \varphi_{xn}$

Cette quantité s'appelle la **valeur du flot**.

Recherche d'un flot complet

Définition 25

- Pour un flot φ dans un réseau de transport $G=(X, A, C)$, on dit qu'un arc est **saturé** si l'on a $\varphi(a) = C(a)$.
- Le flot est dit **complet** si tout chemin allant de x_1 à x_n contient au moins un arc saturé.

12

Réseaux, réseaux de transport et flots

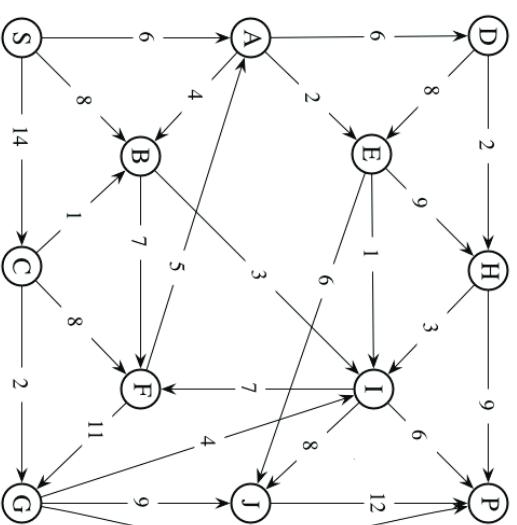
Recherche d'un flot complet

- On considère le graphe partiel engendré par les arcs non saturés par le flot.
- Si le flot n'est pas complet, il existe nécessairement un chemin μ allant de l'entrée à la sortie.
- On peut alors définir un nouveau flot pour le réseau en augmentant de 1 le flot de chacun des arcs constituant le chemin μ .
- La valeur du flot est alors également augmentée de 1.
- On peut donc progressivement augmenter la valeur d'un flot incomplet jusqu'à ce qu'il soit complet.
- En tenant compte des différences entre les capacités et la valeur du flot sur les arcs de μ , on peut connaître d'avance l'augmentation possible du flot.
- Cependant, le flot complet ainsi obtenu n'est pas, en général, le flot maximal.

13

Réseaux, réseaux de transport et flots

Recherche d'un flot complet : exemple

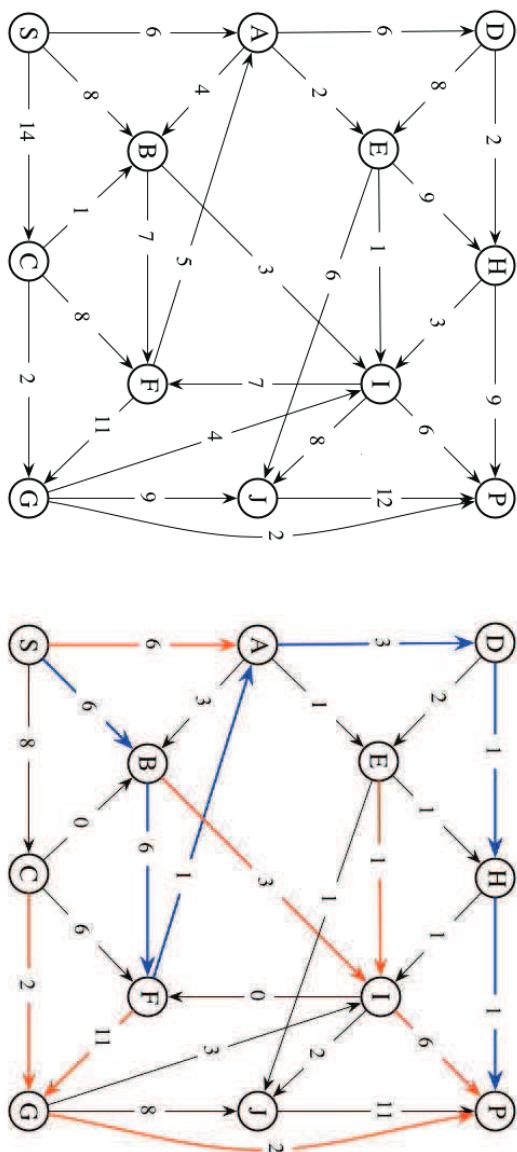


Capacités

14

Réseaux, réseaux de transport et flots

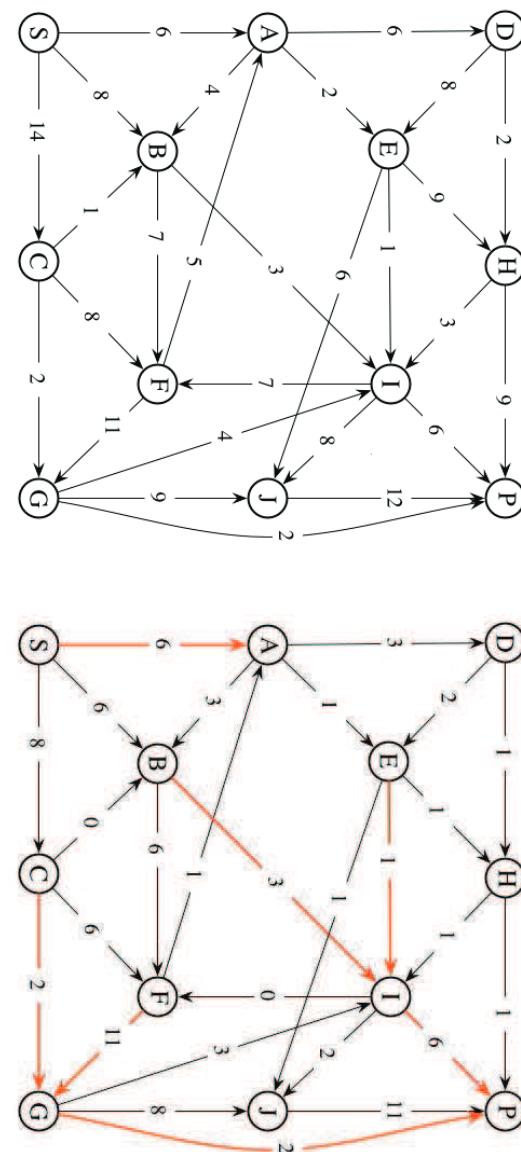
Recherche d'un flot complet : exemple



Capacités

Réseaux, réseaux de transport et flots

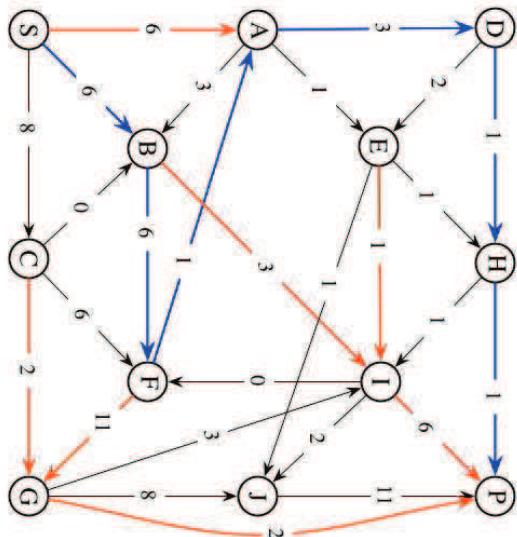
Recherche d'un flot complet : exemple



Capacités

Flot au jugé

15

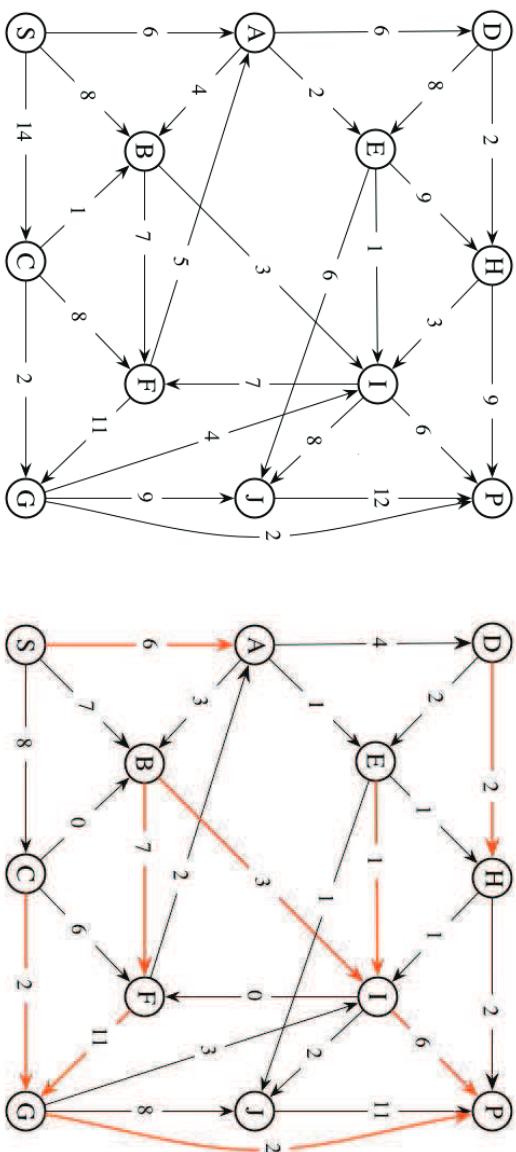


Amélioration du flot

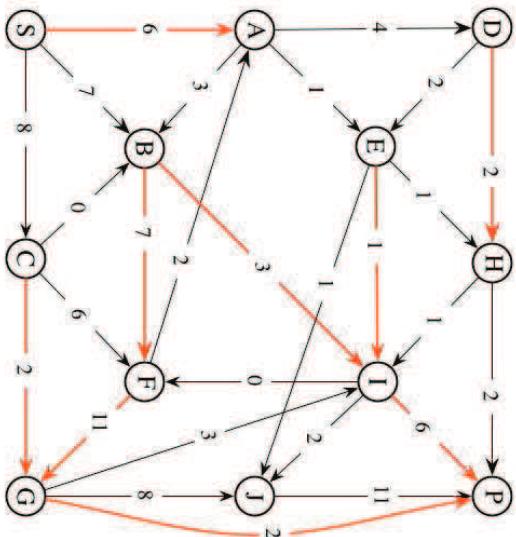
16

Réseaux, réseaux de transport et flots

Recherche d'un flot complet : exemple



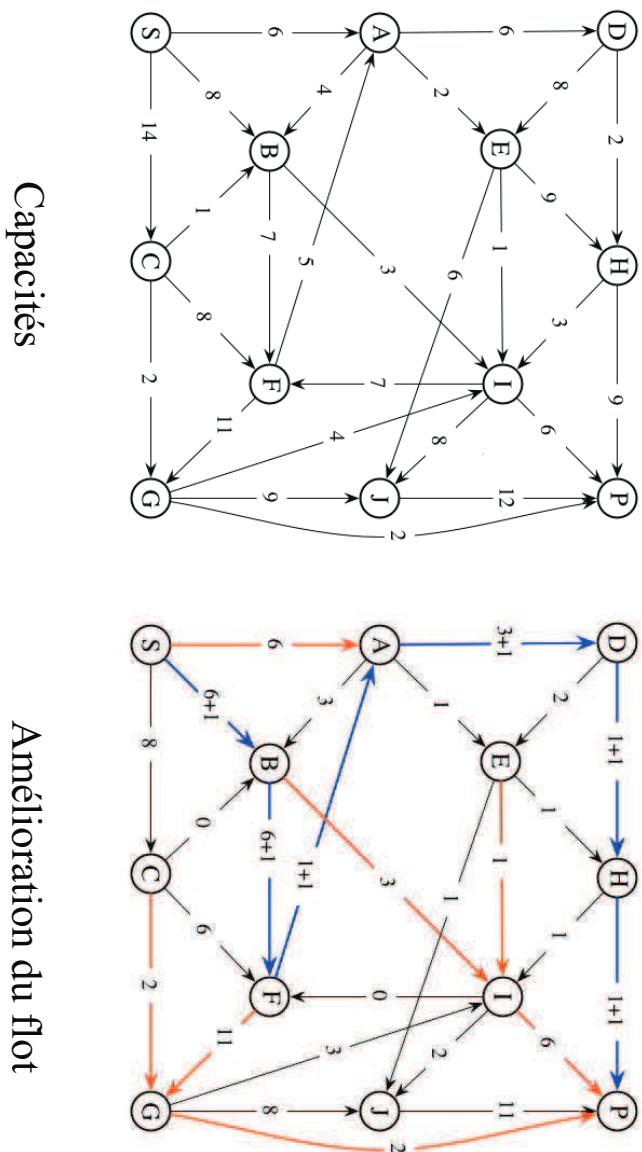
Capacités



Nouveau flot

Réseaux, réseaux de transport et flots

Recherche d'un flot complet : exemple



Capacités

Amélioration du flot

Réseaux, réseaux de transport et flots

Amélioration du flot

Processus d'étiquetage à partir du flot complet

On place une étiquette $\boxed{+}$ en x_1 ; soit x un sommet déjà marqué

- On marque avec une étiquette $\boxed{+x}$ tout successeur y non marqué de x pour lequel le flot n'est pas à son maximum

- On marque avec une étiquette $\boxed{-x}$ tout prédecesseur y non marqué de x pour lequel le flot n'est pas nul

Role de l'étiquette

indiquer si le flot peut être augmenté dans le sens de parcours (étiquette $\boxed{+x}$) ou diminué s'il est dans le sens contraire (étiquette $\boxed{-x}$).

