Graphes - Propriétés

Institut National des Sciences Appliquées – Rouen
Département Architecture des Systèmes d'Information
michel.mainguenaud@insa-rouen.fr

Topologie

- Chaîne : séquence d'arête / arc (sans le sens)
- Cycle : chaîne dont les extrémités coïncident
- Cycle élémentaire : cycle minimal (pour l'inclusion). Ne contient pas strictement un autre cycle.
  - En parcourant un cycle élémentaire on ne rencontre pas deux fois le même noeud – sauf celui choisi comme origine du parcours

Topologiques (2)

- Chemin (x, y) dans le graphe G = (X, U)
  - $x \in X$, $y \in X$
  - Il existe un séquence d’arcs de G telle que
    - L’extrémité initiale du premier arc est x
    - L’extrémité terminale du dernier arc est y
    - L’extrémité terminale d’un arc est l’extrémité initiale de l’arc qui le suit dans la séquence (sauf pour le dernier).

Chemins

- Un graphe G = (X, U) orienté Ω-valué est
  - Un ensemble de nœuds, X
  - Un ensemble de triplets, U, (nœud, nœud, valuation) où valuation $\in \Omega$. Les éléments de U sont appelés des arcs valués
- Les valuations (des nombres) sont appelées coût ou poids de l'arc
- Dans un contexte base de données la valuation est hétérogène

Chemins (2)

- Elémentaire : ne passe pas plus d’une fois par le même noeud (les noeuds de G sont adjacents à deux arcs du chemin au plus)
- Simple : ne passe pas plus d’une fois par le même arc (simple n'implique pas élémentaire)
- Eulérien : passe par chaque arc une fois et une seule
- Hamiltonien : passe par chaque noeud une fois et une seule
- Circuit : chemin dont les deux extrémités coïncident (cycle pour le non orienté)

Connexité

- Connexité : Connexité (x, y) $\Leftrightarrow$ $\exists$ Chaîne (x, y) $\vee$ x = y
- Forte connexité : ForteConnexité (x, y) $\Leftrightarrow \exists$ [Chemin (x, y) $\wedge$ Chemin (y, x)] $\vee$ x = y
- Graphe connexe si tous les nœuds sont deux à deux en connexité
- Graphe fortement connexe si tous les nœuds sont deux à deux en forte connexité (relation réflexive, symétrique, transitive $\Leftrightarrow$ relation d'équivalence)

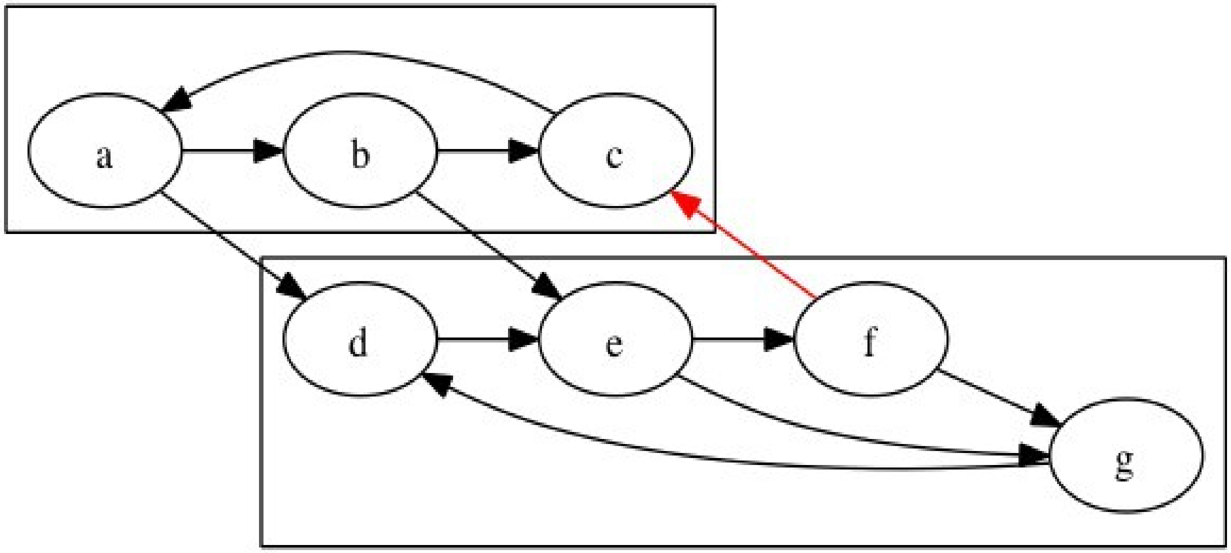
Composante

- Composante (fortement)connexe : ensemble de nœuds qui ont deux à deux la relation de (forte)connexité. De plus tout nœud en dehors de la composante n'a pas de relation de (forte)connexité avec aucun des éléments de la composante
- Un graphe est connexe s'il ne possède qu'une composante connexe
- Classe d'équivalence : sous-graphe fortement connexe maximal

Exemple

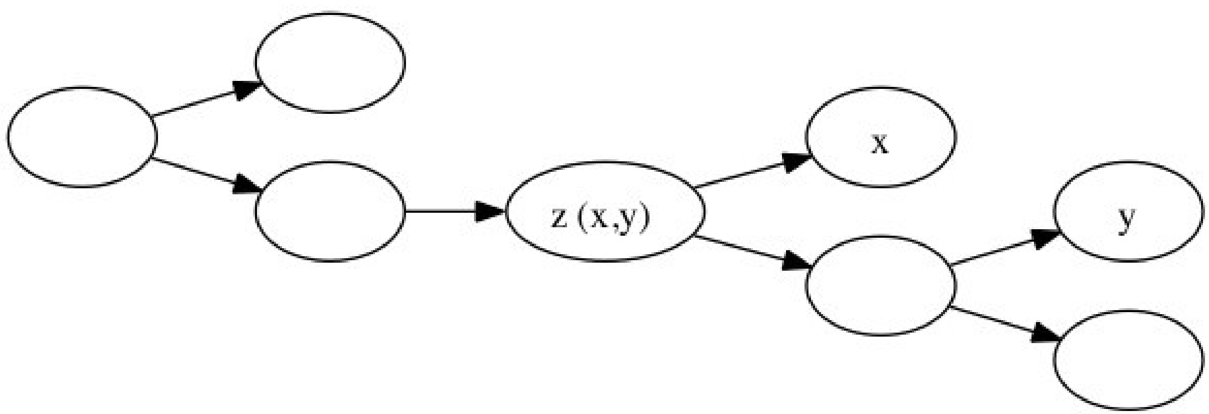
Semi-fortement connexe

Fortement connexe



Quasi-fortement connexe

- Pour tout couple de noeuds (x,y), il existe un noeud z(x,y) d'où part un chemin allant à x et un chemin allant à y



Propriétés

- Graphe réduit : graphe limité aux classes d'équivalence
- Nombre de connexité : nombre de classes d'équivalence :
  - graphe connexe $\Leftrightarrow$ le nombre de connexité = 1
- Complexité de la vérification de la connexité : O(M)

Arbre

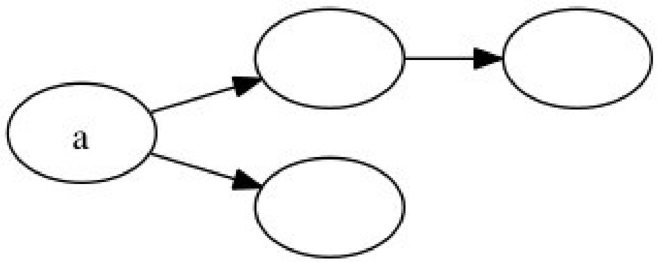
- Définitions (équivalentes)
  - Connexe et sans cycle
  - Sans cycle et admet (n - 1) arcs
  - Connexe et admet (n - 1) arcs
  - Sans cycle et l’ajout d’un arc crée un cycle et un seul
  - Connexe minimal et si on supprime un arc il ne l’est plus
  - Tout couple de noeud est relié par une chaîne et une seule
- Forêt
  - Graphe dont chaque composante connexe est un arbre

Arbre (2)

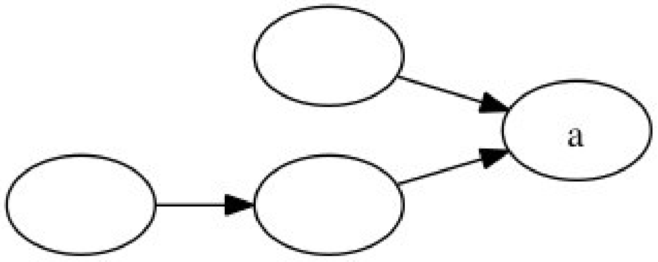
- Racine : s’il existe un chemin joignant ce nœud à chaque nœud du graphe (accessibilité)
- Anti-racine : s’il existe un chemin joignant chaque nœud du graphe à ce nœud
- Arbre de G : graphe partiel maximal (au sens de l’inclusion) sans cycle de G
- Coarbre de G : est un graphe partiel maximal sans cocyle de G – même ensemble de noeuds - (peut être un arbre) (complémentaire d'un arbre)

Racine et Anti-racine

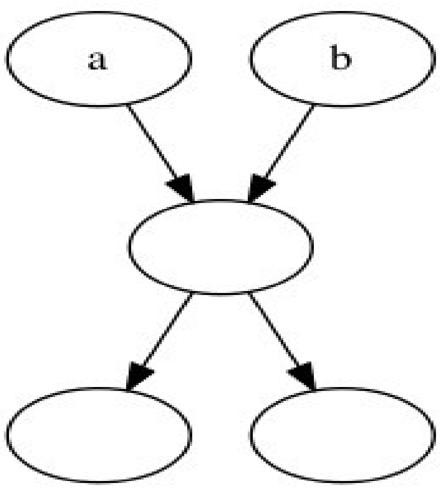
Arbre: racine a

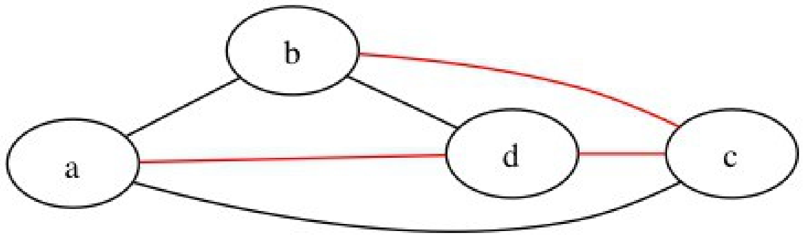


Arbre: anti-racine a



Arbre : racine a et b

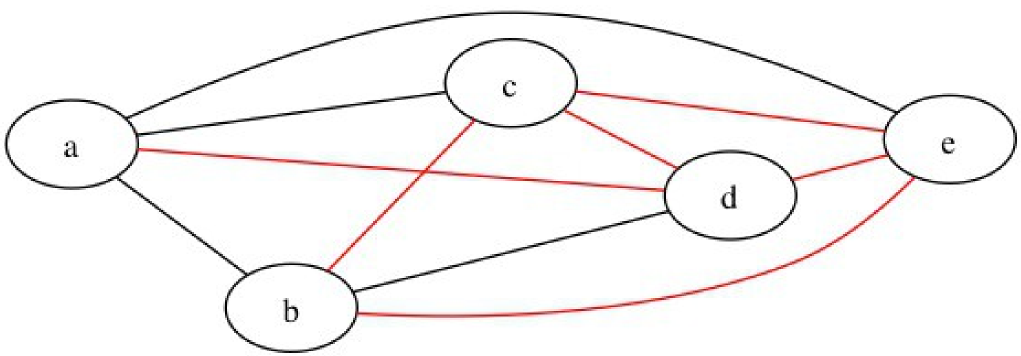




Arbre H'

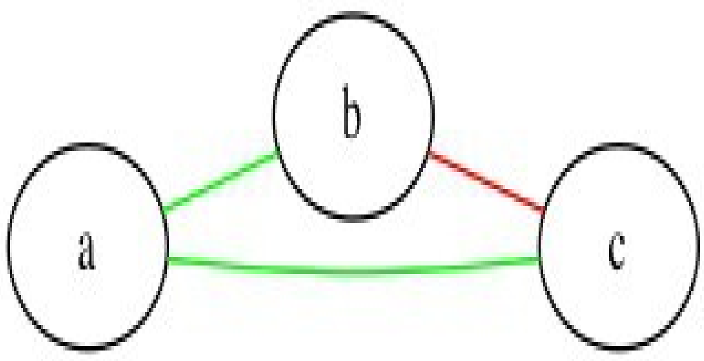
Co-arbre H (ici un arbre)

G (X, U) : connexe
Sans cocycle => même ensemble de noeuds, X, entre H et H'
H (X, V) co-arbre ssi
H' (X, U-V) est un arbre



Arbre H'

Co-arbre H



X = {a, b, c} : Arbre (vert) / complémentaire (rouge)
mais {(b,c)} n'est pas un co-arbre car pas de cocycle
(A → X-A) A = {b,c} X-A car même ensemble de noeuds

Arborescence

- Un graphe G (≥ 2 noeuds) est une arborescence de racine a (resp. une anti-arborescence d'anti-racine a) si :
  - a est une racine (resp. anti-racine) de G
  - G est un arbre.
- Une arborescence est un arbre orienté dans le même sens.
- Si on inverse le sens des arcs d'une arborescence on obtient une anti-arborescence

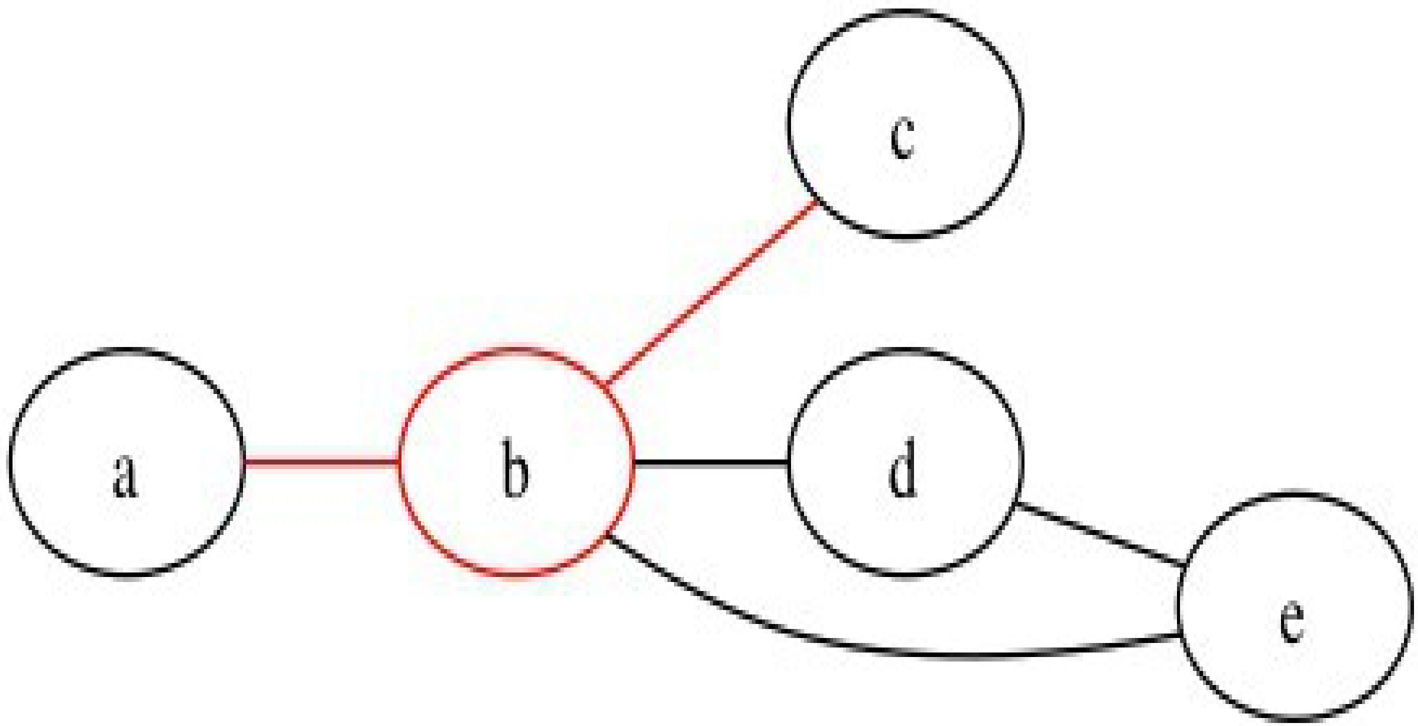
Arborescence (2)

- Les conditions suivantes sont équivalentes et caractérisent une arborescence de racine a :
  - G est un arbre admettant le noeud a comme racine
  - $\forall$ x $\in$ X il existe un chemin unique dans G joignant a à x
  - G admet a comme racine et est minimal pour cette propriété (si on supprime un arc, a n'est plus racine)
  - G est connexe et Γ- (a) = 0 et Γ- (x) = 1 $\forall$ x $\neq$ a
  - G est sans cycle et de plus les relations (1) sont vérifiées
  - G admet a comme racine et est sans cycle
  - G admet a comme racine et possède (n-1) arcs.

