Algorithme de détermination d'un ensemble indépendant maximal dans un graphe

Résumé

Un des problèmes classiques de la théorie des graphes est la détermination d'un stable maximum . En effet , ce mémoire s'intéresse à la mise en œuvre d'un algorithme de détermination d'un ensemble indépendant maximal . Nous ne saurions traiter ce sujet sans parler du stable maximum car étant le plus grand indépendant maximal . La détermination de ce dernier est classée parmi les problèmes NP-complets . Cependant , si on se restreint à certaines classes de graphe , il existe des procédés particulièrement efficaces pour trouver cet ensemble . Ainsi , dans ce mémoire , on a étudié le cas des arbres , qui sont des graphes connexes sans cycles , les graphes bipartis et les k-cliques . De façon générale , le sujet répond à la question : comment dans un ensemble d'objets dont certains sont liés par une certaine « relation » , avoir un sous-ensemble pour lequel ses éléments ne sont pas en relation mutuelle et qui soit maximal pour cette propriété ? Dans le premier chapitre, on a fait le point sur les principales définitions et « objets » que l'on retrouve en Théorie des Graphes et ceci pour se fixer les idées . Dans le second chapitre, on a introduit la notion de coloration et quelques types de sous-ensembles d'un graphe . Au niveau du chapitre III , on trouve les coefficients caractérisant les sous-ensembles précédemment mentionnés . L'algorithme de détermination d'un ensemble indépendant maximal ne sera développé qu'au chapitre IV. Nous avons présenté un programme en langage C permettant , d'obtenir entre autres résultats , un ensemble indépendant maximal , puis de pouvoir vérifier si un sous-ensemble en notre possession est oui ou non indépendant maximal . Enfin dans le dernier chapitre , on a présenté quelques applications pratiques . En conclusion , nous avons parlé des perspectives que ce travail offre . J'ajoute que le programme mis en annexe , pourra faire l'objet de modification afin d'obtenir d'autres résultats intéressants .

Remerciements

Je remercie Dieu de m'avoir permis d'accomplir ce travail .
Je remercie Youssou Gningue qui a été à l'origine de notre intérêt pour les graphes .
Je salue sa disponibilité et son esprit d'ouverture . Je voudrais lui témoigner toute ma reconnaissance .
Je remercie Mamadou Sangharé pour avoir accepté de présider la séance de soutenance , ainsi que tous les membres du jury pour leur disponibilité .
Je remercie également Sada Sory Thiam et Sérigne Aliou Lô pour les critiques et remarques constructives qu'ils nous ont faites lors de l'élaboration du mémoire .
Je tiens à remercie mes parents , ainsi que mes frères et sœurs pour leur soutien sans faille .
J'exprime ma profonde gratitude à l'égard de tous mes collègues de promotion pour leur aide et leurs encouragements .
Enfin , mes remerciements vont à tous les enseignants ainsi qu'à tout le personnel administratif du département de Mathématiques et Informatique .

Sommaire

Introduction ……. …….……..……4

Chapitre I Eléments de Théorie des Graphes ……6
I-1 Définitions et concepts de base ……..…….……..…6
A) Graphe orienté……..……..…….……..…6
B) Graphe non orienté……...……..…….……..8
C) Isomorphisme de graphe……..…..…….……..8
I-2 Représentation d'un graphe …..…….…..……10
I-3 Caractérisation d'un graphe ……..…….…….……...11
A) Arbre et arborescence……15

Chapitre II Sous-ensembles d'un graphe et coloration..……17
II-1 Sous-ensemble indépendant……...17
II-2 Sous-ensemble dépendant ……..……...17
II-3 Sous-ensemble recouvrant ……..……...……18
II-4 Coloration ….……...…….…….……..19
A) Nombre chromatique d'un graphe……..…19

Chapitre III Ensemble indépendant maximal ……23
III-1 Définitions……..……...23
III-2 Relation entre les coefficients caractéristiques……..……24

Chapitre IV Algorithme de détermination d'un ensemble indépendant maximal ..…...……...……27
IV-1 Construction d'un stable maximal dans le cas général ……..27
IV-2 Présentation de l'algorithme……..28
IV-3 Ensemble indépendant maximum d'un graphe …….……...….36
IV-4 Cas particuliers …….……..….38
A) Stable maximum dans un arbre……..38
IV-5 Recherche d'une clique maximum……42

Chapitre V Applications ……..…….……44
V-1 Application à la localisation ……...…..……..…….44
V-2 Application à la recherche opérationnelle ……..…….49
V-3 Ensemble indépendant de coût maximum ……..…….51

Conclusion ……..……54

Annexe …….……55

Bibliographie ……...……...…….60

Introduction

La théorie des graphes est née en 1736 quand Euler démontra qu'il était impossible de traverser chacun des sept ponts de la ville de Königsberg une fois exactement et de revenir au point de départ .
Le terme lui même vient du mot grec $\gamma\rho\alpha\in\iota\nu$ qui veut dire écrire , dessiner . Il a été employé semble t-il la première fois , en langue Française par Sainte – Lague en 1926 .
Pendant longtemps , les travaux sur les graphes ne déborderont guère le cadre des récréations mathématiques .
Des recherches d'Euler au XVIIIème siècle , elle est devenue une branche des Mathématiques au début du XXème siècle , grâce aux travaux de König , de Kuratowski de Cayley et , plus récemment de Berge , d'Erdös et de Harary . Les recherches récentes en informatique et surtout en algorithmique lui donne un nouveau souffle .
La théorie des graphes constitue un domaine des mathématiques qui , historiquement , s'est développée au sein de disciplines diverses telle que la chimie (modélisation de structures) , la biologie (génome) , les sciences sociales (modélisation des relations) ou en vue d'applications industrielles (problème du voyageur de commerce) .
Elle est devenue l'un des instruments les plus efficaces pour représenter (modéliser) ; puis résoudre de nombreux problèmes discrets que pose la recherche opérationnelle.
La théorie des graphes sert avant tout à représenter et à organiser les taches de façon optimale : après avoir traduit un problème sous forme de graphe , on cherche des méthodes systématiques qui permettent de trouver la succession la plus rapide ou la moins coûteuse pour effectuer toutes les tâches.
Un graphe permet de représenter simplement la structure , les connexions , les cheminements possibles d'un ensemble complexe comprenant un grand nombre de situations , en exprimant les relations , les dépendances entre ses éléments (réseau de communication , réseaux ferroviaire ou routier , arbre généalogique , diagramme de succession de tâches en gestion de projet etc…) .
Elle permet de résoudre efficacement une grande variété de problèmes pratiques ou récréatifs en les ramenant à des configurations qui se dessinent simplement à l'aide de points et de liaisons entre ces points.
Les problèmes classiques de la théorie des graphes sont : flots et connectivité , (« fiabilité d'un réseau ») ,couplage (affectation) , parcours eulérien (qui traverse chaque arête : problème du « postier chinois ») , parcours hamiltonien (qui traverse chaque sommet :
« problème du voyageur de commerce ») , coloration , ensemble stable , absorbant . Certains de ces problèmes (flots maximum , couplage maximum parcours eulérien) peuvent être résolus « efficacement » , mais la plupart sont NP-complets .
En effet , de l'un de ces problèmes classiques , à savoir « ensemble stable » ressort le sujet de notre mémoire qui s'intitule comme suit : Algorithme de détermination d'un ensemble indépendant maximal .
Ce thème est d'autant plus actuel à cause de ses nombreuses applications .
Si l'on s'intéresse par exemple au domaine des Télécommunications, nous avons des solutions efficaces dans les problèmes de partage de ressources, l'affectation de ressources, le multiplexage des longueurs d'ondes (WDM) utilisé dans la technologie de la fibre optique, l'allocation des fréquences qui est un problème récurrent dans ce domaine etc.…
La solution à la plus part de ces problèmes passent par le biais de la détermination d'un ensemble indépendant maximal dans un graphe bien déterminé .
Il existe d'autres applications dans plusieurs domaines, comme l'ordonnancement des travaux , la gestion des emplois de temps etc.….

De façon générale , le sujet répond à la question : comment dans un ensemble d'objets dont certains sont liés par une certaine « relation » , avoir un sous-ensemble pour lequel ses éléments ne sont pas en relation mutuelle et qui soit maximal pour cette propriété ?

Ce thème est connexe à la célèbre notion de coloration des graphes , mais aussi à la localisation et à la recherche opérationnelle notamment en nombres entiers et donc pour lequel la méthode du simplexe n'est pas toujours applicable .

Ainsi, pour aborder le sujet , on a divisé le travail en cinq chapitres qui s'échelonnent comme suit :

Dans le premier chapitre, on a fait le point sur les principales définitions et « objets » que l'on retrouve en Théorie des Graphes et ceci pour se fixer les idées .

Dans le second chapitre, on a introduit la notion de coloration et quelques types de sous-ensembles d'un graphe . Si nous avons abordé ces notions, c'est parce qu'elles sont étroitement liées à notre sujet et contribuent d'une certaine manière à l'éclaircir .

Au niveau du chapitre III , dans la logique de la présentation , on trouve les coefficients caractérisant les sous-ensembles précédemment mentionnés .

L'algorithme de détermination d'un ensemble indépendant maximal ne sera développé qu'au chapitre IV . Nous avons présenté un programme en langage C permettant , une fois qu'on dispose d'un graphe , après saisie sur une console de sa matrice d'adjacence , d'obtenir entre autres résultats , un ensemble indépendant maximal , puis de pouvoir vérifier si un sous-ensemble en notre possession est oui ou non indépendant maximal .