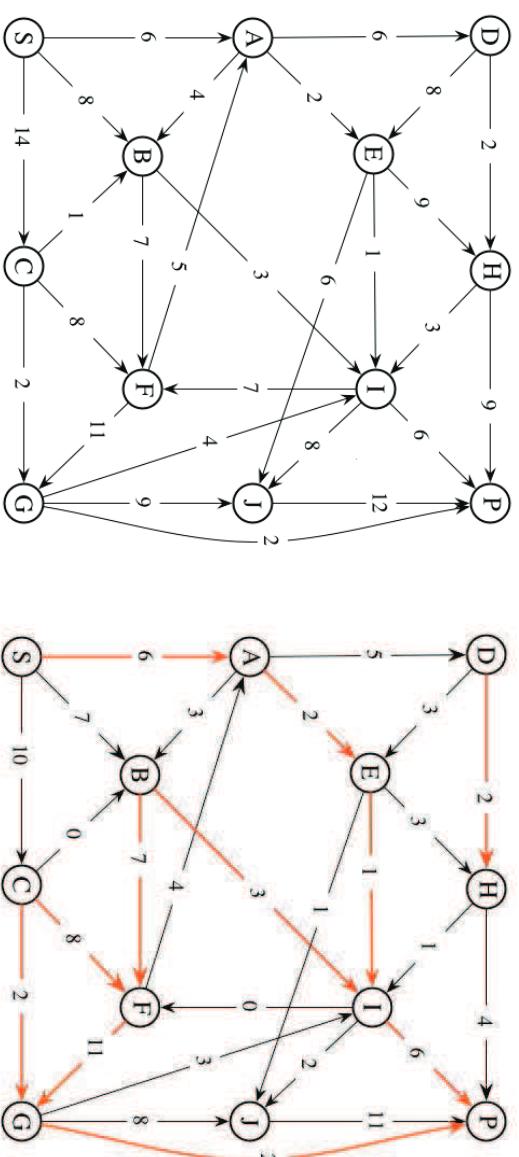
Si l'on parvient jusqu'au marquage du sommet x_n avec cette procédure, c'est qu'il existe une chaîne μ de x_1 à x_n dont tous les sommets sont marqués avec l'indice du sommet précédent au signe près.

Le flot peut alors être augmenté d'une unité.

19

Réseaux, réseaux de transport et flots

Recherche d'un flot maximal : exemple



Capacités

Flot complet

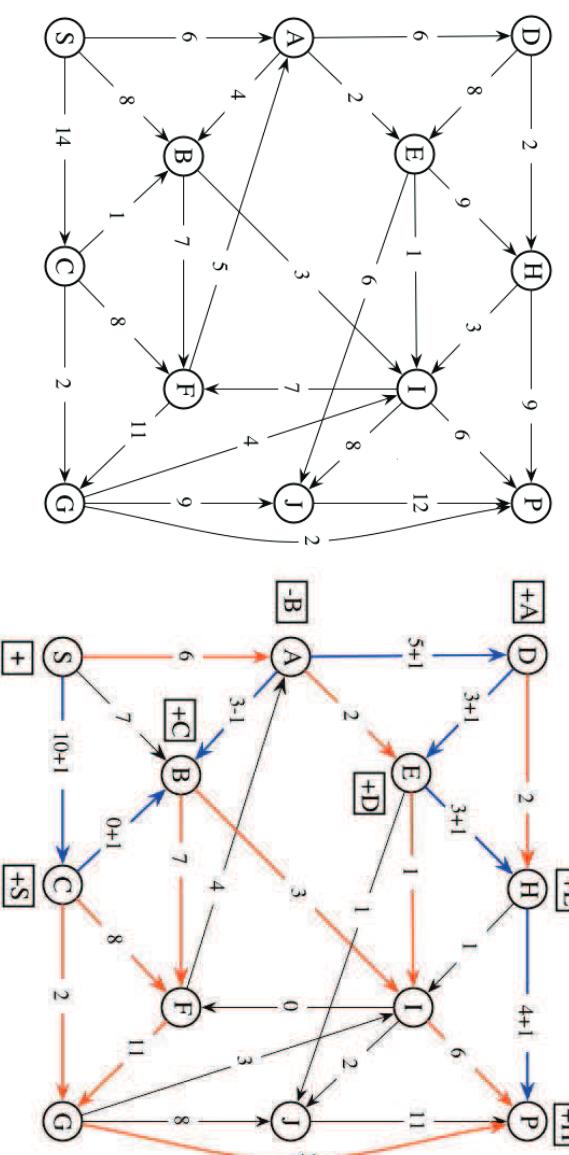
20

Réseaux, réseaux de transport et flots

Recherche d'un flot maximal : exemple

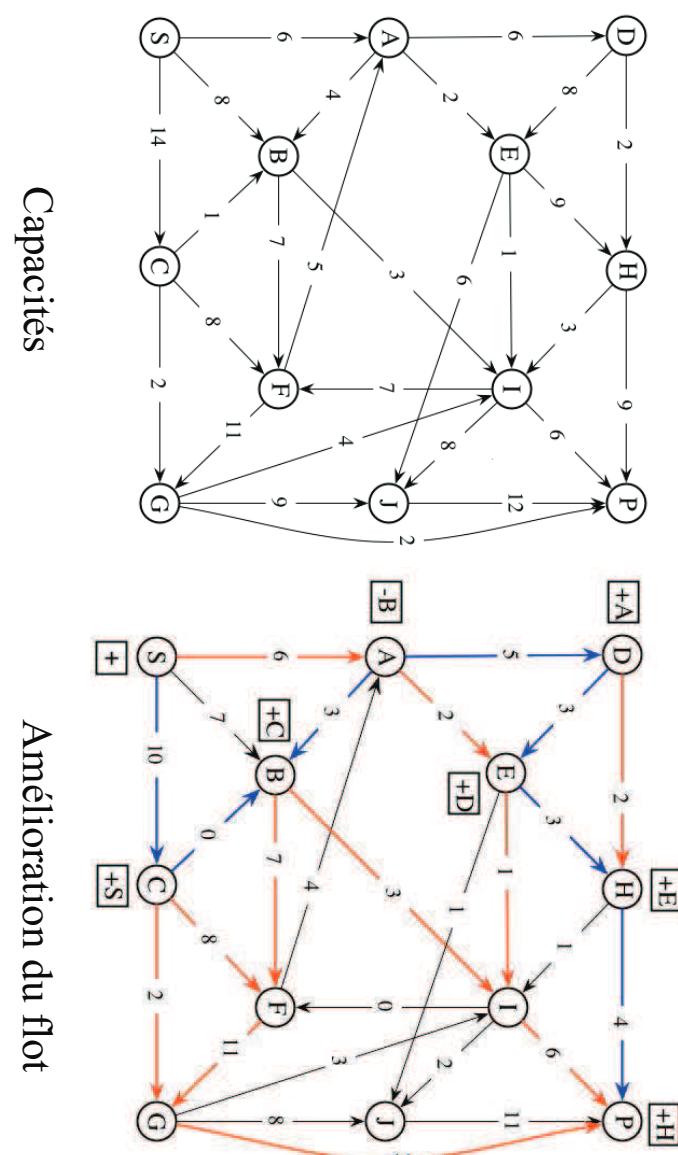
Réseaux, réseaux de transport et flots

Recherche d'un flot maximal : exemple



Capacités

Amélioration du flot

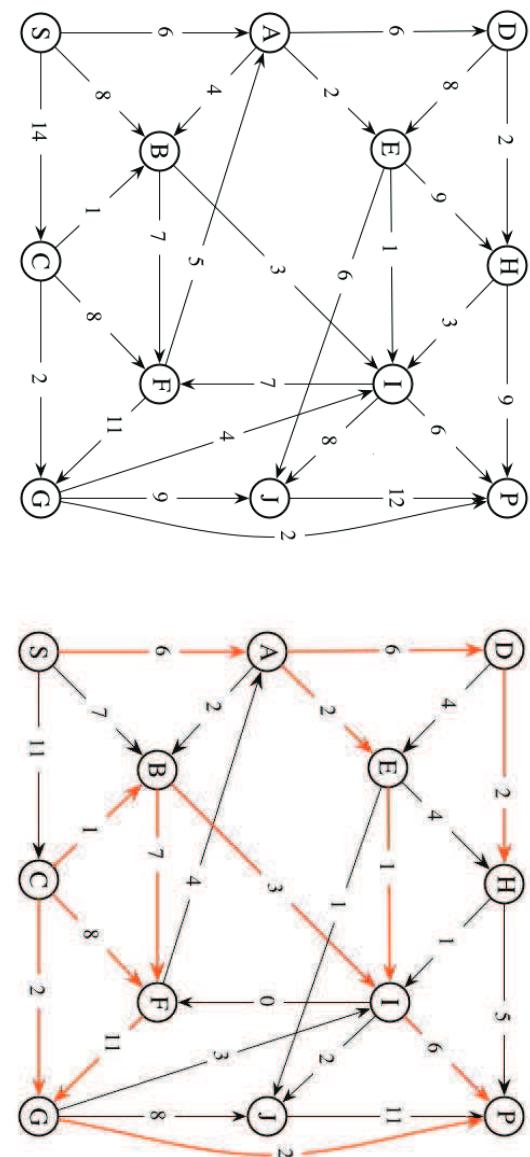


Capacités

Amélioration du flot

Réseaux, réseaux de transport et flots

Recherche d'un flot maximal : exemple



Capacités

Flot maximal

23

Réseaux, réseaux de transport et flots

Amélioration du flot : graphe d'écart ou réseau résiduel

Définition 26

- Soit un réseau de transport $G=(X, A, C)$ possédant un flot complet ϕ . On appelle **graphe d'écart** ou **réseau résiduel**, le réseau $G'(\phi)=(X, A', C')$ tel que :
 - si $a \in A$ et $\phi(a) < C(a)$ alors $a \in A'$ et $C'(a) = C(a) - \phi(a)$
 - si $a \in A$ et $\phi(a) = C(a)$ alors $a \notin A'$,
 - si $a = (x, y) \in A$ et $\phi(a) > 0$ alors $a^{-1} = (y, x) \in A'$ et $C'(a^{-1}) = \phi(a)$

Le réseau résiduel indique le long de quels arcs on peut augmenter ou diminuer le flot.
L'intérêt du graphe d'écart apparaît dans le théorème suivant.

Théorème 4

- Soit ϕ un flot de G (de x_1 à x_n) et $G'(\phi)$ le réseau résiduel associé à ϕ .
Une condition nécessaire et suffisante pour que le flot ϕ soit maximal est qu'il n'existe pas de chemin de x_1 à x_n dans $G'(\phi)$.

24

Réseaux, réseaux de transport et flots

Algorithme 4 (Ford-Fulkerson) Recherche d'un flot maximum

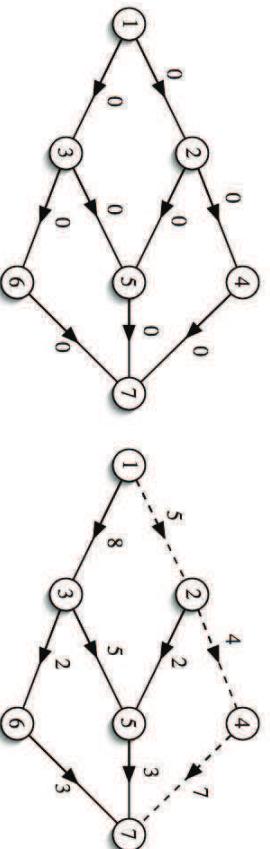
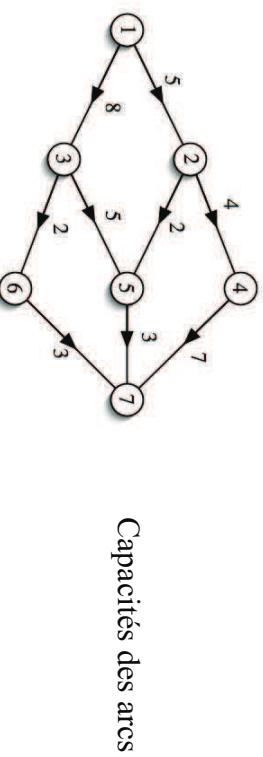
- Itération $k = 0$
- Partir d'un flot initial ϕ^0 compatible avec les contraintes de capacité par exemple $\phi^0 = (0 \ 0 \ 0 \ 0 \dots 0)$