Points sensibles

- Point d’articulation (resp. ensemble) : nœud (resp. ensemble de) dont la suppression augmente le nombre de composantes connexes (perte de connexité) - ex noeud b
- Isthme : arête dont la suppression augmente le nombre de composantes connexes - ex arêtes [a,b], [b,c]
- Configuration classique dans les réseaux de transport, communication, …

Exemple : Articulation / Isthme

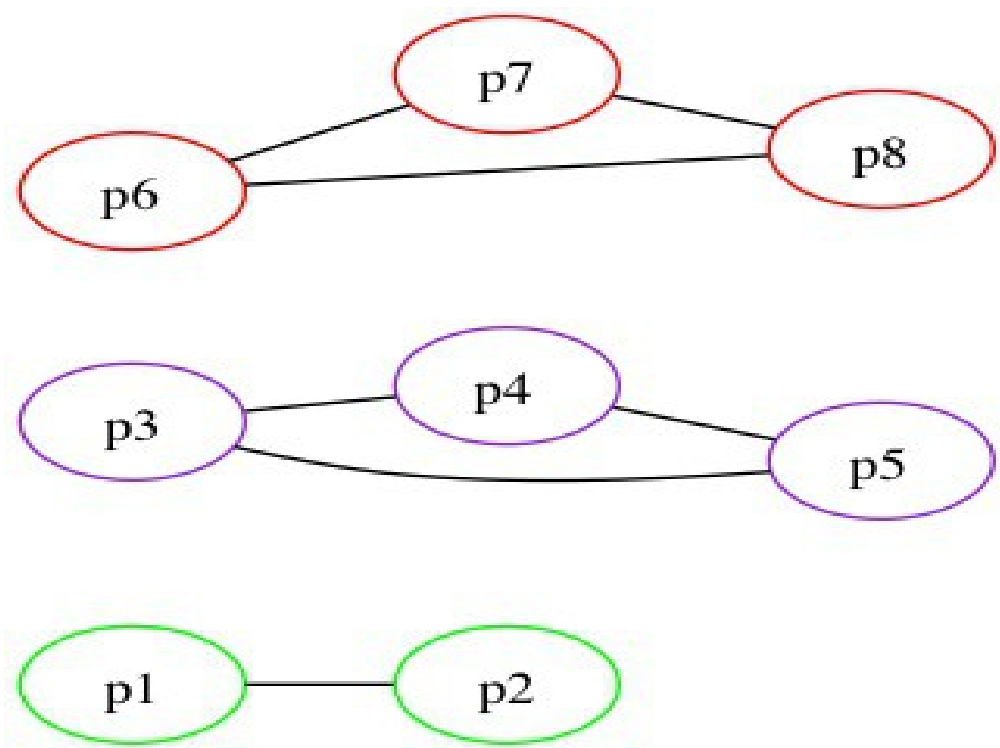


Couplages

- Définition : deux arêtes quelconques du couplage ne sont pas adjacentes (n'ont jamais un noeud en commun)
- Contraintes :
  - Avions bi-places
  - Pilotes ne pouvant pas faire équipes (langue)
- Nombre maximum d’avions qui peuvent voler simultanément ? (couplage de cardinalité maximum)

Exemple : Couplage

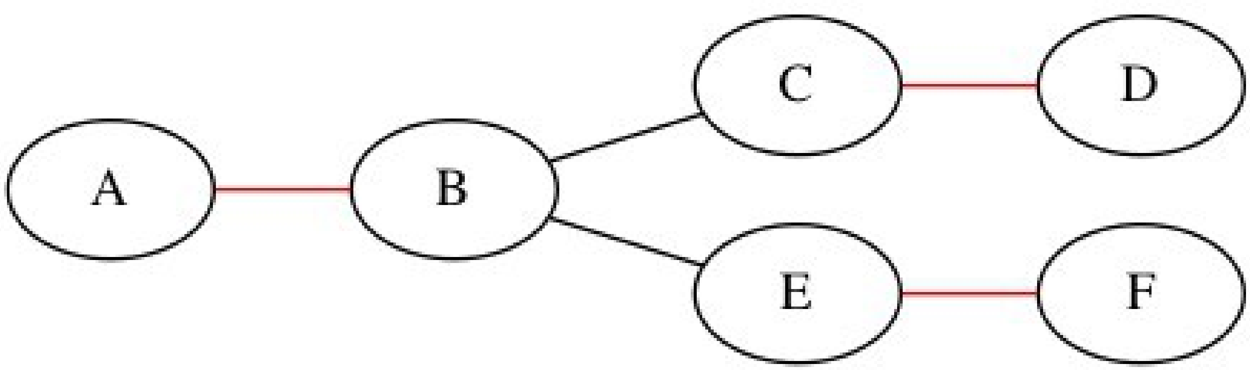
- Nœuds : P1, … P8 (pilotes)
- Arc : les pilotes peuvent voler ensemble



3 avions

Vocabulaire

- Couplage maximum : parmi tous les couplages il possède le plus grand nombre d'arêtes (pas unique)
- Un nœud x est saturé par un couplage C s'il existe une arête de C dont x est une extrémité (resp. insaturé - p5).
- Couplage parfait : sature tous les nœuds du graphe.



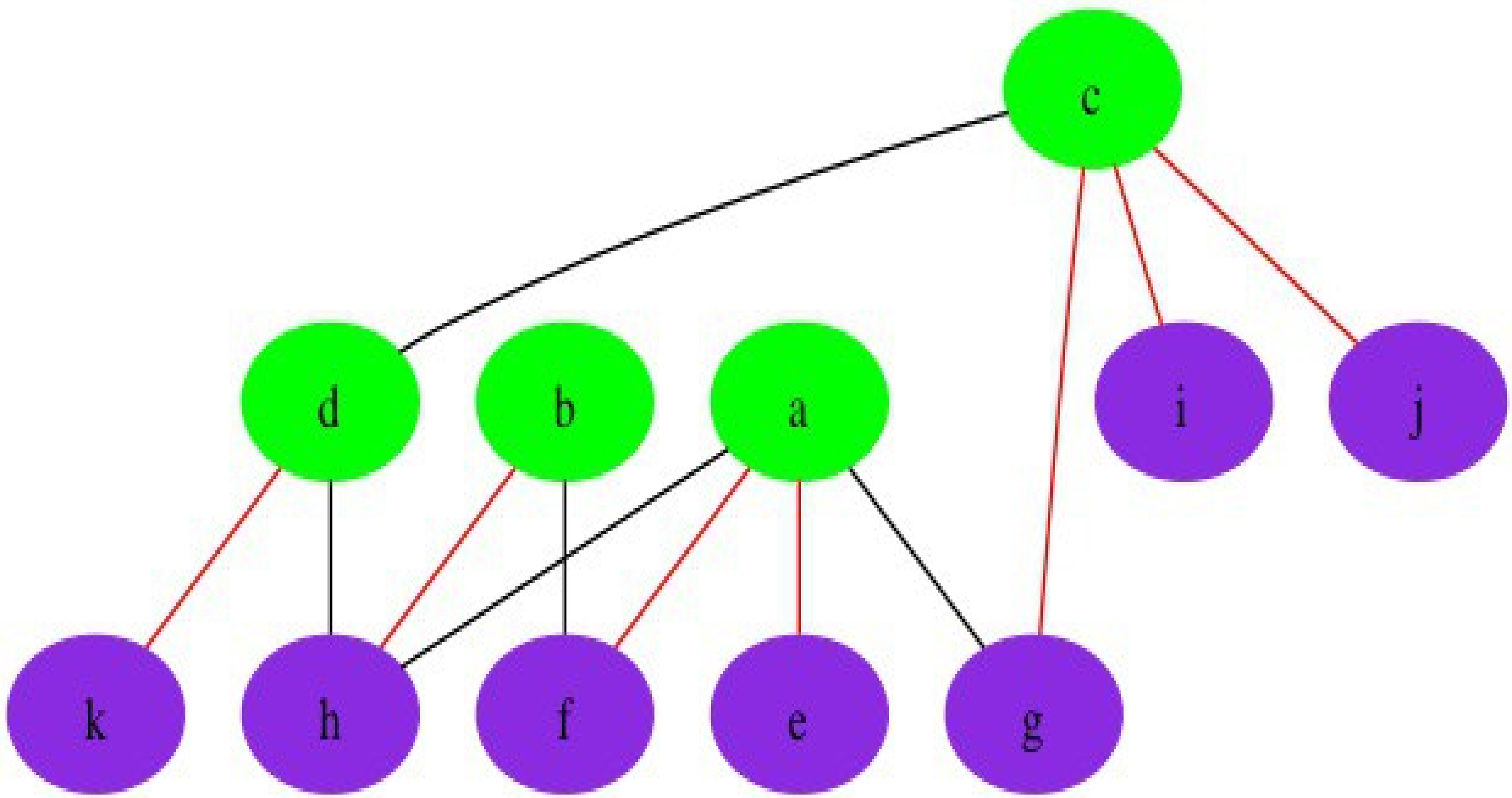
Complexité

- Un couplage parfait n'existe pas toujours (ordre impair)
- Le nombre de couplage est une fonction exponentielle du nombre de nœuds (mais il existe des heuristiques).
- Couplage de cardinalité maximale : → complexité polynomiale

Recouvrement (minimum)

- Recouvrement : famille d'arêtes telle que tout noeud du graphe soit l'extrémité d'au moins une arête de la famille
- Contrainte :
  - Caméra placée au milieu d'un couloir (une arête) sait surveiller les deux carrefours qui constituent les extrémités d'un couloir.
- Quel est le nombre minimum de caméra ?

Exemple : Recouvrement

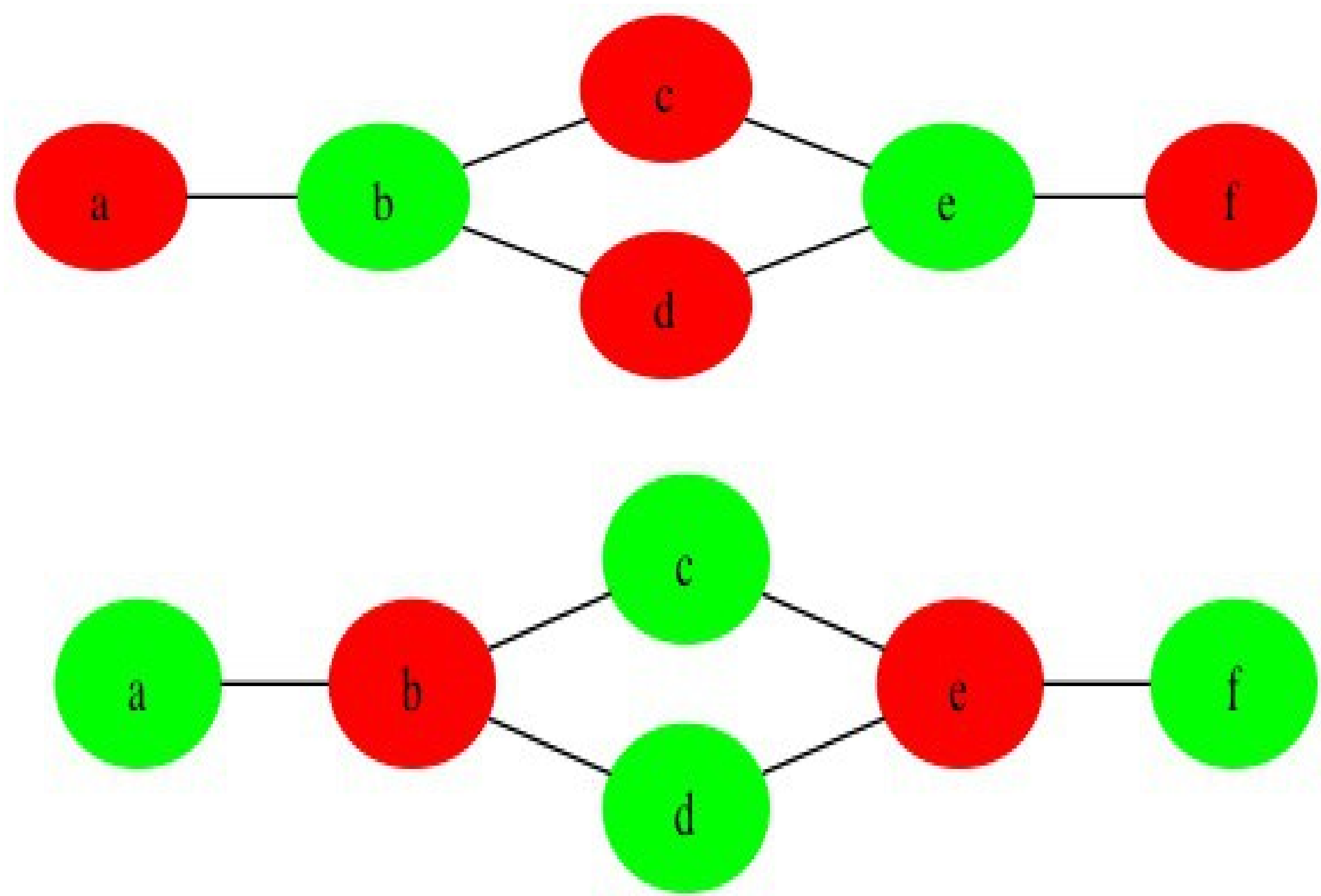


- 7 arêtes : 7 caméras

Stable

- Principe : Graphe orienté ou non
- Définition
  - 2 noeuds de S ne sont pas adjacents
- Formulation :
  - $S \subseteq X$ Stable $\Leftrightarrow \forall x \in S, \Gamma(x) \cap S = \emptyset$
- En général non unique
- Ensemble stable maximal ($\alpha = \max |S|$)

Exemple : Stable

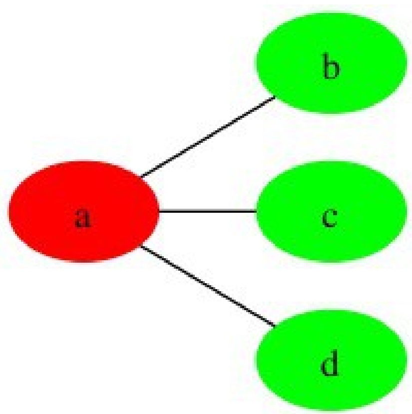


Exemple (2)

- Problème des 8 dames (échec)
  - 64 noeuds (⇔ une case = 1 noeud)
  - y $\in \Gamma(x)$ si les cases x et y sont sur la même rangée ou la même diagonale
  - Stable maximum … 92 solutions

Bi-parti et Stable

- On peut partitionner l'ensemble des nœuds en deux stables : G(X,Y;E) de cardinalité p et q
- Si tout nœud de X est relié à tout nœud de Y par une arête : bi-parti complet : $K_{p,q}$