Supposant que tous les lecteurs ne sont pas forcément outillés en programmation , nous avons préféré placer le code source du programme en annexe . Cependant , le langage C étant portable il est aisé de faire soi-même les tests et de faire tourner l'algorithme .

Enfin dans le dernier chapitre , on a présenté quelques applications pratiques .

En conclusion , nous avons parlé des perspectives que ce travail offre .

Chapitre I Eléments de Théorie des Graphes

On trouve dans ce chapitre la définition d'un graphe orienté ou non . Les différentes forment de représentation d'un graphe y ont pris une place de choix . En effet , après une définition formelle, il fallait trouver les meilleurs moyens pour représenter les graphes en ordinateur et ainsi automatiser certaines types de taches . On trouvera aussi quelques classes de graphes particulièrement intéressants .

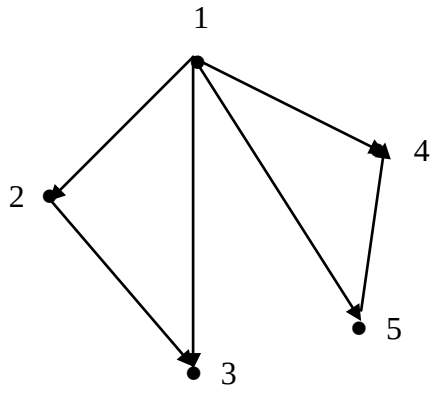
I-1 Définitions et concepts de base

A) Graphe orienté

Un graphe G = (S ; A) est déterminé par la donnée :
d'un ensemble S dont les éléments sont appelés des sommets ou des nœuds ,
d'un ensemble A dont les éléments a $\in$ A sont des couples ordonnés de sommets appelés des arcs .
Si n = | S | est le nombre de sommets (de nœuds) on dit que G est d'ordre n .
On supposera que les sommets sont numérotés de 1 à n (i = 1 , ….. , n) .
Si a = (i ,j) est un arc de G , i est l'extrémité initiale de a et j est l'extrémité terminale de a
On notera souvent | A | = m .
On représente graphiquement un graphe en plaçant dans un plan des points qui représentent les éléments de S et des flèches reliant tout couple de sommets élément de A dont le sens sera de l'extrémité initiale vers l'extrémité terminale .

Exemple 1

G = (S , A) ; S = { 1 , 2 , 3 , 4 ,5} et A = { (1,2) ; (2,3) ; (1,4) ; (1,3) ; (1,5) ; (5,4) }



Formellement , un graphe orienté est constitué :
D'un ensemble , non vide S dont les éléments sont les sommets du graphe
D'un ensemble A fini qui peut être vide dont les éléments sont appelés arcs du graphe ; d'une fonction

g : A $\longrightarrow$ S^2 qui associe à tout arc un couple de sommets

Notation G = (S , A , g)

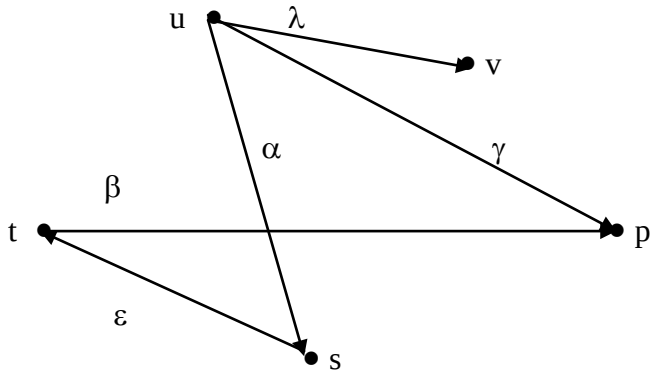
Remarque 1 L'application g est injective s'il n'existe pas deux arcs distincts ayant à la fois le même début et la même fin (on parle alors d'arcs parallèles).

Exemple 2

A = { ε ; γ ; λ ; α ; β }

S = { s ; t ; p ; u ; v }

g(ε) = (s , t) g(γ) = (u , p) g(λ) = (u , v) g(α) = (u , s) g(β) = (t , p)



Le sommets s est appelé le début de l'arc ε tandis que t est la fin .

s et t sont des sommets extrémités de ε

s et t sont adjacents s'il existe un arc ε tel que g(ε) = (s , t) .

Deux arcs qui ont même début et même fin sont des arcs parallèles . Dans ce cas , on dit que G est un multigraphe .

Un arc tel que le début et la fin coïncide est appelé boucle . On dit que G est un pseudographe.

Un graphe qui n'a ni arc parallèle , ni boucle est un graphe simple .

Remarque 2

En fait les graphes orientés dépourvus d'arcs parallèles correspondent aux relations binaires sur un ensemble fini et les relations binaires sur un ensemble fini correspondent aux graphes orientés dépourvus d'arcs parallèles .

La possibilité d'avoir des arcs parallèles est donc une généralisation de la notion de relation binaire , qui autorise des objets à être liés de plusieurs façons en même temps .

Théorème

G = (S , A , g) est un multigraphe si et seulement si g n'est pas injective .

Preuve

Si G est un multigraphe , alors il admet au moins deux arcs parallèles .

Par définition d'arc parallèle ; $\exists\ \mu \in A$; $\nu \in A$; $\mu \neq \nu$ et $\exists$ i ; j des éléments de S tel que : g(μ) = (i , j) et g(ν) = (i , j) c'est à dire g(μ) = g(ν) . Ce qui entraîne que g n'est pas injective.

Supposons que g n'est pas injective :

Alors il existe au moins deux arcs u et v distincts tel que g(u) = g(v) .

Ceci entraîne que les arcs u et v sont parallèles et ainsi G est un multigraphe .

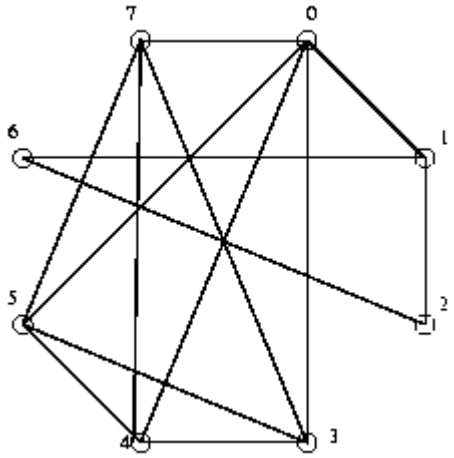
Exemple 3

Considérer le graphe orienté dont les sommets sont les entiers compris entre 1 et 12 et dont les arcs représentent la relation << être diviseur de>>

B) Graphe non orienté

Dans l'étude de certaines propriétés des graphes , il arrive parfois que l'orientation des arcs c'est à dire la distinction entre extrémité initiale et extrémité terminale , ne joue aucun rôle .
Ce qui importe ; c'est seulement l'existence ou la non existence d'un ou de plusieurs arcs entre deux sommets . Dans ce cas , l'ordre n'est plus important .
Un graphe non orienté est ce que l'on obtient en effaçant le sens de parcours sur les arcs d'un graphe orienté .
A tout arc (i , j) ; à tout couple ordonné (i , j) ; on associe le couple non ordonné (i , j) qui est appelé arête .
Formellement un graphe non orienté est constitué :
D'un ensemble fini non vide S dont les éléments sont appelés les sommets du graphe .
D'un ensemble fini mais qui peut être vide A dont les éléments sont les arêtes du graphe
D'une fonction $g : A \longrightarrow (S_2 \cup S)$ avec S_2 l'ensemble des parties à deux éléments de S .

Exemple 4



S ={ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 }
A={ {0,1} ; {0,3} ; {0,4} ; {0,5} ; {0,7} ; {1,2} ; {1,6} ; {2,6} ; {3 ,7} ; {3,5} ; {3,4} ; {4,7} ; {4,5} ; {5,7} }

Remarque 3

Il ne faut pas confondre un graphe et son dessin : un même graphe peut être dessiné de plusieurs façons . La lisibilité de la visualisation est une question importante , et le dessin de graphes est à lui seul un domaine de recherche dont les applications sont nombreuses : fabrication de cartes routières , représentation d'interactions entre sous-systèmes , aide graphique à l'ordonnancement des taches pour ne prendre que quelques exemples .