- A l'itération k , soit ϕ^k le flot courant
 - Rechercher un chemin μ^k de x_1 à x_n dans le graphe d'écart $G'(\phi^k)$.
S'il n'en existe pas, FIN : le flot ϕ^k est maximal
 - Soit ε^k la capacité résiduelle du chemin μ^k (minimum des capacités résiduelles)
 - Définir le flot ϕ^{k+1} par :
 - $\phi_a^{k+1} = \phi_a^k + \varepsilon^k$ si $a \in \mu^k$ et si a est orientée dans le sens de μ^k
 - $\phi_a^{k+1} = \phi_a^k - \varepsilon^k$ si $a \in \mu^k$ et si a est orientée dans le sens contraire de μ^k
 - Faire $k \leftarrow k + 1$

25

Réseaux, réseaux de transport et flots

Exemple de recherche d'un flot maximum par l'algorithme de Ford et Fulkerson

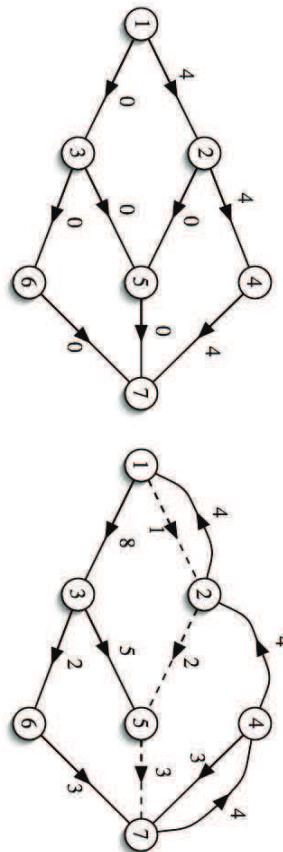
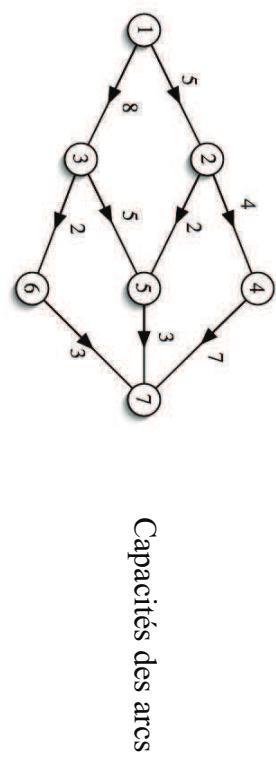


Flot initial et graphe d'écart

26

Réseaux, réseaux de transport et flots

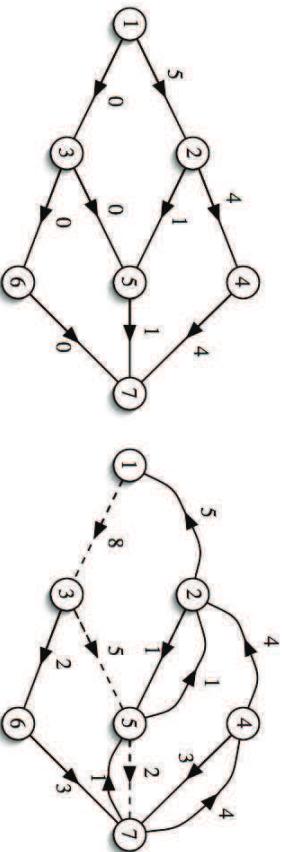
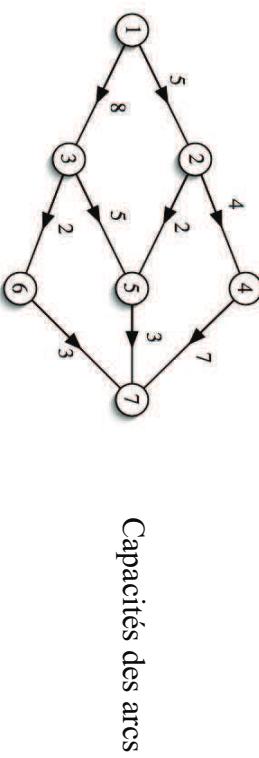
Exemple de recherche d'un flot maximum par l'algorithme de Ford et Fulkerson



27

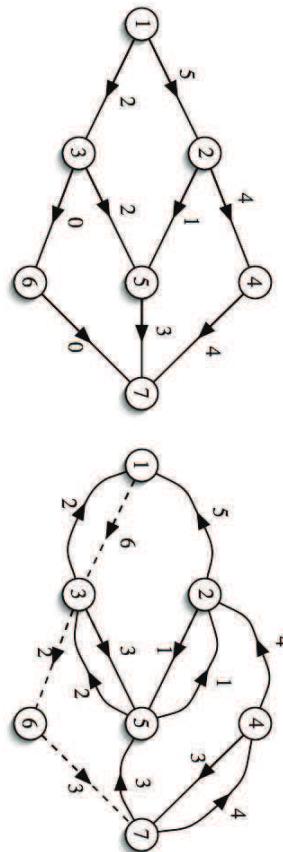
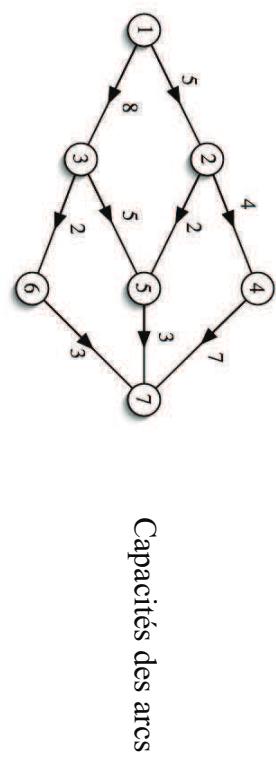
Réseaux, réseaux de transport et flots

Exemple de recherche d'un flot maximum par l'algorithme de Ford et Fulkerson



Réseaux, réseaux de transport et flots

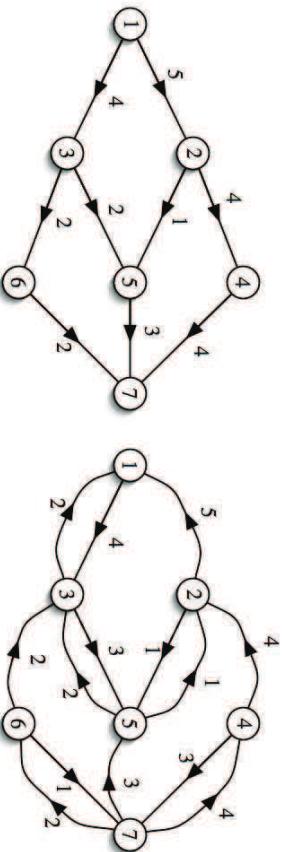
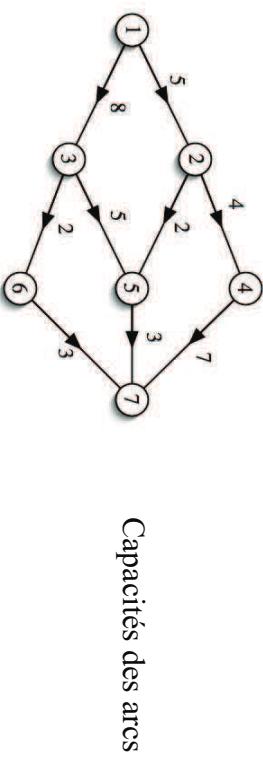
Exemple de recherche d'un flot maximum par l'algorithme de Ford et Fulkerson



29

Réseaux, réseaux de transport et flots

Exemple de recherche d'un flot maximum par l'algorithme de Ford et Fulkerson



Réseaux, réseaux de transport et flots

Recherche d'un flot maximal à coût minimal

- En pratique, on associe à la plupart des réseaux de transport, une notion de coût unitaire du transport qui se traduit par l'association, à chaque arc a du réseau, d'un nombre réel $w(a)$ représentant ce coût unitaire.

- Le coût total d'un flot φ est défini comme la somme des coûts associés aux flots élémentaires

$$W(\varphi) = \sum_{a \in A} w(a)\varphi(a)$$

- \Rightarrow Recherche, parmi les flots à valeur maximale, de celui qui est à coût minimal.

- Adaptation de l'algorithme de Ford et Fulkerson.

31

Réseaux, réseaux de transport et flots

Recherche d'un flot maximal à coût minimal

Algorithme de Busaker-Gowen

- A l'itération numéro k , le flot courant φ^k est supposé être un flot de valeur φ_0^k et de coût minimal parmi l'ensemble de tous les flots de valeurs φ_0^k .
- Soit $G'(\varphi^k)$ le graphe d'écart relatif à φ^k ; on attribue aux arcs de $G'(\varphi^k)$ les coûts w' et les capacités c' suivantes :
 - si $a = (x_i, x_j) \in A$ et $\varphi(a) < c(a)$,
l'arc a^+ = (x_i, x_j) a un coût $w'(a) = w(a)$ et une capacité $c'(a) = c(a) - \varphi(a) > 0$
 - si $a = (x_i, x_j) \in A$ et $\varphi(a) > 0$,
l'arc a^- = (x_j, x_i) a un coût $w'(a) = -w(a)$ et une capacité $c'(a) = \varphi(a)$.
- Soit alors μ^k un chemin de coût minimal relativement aux coûts w' de x_1 à x_n sur le graphe d'écart $G'(\varphi^k)$; on note ε^k la capacité résiduelle de ce chemin.
- On définit alors le flot φ^{k+1} comme dans l'algorithme de Ford et Fulkerson.
Si μ est le vecteur associé au chemin μ^k , on a $\varphi^{k+1} = \varepsilon^k \mu$

32

Couplages

Couplages

Problème important en particulier dans les graphes **biparti** (graphes dont les sommets peuvent être classés en deux sous-ensembles disjoints).

Permet de résoudre des problèmes d'affectation.

Lien étroit avec une classe de problèmes de programmation linéaire en nombres entiers.

Définition 27

- Etant donné un graphe simple non orienté $G=(X, A)$, un **coupillage** est un sous-ensemble d'arêtes $K \subset A$ tel que deux arêtes quelconques de K ne sont pas adjacentes.
- Un sommet $i \in X$ est dit **saturé** par le coupillage $K \subset A$ s'il existe une arête de K incidente à i .
- Un coupillage K qui sature tous les sommets du graphe est appelé **coupillage parfait**.
- Un **coupillage maximal** est un coupillage de cardinalité maximale.

33

Couplages

Couplage maximal

Définition 28

- Etant donné un graphe simple non orienté $G=(X, A)$, on appelle **chaîne alternée** (relativement à K) une chaîne élémentaire de G dont les arêtes appartiennent alternativement à K et à $A-K$.
- Une **chaîne alternée augmentante** ou **chaîne améliorante** est une chaîne alternée joignant deux sommets insaturnés.

Définition 29

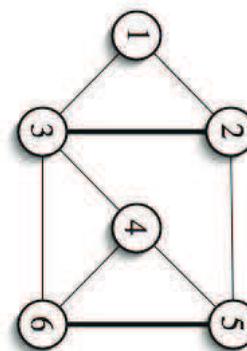
- Etant donné un coupillage $K \subset A$, considérons une chaîne alternée L dont chaque extrémité est un sommet insaturé.
- Le coupillage K' obtenu en échangeant le rôle des arêtes appartenant et n'appartenant pas au coupillage le long de la chaîne L est encore un coupillage de G .
- Cette opération qui fait passer du coupillage K au coupillage K' est appelé **transfert** le long de la chaîne alternée L .

34

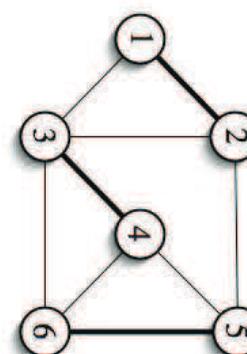
Couplages

Couplage maximal

- On choisit de dessiner en trait épais, les arêtes du couplage (arêtes (2, 3) et (5, 6)) et en trait fin les arêtes n'appartenant pas au couplage.
- La chaîne $L = \{1, 2, 3, 4\}$ est une chaîne alternée dont les sommets sont insaturés ; c'est donc une chaîne augmentante.
- Un transfert le long de cette chaîne produit le nouveau couplage $K' = \{(1, 2), (3, 4), (5, 6)\}$ de cardinalité égale à 3.



Couplage de cardinalité égale à 2



Couplage de cardinalité égale à 3

35

Couplages

Couplage maximal

Théorème 5

- Un couplage est maximal si et seulement s'il n'existe pas de chaîne alternée augmentante relativement à K .

Procédure de construction d'un couplage maximal

- Initialiser $K = \emptyset$.
- Trouver une chaîne améliorante K_a de K , effectuer le transfert le long de cette chaîne et remplacer le couplage K par celui ainsi obtenu (dont la cardinalité a nécessairement augmenté de 1).
- Répéter l'étape précédente jusqu'à ce qu'il ne soit plus possible de créer une nouvelle chaîne améliorante ; K est alors un couplage maximal.

→ Problème : savoir construire une chaîne améliorante

Couplages

Couplage maximal

Construction d'une chaîne alternée augmentante

⇒ Arbre alterné T' de racine i_0 tel que

- $T' = (Y, T)$ est un sous-graphe partiel connexe et sans cycle de G .
- Pour tout $j \in Y$, la chaîne (unique) $L_{T'}(j)$ de l'arbre entre i_0 et j est une chaîne alternée.
- Les chaînes reliant i_0 aux sommets pendants ont un nombre impair d'arêtes.