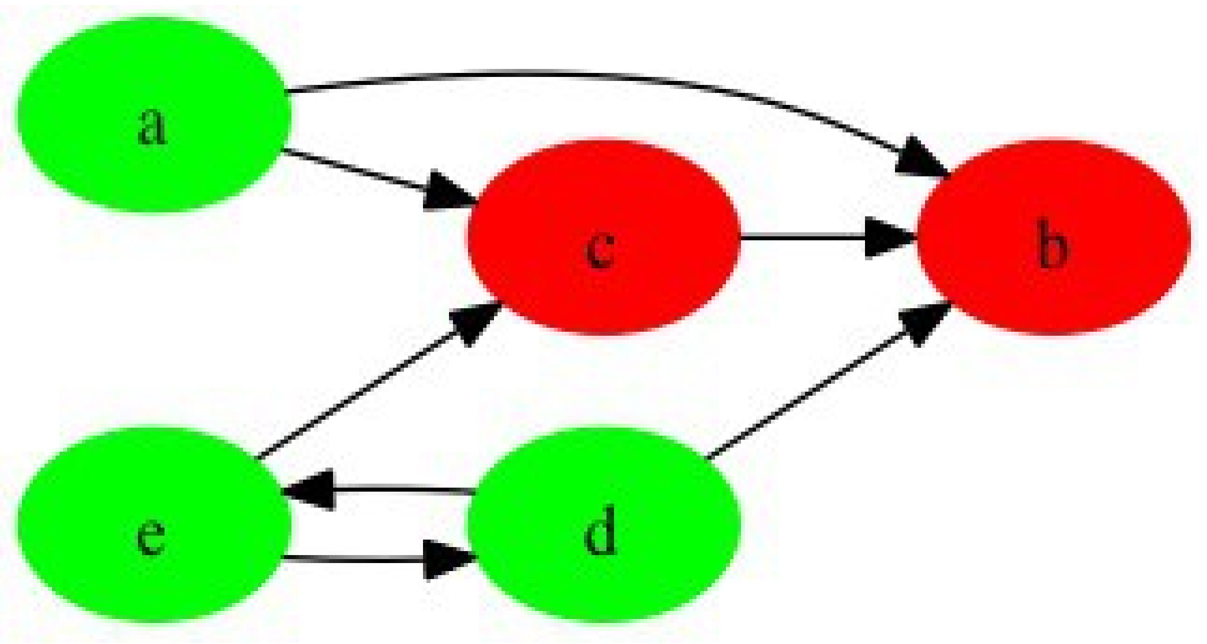
$K_{1,3}$

Absorption

- Principe : Graphe orienté
- Définition :
  - Tous les noeuds n'appartenant pas à A ont au moins un successeur dans A
- Formulation :
  - A est absorbant $\Leftrightarrow \forall\, x \notin A, \Gamma+(x) \cap A \neq \emptyset$
- Objectif :
  - Ensemble absorbant minimal ($\beta = \min |A|$) / maximal

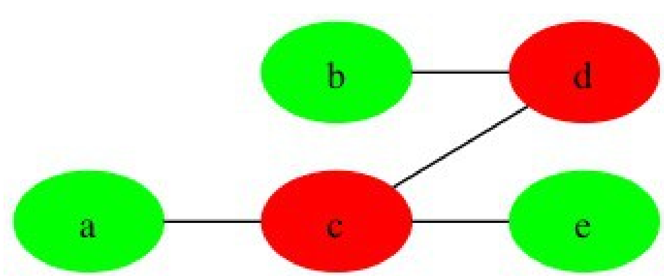
Exemple : Absorption

- Graphe :
  - Noeud : point stratégique (altitude variable)
  - Arc : est surveillé par (obtention de Γ^{+})
  - Problème : valeur de β (ici 2)



Support (Transversal)

- Principe :
  - non orienté
  - quel que soit le « successeur »
- Définition
  - Toute arête de G a au moins une extrémité dans T
- Formulation
  - T support de G ⇔ (X – T) stable de G



Noyau

- Un noyau est un ensemble S
  - Stable : $x \in S \Rightarrow \Gamma(x) \cap S = \varnothing$
  - Absorbant : $x \notin S \Rightarrow \Gamma(x) \cap S \neq \varnothing$
- Tout graphe sans circuit a un noyau unique
- Exemple :
  - Marienbad

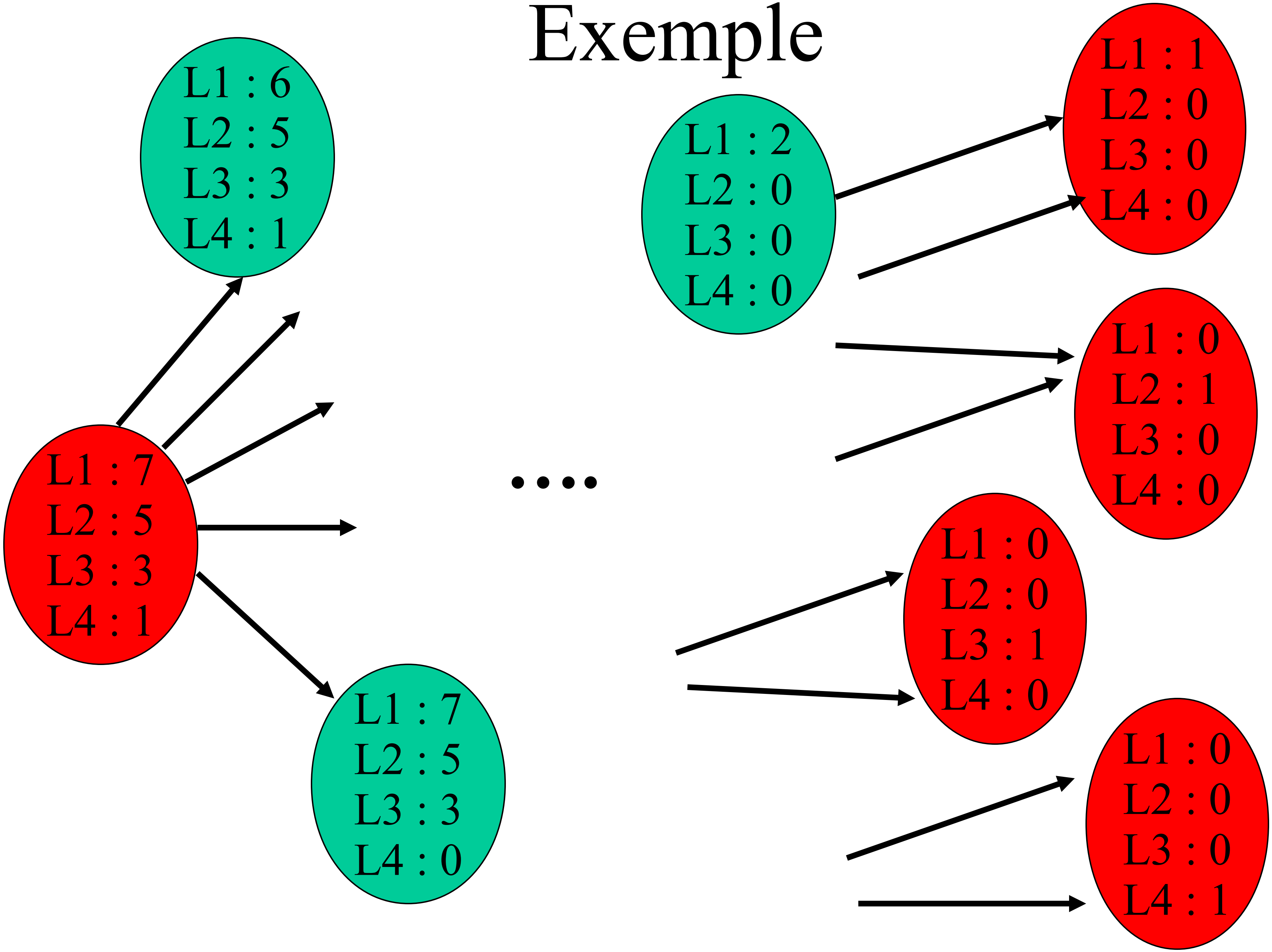
L'année dernière à Marienbad

(Alain Robbe-Grillet / A. Resnais)

- Voix de M : […] je connais un jeu auquel je gagne toujours...
- Voix de X : Si vous ne pouvez pas perdre, ce n'est pas un jeu !
- Voix de M : Je peux perdre … Mais je gagne toujours
- X : essayons

(… et il perd !)

Exemple

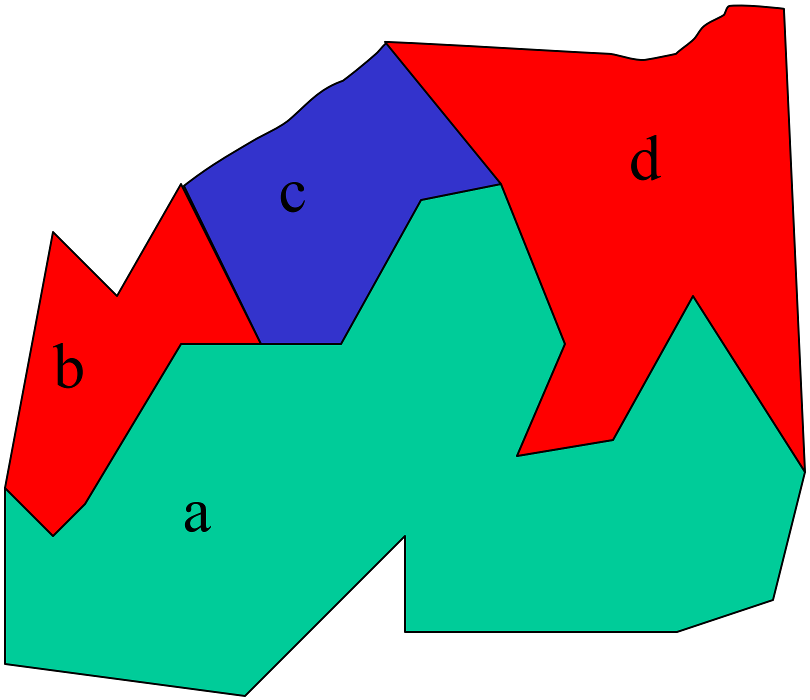

L1 : 6
L2 : 5
L3 : 3
L4 : 1
L1 : 2
L2 : 0
L3 : 0
L4 : 0
L1 : 1
L2 : 0
L3 : 0
L4 : 0
L1 : 0
L2 : 1
L3 : 0
L4 : 0
L1 : 7
L2 : 5
L3 : 3
L4 : 1
....
L1 : 0
L2 : 0
L3 : 1
L4 : 0
L1 : 7
L2 : 5
L3 : 3
L4 : 0
L1 : 0
L2 : 0
L3 : 0
L4 : 1

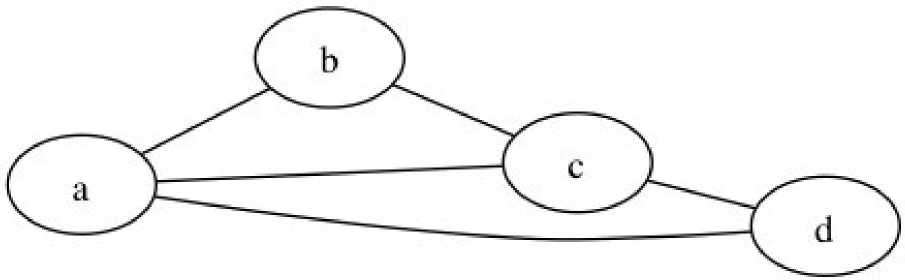
Nombre chromatique

- Le nombre chromatique (γ) est le nombre minimum de couleurs nécessaires pour colorier les noeuds d'un graphe sans que deux noeuds adjacents soient de la même couleur (resp. indice chromatique / arête).
- Nombre chromatique = nombre minimum de stables dont l'union est X (liaison : α / γ). Détermination difficile.
- Question : combien faut-il de couleurs pour colorier une carte sans que deux régions adjacentes aient la même couleur ?

Exemple

- Un noeud = 1 face
- Un arc = une adjacence de face

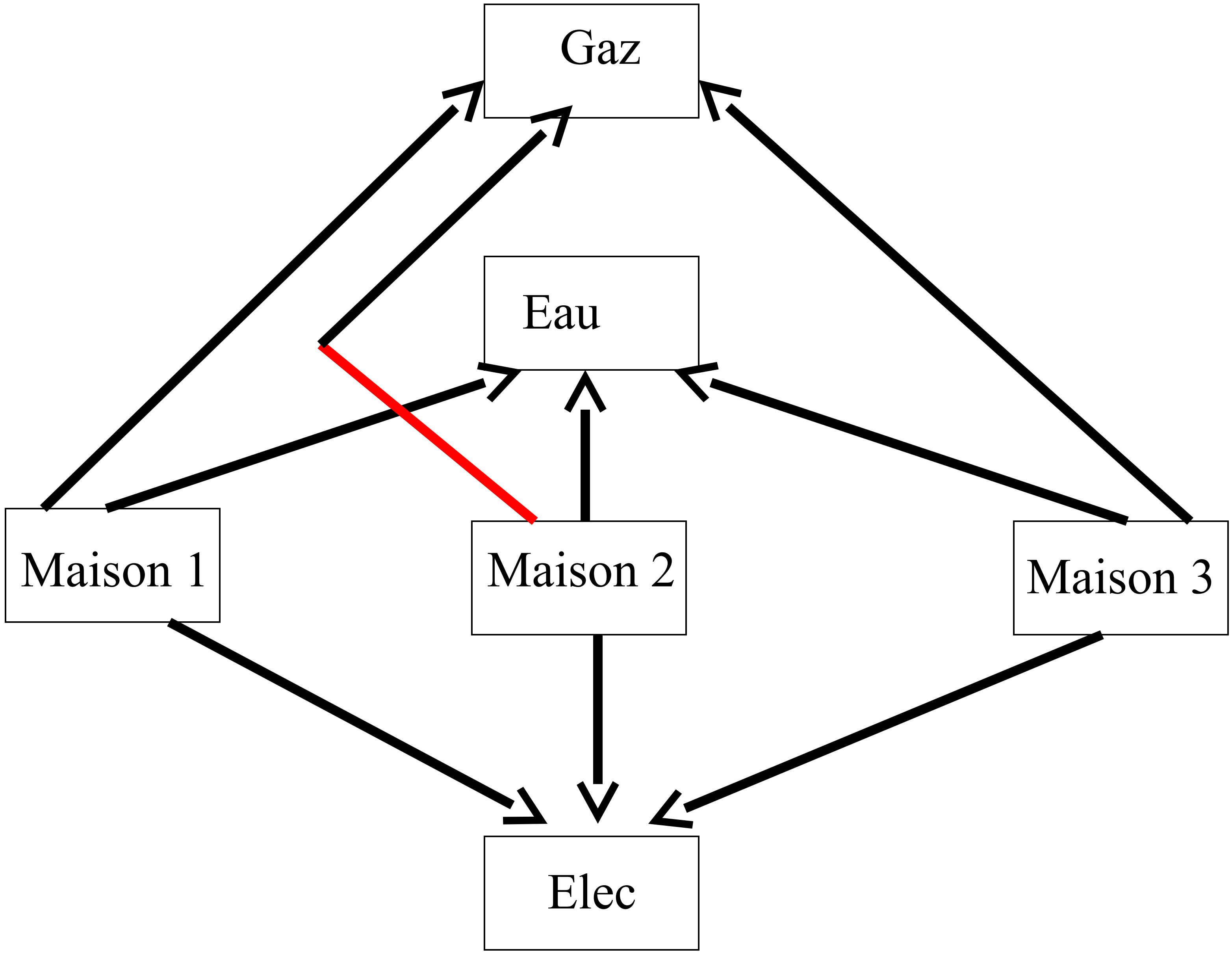




Planarité

- Définition : Un graphe est planaire lorsqu’il admet une représentation sur un plan telle que deux arêtes ne se rencontrent pas en dehors de leurs extrémités
- Objectif : lisibilité du graphe, propriétés physiques incompatibles
- Exemple : villas / usines