C) Isomorphisme de graphe

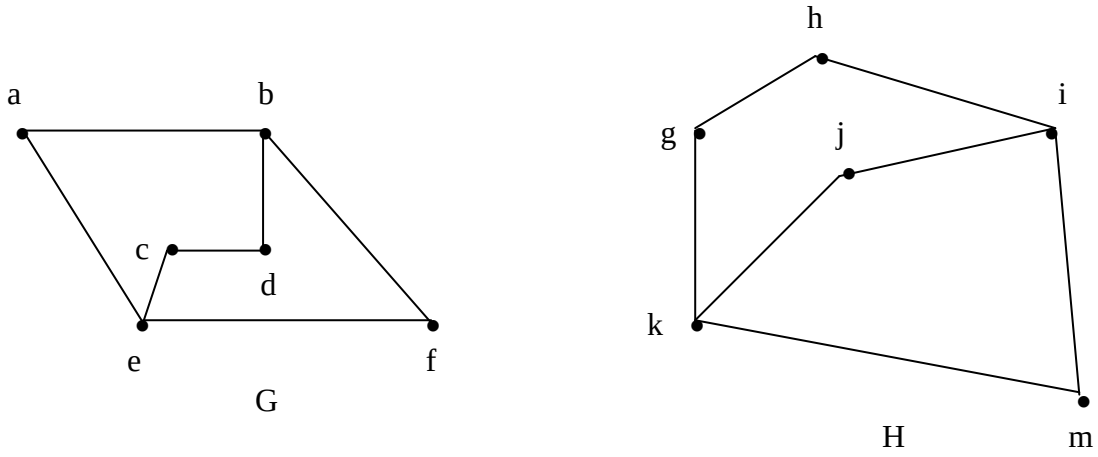
Définition

Soit $G = (S , A , g)$ et $H = (T , B , h)$
On dit qu'ils sont isomorphes quant il existe une bijection
$\phi : S \longrightarrow T$ et une bijection
$\Psi : A \longrightarrow B$ ayant la propriété suivante :

Si on prend une arête quelconque ξ de A et si $g(\xi) = (c , d)$; alors $\Psi(\xi)$ a pour début $\phi(c)$ et pour fin $\phi(d)$.

Exemple 5

Les graphes suivants sont isomorphes .



On peut construire une bijection w par :
w(a)=j ; w(b)=i ; w(f)=m ; w(e)=k ; w(d)=h

Remarque 4
Il sort de cette définition que deux graphes isomorphes ont nécessairement même nombre de sommets et même nombre d'arcs .
Dans certains cas , le nom particulier donné à un sommet , ou à une arête d'un graphe n'a vraiment pas d'importance et on peut considérer que deux graphes qui ne diffèrent que par les noms de leurs sommets et de leurs arêtes sont en fait identiques . C'est la notion d'isomorphisme qui permet d'exprimer formellement cette idée .

Terminologie
Le degré d'un sommet de G est le nombre d'arcs (d'arêtes) qui lui sont incident(e)s . Il est noté d(i) pour tout sommet i du graphe .
Le demi- degré extérieur du sommet i , noté d^+ (i) est le nombre d'arcs ayant i comme extrémité initiale .
Le demi-degré intérieur , noté d^- (i) est le nombre d'arcs ayant i comme extrémité terminale .
Par définition du degré du sommet i , on a $d(i) = d^+(i) + d^-(i)$.
Un graphe est dit régulier de degré r si tous ses sommets sont de degré r .
Le graphe nul est régulier . Sa suite de degré est (0 ,……., 0) .
Le graphe G à n sommets , constitué d'arcs tous non adjacents est régulier ;
c'est le graphe constitué d'arcs dispersés . Sa suite de degré est de la forme (1 ,…., 1) .
Dans un graphe , on appelle chemin , une suite d'arcs dont l'extrémité terminale de chacun , sauf pour le dernier est l'extrémité initial du suivant .
Un chemin qui se ferme sur lui même est un circuit .
Un chemin est simple s'il ne passe qu'une fois par chacun de ses arcs .
Un chemin est élémentaire s'il ne rencontre pas plus d'une fois chacun de ses sommets (un chemin élémentaire est nécessairement simple) .
La longueur d'un chemin est le nombre de ses arcs .
On appelle chaîne ; une suite d'arêtes dont chacune a une extrémité commune avec l'arête précédente (sauf la première) et l'autre avec l'arête suivante (sauf la dernière) .

Exemple 6

On montre que dans un groupe de personnes , il y a toujours deux personnes qui ont le même nombre d'amis :

Construisons un graphe dont les sommets représentent les personnes et plaçons une arête entre deux sommets lorsque les personnes correspondantes sont amies .

Dire que deux personnes ont le même nombre d'amis , revient à dire que deux sommets dans le graphe ont le même degré .

Nous allons montrer qu'il n'existe aucun graphe dont tous les sommets ont des degrés distincts . Supposons qu'un tel graphe existe et qu'il possède n sommets .

Le degré maximal d'un sommet est donc n-1 . Si tous les degrés des sommets sont distincts , on a donc nécessairement un sommet de degré 0 , un sommet de degré 1 , un de degré 2 , ainsi de suite j'usqu'au sommet de degré n-1 . La présence d'un sommet de degré 0 , disons x_0 , fait qu'il soit impossible d'avoir un sommet de degré n-1 !

(en effet celui-ci devrait être lié à tous les autres , y compris x_0 . On obtient une contradiction)

I-2 Représentation d'un graphe

Pour décrire un graphe G ; un certain nombre de représentations peuvent être utilisées.

Cette idée de représentation des graphes vient répondre à certaines interrogations qui étaient : ayant un graphe G = (S,A ,g) ; représenté sur le papier par des points et segments les joignant (qui peuvent être orientés ou non) ; comment pourrait –on représenter ceci en ordinateur de manière à ce que de ces représentations ; puissent sortir des traitements efficaces de ces graphes . En pratique , ces représentations ne sont généralement pas équivalentes du point de vu de l'efficacité des algorithmes .

Matrice d'adjacence

Une structure de données simple pour représenter un graphe est la matrice d'adjacence M .

Pour obtenir M , on numérote les sommets du graphe de façon quelconque :

$S = \{ x_1 , x_2 , ….., x_n \}$

M est une matrice carrée n*n dont les coefficients sont 0 ou 1 telle que :

$M_{i,j} = 1$ si $(x_i , x_j) \in A$ et $M_{i,j} = 0$ si $(x_i , x_j) \notin A$

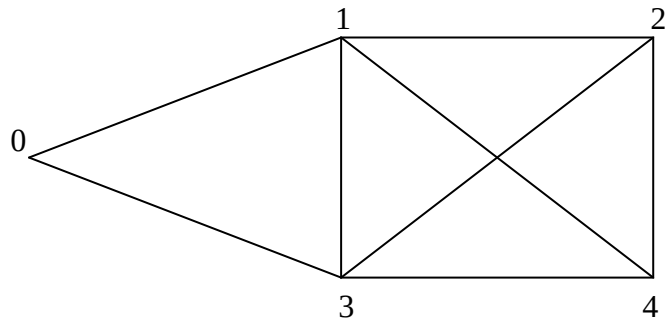
La matrice d'adjacence permet donc de décrire soit des 1-graphes (orientés) soit des graphes simples (non orientés) .

On voit en outre que la quantité d'informations nécessaires est n^2 .

La représentation d'un graphe par sa matrice d'adjacence est préférée quand le graphe est dense (beaucoup d'arcs) , car elle comporte toujours les $|S|^2$ éléments d'un tableau bidimensionnel .

Exemple 7

Considérons le graphe suivant :



Sa matrice d'adjacence est :

0 1 0 1 0
1 0 1 1 1
0 1 0 1 1
1 1 1 0 1
0 1 1 1 0

Matrice d'incidence sommets-arcs

Une autre manière de décrire un graphe est d'utiliser sa matrice d'incidence sommets-arcs .
Chaque ligne de cette matrice est associée à un sommet et chaque colonne à un arc du graphe .
Dans une colonne correspondant à un arc (i , j) , autre qu'une boucle , figure la valeur +1 et sur la ligne correspondant au sommet i ; la laveur –1 sur la ligne correspondant au sommet j et la valeur 0 sur toutes les autres lignes .
Dans une colonne correspondant à une boucle (i , i) , figure la valeur +1 sur la ligne correspondant au sommet i et la valeur 0 sur toutes les autres lignes .

Matrice d'incidence sommets-arêtes

Soit un graphe G = (S , A , g) où A est un ensemble d'arêtes (on ne s'intéresse pas à l'orientation du graphe) .
La matrice d'incidence sommets-arêtes de G est une matrice à coefficient 0 ou 1 où chaque ligne i correspondant à un sommet i de G et chaque colonne à une arête u = (i , j) de G ; de plus si u = (i , j) , la colonne u a tous ses coefficients nuls sauf ceux correspondant aux lignes i et j .

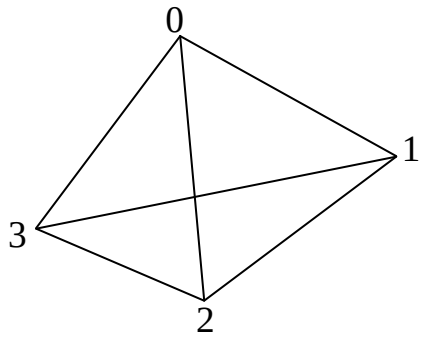
I-3 Caractérisation d'un graphe

Complétude

Un graphe G = (S, A , g) est dit complet si toute paire de sommets est reliée par au moins une arête . En d'autre termes l'application g surjective .
Si g est bijective , alors le graphe est complet sans arc parallèle .
Le graphe complet K_n est régulier de degré n-1

Exemple 8

Considérons le graphe K_4 . Il est complet d'ordre 4 . Chaque sommet est de degré 3 .



Connexité

Un graphe est fortement connexe lorsqu'il existe un chemin entre chaque paire de sommets .
Un graphe est connexe lorsqu'il existe une chaîne entre chaque paire de sommets .

Couplage

Un couplage est un ensemble d'arêtes deux à deux non adjacentes . On sait trouver un couplage de taille maximum .