37

Couplages

Couplage maximal

Construction d'un arbre alterné

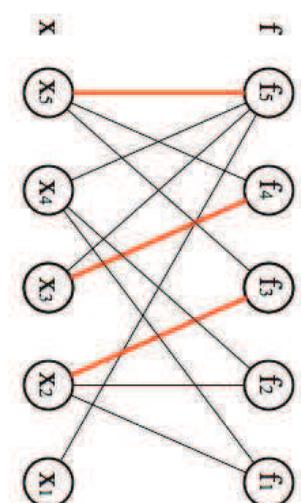
- On choisit comme racine de l'arbre (niveau 0) un sommet insature.
- Au niveau i impair, on insère dans l'arbre un sommet adjacent à l'un des sommets insérés au niveau $i - 1$ par le biais d'une arête **n'appartenant pas au couplage** ainsi que cette arête.
- Au niveau i pair, on insère dans l'arbre un sommet adjacent à l'un des sommets insérés au niveau $i - 1$ par le biais d'une arête **appartenant au couplage** ainsi que cette arête.
- On continue à construire progressivement cet arbre jusqu'à l'**insertion**, à un niveau impair, d'un **sommet insaturné**, ou bien jusqu'à ce que l'on ne puisse plus insérer de sommet.
- S'il existe une chaîne alternée augmentante du couplage courant, on finit nécessairement par insérer un sommet insaturné. La chaîne reliant s à la racine de l'arbre est alors une **chaîne alternée augmentante**.

38

Couplages

Couplage maximal

- Construction d'un arbre alterné : exemple



x_1

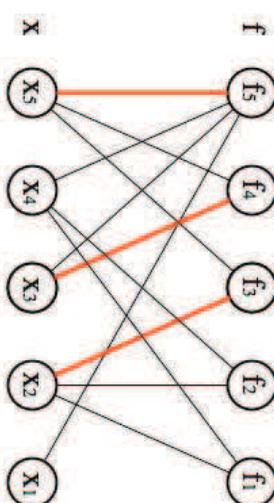
Construction d'un arbre alterné de racine x_1 .

39

Couplages

Couplage maximal

- Construction d'un arbre alterné : exemple



x_1

f_5

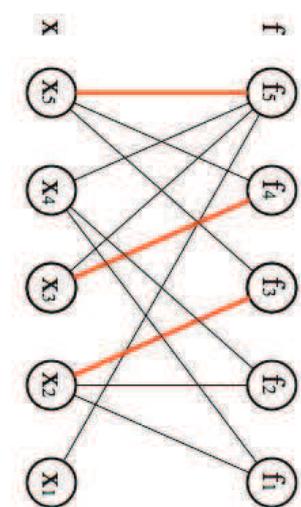
Au niveau 1 de l'arbre, la seule possibilité est d'insérer l'arête (x_1, f_5) et le sommet f_5 .

40

Couplages

Couplage maximal

- Construction d'un arbre alterné : exemple



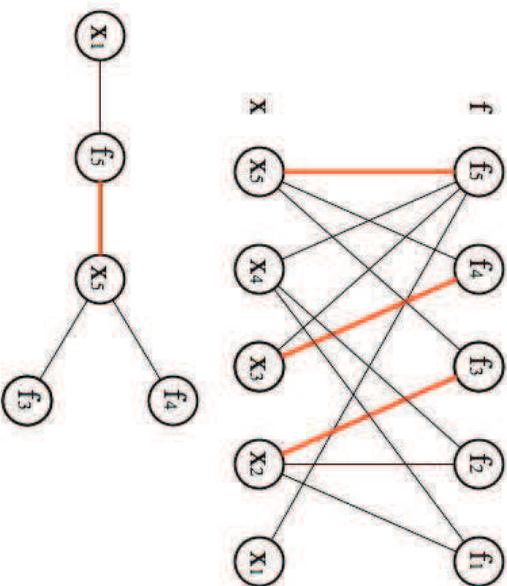
Au niveau 2, on insère l'arête (f_5, x_5) appartenant au couplage (c'est encore la seule possibilité).

41

Couplages

Couplage maximal

- Construction d'un arbre alterné : exemple



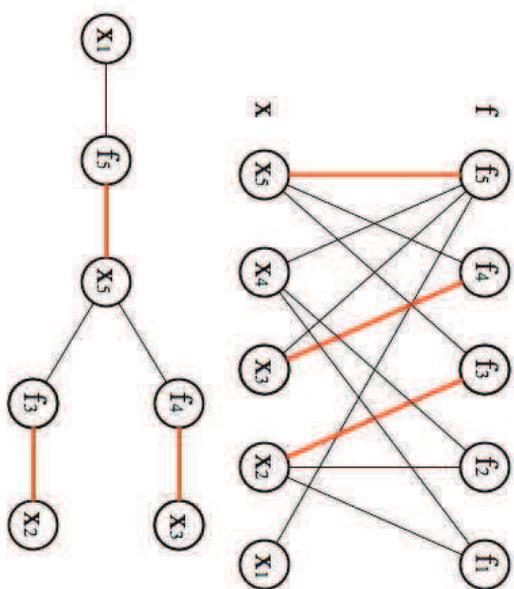
Au niveau 3, les 2 arêtes (x_5, f_3) et (x_5, f_4) n'appartenant pas au couplage peuvent être insérées.

42

Couplages

Couplage maximal

Construction d'un arbre alterné : exemple



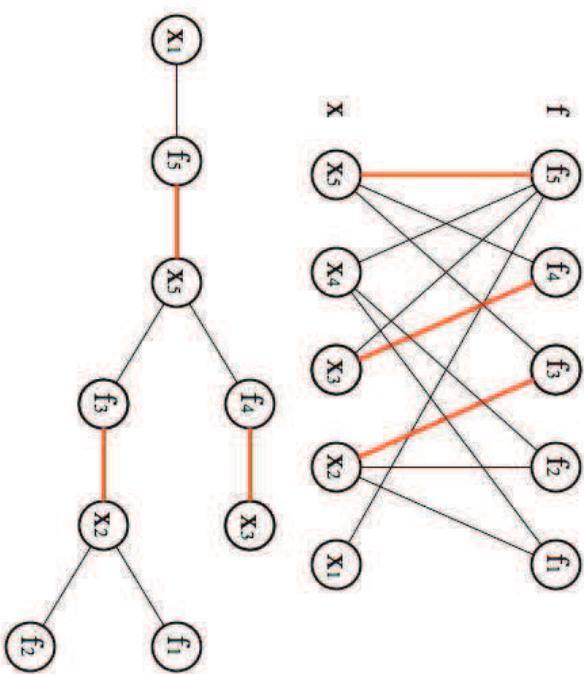
On insère ensuite les arêtes (f_4, x_3) et (f_3, x_2) ainsi que les sommets (saturés) correspondants.

43

Couplages

Couplage maximal

Construction d'un arbre alterné : exemple



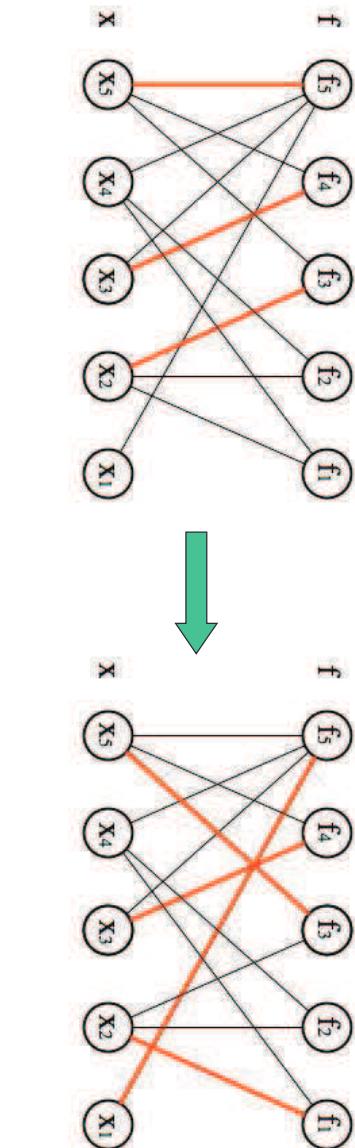
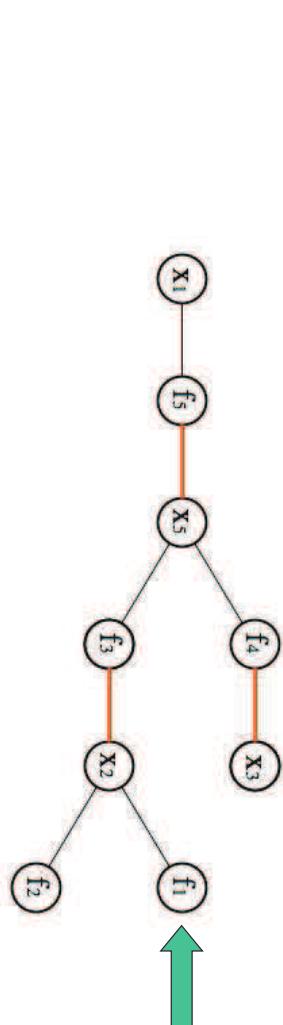
Le sommet x_3 est alors un sommet pendant. On insère, au niveau 5, les arêtes (x_2, f_1) et (x_2, f_2) .

44

Couplages

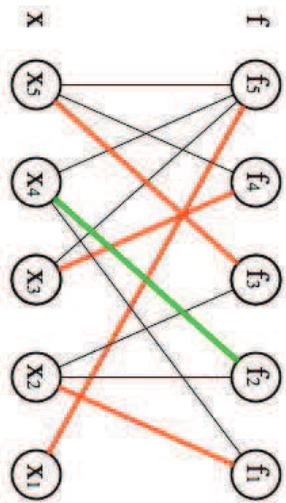
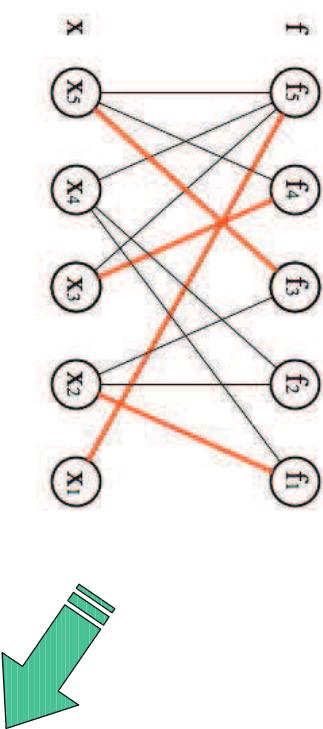
Couplage maximal

- Construction d'un arbre alterné : exemple



Couplages

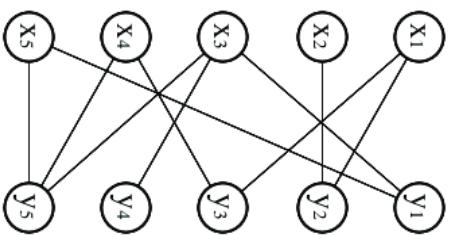
Couplage maximal



Couplages

Couplage maximal et flot maximal

- Un couplage maximal sur un graphe biparti peut être déterminé en recherchant un flot maximal dans un réseau de transport particulier.

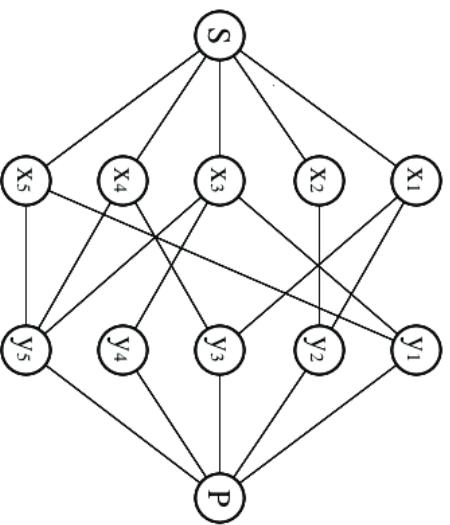


4/7

Couplages

Couplage maximal et flot maximal

- On construit un réseau de transport en ajoutant au graphe une origine S reliée à tous les sommets de X et une destination P liée à tous les sommets de Y . On fixe à un les capacités de tous les arcs de ce réseau.



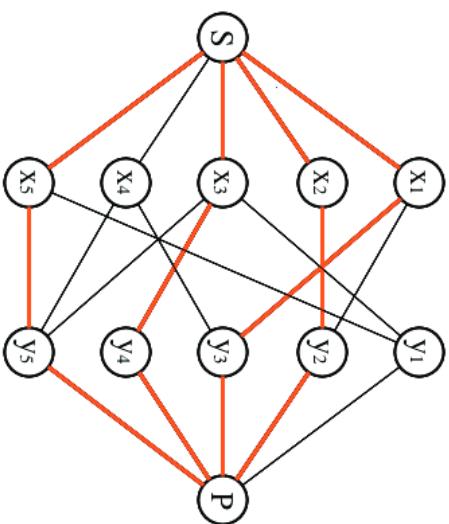
48

Couplages

Couplage maximal et flot maximal

- On cherche alors un flot maximal entre S et P .