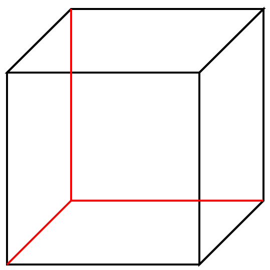
Exemple : Planarité

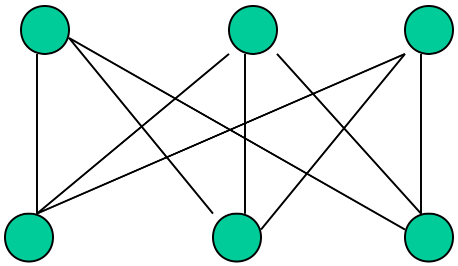


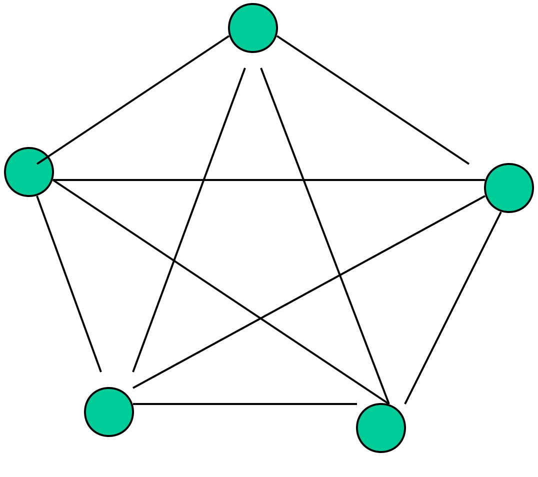
Planarité (2)

- Planaire saturé : l'ajout d'un arc fait perdre la propriété de planarité
- Dans un graphe planaire saturé les faces délimitées par les arcs sont triangulaires
- Dim > 2 : Certains graphes peuvent être à 3 dimensions (ex : cube)
- Graphe non planaire <=> contient un sous-graphe homéomorphe à K3,3 ou à K5

Exemple : Planarité (2)



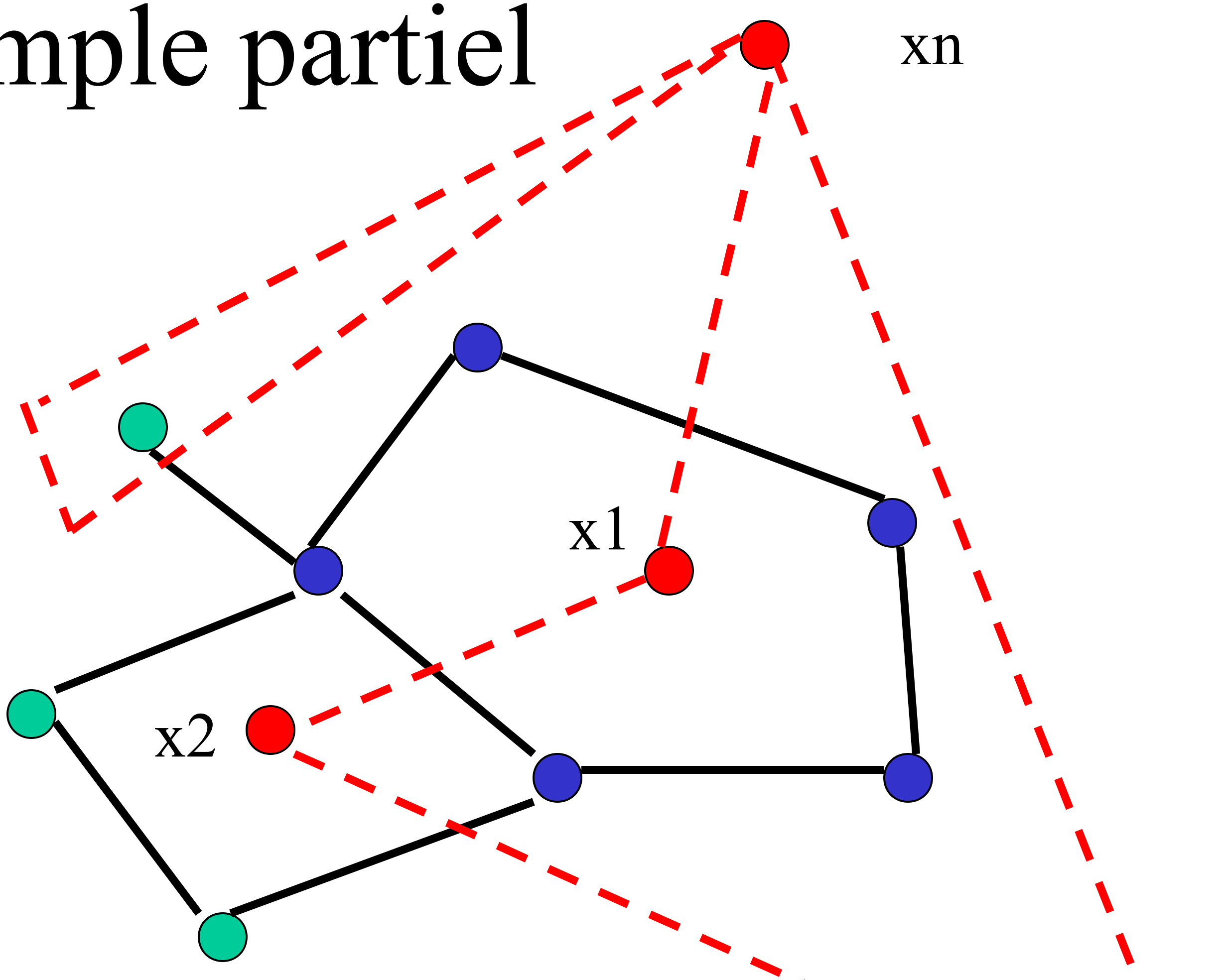




Dualité

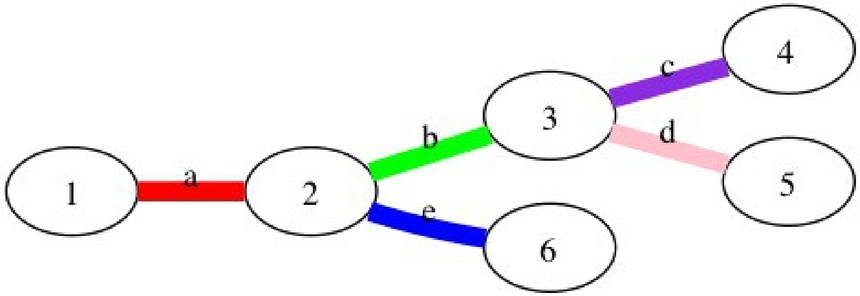
- Faire correspondre à un graphe G (planaire, connexe et sans noeud isolé) un graphe G* tel que :
  - A l'intérieur de toute face s de G on place un noeud x* de G* (avec une face extérieure)
  - A toute arête e de G on fera correspondre une arête e* de G* qui reliera les noeuds x* et y* correspondant aux faces s et t de part et autre de l'arête e
  - G* est planaire, connexe et sans noeud isolé

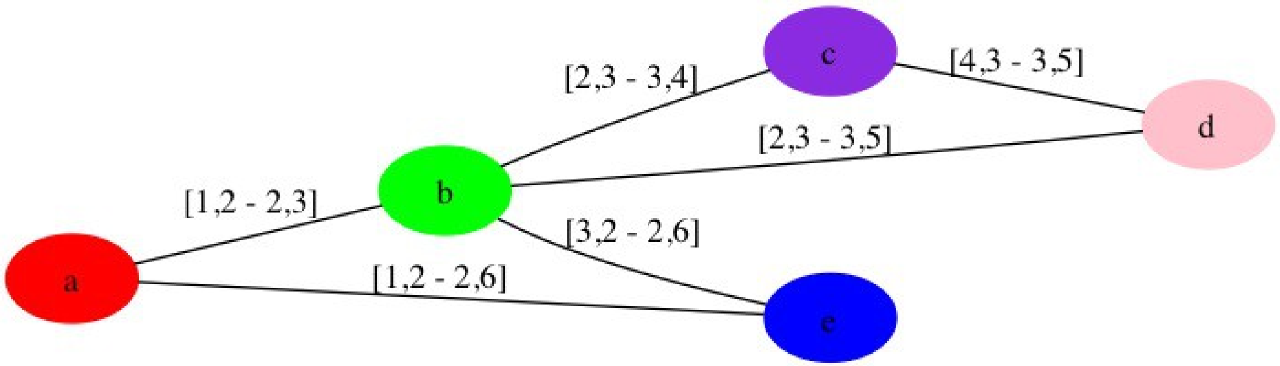
Exemple partiel



Graphe de Ligne

- Chaque arête de G est un noeud du graphe G'.
- Arête dans le graphe G' entre deux noeuds si les arêtes correspondantes dans G sont adjacentes.



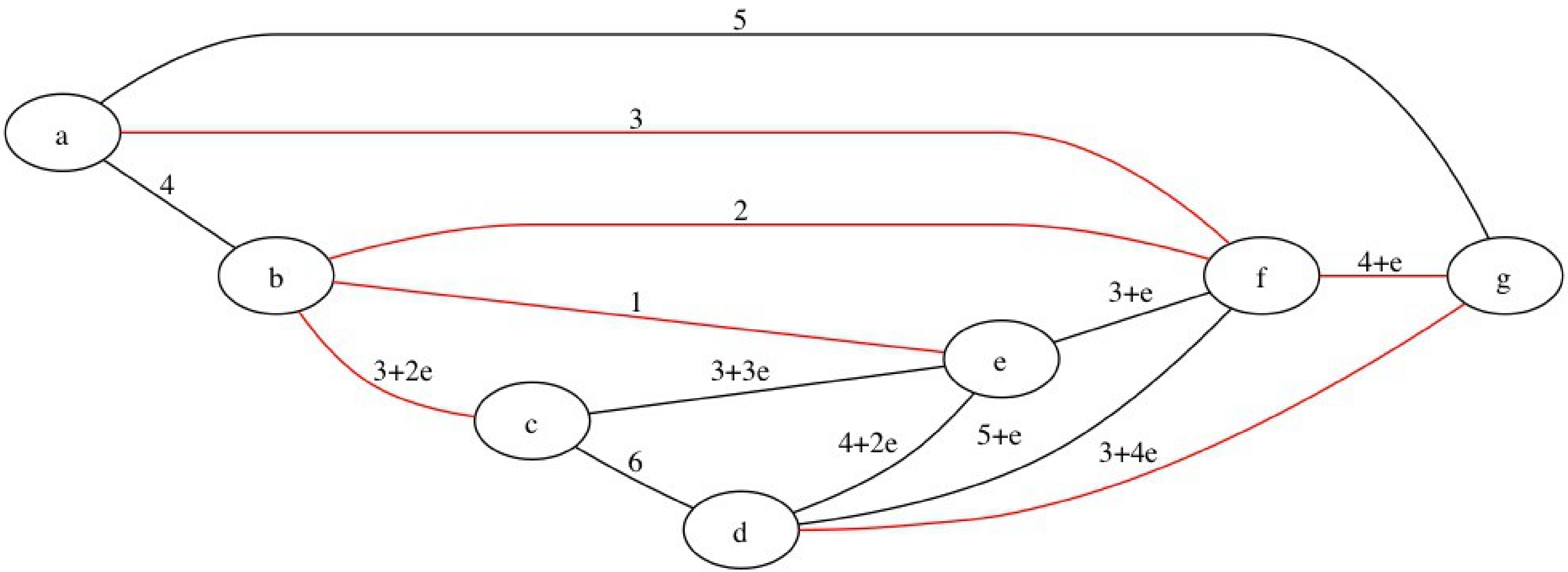


Arbre de coût minimal

- Définition : Trouver un arbre (graphe partiel de G) dont la somme des poids des arcs est minimal
- Objectif : limitation du nombre de connections entre nœuds (arbre) avec une valuation sur les arcs (par exemple une longueur ou un coût)
- Multiplier les valuations par -1, poids minimum -> poids maximum

Exemple : Arbre coût minimal

- Réseau de communication
- Les ε sont là pour différentier les coûts (sans que la somme dépasse la valuation de l’arête suivante dans l’ordre)



Gestion des flots

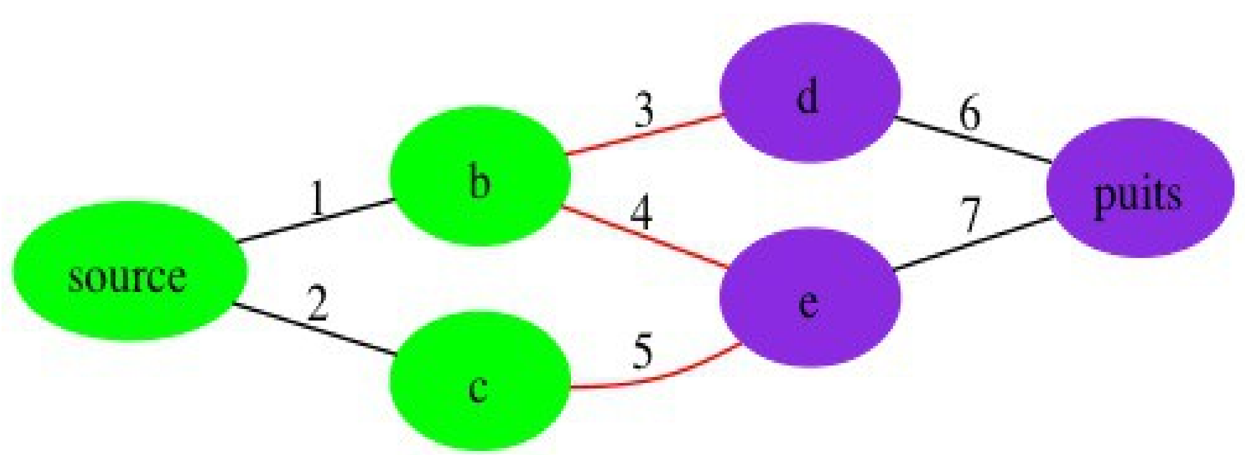
- Graphe → Réseau :
  - Graphe sans boucle
  - A chaque arc est associé une capacité ($c(u) \geq 0$)
  - Un seul nœud n'a pas de prédécesseur (source)
  - Un seul nœud n'a pas de successeur (puits)
- Flot :
  - Fonction qui associe à chaque arc une quantité ($f(u)$) dans le cadre du flot de la source au puits.
  - Pas de perte (autant en entrée qu'en sortie)

Propriétés

- Flot compatible :
  - Le flot qui traverse un arc ne doit pas dépasser la capacité de l'arc
    - $\forall\ u \in G,\ 0 \leq f(u) \leq c(u)$
- Flot complet :
  - Le flot traverse un arc en saturant sa capacité
    - $\exists\ u \in G,\ f(u) = c(u)$

Coupe

- Ensemble d'arcs {3, 4, 5} tel que si on les retire le graphe partiel ne contient plus de chemin de la source au puits.
- Partitionne l'ensemble des nœuds en deux parties (S et X-S). Coupe = {Arc / extrémité initiale dans S, terminale dans X-S}