Valuation

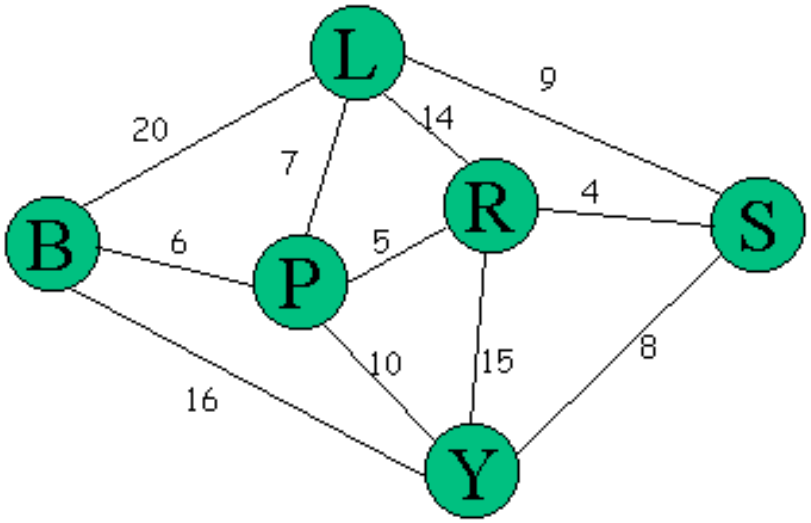
On dit qu'un graphe est valué ou pondéré si , à tout arc (arête) qui le constitue correspond une valeur numérique qu'on appelle poids ou coût , qu'on écrit simplement à proximité de cet arc (arête)
Ces valeurs peuvent être des quantités transportées , des débits des coûts des durées des distances etc.…
Formellement donc , un graphe G = (S , A , g) est valué s'il existe une application

$$C : A \longrightarrow IR+$$

$$(u , v) \longrightarrow c$$

Une généralisation de ce concept consiste à valuer aussi bien les arêtes que les sommets du graphe avec des nombres réels .

Exemple 9



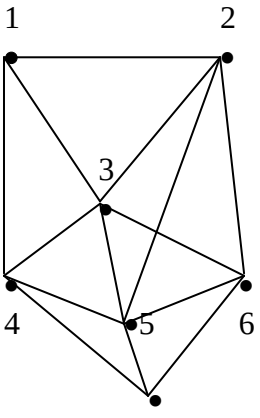
Existe-t-il un chemin partant de L, passant par toutes les villes, et faisant moins de 40kms???? Problème difficile...
Le trajet L-P-Y-S-R-P-B convient. Evident à vérifier!

Graphe ligne

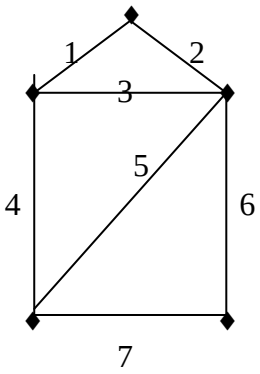
Soit le graphe non orienté G = (S , A , g) .
On appelle graphe ligne G_l = (S_l , A_l , g_l) de G ; le graphe défini par :
Chaque élément v ∈ S_l corresponde à une arête de G .
L'arête (u , v) ∈ A_l si les arêtes associées a_u et a_v sont adjacentes dans G .
Ainsi ; nous pouvons dire que tout graphe G admet un graphe ligne G_l transformant les arêtes de G en sommets de G_l .
Cependant ; la réciproque n'est pas vrai : la transformation des sommets d'un graphe en arêtes n'est pas toujours vrai .

Exemple 10

Considérons le graphe G et son graphe ligne G_l .



G_l



G

Graphe biparti

On dit qu'un graphe G = (S , A , g) est biparti si l'ensemble des sommets S peut être partitionné en deux-ensembles S_1 et S_2 de telle sorte que , pour toute arête (i , j) $\in$ A :

1) $i \in S_1 \Rightarrow j \in S_2$
2) $i \in S_2 \Rightarrow j \in S_1$

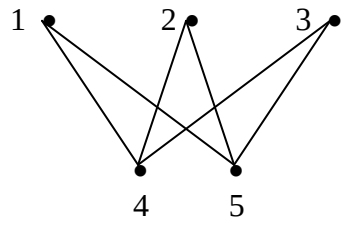
Graphe biparti complet

Un graphe est biparti complet s'il est biparti avec l'existence d'une partition de S telle que $S = S_1 \cup S_2$ et tout couple de sommets de $S_1 * S_2$ forme un arc .

Si les nombres de sommets de S_1 et de S_2 sont respectivement r et s , alors le graphe biparti complet est noté $K_{r , s}$.

Exemple 11

Considérons $K_{3 , 2}$

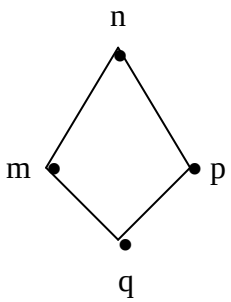


Cycle

Un cycle d'ordre n est un graphe de n sommets dont chacun est de degré 2 .

Exemple 12

Le graphe suivant est le C_4



Remarque 5

Il existe d'autres types de graphes bipartis réguliers et non complets . Il suffit de se rappeler que tout cycle C_n avec n pair admet une représentation en graphe biparti qui est donc régulier

Graphe partiel et sous graphe

Intuitivement , on obtient un graphe partiel d'un graphe G = (S , A , g) en retirant des arêtes (ou des arcs dans le cas orienté) et on obtient un sous-graphe de G = (S , A , g) en retirant des sommets ainsi que toutes les arêtes dont ces sommets sont les extrémités .

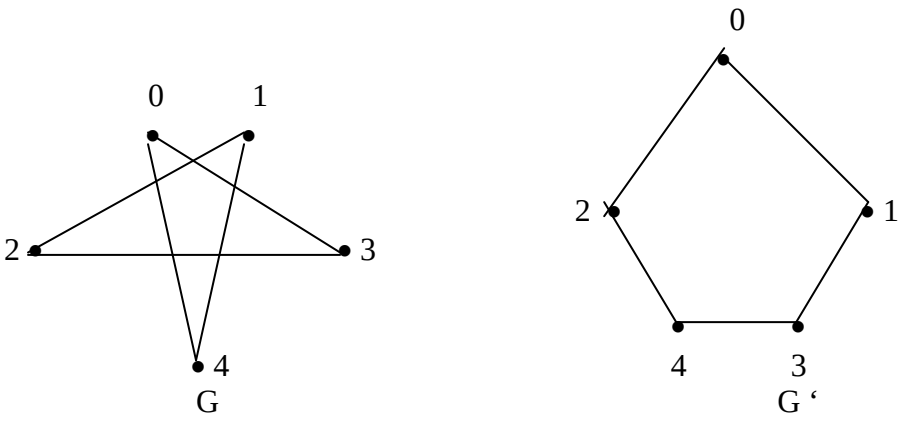
Graphe complémentaire

Etant donné un graphe simple G = (S , A , g) , le graphe complémentaire G ' = (S', A' , g') de G = (S , A , g) a le même ensemble de sommets que G et comme arcs , les arcs complémentaires à A . D'où on a :

(i , j) $\in$ A $\Rightarrow$ (i , j) $\notin$ A'

(i , j) $\notin$ A $\Rightarrow$ (i , j) $\in$ A'

Exemple 13



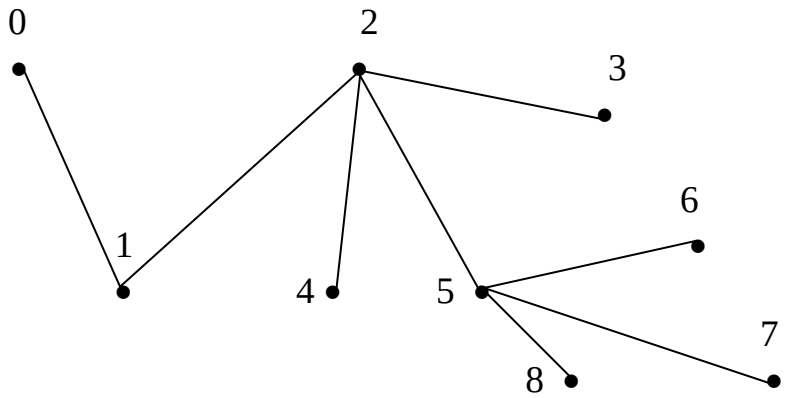
A) Arbre et arborescence

Un arbre est un graphe connexe sans cycle .

La notion d'arbre ne fait pas appel à l'orientation du graphe .

Un arbre de n sommets possèdent (n-1) arcs .

Exemple 14



Un arbre peut être vu comme un graphe biparti avec l'une des bipartitions composée des sommets de poids pairs et l'autre des sommets de poids impairs .

Une arborescence est un arbre comportant un sommet particulier r , nommé racine de l'arborescence : Depuis r , il existe dans l'arborescence un chemin (et un seul) vers tout autre sommet (cette notion fait appel à l'orientation du graphe) .