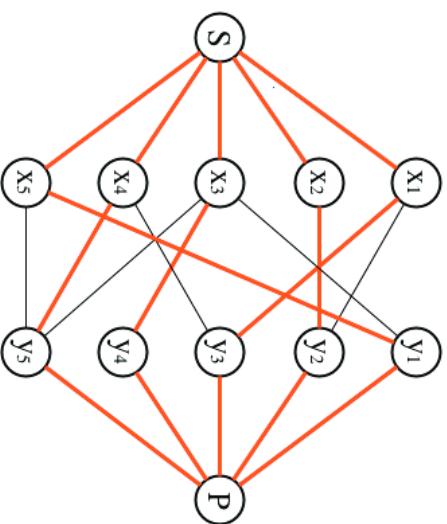
On commence par la recherche d'un flot complet :



Couplages

Couplage maximal et flot maximal

- On améliore ensuite ce flot complet pour obtenir un flot maximal (transfert le long de la chaîne améliorante x_4, y_5, x_5, y_1)



Couplages

Couplage de poids maximal

- A chaque arête $\alpha \in A$ du graphe $G(X, A)$ on associe maintenant un nombre réel $w(\alpha)$ appelé poids de l'arête α .
- Le poids d'un couplage $K \subset A$ est la somme des poids des arêtes qui le constituent.
- Le problème du couplage de poids maximal est la recherche d'un couplage tel que son poids soit maximal sur l'ensemble de tous les couplages de G .
- Un couplage de poids maximal n'est pas nécessairement un couplage maximal

Définition 30

- Soit $K \subset A$, un couplage de $G(X, A)$, on rappelle qu'une **chaîne alternée augmentante** est une chaîne alternée joignant deux sommets insaturnés.
- Une **chaîne alternée réductrice** est une chaîne alternée impaire dont les arêtes extrêmes sont dans K .
- Une **chaîne alternée conservative** est une chaîne alternée paire dont une extrémité est un sommet isolé.

51

Couplages

Couplage de poids maximal

Définition 31

- Le coût réduit d'une chaîne alternée L (augmentante, réductrice ou conservatrice) relative au couplage K vaut :
$$\delta(L) = w(L \cap \bar{K}) - w(L \cap K)$$
- Un cycle alterné pair est une chaîne alternée paire dont les extrémités coïncident. Un transfert le long d'un cycle alterné pair conserve la cardinalité du couplage.
- Le coût réduit du cycle μ est :
$$\delta(\mu) = w(\mu \cap \bar{K}) - w(\mu \cap K)$$

52

Couplages

Couplage de poids maximal

Théorème 6

- Un couplage K est de poids maximal si et seulement s'il n'existe pas de chaîne alternée, ni de cycle alterné pair de coût strictement positif relativement à K .

On peut donc construire, de proche en proche, des couplages dont la cardinalité augmente et de poids maximal par des transferts successifs le long de chaînes alternées augmentantes et de coût réduit maximal.

Théorème 7

- Soit K un couplage de cardinalité p de poids maximal (sur l'ensemble des couplages de cardinalité p) et soit L une chaîne alternée augmentante relativement à K de coût réduit maximal.
- Le couplage K' obtenu à partir de K par transfert le long de L est de poids maximal parmi tous les couplages de cardinalité $p + 1$.

53

Couplages

Problèmes d'affectation

Etant donné n tâches à réaliser et n machines pour les réaliser et sachant que l'on connaît le coût de réalisation C_{ij} de la tâche t_i par la machine m_j ,

on cherche une permutation σ de $\{1, 2, \dots, n\}$ conduisant à un coût total

$$\sum_{i=1}^n C_{i,\sigma(i)}$$

minimal (sur l'ensemble de toutes les $n!$ permutations σ possibles).

- Cas particulier d'un problème de transport (sans capacité)
- Cas particulier d'un problème de couplage parfait de poids minimum

54

Couplages

Problèmes d'affectation

- Constat : on ne change pas la ou les solutions optimales en augmentant ou en diminuant d'une même quantité tous les éléments d'une même ligne (ou d'une même colonne) de la matrice des coûts.
- Si l'on fait apparaître, par ce type de transformation, suffisamment de zéros dans le tableau (mais pas de coût négatifs) et qu'il existe n zéros indépendants (c'est-à-dire un seul zéro dans chaque ligne et dans chaque colonne), on aura trouvé l'affectation optimale.
- **Algorithme Hongrois** (algorithme itératif comportant trois phases)

55

Couplages

Problèmes d'affectation : algorithme Hongrois

	1	2	3	4	5
A	7	3	5	7	10
B	6	∞	∞	8	7
C	6	5	1	5	∞
D	11	4	∞	11	15
E	∞	4	5	2	10

■ Première phase : obtention de valeurs nulles

A tous les éléments de chacune des colonnes, on enlève le plus petit élément de cette colonne, puis on effectue la même opération sur les lignes

	1	2	3	4	5
A	1	0	4	5	3
B	0	∞	∞	6	0
C	0	2	0	3	∞
D	5	1	∞	9	8
E	∞	1	4	0	3

Couplages

Problèmes d'affectation : algorithme Hongrois

■ Deuxième phase : recherche du maximum d'affectations possibles

On cherche à former une solution à coût nul.

On considère la ligne ayant un nombre minimal de zéros, on encadre l'un des zéros de cette ligne, puis on barre les zéros qui se trouvent sur la même ligne ou la même colonne.

On procède de même pour toutes les lignes.

	1	2	3	4	5
A	1	0	4	5	3
B	0	∞	∞	6	0
C	0	2	0	3	∞
D	4	0	∞	8	7
E	∞	1	4	0	3

57

Couplages

Problèmes d'affectation : algorithme Hongrois

■ Troisième phase : détermination des affectations intéressantes

- On marque toutes les lignes qui ne contiennent aucun zéros encadré
- On marque les lignes qui ont un zéro encadré dans une colonne marquée
- On marque les colonnes qui ont un ou plusieurs zéros barrés dans une ligne marquée
- On répète jusqu'à ce que l'on ne puisse plus faire de nouveaux marquages

	1	2	3	4	5
A	1	0	4	5	3
B	0	∞	∞	6	0
C	0	2	0	3	∞
D	4	0	∞	8	7
E	∞	1	4	0	3

*

58

Couplages

Problèmes d'affectation : algorithme Hongrois

■ Troisième phase : détermination des affectations intéressantes

- Les affectations intéressantes à considérer sont celles issues des sommets correspondant aux lignes marquées ; on barre donc les lignes non marquées
- Les sommets correspondant au colonnes marquées ne sont pas intéressants car il ne peuvent être ré-affectés ; on barre donc les colonnes marquées

	1	2	3	4	5	*
A	1	0	4	5	3	*
B	0	∞	∞	6	∅	
C	∅	2	0	3	∞	
D	4	∅	∞	8	7	*
E	∞	1	4	0	3	

*

59

Couplages

Problèmes d'affectation : algorithme Hongrois

■ Troisième phase : détermination des affectations intéressantes

- Dans le tableau réduit obtenu (cases non barrées), on recherche l'élément le plus petit (nécessairement non nul par construction)
- On retranche sa valeur aux colonnes non barrées
- Et on l'ajoute aux lignes barrées

	1	2	3	4	5		1	2	3	4	5		1	2	3	4	5		
A	1	0	4	5	3	*	A	0	0	3	4	2	A	0	0	3	4	2	
B	0	∞	∞	6	∅		B	-1	∞	∞	5	-1	B	0	∞	∞	6	0	
C	∅	1	0	3	∞		C	-1	2	-1	2	∞	C	0	3	0	3	∞	
D	4	∅	∞	8	7	*	D	3	0	∞	7	6	D	3	0	∞	7	6	
E	∞	1	4	0	3		E	∞	1	3	-1	2	E	∞	2	4	0	3	

*

60

Couplages

Problèmes d'affectation : algorithme Hongrois

■ Deuxième phase : recherche du maximum d'affectations possibles

On cherche à former une solution à coût nul.

On considère la ligne ayant un nombre minimal de zéros, on encadre l'un des zéros de cette ligne, puis on barre les zéros qui se trouvent sur la même ligne ou la même colonne.

On procède de même pour toutes les lignes.

	1	2	3	4	5
A	0	0	3	4	2
B	0	∞	∞	6	0
C	0	3	0	3	∞
D	3	0	∞	7	6
E	∞	2	4	0	3

61

Couplages

Problèmes d'affectation : algorithme Hongrois

■ Solution optimale

	1	2	3	4	5
A	0	0	3	4	2
B	0	∞	∞	6	0
C	0	3	0	3	∞
D	3	0	∞	7	6
E	∞	2	4	0	3

■ Permutation

$$\sigma = \begin{pmatrix} A & B & C & D & E \\ 1 & 5 & 3 & 2 & 4 \end{pmatrix}$$

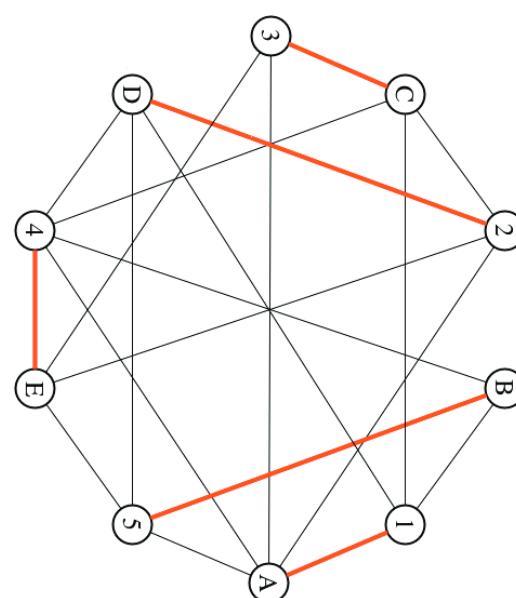
Couplages

Problèmes d'affectation : algorithme Hongrois

Solution optimale

$$C_{A1} + C_{B5} + C_{C3} + C_{D2} + C_{E4} = 7 + 7 + 1 + 4 + 2 = 21$$

	1	2	3	4	5
A	7	3	5	7	10
B	6	∞	∞	8	7
C	6	5	1	5	∞
D	11	4	∞	11	15
E	∞	4	5	2	10



63

Problèmes d'ordonnancement

Objectif

- Faciliter la mise en œuvre et guider l'exécution d'un ensemble complexe de tâches.
- Etant donné un objectif qu'on se propose d'atteindre et dont la réalisation suppose l'exécution préalable de multiples tâches, soumises à de nombreuses contraintes, déterminer l'ordre et le calendrier des diverses tâches.
- Le critère d'optimalité peut porter sur la minimisation de la durée et/ou du coût de la réalisation du projet.

Problème central de l'ordonnancement

- Le projet est décomposable en un nombre fini de tâches, caractérisées par leur durée d'exécution (éventuellement également également par leur coût d'exécution)
- Ces tâches sont soumises à des contraintes de postériorité stricte (une tâche ne peut commencer que si certaines autres tâches sont complètement terminées)

Problèmes d'ordonnancement

Le graphe potentiels-tâches



A partir de la description des tâches du projet, on construit le graphe suivant

- A chaque tâche i , on associe un sommet i du graphe
- On définit un arc entre i et j de longueur d_i (la durée de la tâche i) si la tâche i doit précéder la tâche j
- On introduit deux sommets correspondant à deux tâches fictives de durée nulle, la tâche de début des travaux α , antérieure à toutes les tâches et la tâche de fin des travaux ω postérieure à toutes les tâches
- Le graphe ainsi défini est sans circuit
- Le travail commençant à la date 0, on cherche un ordonnancement qui minimise la durée totale du projet
- Pour qu'une tâche puisse commencer, il est nécessaire que toutes les tâches qui la relient à la tâche début du projet soient réalisées

65

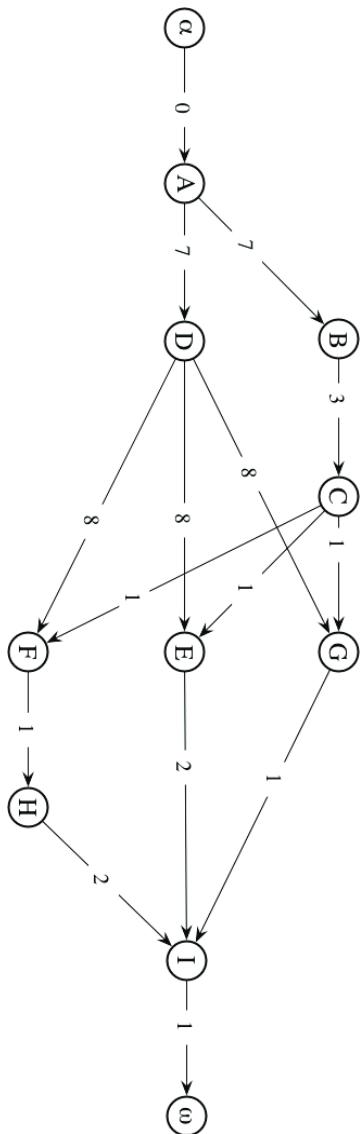
Problèmes d'ordonnancement

Le graphe potentiels-tâches

Tâche		Durée	Tâches ant.
A	Maçonnerie	7	-
B	Charpente	3	A
C	Toiture	1	B
D	Sanitaire et électricité	8	A
E	Façade	2	D,C
F	Fenêtres	1	D,C
G	Aménagement jardin	1	D,C
H	Peinture	2	F
I	Emménagement	1	E,G,H