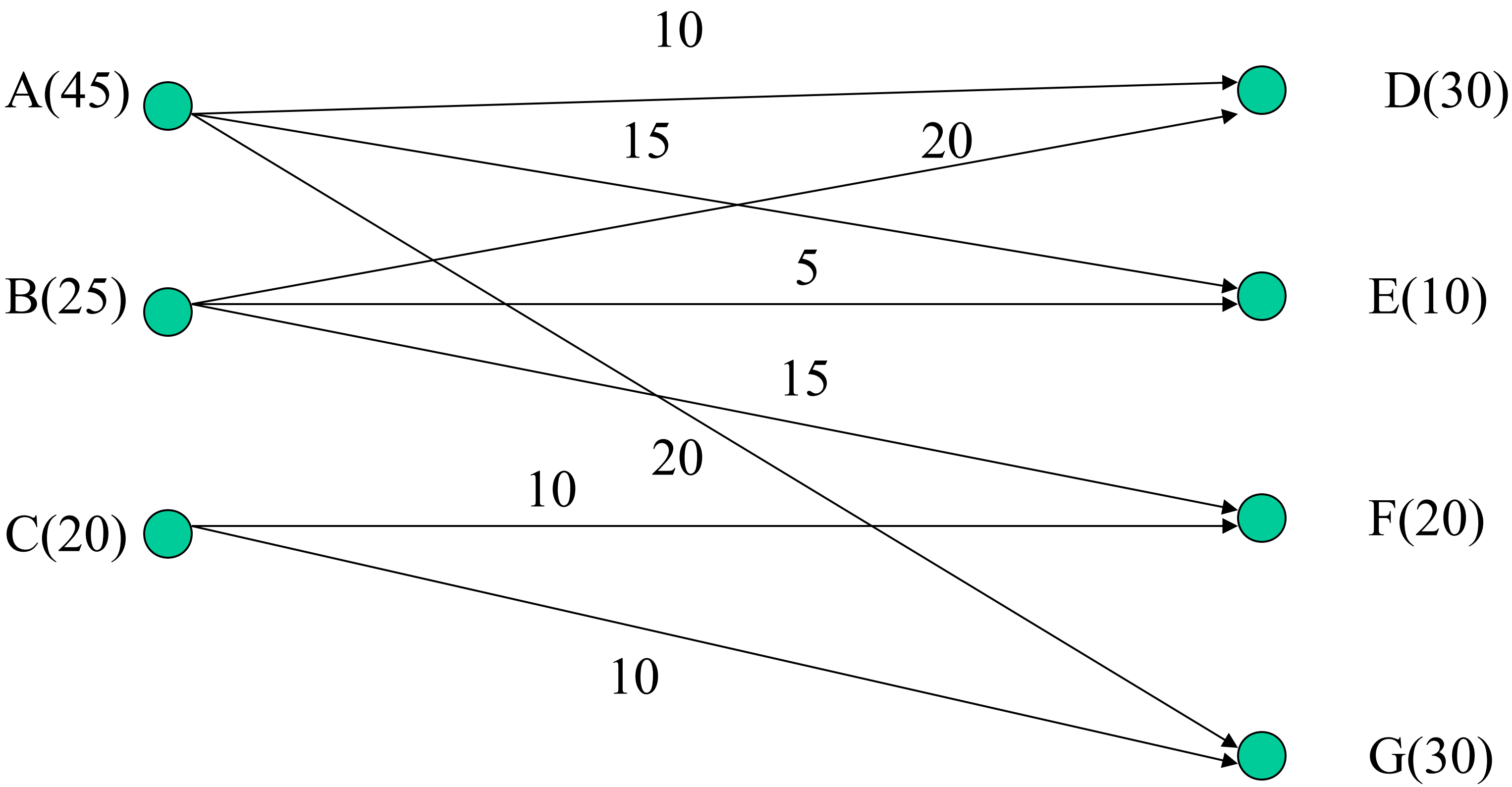
Propriétés

- Capacité d’une coupe : somme de la capacité des arcs de la coupe
- La valeur d’un flot sera inférieure ou égale à la capacité de n’importe quelle coupe
- La valeur maximum d’un flot est toujours égale à la capacité minimum d’une coupe (valide l’optimalité du flot)
- Problème : le nombre de coupe est exponentiel.

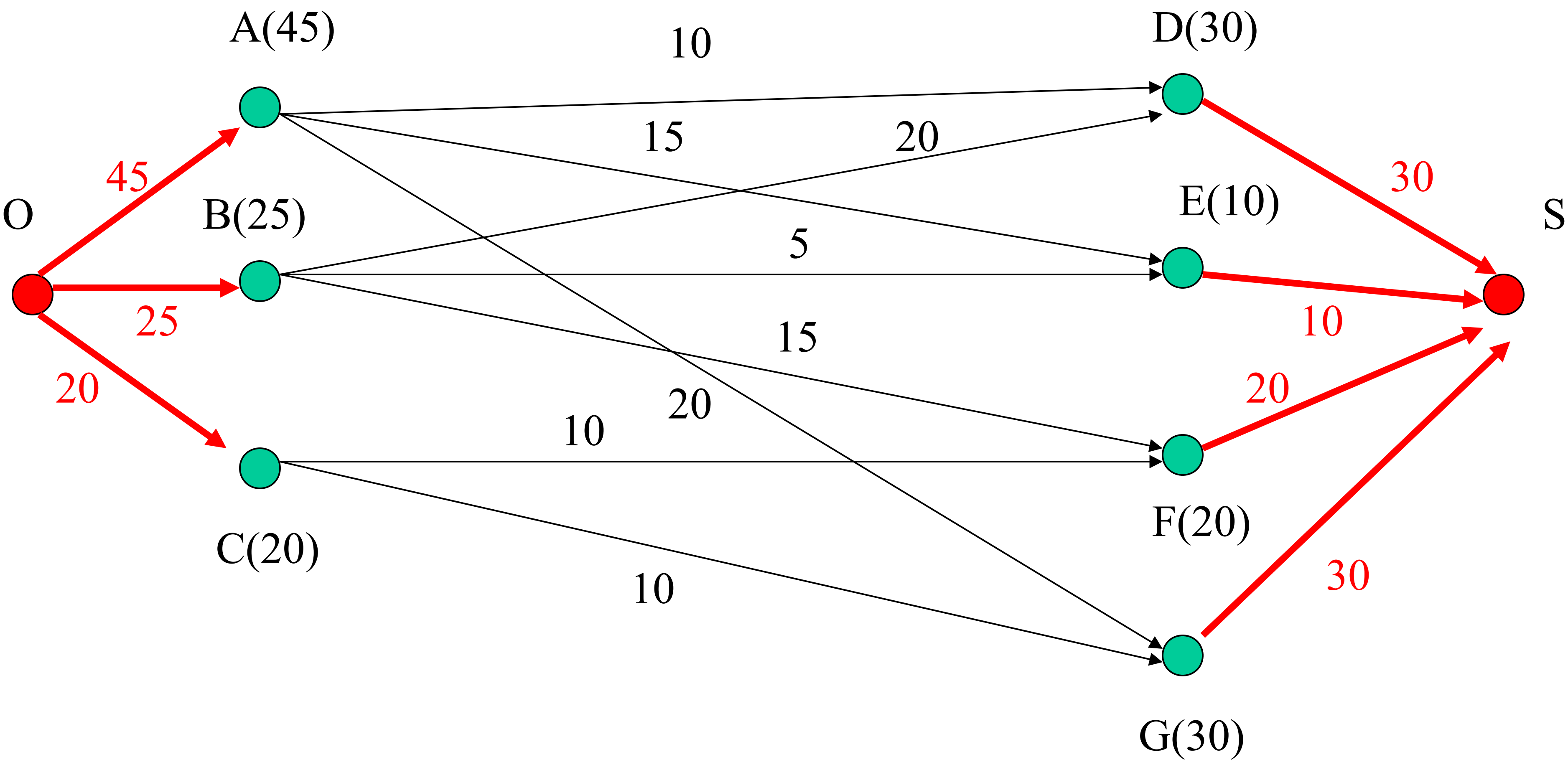
Objectif : Flot maximal

- Algorithme de référence : Ford-Fulkerson
- Principe :
  - Etape 1 : Flot nul
  - Etape 2 : Chercher une chaîne source → puits
  - Etape 3 (répétée) : augmente le flot
- Différence avec les chemins :
  - On va chercher à faire varier l'étiquette de l'arc
  - Chemin impose arcs dans le même sens (direct)
  - Ce n'est plus un problème base de données

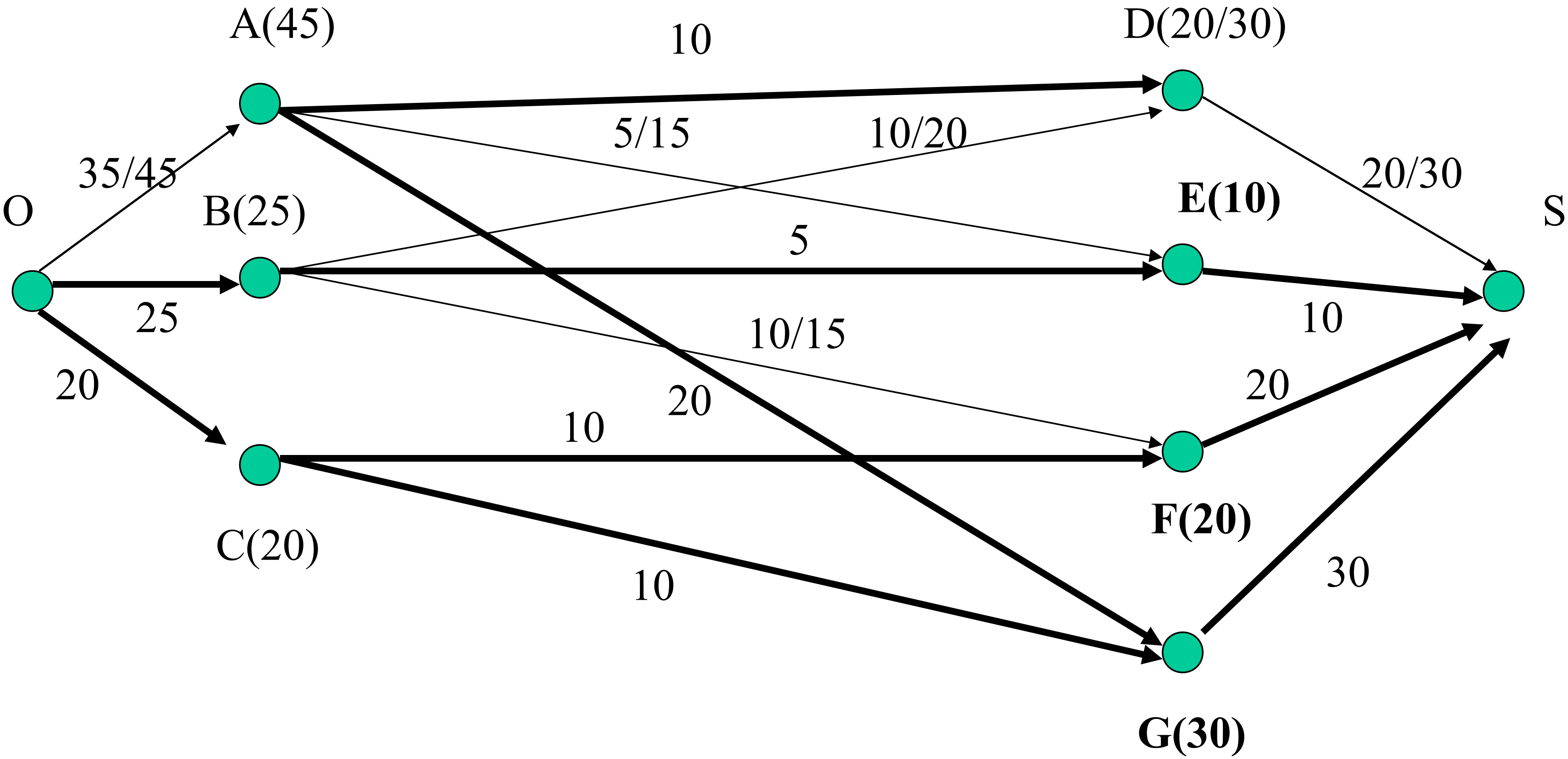
Exemple : Flot (1)



Exemple – Construction du réseau (2)

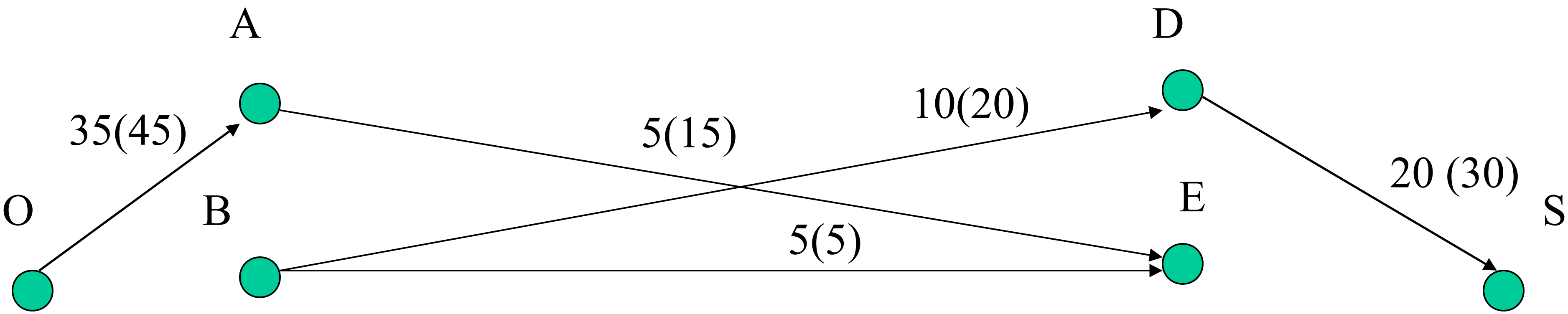


Exemple – Itérative (3)



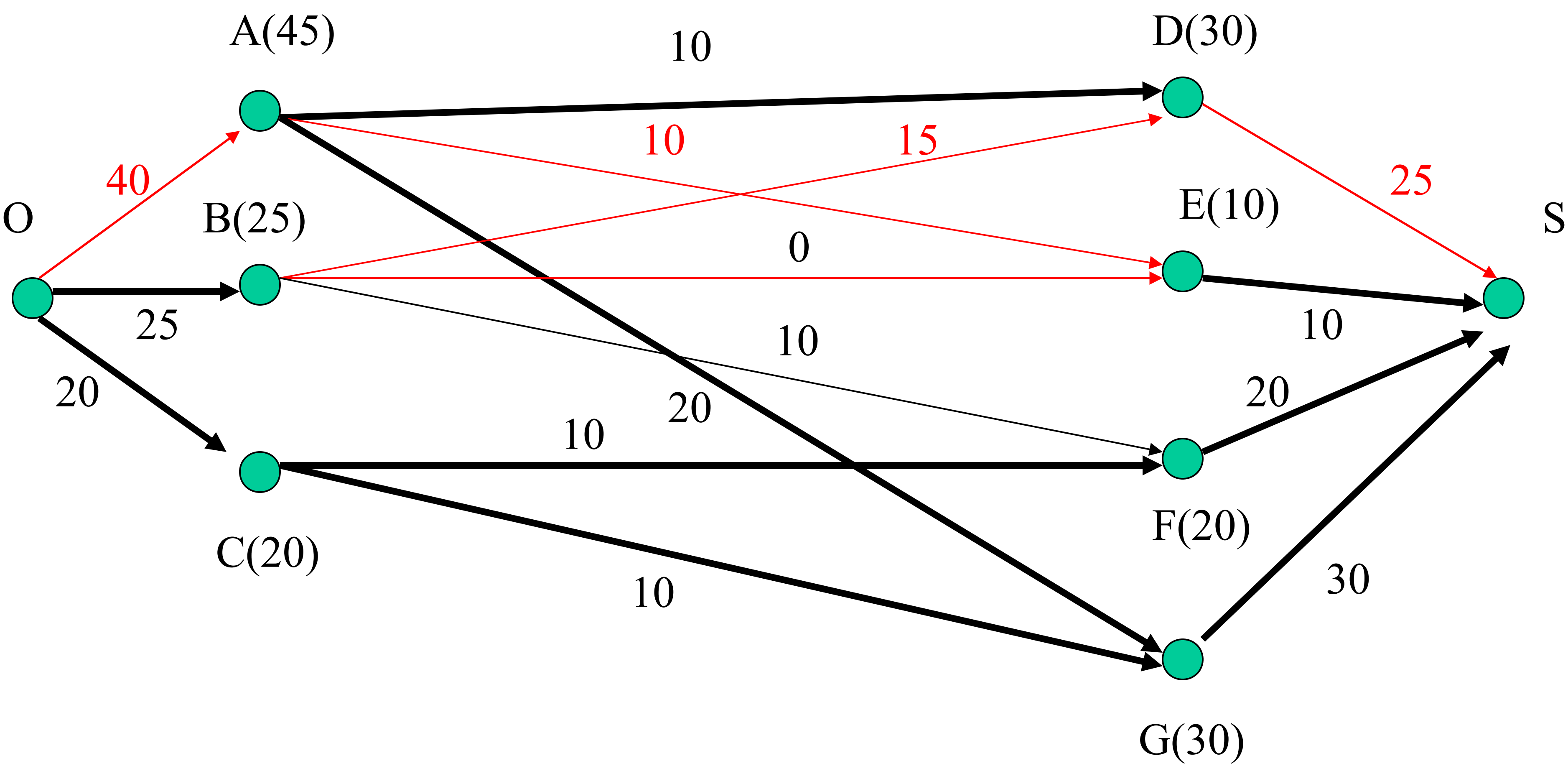
C (20) / A (35) / B (25) : Amélioration ?