Une généralisation récente du concept de graphe , introduite par Claude Berge dans les années 60 , est celle d'hypergraphe , où les arêtes peuvent être de taille arbitraire et non plus seulement de taille deux . Cette notion est définie comme suit :
On se donne un ensemble $S = \{ s_1 , s_2 ,\ldots\ldots, s_n \}$ et une famille $E = \{ e_1 , e_2 ,\ldots, e_n \}$ de parties de S . On dira que E constitue un hypergraphe sur S si l'on a :

$e_j \neq \varnothing$ j =1 ,….., n

$\cup_j e_j = S$

Le couple H = (S , E) s'appelle un hypergraphe .
Les éléments $s_1 , s_2 ,\ldots\ldots, s_n$ de S sont les sommets de l'hypergraphe et les éléments $e_1 , e_2 ,\ldots\ldots , e_n$ de sont les arêtes de l'hypergraphe .
Dans un hypergraphe , deux sommets s et t sont adjacents s'il existe une arête e_j qui les contient tous les deux , de même deux arêtes sont dites adjacentes si leur intersection est non vide .
Cette notion est particulièrement importante car un grand nombre de problèmes concrets peuvent se formaliser de cette façon .

<u>Remarque</u> 6

Vous avez remarqué que ce chapitre est enrichi par beaucoup d'exemples et de remarques. Cela est motivé par le fait qu'un bon dessin vaut mieux qu'un long discours . Le langage des graphes essaient de mettre en pratique cette idée . Bien souvent , en effet , c'est un réflexe naturel qui pousse à abstraire une situation donnée en traçant , sur une feuille de papier , des points pouvant représenter des individus , des localités , des corps chimiques etc …, reliés entre eux par des lignes ou des flèches symbolisant une certaine relation .
Cette représentation figurative de la réalité présente un double intérêt :
-elle permet de mettre en évidence la structure profonde de la situation donnée,
-d'un point de vu pratique, elle fournit une vision globale du problème , ce qui constitue un guide précieux pour l'intuition et le raisonnement .

Chapitre II Sous-ensembles d'un graphe et coloration

Dans la première partie ce chapitre, nous allons faire le point sur quelques sous-ensembles, dont les cliques , qui correspondent aux sous-ensembles indépendants quand on passe au complémentaire du graphe. L'autre grande partie aborde la coloration qui oblige deux sommets ou arcs adjacents à être identifiés différemment . Cette identification qui se fait souvent avec un ensemble de couleurs usuelles (d'où le nom de coloration donné à cette notion), pourra aussi se faire à l'aide d'entiers .
Dans la suite du mémoire , sauf indication contraire , nous noterons G = (S ; A)
les graphes , qui sont supposés finis avec S et A de cardinaux respectifs n et m. Pour d'avantage de simplification , à la place de G = (S ; A) , on écrira simplement G .

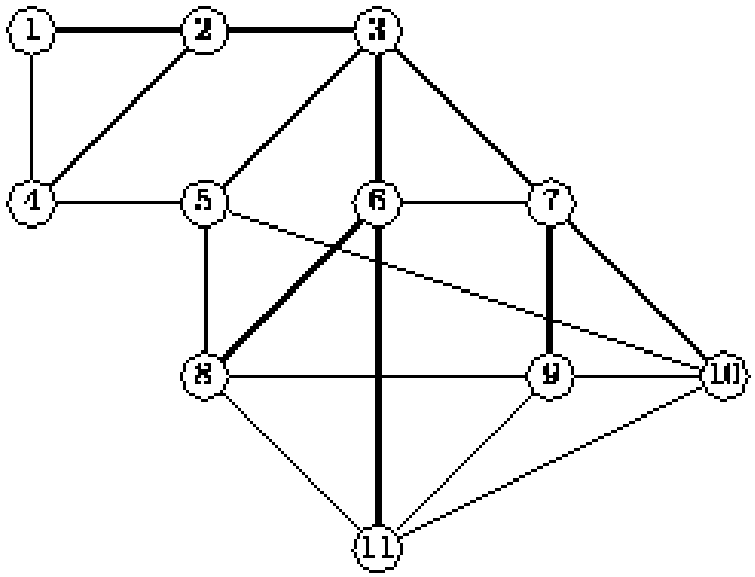
II-1 Sous-ensemble indépendant

Définition

Considérons un graphe G . Un sous-ensemble I de S est dit indépendant ou stable si et seulement si : $\forall\ i \in I\ ;\ \forall\ j \in I$, alors $(i, j) \notin A$.
Donc un sous-ensemble de S est indépendant si ses éléments ne sont pas mutuellement adjacents . On peut aussi le définir comme suit :
Un ensemble indépendant d'un graphe G est un sous-ensemble $S' \subseteq S$ de sommets tel que chaque arête de A est incidente à au plus un sommet de S' .

Exemple 1

Considérons le graphe ci-dessous :



Les sous-ensembles { 1 , 5 , 6 , 9 } et { 1 , 3 , 8 , 10 } sont indépendants .
Le sous-ensemble { 1 , 5 , 9 , 10 } n'est pas indépendant car { 5 , 10 } est une arête de G

II-2 Sous-ensemble dépendant

Définition

Soit le graphe G et C un sous- ensemble de S . On dit que C est un sous-ensemble dépendant si C est une clique de G c'est à dire un sous-ensemble complet de G .

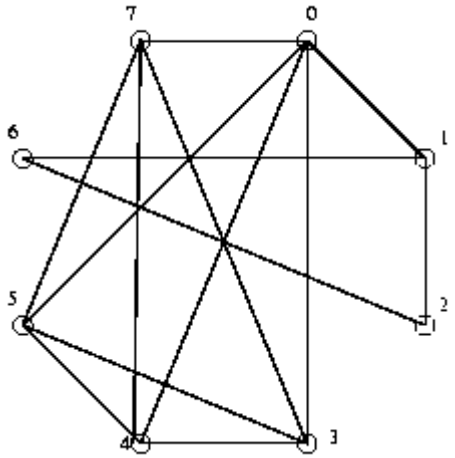
Soit C l'ensemble des cliques de G . On appelle degré de dépendance de G , le nombre

$\theta (G) = \max_{C \in C} \{ \text{card} (C) \}$

C'est le nombre maximal de sommets constituant une clique de G .

Exemple 2

Les sous-ensembles { 0 , 3 , 4 , 5 , 7 } et { 1 , 2 , 6 } sont des cliques du graphe ci-dessous .
Cependant l'ensemble { 1 , 2 , 6 , 7 } n'est pas une clique car { 6 , 7 } n'est pas une arête .



Remarque 1
Un sous-ensemble sommets I est un stable si et seulement si I est une clique du graphe complémentaire G' de G .

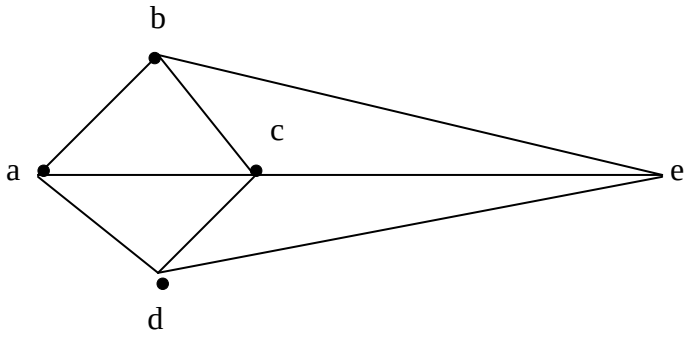
II-3 Sous-ensemble recouvrant

Soit le graphe G et $R \subset S$ un sous-ensemble de S .
On dit que R est recouvrant si : $\forall$ (u , v) $\in$ A alors u $\in$ R ou v $\in$ R .
On appelle degré de recouvrement de G , le nombre $\delta (G) = \min_{R \in R} (|R|)$;

où R est l'ensemble des sous-ensembles recouvrant de G .
Donc un sous-ensemble de G est recouvrant si et seulement si tout arc de G lui est incident .

Exemple 3

L'ensemble { a , c , e } du graphe ci-dessous est recouvrant



Remarque 2
Ces sous-ensembles interviennent très souvent directement ou indirectement dans beaucoup d'applications faisant intervenir la coloration de graphe. Cette dernière est l'objet de la prochaine section .

II-4 Coloration

Définition

On appellera coloration du graphe G , toute coloration des sommets telle que deux sommets adjacents ne soient pas de même couleur .
Formellement , une coloration des sommets d'un graphe non orienté G sans boucle est une application

$$\theta : S \longrightarrow C$$ avec C un ensemble de couleurs .

Une coloration , est donc une partition de G en stables .
On essaie le plus souvent de déterminer le nombre minimal de couleurs nécessaires .

Application 1
L'allocation des fréquences est un problème récurrent en télécommunication . Il consiste à allouer des fréquences à des transmetteurs de manière à éviter les interférences et en utilisant le moins de fréquences possibles .
Une première modélisation de ce problème en terme de coloration d'un graphe est la suivante : Les sommets du graphe représentent les transmetteurs , deux sommets sont reliés par une arête si les transmetteurs correspondant peuvent interférer et les couleurs représentent les fréquences . Affecter les longueurs d'onde revient à trouver une bonne coloration du graphe .

A) Nombre chromatique d'un graphe

Définition

On appelle nombre chromatique d'un graphe G , le nombre minimal de couleurs servant à le colorier . On le notera $\gamma (G)$.