Problèmes d'ordonnancement

Le graphe potentiels-tâches



67

Problèmes d'ordonnancement

Le graphe potentiels-tâches

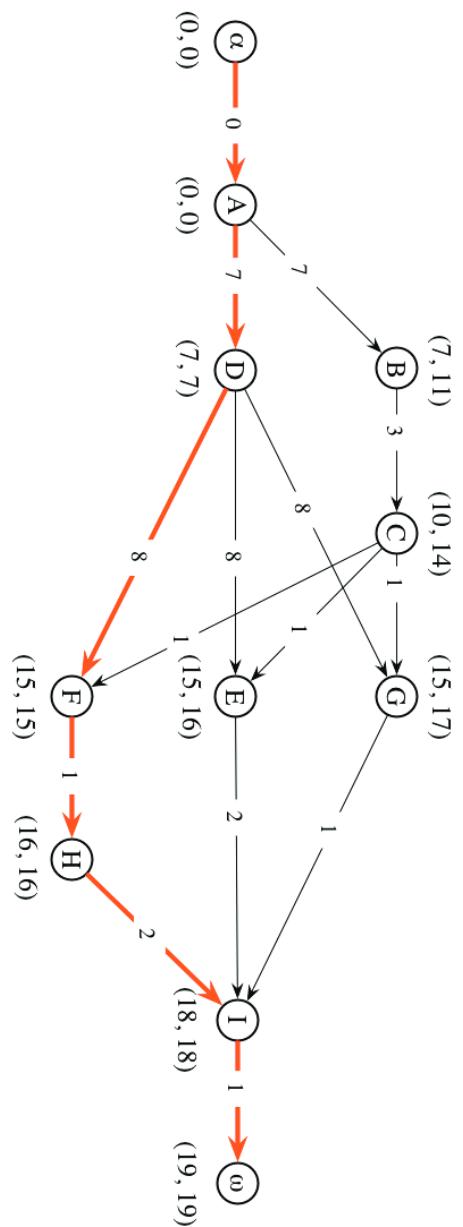
Définition 32

- La **date au plus tôt**, t_i de début de la tâche i est égale à $\max(t_j + d_j)$ pour $j \in \Gamma_i^{-1}$ c'est-à-dire que t_i est égal à la longueur du plus long chemin de α à i .
- La **durée minimale du projet** est égale à t_ω .
- La **date au plus tard** T_i pour commencer la tâche i est égale à $\min(T_j - d_j)$ pour $j \in \Gamma_i$ avec $T_\omega = t_\omega$ c'est-à-dire $T_i = t_\omega - l(i, \omega)$ où $l(i, \omega)$ est la longueur du plus long chemin de i à ω .
- La **marge totale** M_i de la tâche i est définie comme la différence entre la date au plus tôt et au plus tard : $M_i = T_i - t_i$
- Les tâches dont la marge est nulle sont appelées **tâches critiques**. Si un quelconque retard est pris sur la réalisation de l'une de ces tâches, la durée minimale du projet sera augmentée d'autant.
- La **marge libre** m_i de la tâche i est le délai dont on peut retarder cette tâche sans affecter les dates de début au plus tôt des tâches postérieures :
$$m_i = \min(t_j - t_i - d_i) \text{ pour } j \in \Gamma_i$$

68

Problèmes d'ordonnancement

Le graphe potentiels-tâches



— Chemin critique

(date au plus tôt, date au plus tard)

69



Problèmes d'ordonnancement

Le graphe potentiels-étapes ou graphe PERT

Construction du graphe

- Les tâches sont matérialisées par les arcs du graphe
- La longueur de chaque arc est égale à la durée d_i de la tâche correspondante
- Le début et la fin d'une tâche constituent les étapes du projet
- On introduit deux étapes de début et de fin du projet
- Si une tâche j doit succéder à une tâche i, on confond l'extrémité initiale de l'arc a_j (tâche j) et l'extrémité de l'arc a_i (tâche i)

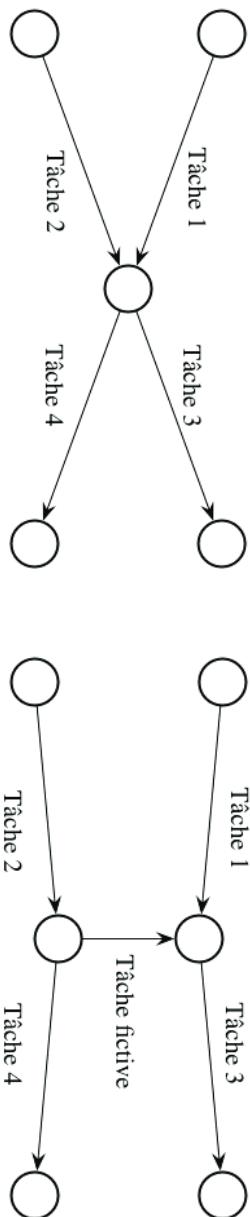
- Le graphe est sans circuit
- Les sommets s'interprètent comme des étapes

Problèmes d'ordonnancement

Le graphe potentiels-étapes ou graphe PERT

Construction du graphe

- Nécessité d'introduire parfois des tâches fictives de durée nulle pour traduire des contraintes de postériorité



Les tâches 3 et 4 doivent succéder aux tâches 1 et 2

71

La tâche 3 doit succéder aux tâches 1 et 2

La tâche 4 doit succéder à la tâche 2

Problèmes d'ordonnancement

Le graphe potentiels-étapes

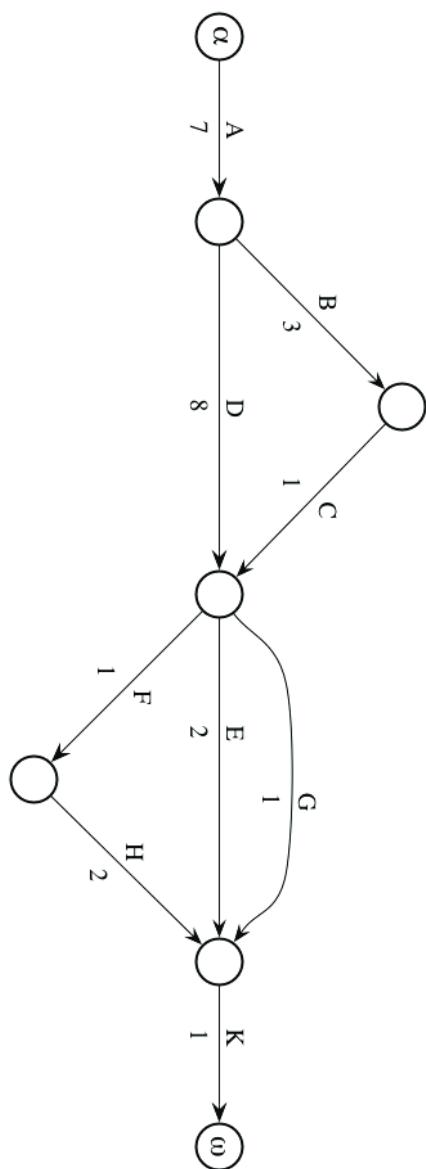
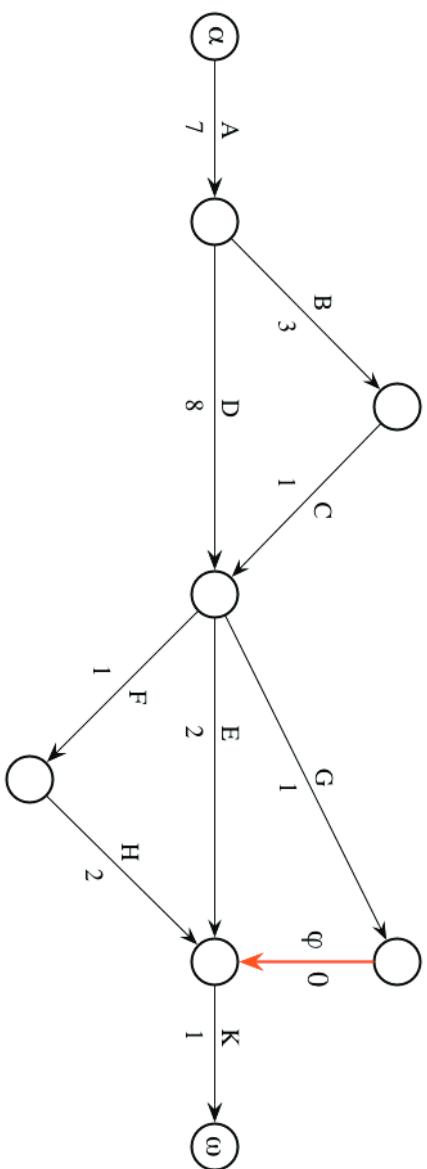
Tâche		Durée	Tâches ant.
A	Maçonnerie	7	-
B	Charpente	3	A
C	Toiture	1	B
D	Sanitaire et électricité	8	A
E	Façade	2	D,C
F	Fenêtres	1	D,C
G	Aménagement jardin	1	D,C
H	Peinture	2	F
I	Emménagement	1	E,G,H

Problèmes d'ordonnancement

Le graphe potentiels-étapes

Problèmes d'ordonnancement

Le graphe potentiels-étapes



Problèmes d'ordonnancement

Résolution

- Un **ordonnancement** est un programme d'exécution qui fixe la date t_i de début d'exécution de chacune des tâches (dans la méthode du potentiel) ou de début de réalisation de chacune des étapes (dans la méthode PERT).
- Un ordonnancement est **optimal** s'il est réalisable, c'est-à-dire si $t_j - t_i \geq d_i$ et s'il minimise la durée de réalisation t_ω du projet.
- La **durée minimale** du projet est égale à la longueur du plus long chemin du sommet α au sommet ω . Ce chemin est appelé **chemin critique**.

- Chaque $t_j = \max(t_i + d_i)$ pour $i \in \Gamma_j^{-1}$ est la longueur du plus long chemin de α à i et représente la **date au plus tôt** de début d'exécution de la tâche ou de l'étape j .
- En définissant $T_j = \min(T_i - d_i)$ pour $i \in \Gamma_j$ avec $T_\omega = t_\omega$, chaque $T_\omega - T_i$ est la longueur du plus long chemin de i à ω et représente la **date au plus tard** de début d'exécution de la tâche ou de l'étape j .

75

Problèmes d'ordonnancement

Compléments

➤ Contraintes additives (par rapport au problème central)

- Type potentiel
 - la tâche j doit commencer après la fin de i ou bien après la moitié de la réalisation de i , ou bien un certain temps après la fin de i , etc.
- Type disjonctif
 - les tâches i et j ne peuvent être réalisées en même temps
- Type cumulatif
 - les moyens nécessaires à l'exécution d'un certain nombre de tâches sont, à chaque instant, limités ; par exemple, le nombre de camions utilisables ou la somme dépensée pour les travaux réalisés à l'instant t ne doit pas dépasser un certain seuil.

➤ Prise en compte

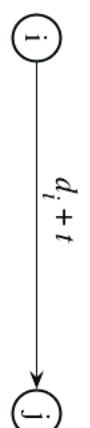
- Contraintes difficiles à prendre en compte dans le graphe potentiels-étapes (introduction d'un grand nombre d'étapes fictives)
- Adjonction d'arcs supplémentaires dans le graphe potentiels-tâches.