Exemple – Itérative (4)



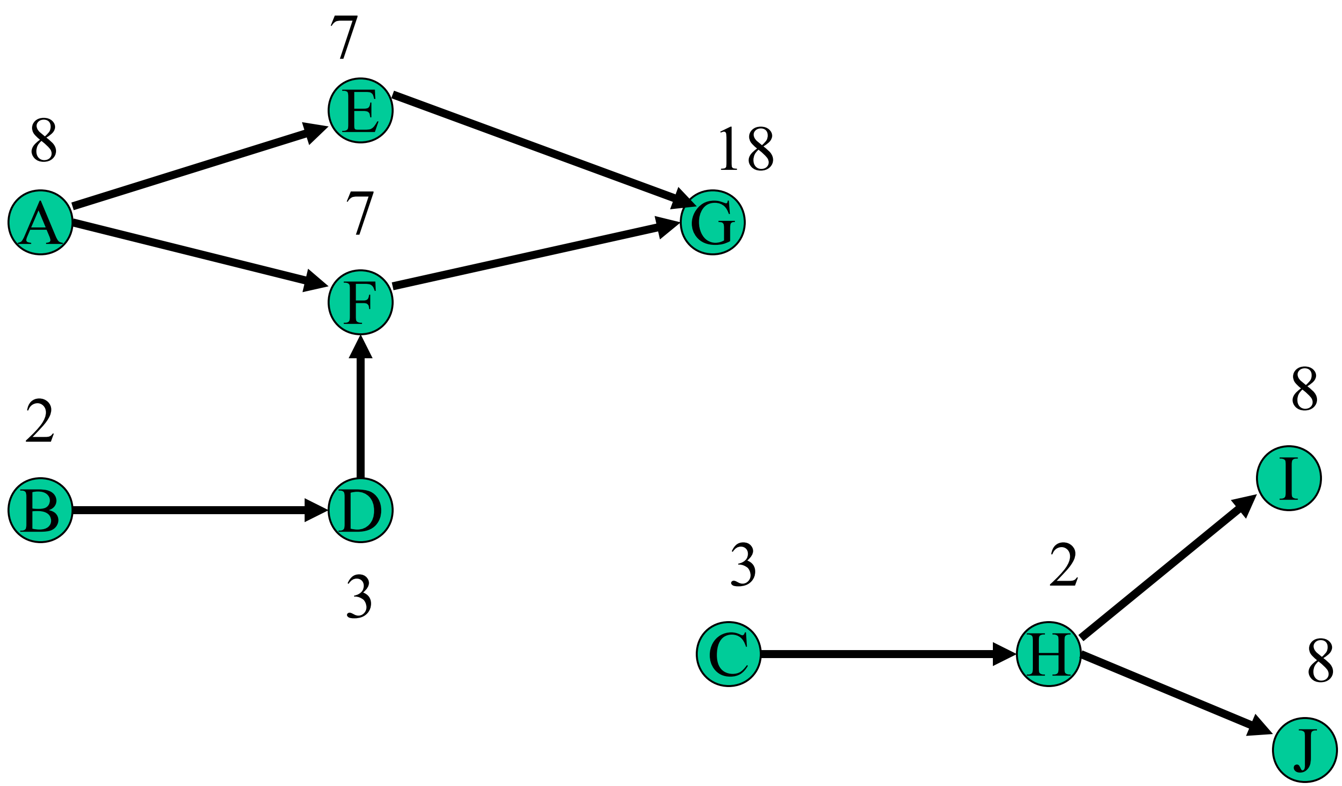
- Chaine : Arc indirect E -> B de 5 maximum
  - => Augmenter (B, D), (A, E) (O, A), et (D, S) de 5

Exemple – Final (5)

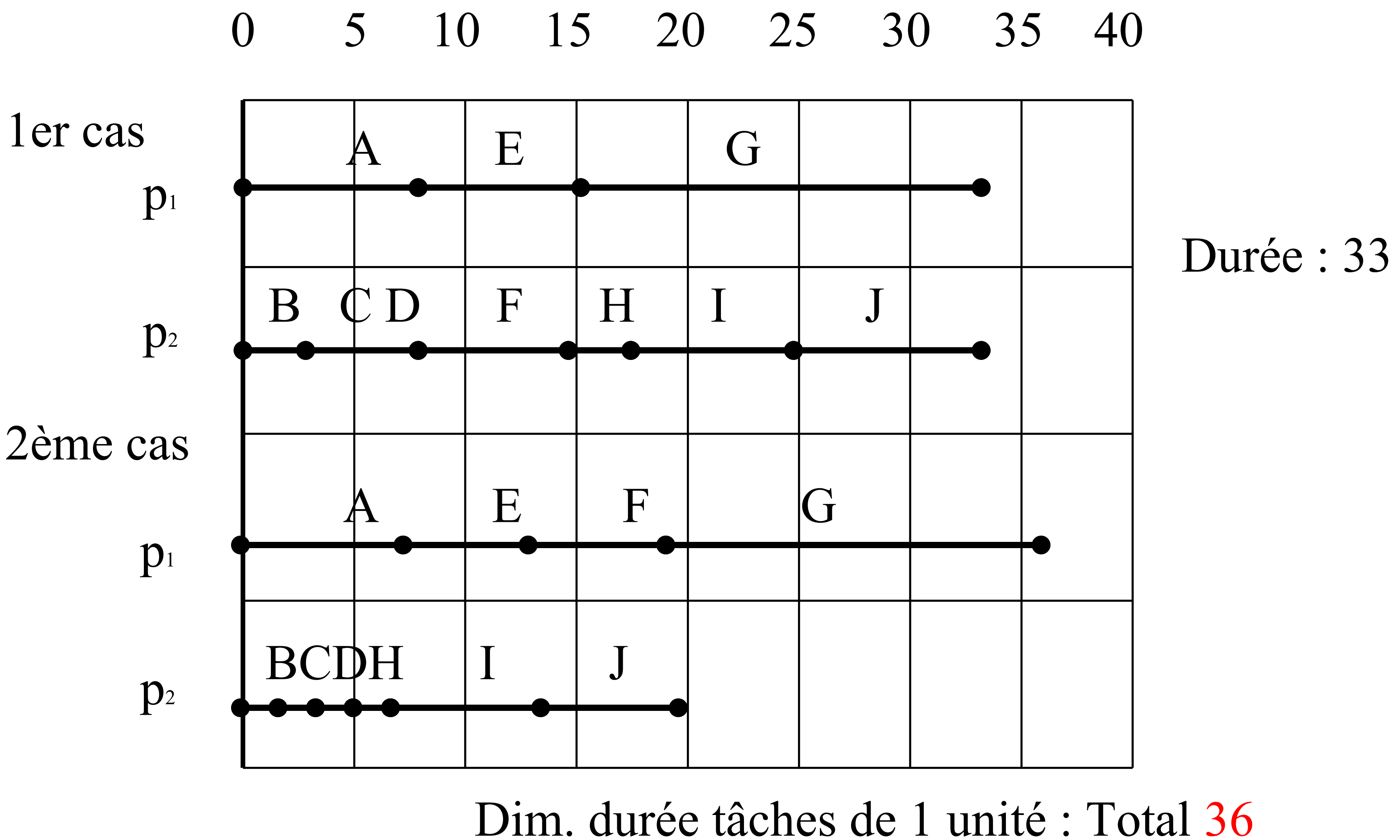


Déterminisme (1)

- Projet : 10 tâches (ex processus informatiques)
- Choix déterministe des tâches et des processeurs
- Pas de temps mort entre les tâches

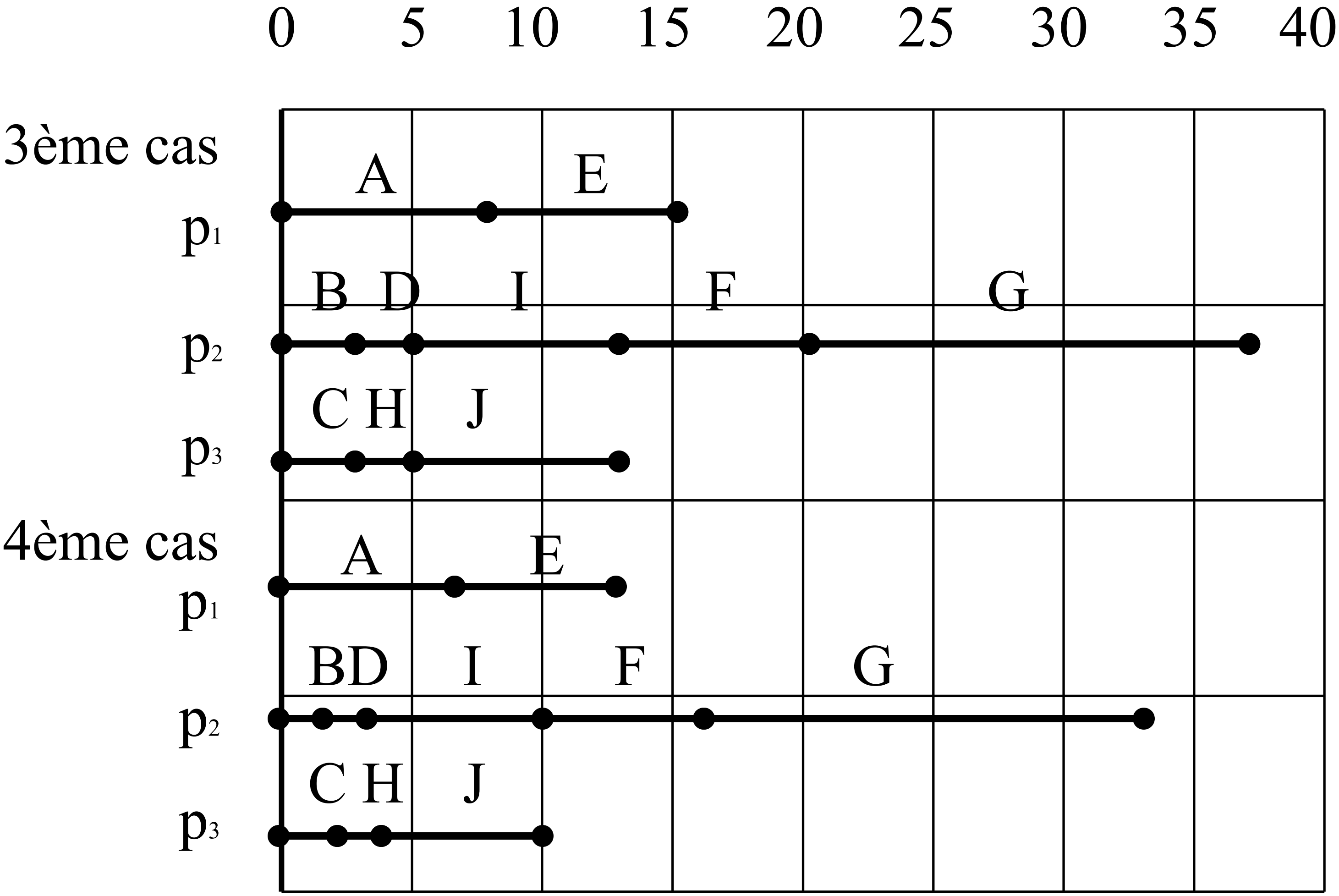


Exemple (1.1)



0 5 10 15 20 25 30 35 40
1er cas
p1
A E G
p2
B C D F H I J
Durée : 33
2ème cas
p1
A E F G
p2
BCDH I J
Dim. durée tâches de 1 unité : Total 36

Exemple (1.2)



Durées initiales
+ de processeurs
Total : 38

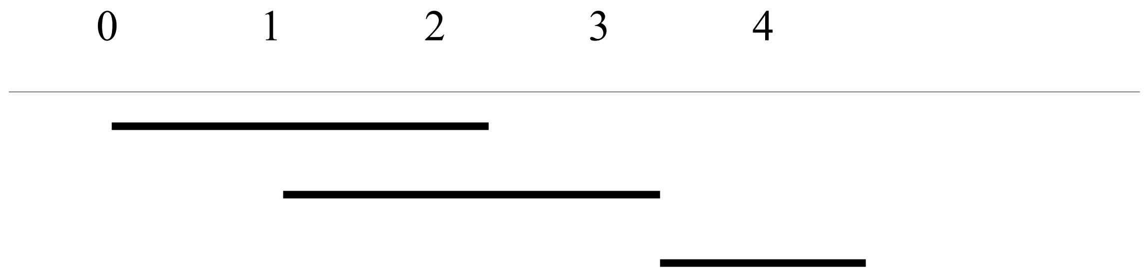
Durées <, nombre de processeurs > : Total 33

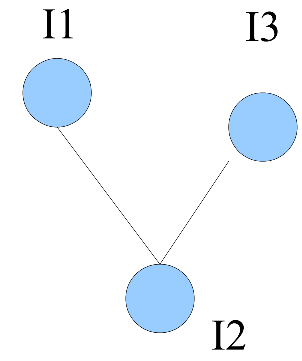
Attention …

- La durée totale n’est :
  - Ni systématiquement croissante de la durée des tâches
  - Ni systématiquement décroissante du nombre de processeurs
- … donc piège

Graphe d'intervalle

- Noeud : Ensemble
- Arête entre deux noeuds : Les deux ensembles ont une intersection non vide
- Exemple
  - N = { I1 = [0,2], I2 = [1, 3], I3 = [3, 4] }