Partition des sommets d'un graphe

Soit le graphe G . Considérons un sommet X_1 et construisons un ensemble S_1 tel que :

- $X_1 \in S_1$
- S_1 est stable
- S_1 est de cardinal maximum

Considérons maintenant un sommet $X_2 \notin S_1$ et construisons un ensemble S_2 tel que :

- $X_2 \in S_2$
- X_2 est stable
- S_1 et S_2 sont disjoints
- S_2 est de cardinal maximum

Considérons maintenant un sommet $X_3 \notin S_1 \cup S_2$ et construisons un ensemble S_3 tel que :

- $X_3 \in S_3$
- X_3 est stable
- S_1 S_2 et S_3 sont disjoints
- S_3 est de cardinal maximum

En continuant ainsi j'usqu'à épuisement de l'ensemble S , on obtient une partition de S : $\{ S_1 , S_2 , ..., S_q \}$. Si on choisit une couleur par sous-ensemble S_i , on obtient ainsi une q-coloration du graphe et on aura nécessairement $\gamma (G) \leq q$.

<u>Application</u> 2

Soit une carte géographique représentant plusieurs pays . On souhaite colorier chaque pays de manière à ce qu'il n'ait pas la même couleur que chacun de ses voisins . Ce problème peut être résolu en cherchant une partition en stables du graphe obtenu en associant chaque pays à un sommet et chaque arc à une frontière commune entre deux pays . Il suffit alors d'associer une couleur à chaque stable obtenu .

<u>Une majoration de $\gamma (G)$</u>

Considérons maintenant un sommet $x \in S_q$. Ce sommet est lié à au moins un sommet appartenant à S_1 (sinon , on pourrait le mettre dans S_1 , ce qui contredirait la maximalité de S_1)

De même , il est lié à au moins un sommet de S_2 (sinon , il serait dans S_2) . On en déduit donc en raisonnant de la même manière sur S_3 , ..., S_{q-1} que $\deg (x) \geq q-1$.

Si on appelle r le degré maximum des sommets du graphe , on a alors :

$r \geq \deg (x) \geq q-1 \geq \gamma (G) - 1$ et donc

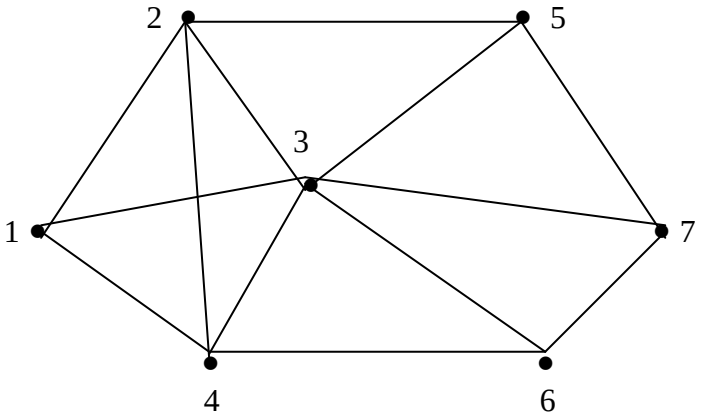
$\gamma (G) \leq r + 1$

<u>Une minoration de $\gamma (G)$</u>

En notant $\theta (G)$ le cardinal de la plus grande clique de G , on a :

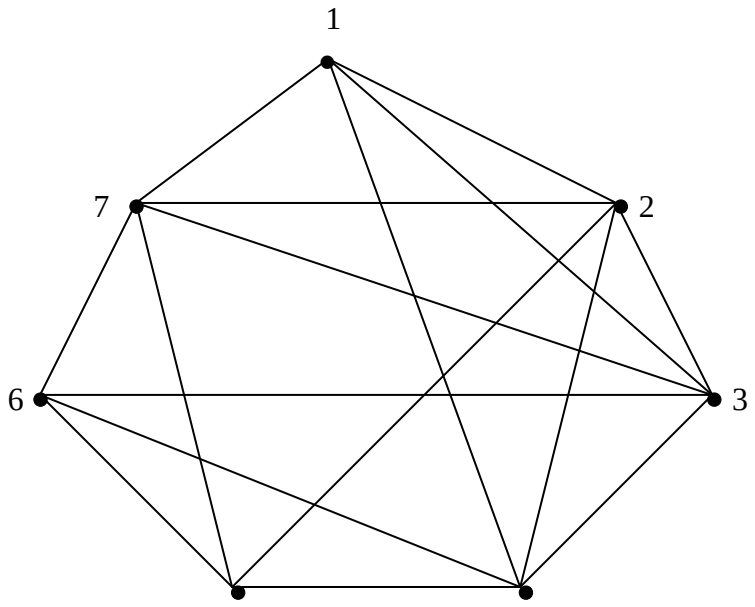
$$\gamma (G) \geq \theta (G)$$

Exemple 4



Le graphe contient un sous-graphe complet d'ordre 4 (1-2-3-4) , donc $\gamma(G) \geq 4$.
Le degré maximum des sommets est 6 , donc $\gamma(G) \leq 6+1 = 7$
Ainsi , on a : $4 \leq \gamma(G) \leq 7$.

Exemple 5 Problème d'emploi du temps
Une université doit organiser les horaires des examens . On suppose qu'il y a 7 épreuves à planifier correspondant aux cours numérotés de 1 à 7 et que les pairs de cours suivants ont des étudiants en commun :
1 et 2 ; 1 et 3 ; 1 et 4 ; 1 et 7 ; 2 et 3 ; 2 et 4 ; 2 et 5 ; 2 et 7 ; 3 et 4 ; 3 et 6 ; 3 et 7 ; 4 et 5 ; 4 et 6 ; 5 et 6 ; 5 et 7 ; 6 et 7 . Comment organiser ces épreuves de façon qu'aucun étudiant n'ait à passer deux épreuves en même temps et cela sur une durée minimale ?
Pour cela , construisons le graphe G dont les sommets sont les épreuves numérotés de 1 à 7 , une arête relie deux sommets lorsque les deux cours correspondant possédant des étudiants en commun .



5 4

Planifier les examens en un temps minimal consiste une k-coloration de G , avec $k = \gamma (G)$. G possède un sous-graphe complet d'ordre 4 (de sommets 1 , 2 , 3 , 4) , donc $\gamma (G) \geq 4$. Déterminons une partition des sommets de G en sous-ensembles stables :

$S_1 = \{ 1 , 6 \}$
$S_2 = \{ 2 \}$
$S_3 = \{ 3 , 5 \}$
$S_4 = \{ 4 , 7 \}$

D'où $\gamma (G) \leq 4$ et finalement $\gamma (G) = 4$. Les examens peuvent être répartis en 4 périodes de la manière suivante :

1ère Période : épreuves des cours 1 et 6
2ème Période : épreuve du cours 2
3ème Période : épreuves des cours 3 et 5
4ème Période : épreuves des cours 4 et 7

Chapitre III Ensemble indépendant maximal

Dans le chapitre précédent , nous avions parlé du sous-ensemble indépendant .
En effet , dans celui-ci , il faudra préciser le sens donné au mot « maximal » .
Ainsi , un sous-ensemble S_1 est maximal (pour l'inclusion) relativement à une propriété $\wp$ si et seulement si , il n'existe aucun sous-ensemble S_2 tel que $S_2 \supset S_1$ possédant la même propriété $\wp$. Le fait qu'un sous-ensemble maximal pour l'inclusion n'est pas nécessairement de cardinalité maximale a permis d'introduire l'ensemble indépendant maximum , qui est un ensemble indépendant maximal de plus grande cardinalité . Il n'est pas facile de déterminer la valeur de ce cardinal , mais nous avons fourni des relations le liant à d'autres valeurs caractéristiques d'un graphe .

III-1 Définitions

Un ensemble indépendant maximal est un ensemble S' indépendant tel que pour tout sommet $v \in S\text{-}S'$; l'ensemble $S' \cup \{v\}$ n'est pas indépendant .
Tout sommet n'appartenant pas à S' est adjacent à un sommet de S' .
Notons I l'ensemble des sous-ensembles indépendants de S . Alors , on appelle degré d'indépendance ou de stabilité de G , le nombre entier défini par :

$$\alpha (G) = \max_{I \in I} (\text{card} (I))$$

C'est donc le nombre maximal de sommets constituant un ensemble indépendant de S .
Le problème est alors le suivant : étant donné un graphe G ; comment déterminer un stable de cardinal $\alpha (G)$.
La connaissance de cet ensemble nous intéressera particulièrement pour résoudre certains problèmes d'optimisation , de recherche opérationnelle , de localisation etc …
Un ensemble indépendant maximum aura donc pour cardinal $\alpha (G)$.

Propriété 1

$$\varnothing \in I \text{ et } [S_0 \in I , A_0 \subset S_0] \Rightarrow [A_0 \in I]$$

Propriété 2

Soit G un graphe et G' son complémentaire .
C est une clique maximale de G' si et seulement si , C est un ensemble indépendant maximal de G .

Preuve

Supposons que C soit une clique maximale de G' .
Ceci entraîne que $\forall (i , j) \in C$; on a $(i , j) \in A'$, donc $(i , j) \notin A$.
Donc C est stable dans G .
C étant maximale dans G' ; alors pour tout $v \notin C$ (donc $v \in S\text{-}C$) ; il existe $u \in C$ tel que $(u , v) \notin A'$; en d'autre termes ; $C \cup \{v\}$ n'est plus une clique .
Or $(u , v) \notin A'$ est équivaut à $(u , v) \in A$.
Ainsi $C \cup \{ v \}$ ne sera plus stable .
Donc C est maximal dans G .
De même supposons cette que C soit un ensemble stable maximal de G .
Ceci entraîne que $\forall (i , j) \in C$; on a $(i , j) \notin A$, donc $(i , j) \in A'$.
Donc C est complet dans G' .