76

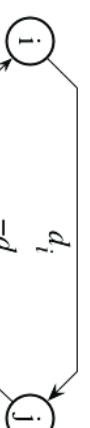
Problèmes d'ordonnancement

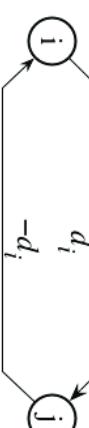
Compléments

- j ne doit pas commencer avant la moitié de la réalisation de la tâche i

- j ne peut commencer qu'un temps t après la fin de la tâche i


- j ne peut commencer qu'après la date b_j

- j doit commencer avant la date c_j


- j doit suivre immédiatement la tâche i


77

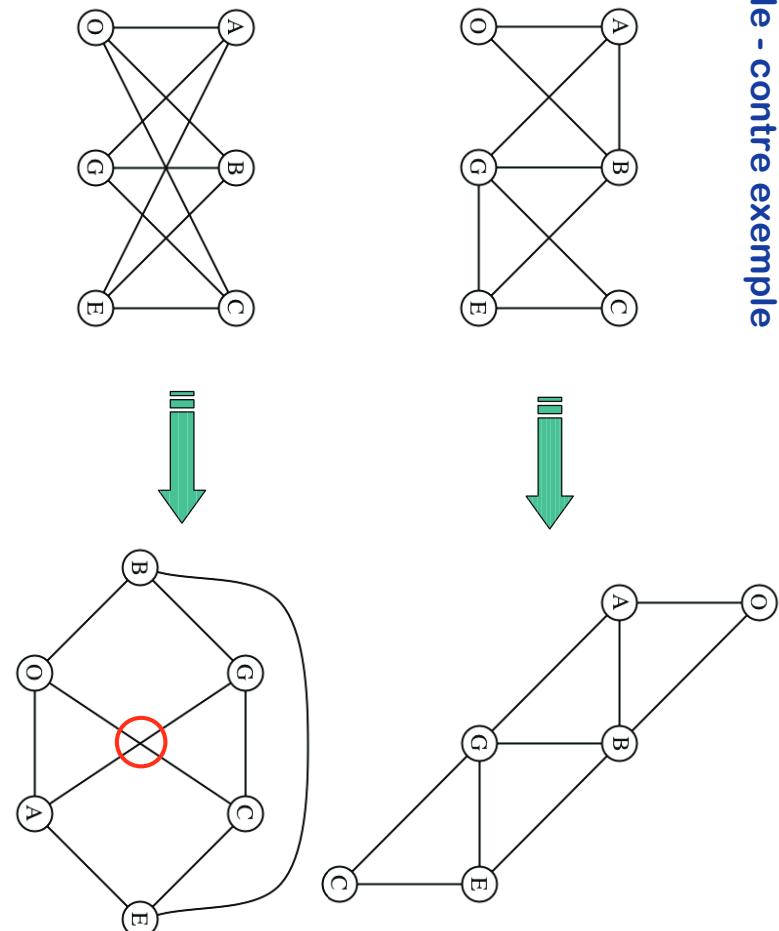
Graphes planaires

Définition 33

- On dit qu'un graphe (simple ou multigraphe) est **planaire** si on peut le « dessiner » dans le plan, en matérialisant les arêtes par des arcs continus qui ne se croisent pas en dehors de leurs extrémités.
- Deux graphes que l'on peut amener à coïncider par déformation élastique du plan sont dits **isomorphes**.
- Les arêtes d'un graphe planaire G délimitent des surfaces planes nommées **faces** de G , telles que deux points arbitrairement choisis dans une même région peuvent toujours être reliés par un trait continu ne rencontrant ni sommet, ni arêtes.
- La **frontière** d'une face de G est l'ensemble des arêtes qui touchent une face z .
- Deux **faces** z et z' sont dites **adjacentes** si leurs frontières ont une arête commune.
- On appelle **contour** de la face z le cycle élémentaire constitué de toutes les arêtes de la frontière de la face z .
- Dans un graphe, il existe toujours une face illimitée et une seule qui constitue la **face infinie**. On ne définit pas de contour pour cette face.

Graphes planaires

Exemple - contre exemple



79

Graphes planaires

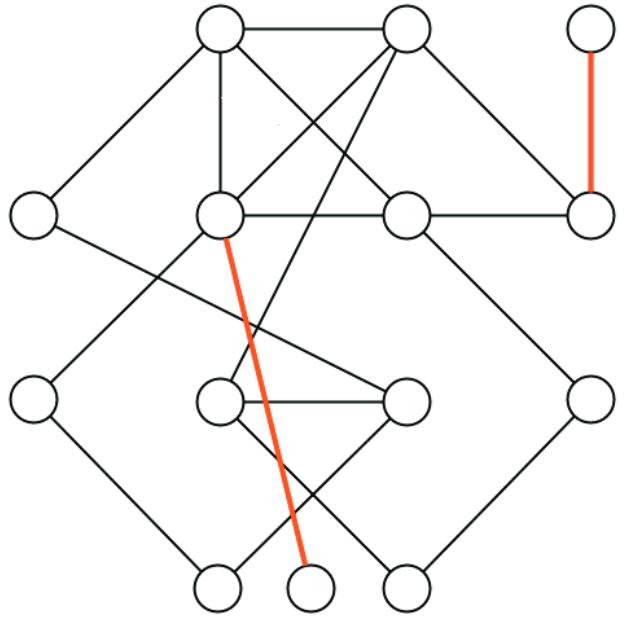
Test de planarité

Définition 34

- Soit un graphe connexe G . On dit qu'un graphe H se déduit par contraction s'il a été obtenu par application répétée des règles suivantes :
 - x étant un sommet de degré 1 de G , H est le sous-graphe de G obtenu en supprimant x .
 - x étant un sommet de degré 2 de G , de voisins y et z , H est le sous-graphe de G obtenu en supprimant x et auquel on ajoute une arête (y, z) .

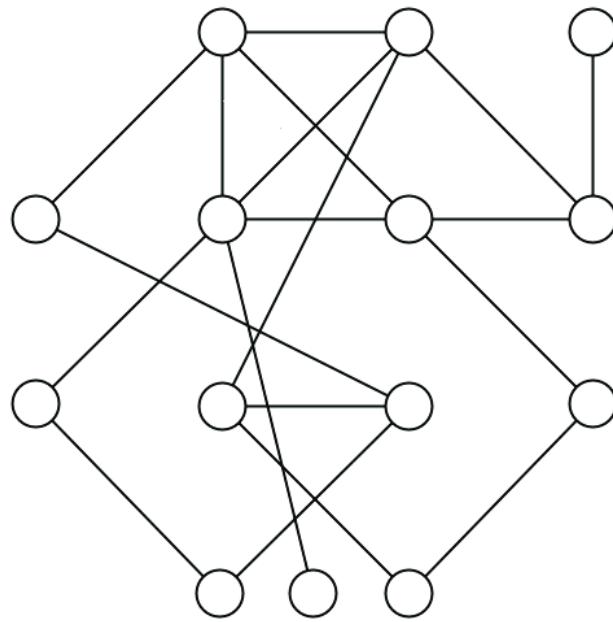
Théorème 8 (Kuratowski)

- La condition nécessaire et suffisante pour qu'un graphe G soit planaire est qu'il n'admette pas de sous-graphe partiel se contractant en un graphe isomorphe à $K_{3,3}$ ou K_5 .



Test de planarité

Graphes planaires

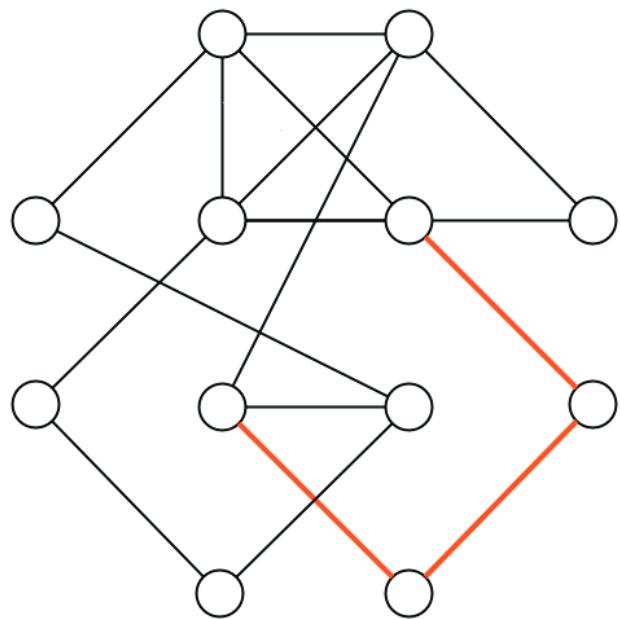


Test de planarité

Graphes planaires

Graphes planaires

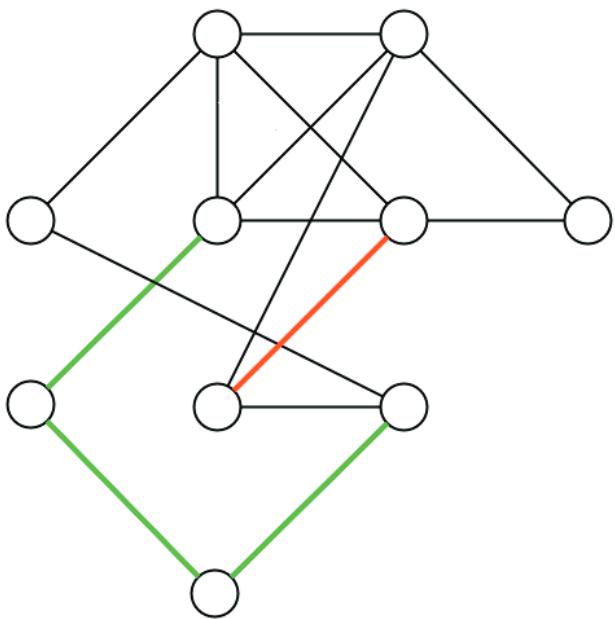
Test de planarité



83

Graphes planaires

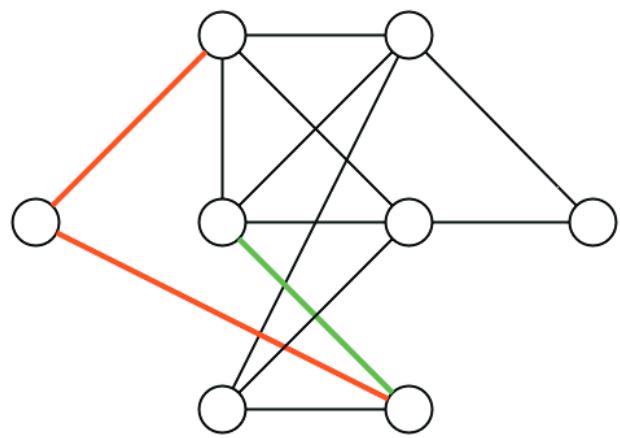
Test de planarité



84

Graphes planaires

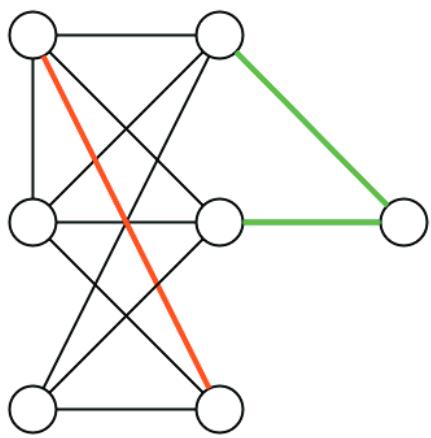
Test de planarité



85

Graphes planaires

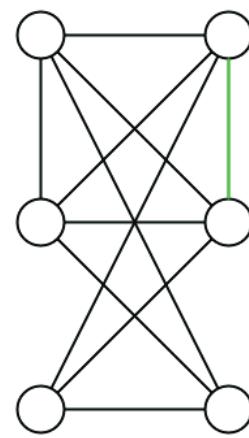
Test de planarité



86

Graphes planaires

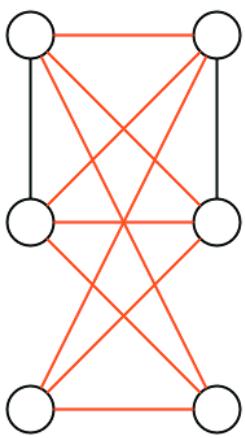
Test de planarité



87

Graphes planaires

Test de planarité



$K_{3,3}$

→ Graphe non planaire

88

Coloration d'un graphe

- On définit deux types de coloration pour un graphe $G=(X, A)$; la **coloration des arêtes** et la **coloration des sommets**.
- La coloration des sommets (resp. des arêtes) d'un graphe G correspond à l'affectation d'une couleur à chacun des sommets (resp. des arêtes) de telle sorte que deux sommets (resp. arêtes) adjacents ne soient pas porteur de la même couleur.
- Un graphe est dit p -chromatique si ses sommets admettent une coloration en p couleurs.
- On appelle **nombre chromatique** $\gamma(G)$ (resp. **indice chromatique** $q(G)$) le nombre minimum de couleurs distinctes nécessaires pour effectuer une coloration des sommets (resp. des arêtes) de G .

89

Coloration d'un graphe

Coloration des sommets

➤ **Définition 34**

- Un sous-ensemble de sommets $I \subset X$ est un **ensemble stable** s'il ne comprend que des sommets non adjacents deux à deux.
- Les sommets coloriés de la même couleur dans une coloration des sommets forment donc un ensemble stable.
- Une coloration des sommets est donc une partition des sommets en ensembles stables.
- Soit $\alpha(G)$ le **nombre de stabilité**, c'est-à-dire le cardinal maximum d'un ensemble stable, alors si $\gamma(G)$ est le nombre chromatique, puisque chaque ensemble de sommets de même couleur a un cardinal inférieur ou égal à $\alpha(G)$, on a :
$$\alpha(G) \gamma(G) \geq N(G)$$

où $N(G)$ est le nombre de sommets du graphe G .