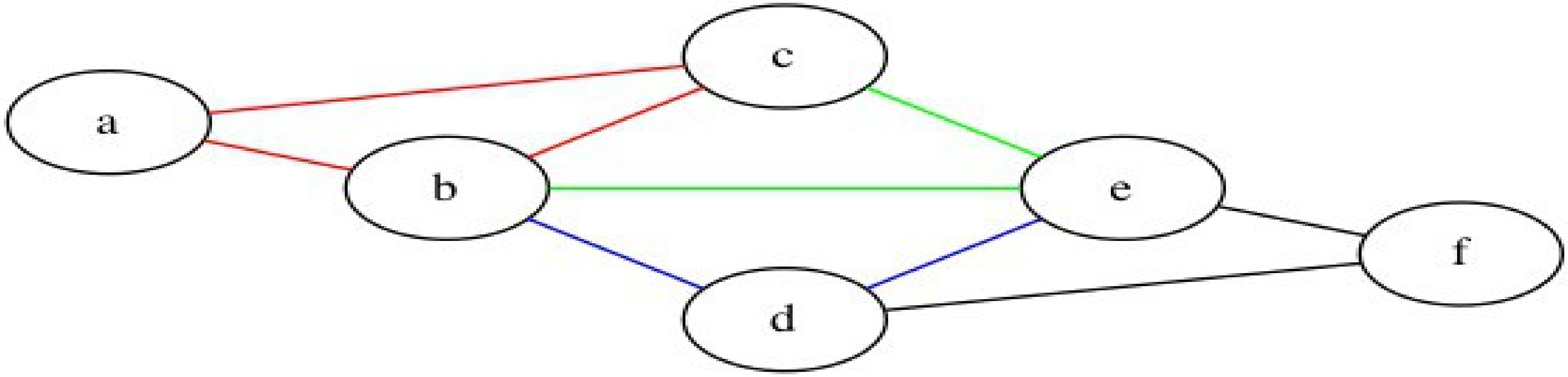




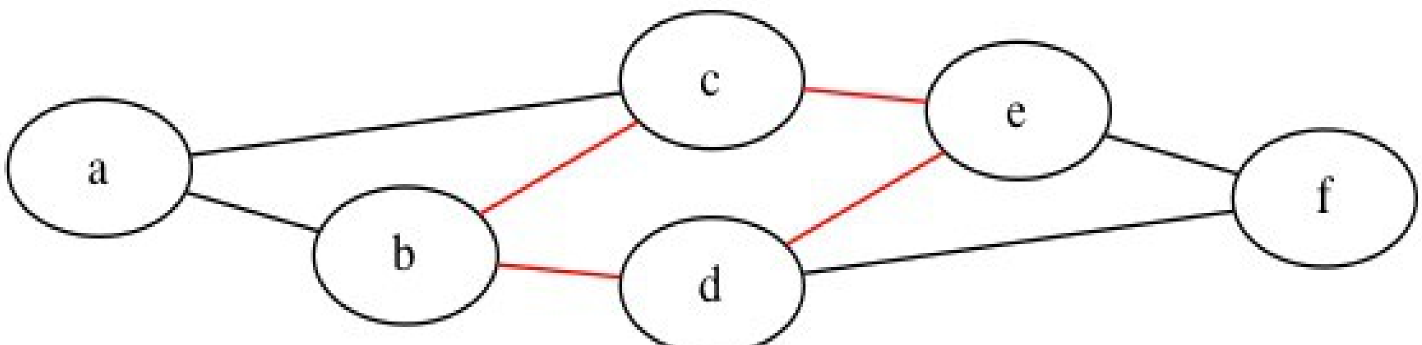
Graphe d'intervalle (2)

- G est un graphe d'intervalle <=>
  - G est triangulé
  - Complémentaire de G est un graphe de comparabilité
- G est triangulé <=
  - Tout cycle de longueur > 3 admet une corde
- Une corde : Une corde d'un cycle élémentaire est une arête qui relie 2 noeuds non consécutifs du cycle
- Tout sous-graphe induit par un sous-ensemble de noeuds doit être un graphe d'intervalle

Triangulé

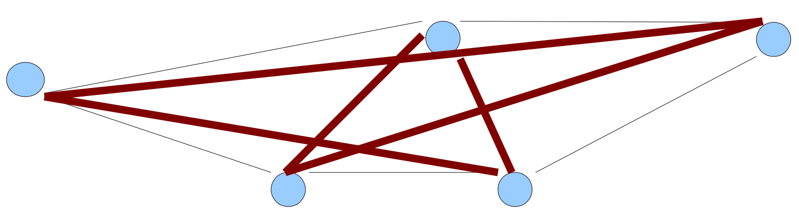


Non triangulé (c,e,d,b,c)



Graphe de comparabilité

- On recherche une relation d'ordre (transitive et anti-symétrique) en orientant les arêtes
- Graphe simple (sans boucle, 1-graphe) est de comparabilité <=>
  - Pour tout pseudo-cycle de longueur impaire (q > 1), Pseudo-cycle : possibilité de plusieurs fois la même arête
    - [a1, ... a2*q, a2*q+1, a1] il existe une arête de la forme :
    - [ai, ai+2] (+ : modulo 2q + 1)



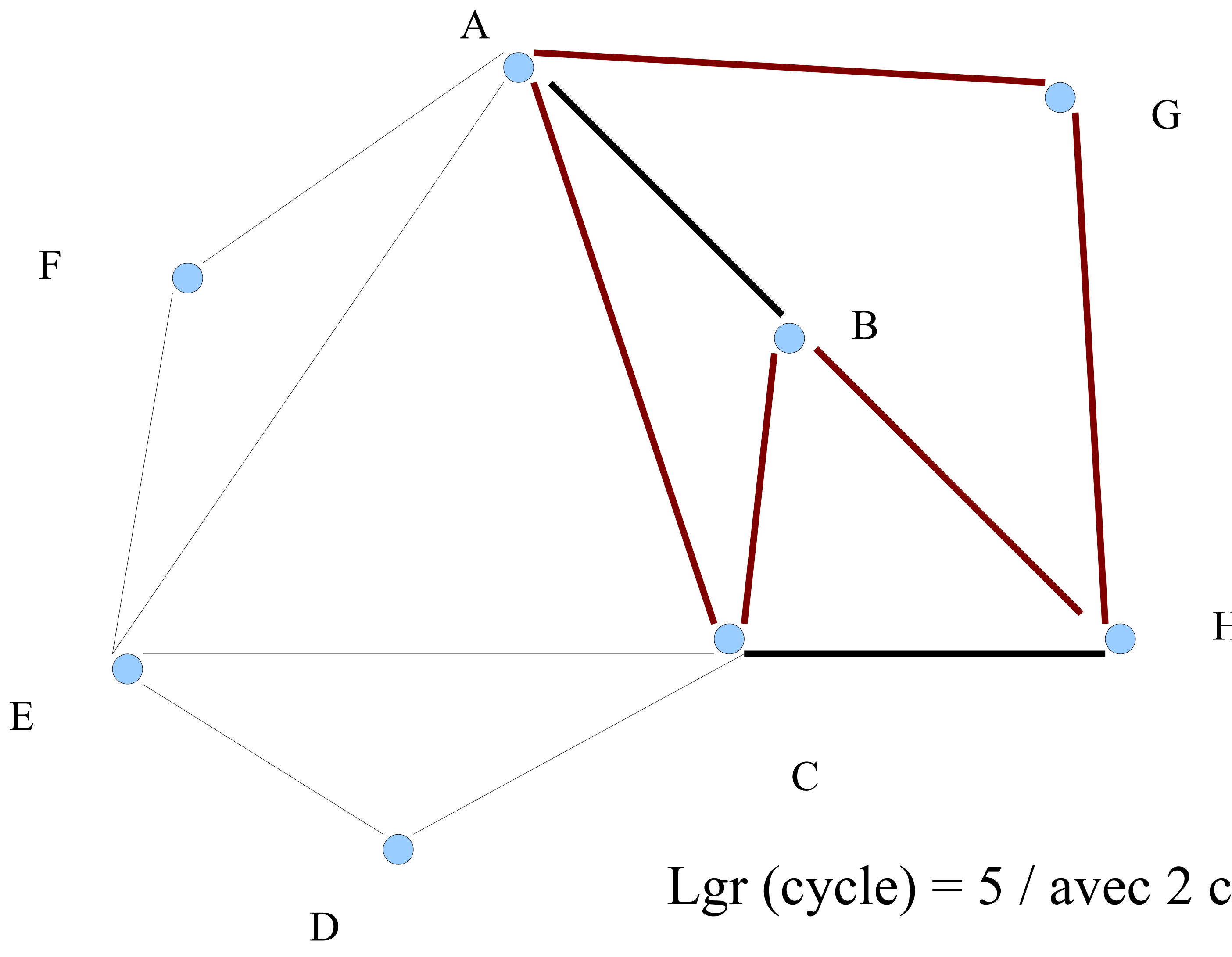
Exemple (Berge)

- Le Duc de Densmore est retrouvé mort dans son château de l'île de White. Le meurtre a nécessité une préparation.
- Avant sa mort, le duc avait invité 8 personnes sur l'île. Celles-ci se rappellent les autres personnes rencontrées, mais ont oublié la date précise de leur présence.
- Le marin qui faisait la navette vers l'île a aussi oublié les dates, mais il se souvient n'avoir transporté chacune que pour un seul aller-retour

Qui a vu qui ? Qui a tué ?

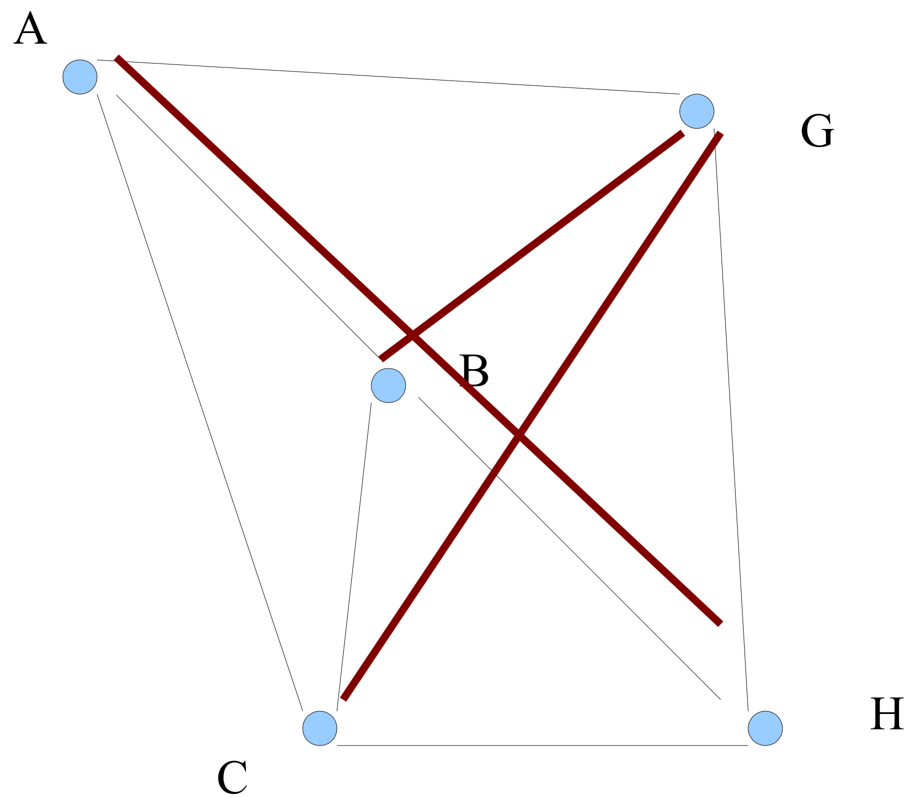
- Ann a vu Betty, Cynthia, Emily, Felicia et Georgia
- Betty a vu Ann, Cynthia et Helen
- Cynthia a vu Ann, Betty, Diana, Emily et Helen
- Diana a vu Cynthia et Emily
- Emily a vu Ann, Cynthia, Diana et Felicia
- Felicia a vu Ann et Emily
- Georgia a vu Ann et Helen
- Helen a vu Betty, Cynthia et Georgia

Représentation du problème « a vu »