C étant maximal dans G ; alors pour tout $v \notin C$ ($v \in S\text{-}C$) ; $C \cup \{v\}$ n'est pas stable .
C'est à dire que il existe $u \in C$ tel que $(u , v) \in A$ et donc $(u , v) \notin A'$.
D'où C est maximale dans G' .

Remarque 1
La recherche des sous-ensembles indépendants maximaux d'un graphe peut donc se ramener à la recherche des cliques maximales du graphe G' ; complémentaire de G .

III-2 Relation entre les coefficients caractéristiques

Proposition 1
Soit G un graphe dont le nombre chromatique est $\gamma (G)$ et le degré de stabilité est $\alpha (G)$.
Alors on a :

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

Preuve
On sait que $\gamma (G)$ est le nombre minimal de couleurs avec lequel on peut colorier le graphe et que chaque sous-ensemble de sommets qui a la même couleur constitue un sous-ensemble indépendant maximal . On obtient ainsi une partition de S en $\gamma (G)$ sous-ensembles , appelons-les $S_1 ; S_2 ; \ldots . ; S_{\gamma (G)}$. Ces ensembles sont alors disjoints et :
$\forall i = 1 , 2 , ..., \gamma (G) ;$ $card (S_i) \leq \alpha (G)$; par définition de $\alpha (G)$.

D'où

$$\sum_{i=1}^{\gamma (G)} card (S_i) \leq \sum_{i=1}^{\gamma (G)} \alpha (G) = \alpha (G). \gamma (G)$$

Les S_i formant une partition de S ; alors :

$$\sum_{i=1}^{\gamma (G)} card (S_i) = n$$. Ce qui termine la preuve .

Ce résultat montre $E(n /\gamma(G))$ est un minorant de $\alpha(G)$.

Proposition 2
Soit G un graphe et $I \subset S$ un sous-ensemble indépendant .
Alors son complémentaire I' est recouvrant .

Preuve
Supposons que $I \subset S$ est indépendant .
Alors $(i , j) \notin A , \forall i \in I$ et $\forall j \in I$.
Donc $(i , j) \in A \Rightarrow i \notin I$ ou $j \notin I$, donc $i \in I'$ ou $j \in I'$.
D'où I' est recouvrant .

Proposition 3
Dans le graphe G on a :

$$\delta (G) + \alpha (G) = n$$

Cette égalité ; par ailleurs très importante , nous permettra , une fois connaissant le degré de recouvrement de G , de pouvoir connaître son degré de stabilité .

Preuve
Si $I \subset S$ est un ensemble indépendant maximum de G (il peut ne pas être unique)

Alors card (I) = α (G) .
D'après la proposition précédente , I' est recouvrant minimal .
Donc card (I') = δ (G) .
D'où le résultat card (I) + card (I') = n .

Théorème

Soit le graphe G et k $\in$ S . On a le résultat suivant :
α (G) = max { α (G–k) , 1 + α (G –k–N(k)) } avec N(k) l'ensemble des sommets adjacents à k .

Preuve

Comme G–k $\subset$ G , alors α (G–k) $\leq$ α (G) . ($1)
De même soit M est un sous-ensemble indépendant de G–k–N(k) tel que
card (M) = α (G –k–N(k)) .
Par définition k $\notin$ M et M $\cap$ N(k) = $\varnothing$; alors le sous-ensemble M $\cup$ {k} est indépendant .
Donc card (M) < card (M $\cup$ {k}) $\leq$ α (G) et 1 + α (G –k–N(k) $\leq$ α (G). ($2)
($1) et ($2) entraînent que α (G) $\geq$ max { α (G–k) , 1 + α (G –k–N(k)) }.
Soit U un ensemble indépendant de G tel que card (U) = α (G) , alors on a deux cas possibles:
Si k $\notin$ U alors U $\subset$ S–k et α (G) = card (U) $\leq$ α (G–k) .
Si k $\in$ U alors aucun élément de N(k) n'est dans U . Donc :
U-k $\subset$ S–k–N(k) et card (U–k) $\leq$ α (G –k–N(k)) .
D'où α (G) = card (U) $\leq$ 1 + card (U–k) $\leq$ 1 + α (G –k–N(k)) .

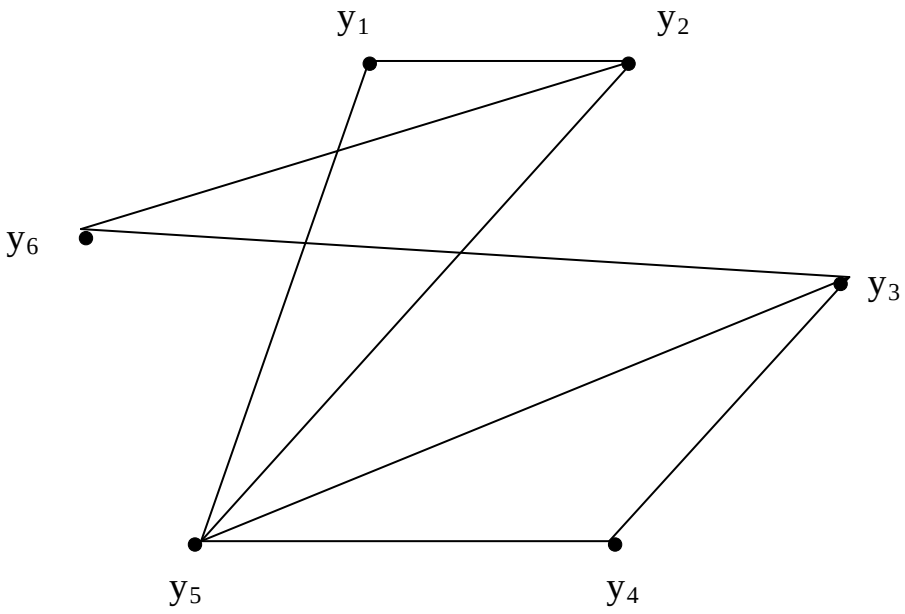
Exemple 1

Les étudiants x_1 ; x_2 ; x_3 ; x_4 et x_5 doivent passer certaines épreuves parmi les suivantes : y_1 ; y_2 ; y_3 ; y_4 ; y_5 et y_6 . L'examen se déroulant par écrit , on désire que tous les étudiants qui doivent subir une même épreuve le fassent simultanément . Chaque étudiant ne peut se présenter qu'à une épreuve au plus chaque jour .
voici la liste des épreuves que doit passer chaque étudiant :

Etudiants	Epreuves
x_1	y_1 ; y_2 ; y_5
x_2	y_3 ; y_4
x_3	y_2 ; y_6
x_4	y_3 ; y_4 ; y_5
x_5	y_3 ; y_6

Le problème est de déterminer le nombre maximal d'épreuves que l'on peut organiser le même jour .
Pour cela , on peut associer au problème le graphe G défini par :
S = { y_1 ; y_2 ; y_3 ; y_4 ; y_5 ; y_6 } ; c'est à dire qu' à toute épreuve , on fait correspondre un sommet de ce graphe .

$V \in A$; c'est à dire que V est une arête du graphe si et seulement si un même étudiant doit subir les épreuves correspondantes .



Nous avons les arcs (y_1 , y_5) , $(y_1 ; y_2)$, (y_2 , y_5) , (y_4 , y_5) , (y_3 , y_5) , (y_3 , y_6) , (y_2, y_6) , (y_4 , y_3) .
Le nombre maximal d'épreuves que l'on peut organiser le même jour est alors le nombre de stabilité de G .
$\alpha (G) = 3$ correspondant à $\{ y_1 ; y_4, y_6 \}$.
D'où les épreuves $y_1 ; y_4, y_6$ pourront avoir lieu le même jour .
Par extension , le nombre minimal de jours nécessaires est le nombre chromatique de G c'est à dire $\gamma (G) = 3$.
Par exemple , le premier jour $y_1 ; y_4; y_6$, le deuxième jour $y_2 ; y_3$ et enfin le troisième jour y_5 .

Chapitre IV Algorithme de détermination d'un ensemble indépendant maximal

Introduction

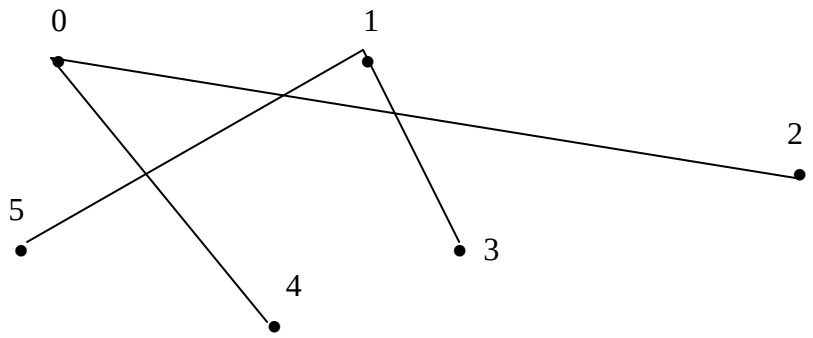
Dans ce chapitre , il sera question de mettre sur pied un algorithme qui permet de trouver un ensemble indépendant maximal dans un graphe G . Dans le souci de fournir les meilleurs résultats , on a introduit une certaine hiérarchie au niveau des sommets du graphe par l'intermédiaire de leur degré respectif ; et l'algorithme a été construit suivant cette logique . Pour voir ce dernier tourner , nous avons écrit le programme correspondant dans l'un des langages Informatiques les plus courant à savoir le C . En effet , ce programme permet , à partir de la matrice d'adjacence sommets-sommets du graphe d'avoir comme résultats un sous-ensemble indépendant maximal construit à partir du sommet de plus faible degré . IL permet également d'obtenir la suite des degrés de G , ordonnée de façon croissante ainsi que la liste des sommets elle aussi disposée suivant l'ordre des degrés . D'autres résultats intéressants pourront aussi être fournis par le programme comme le fait de pouvoir vérifier si un ensemble donné est oui ou non indépendant maximal . En plus , si la réponse est négative , il essayera de trouver dans l'ensemble fourni , les sommets qui vérifient la propriété . Le code source du programme est mis en annexe de ce mémoire et pourra être amélioré en vu d'obtenir d'autres résultats .