On en déduit : $\gamma(G) \geq N(G) / \alpha(G)$

90

Coloration d'un graphe

Coloration des sommets

- La détermination du nombre chromatique d'un graphe ainsi que l'obtention d'une coloration minimale des sommets constituent un problème assez complexe..
- En pratique, on utilise souvent des algorithmes de coloration heuristiques.
- Le nombre chromatique $\gamma(G)$ possède des bornes inférieures et supérieures
Pour un graphe à n sommets et m arêtes, on a :

$$\begin{aligned}\gamma(G) &\geq n / (n - d_{min}) \text{ avec } d_{min} \text{ degré minimum des sommets de } G \\ \gamma(G) &\geq \text{cardinal de la plus grande clique de } G \\ \gamma(G) &\geq n^2 / (n^2 - 2m)\end{aligned}$$

$$\begin{aligned}\gamma(G) &\leq n + 1 - \alpha(G) \text{ avec } \alpha(G) \text{ nombre de stabilité du graphe } G \\ \gamma(G) &\leq d_{max} + 1 \text{ avec } d_{max} \text{ degré maximum des sommets de } G\end{aligned}$$

$\alpha(G)$ nombre de stabilité : maximum des cardinaux des ensembles stables d'un graphe
un sous-ensemble S de sommets est stable si deux sommets quelconques de S ne sont pas adjacents

91

Coloration d'un graphe

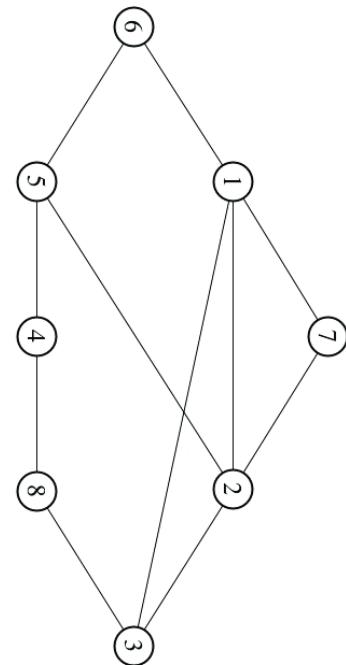
Algorithme de Welsh et Powell

- Etape 0 : Initialisation
 M : matrice d'adjacence du graphe dont les sommets sont rangés par ordre de degré décroissants
- $k = 1$
- Etape 1
 $N = M$
- Etape 2
 - Colorer par la couleur c_k la première ligne non encore colorée de N ainsi que la colonne correspondante
 - N = ensemble des lignes non encore colorées ayant un zéro dans les colonnes de couleur c_k
 - si $N \neq \emptyset$ aller à l'étape 2 sinon aller à l'étape 3
- Etape 3
 - Si toutes les lignes sont colorées alors STOP (on a une k -coloration)
 - $k = k + 1$; (changer de couleur)
 - aller à l'étape 1

92

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple

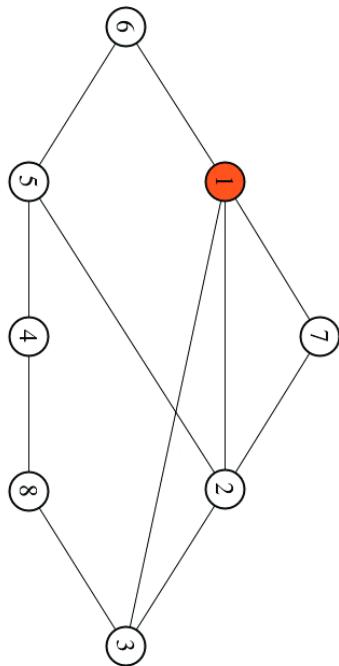


.	1	1	1	.	.	1	1
.	1	.	1	.	1	.	1
1	.	.	.	1	.	1	1
1	.	.	.	1	.	1	.
1	.	.	.	1	.	1	.
.	.	1	.	1	.	1	1
.	.	.	.	1	.	.	1
.	1
.	1	.	.

93

Coloration d'un graphe

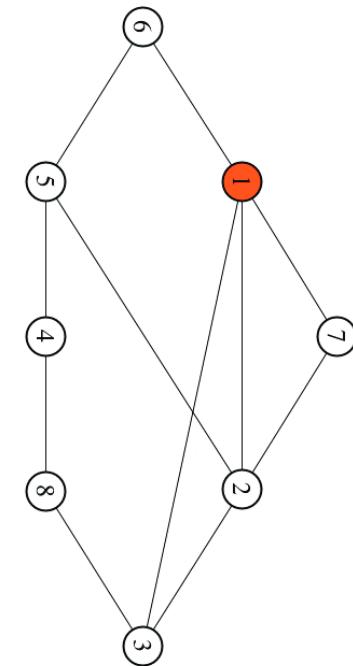
Algorithme de Welsh et Powell : exemple



94

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple

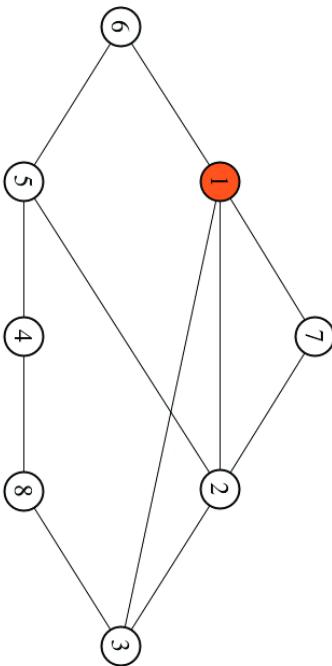


.	1	1	1	1	1	1	1	1	1
1	.	1	1	1	1	1	1	1	1
1	1	.	1	1	1	1	1	1	1
1	1	1	.	1	1	1	1	1	1
1	1	1	1	.	1	1	1	1	1
1	1	1	1	1	.	1	1	1	1
1	1	1	1	1	1	.	1	1	1
1	1	1	1	1	1	1	.	1	1
1	1	1	1	1	1	1	1	.	1
1	1	1	1	1	1	1	1	1	.

94

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple

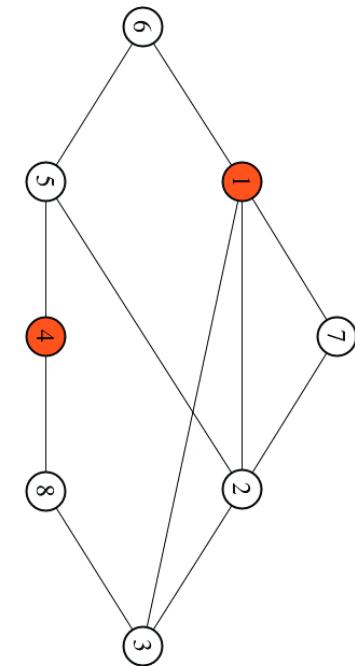


.	1	1	1	1	1	1	1	1	1
1	.	1	1	1	1	1	1	1	1
1	1	.	1	1	1	1	1	1	1
1	1	1	.	1	1	1	1	1	1
1	1	1	1	.	1	1	1	1	1
1	1	1	1	1	.	1	1	1	1
1	1	1	1	1	1	.	1	1	1
1	1	1	1	1	1	1	.	1	1
1	1	1	1	1	1	1	1	.	1
1	1	1	1	1	1	1	1	1	.

95

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple

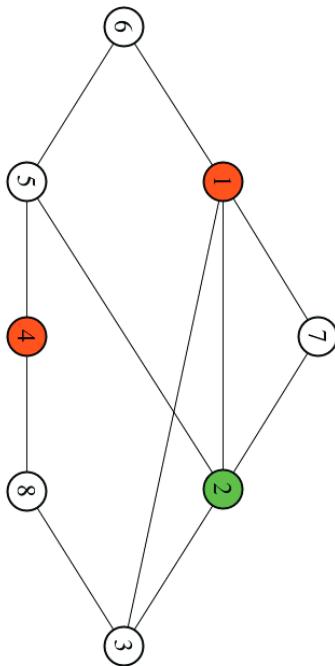


.	1	1	1	1	1	1	1	1	1
1	.	1	1	1	1	1	1	1	1
1	1	.	1	1	1	1	1	1	1
1	1	1	.	1	1	1	1	1	1
1	1	1	1	.	1	1	1	1	1
1	1	1	1	1	.	1	1	1	1
1	1	1	1	1	1	.	1	1	1
1	1	1	1	1	1	1	.	1	1
1	1	1	1	1	1	1	1	.	1
1	1	1	1	1	1	1	1	1	.

96

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple

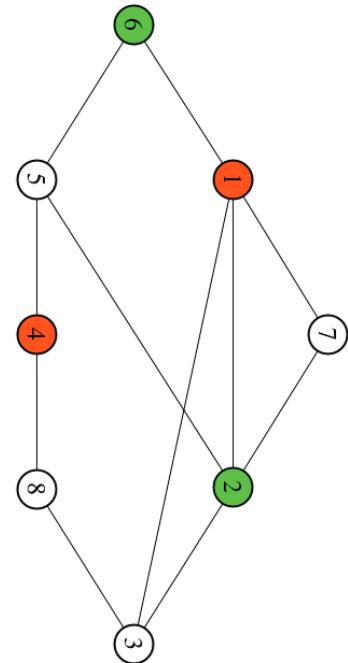


.	1	1	1	1	1	1	1	1	1
1	.	1	1	1	1	1	1	1	1
1	1	.	1	1	1	1	1	1	1
1	1	1	.	1	1	1	1	1	1
1	1	1	1	.	1	1	1	1	1
1	1	1	1	1	.	1	1	1	1
1	1	1	1	1	1	.	1	1	1
1	1	1	1	1	1	1	.	1	1
1	1	1	1	1	1	1	1	.	1
1	1	1	1	1	1	1	1	1	.

97

Coloration d'un graphe

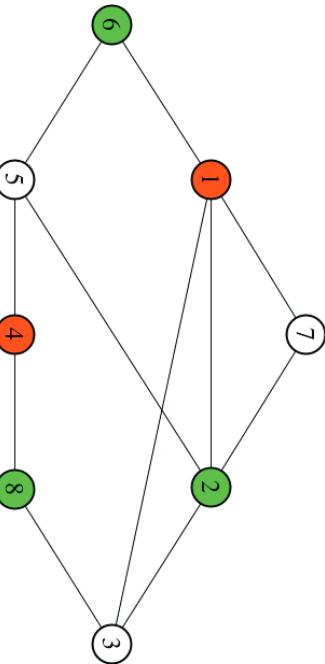
Algorithme de Welsh et Powell : exemple



98

Coloration d'un graphe

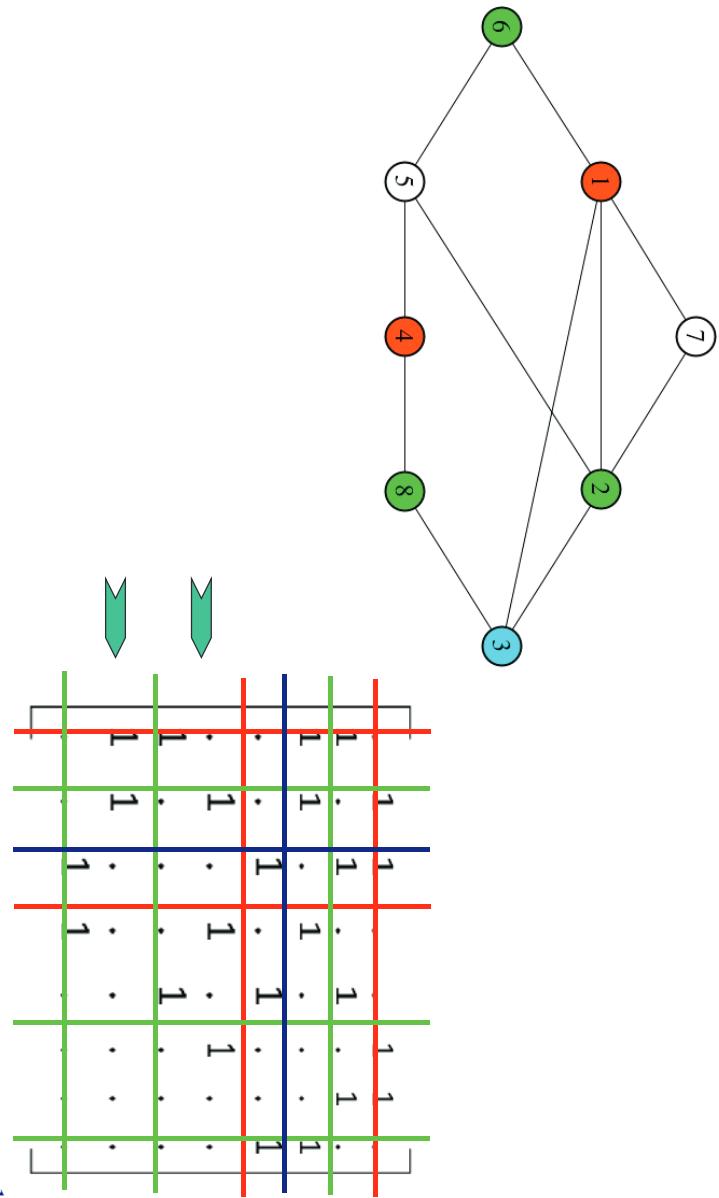
Algorithme de Welsh et Powell : exemple



99

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple



100

Coloration d'un graphe

Algorithme de Welsh et Powell : exemple