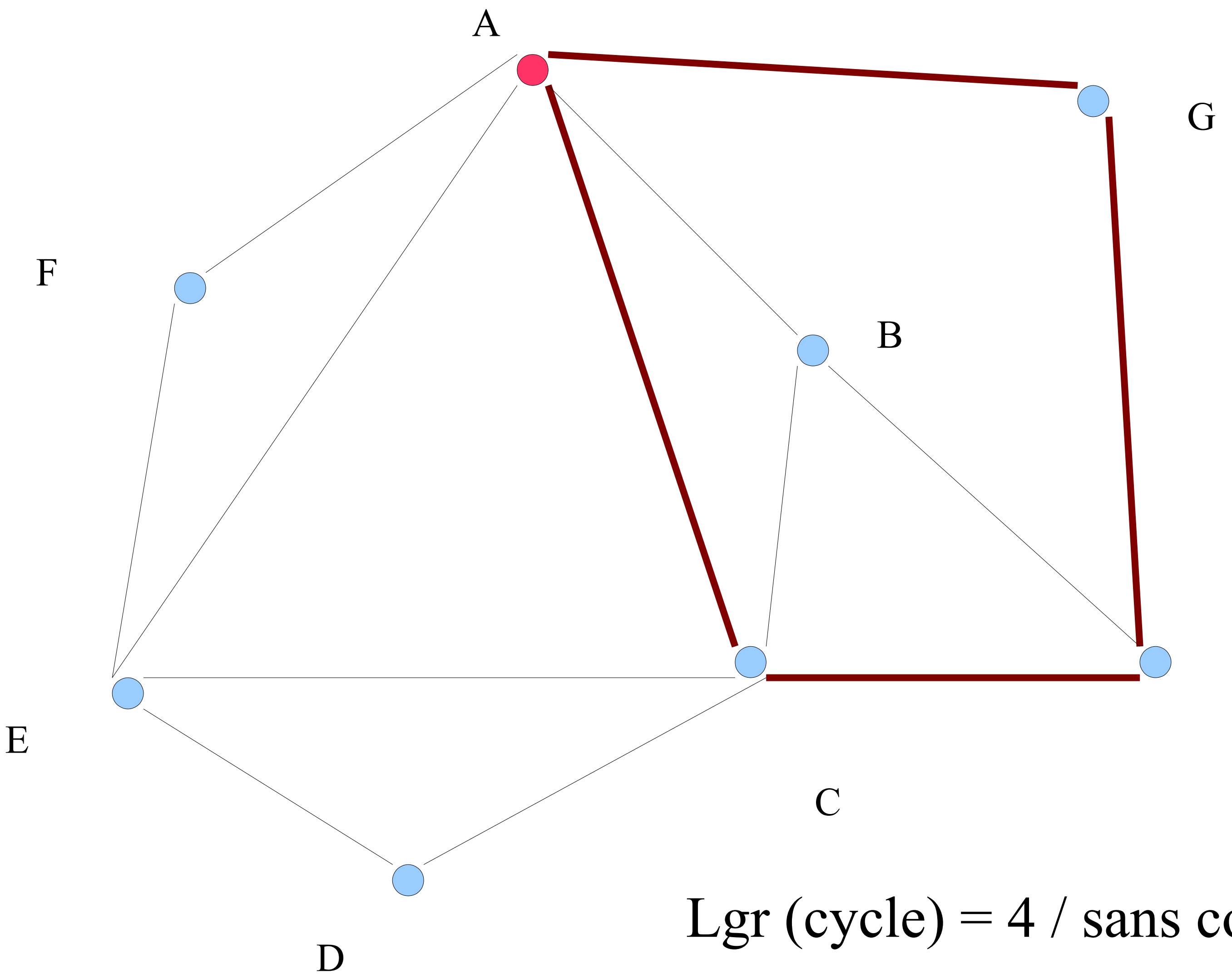
Lgr (cycle) = 5 / avec 2 cordes

Passage au complémentaire



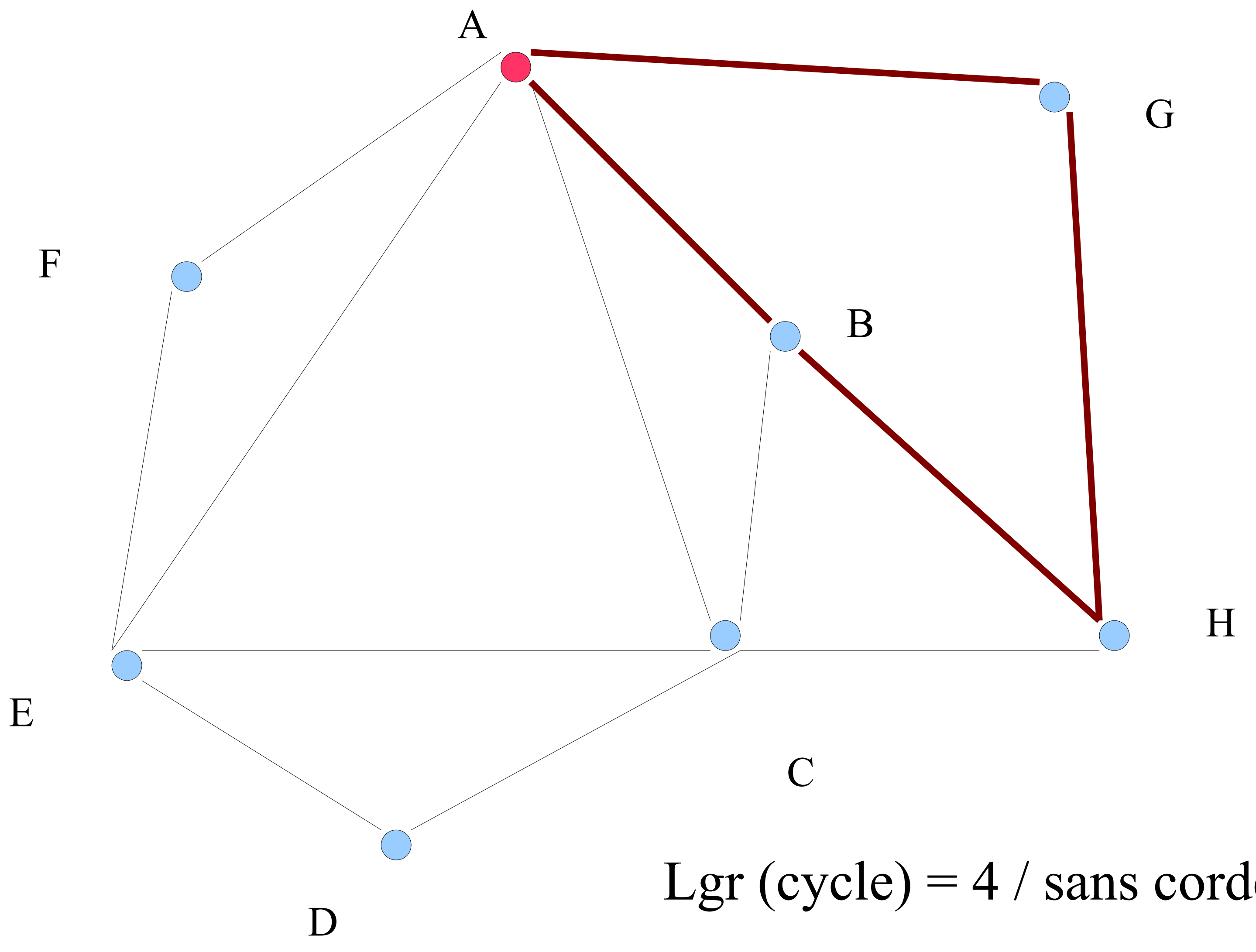
Pas de pseudo-cycle de longueur impaire présent

Représentation du problème (2)



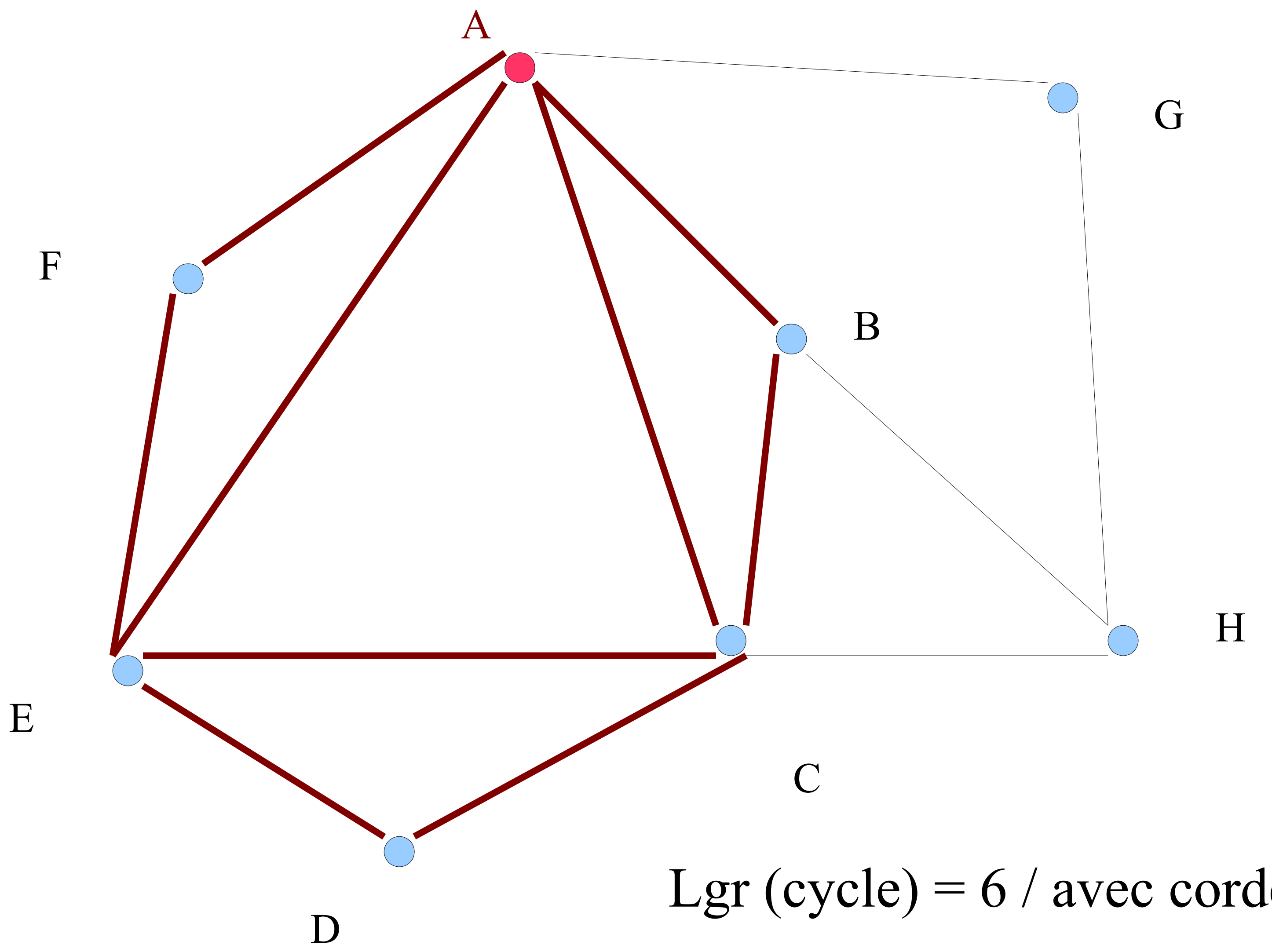
Lgr (cycle) = 4 / sans corde

Représentation du problème (2)



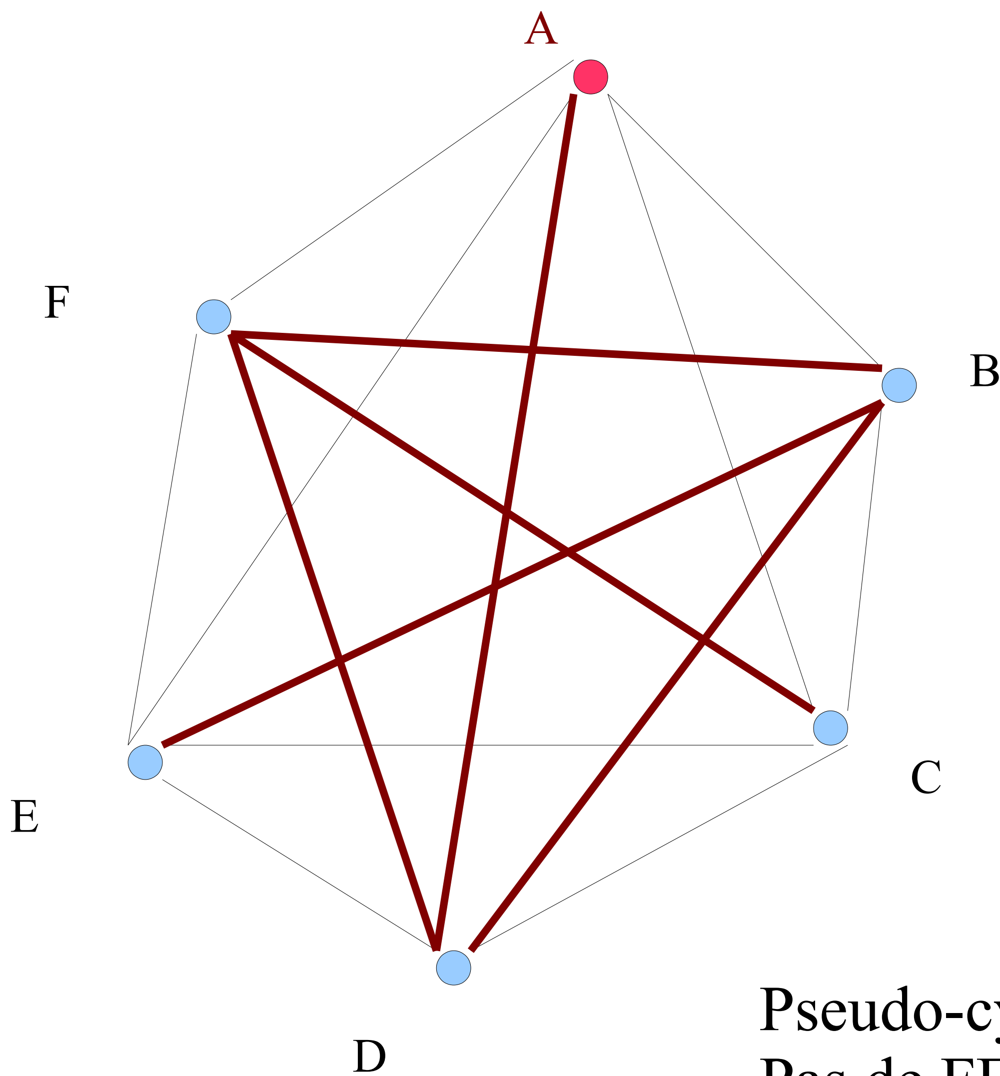
Lgr (cycle) = 4 / sans corde

Représentation du problème (3)



Lgr (cycle) = 6 / avec cordes

Complémentaire (3)



Pseudo-cyle : FBEBDF Lgr = 5
Pas de FE, ED

Temporel

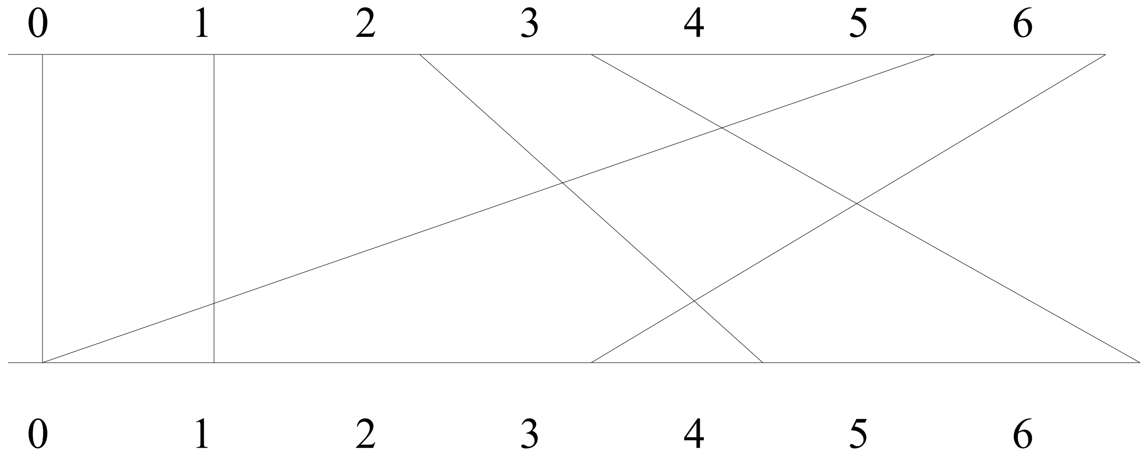
t	1	2	3	4	5	6	7	8	9	10	11	12	13	
F...	X													
E...	X	X	X	X	X									
A...	X	X	X				X						X	
C			X	X	X	X	X	X	X					
D					X									
H									X	X	X			
B							X	X	X					
G											X	X	X	

2 : F doit partir avant l'arrivée de C
4 : A doit partir avant l'arrivée de D
6 : E doit partir avant l'arrivée de B

7 : A revient → B mais pas H
9 : B doit partir avant arrivée de G
11 : H doit partir avant que A
Revienne pour voir G

Intérêt

- Algorithme linéaire de reconnaissance
- Modélise les intervalles de temps
- Trapézoïdaux pour les deux dimensions
  - N ={ ([0,1], [0, 1]), ([2,3], [4,6]), ([5,6], [0,3]) }



Quelques problèmes

- Affectation
  - Consiste à associer chaque objet d'un premier ensemble à un objet d'un deuxième ensemble (place). Simplification : même cardinalité
  - Chaque affectation à un coût
  - Objectif : réalisation l'affectation (chaque objet a une place et chaque place a un objet) en minimisant la somme des coûts
  - Résolution « facile »

Quelques problèmes (2)

- Affectation généralisée :
  - Affecter un ensemble d'objets à un ensemble de places mais une même place peut accueillir plusieurs objets
  - Chaque place ne peut pas accueillir tous les objets
  - Problème difficile à résoudre
  - Exemples :
    - Sac à dos : un poids et une valeur sont associés à un objet. Objectif : remplir un sac de capacité limitée en maximisant la valeur du contenu
    - Bin-packing : chaque objet est caractérisé par un volume. Objectif ranger tous les objets dans des boites identiques ou non de capacité finie en minimisant le nombre de boites.

Quelques problèmes (3)

- Transport :
  - Modélisation par un graphe, l'objectif est de trouver une structure sur ce graphe
  - Exemples :
    - Nœud : Voyageur de commerce : rechercher un parcours de longueur minimale passant par des villes qu'un voyageur doit visiter (circuit hamiltonien de longueur minimale). Difficile à résoudre
    - Arc : Postier chinois : réaliser une tournée passant par toutes les arêtes au moins une fois en minimisant la longueur totale de la tournée (exemple : ramassage d'ordures). Facile à résoudre s'il n'y pas de poids sur les arcs

Quelques problèmes (4)

- Localisation :
  - Placer un ensemble de services sur un territoire donné avec pour objectif de minimiser une fonction de coût
  - Exemple :
    - P-centre : un nombre fixe de services est à déposer sur les arêtes ou sur les nœuds d'un graphe. Quand le service est placé sur une arête, la distance entre ce service et chaque sommet adjacent est calculée en fonction de l'endroit où est placé le service sur l'arête.

Quelques problèmes (5)

- Ordonnancement
  - L'utilisation des ressources doivent être planifiées dans le temps
  - Difficile à résoudre
  - Exemples:
    - Mono-Machine : un nombre connu de tâches doit être ordonnancées (une séquence à fixer). La machine ne peut exécuter qu'une tâche à la fois. Objectif : minimiser une fonction de coût
    - Multi-machines : (1) affecter les tâches à une machine, (2) déterminer la séquence
    - Planification d'équipes : (1) constitution d'équipes (2) affectation dans le temps de ces équipes dans différentes tâches
    - Emploi du temps : sous contraintes

Quelques faiblesses

- Absence de règles de réalisation des graphes alors qu'il existe une sémiologie graphique pour la cartographie des réseaux
- Compte tenu de la complexité de certains opérateurs, il faut réduire la taille des graphes (reproductible et vérifiable)
- Noeud neutre alors que dans la réalité ce n'est pas vrai (importance des carrefours dans l'évolution du trafic routier)
- Hypothèse de parfaite rationalité et de connaissance totale du réseau