IV-1 Construction d'un stable maximal dans le cas général

Dans le cas le plus général , où aucun critère n'est privilégié , la détermination d'un ensemble indépendant maximal se fait de façon simple .
Il existe deux façons de présenter un algorithme : d'une manière assez « littéraire » ou d'une manière très proche de la mise en œuvre informatique . Généralement l'exposé « littéraire » est beaucoup plus clair pour la compréhension de l'algorithme . Cependant , à partir d'un tel exposé , de nombreux choix importants restent à faire pour la mise en œuvre sur calculateur (choix d'une représentation du graphe , choix des structures de données etc…) lesquels ont une incidence directe sur l'efficacité de l'algorithme . Nous nous sommes efforcés , de présenter successivement ces deux descriptions , en raison de leur intérêt complémentaire .
Pour comparer les différents algorithmes existants et ne retenir que les plus intéressants , beaucoup de critères sont possibles . L'étude des performances des algorithmes suivant ces critères conduit à définir la complexité d'un algorithme .
En effet , prenons un sommet quelconque i ;
et posons $I = \{ i \}$;
il faut ensuite considérer l'ensemble S_i constitué des sommets non adjacents à i .
Deux cas se présentent alors :
1) si $S_i = \varnothing$; c'est la fin et I est un singleton ;
2) si $S_i \neq \varnothing$; on choisit un sommet quelconque j dans S_i , puis on l'ajoute dans I .
La prochaine étape est similaire à la première . En effet , il suffit de déterminer l'ensemble S_j constitué des sommets non adjacents à i et à j .
De la même manière , on a deux cas :
1) si $S_j = \varnothing$; c'est la fin et I est un couple composé de i et de j ;
2) si $S_i \neq \varnothing$; on continue le processus , de proche en proche j'usqu'à avoir un S_k qui est vide . L'ensemble I obtenu est alors indépendant , et il est maximal par construction .

Exemple 1

Soit le graphe suivant :



Considérons le sommet 4 et posons $I = \{ 4 \}$.
Soit S_4 l'ensemble des sommets non adjacents à 4 .
Alors , $S_4 = \{ 1 , 2 , 3 , 5 \}$.
Choisissons le sommet 2 et posons $I = \{ 4 , 2 \}$.
L'ensemble des sommets non adjacents à I est $S_2 = \{ 1 , 3 , 5 \}$.
Prenons le sommet 3 et posons $I = \{ 4 , 2 , 3 \}$.
Seul le sommet 5 est non adjacent à I ; donc $S_3 = \{ 5 \}$.
Prenons le sommet 5 et posons $I = \{ 4 , 2 , 3 , 5 \}$.
Il n'existe pas de sommet non adjacent à I . Alors $S_5 = \varnothing$. C'est la fin du processus et l'ensemble indépendant maximal construit est $I = \{ 4 , 2 , 3 , 5 \}$.

IV-2 Présentation de l'algorithme

Numéroter les sommets de G de 0 à n-1 (G étant supposé fini d'ordre n) .
Dresser la suite croissante des degrés de G .
Ranger les sommets du graphe G par ordre croissant de leur degré et de leur numéro .

Initialisation :

$I_m = \varnothing$;

$I_m \leftarrow$ { sommet de plus petit degré et de plus faible numéro} ;

Pour chaque nœud N , par degré et numéro croissants ;

S'il n'existe aucune arête reliant N à aucun sommet de I_m ;

Alors , faire ;

$I_m \leftarrow I_m \cup \{ N \}$;

Retourner I_m ;

Remarque 1

Dans le cas où plusieurs sommets ont le même degré , on choisit celui de plus faible numéro .
On a montré au chapitre I que dans un graphe , il y a au moins deux sommets qui ont le même degré .

Justification de l' algorithme

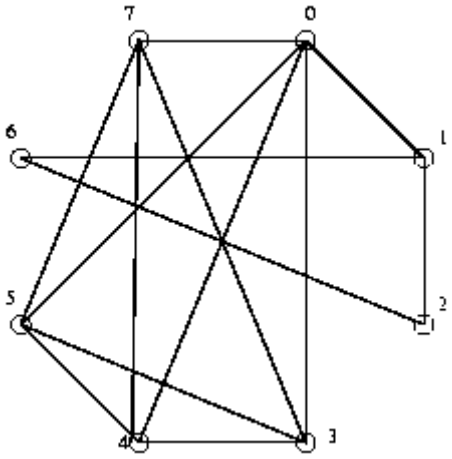
Les éléments de G sont considérés les un après les autres par ordre croissant de leur degré et de leur numéro .
Si l'élément N considéré peut être ajouté à I_m sans nuire à l'indépendance de ce dernier , il l'est ; sinon , il est écarté .
$\varnothing$ est indépendant et comme , N n'est ajouté à I_m que si
$I_m \cup \{ N \}$ est indépendant , le sous-ensemble I_m est toujours indépendant par induction .
Par suite l'algorithme retourne toujours un ensemble indépendant I_m .
De même I_m est maximal par construction .

Remarque 2

Le temps d'exécution de l'algorithme est facile à analyser .
La phase de tri prend un temps en 0(nlgn) .
La ligne 7 est exécutée exactement (n-1) fois .
Chaque exécution de la ligne 7 impose de vérifier si $I_m \cup \{ N \}$ est indépendant .

Exemple 2

S = { 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 }



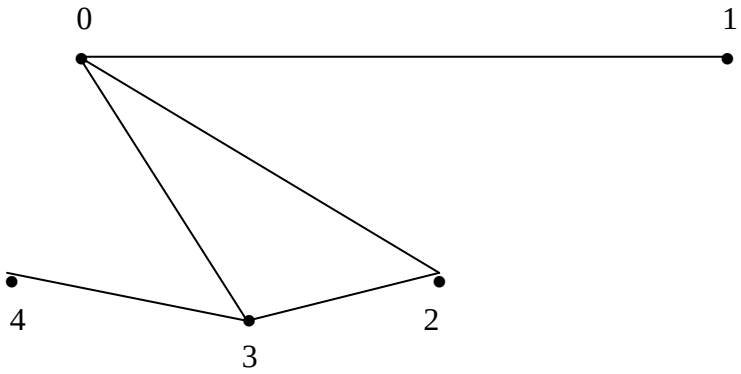
Les sommets du graphe sont déjà numérotés de 0 à 7 .
La suite des degrés de G , ordonnée de façon croissante est : 2 , 2 , 3 , 4 , 4 , 4 , 4 , 5 .
Ceci nécessite de faire le tour de tous les sommets et de déterminer le degré de chacun d'eux .
La suite des sommets du graphe , rangée par ordre croissant de leur degré et de leur numéro est : 2 , 6 , 1 , 3 , 4 , 5 , 7 , 0 .

Les sommets de plus faibles degrés sont : 2 et 6 . Ils ont pour degré 2 . On choisit le sommet 2 qui répond au critère demandé . Ainsi $I_m = \{ 2 \}$
Les sommets non adjacents à 2 , ordonnés de la même manière que précédemment sont :
0 , 3 , 4 , 5 , 7 .
Parmi ces derniers , choisissons celui de plus petit degré et de plus petit numéro .
Il s'agit de 3 .
Ainsi $I_m = \{ 2 , 3 \}$.
De même , on détermine les sommets non adjacents à 2 et à 3 . Il n'y en a pas .
C'est la fin de l'algorithme .
En effet l'ensemble indépendant construit est { 2 , 3 } . Il est maximal puisque l'ajout de n'importe quel autre sommet casse l'indépendance de l'ensemble .

Remarque 3
Le temps d'exécution de l'algorithme pour ce graphe est assez court car il possède deux cliques qui sont : { 0 , 3 , 4 , 5 , 7 } et { 1 , 2 , 6 } . Il suffit de prendre un élément par clique et c'est fini .
Vous avez remarqué que l'algorithme n'est pas simple à réaliser à la main dans le cas général, car même pour n assez petit on devra faire beaucoup d'opérations , à fortiori dans le cas où n est suffisamment grand .
Supposons que nous ayons un graphe avec des centaines de sommets (ce qui est le cas pour beaucoup d'applications); alors il serait fastidieux , voir irréalisable de tenter de faire toutes les opérations menant à la solution avec la main .
D'où une des motivations pour faire tourner l'algorithme . Voici à cet effet des captures d'écran montrant quelques exemples d'exécutions :

Exemple 3

Considérons le graphe suivant :



Il faut préciser que l'exemple est réalisé sous le système Linux et que les données du programme demeurent l'ordre du graphe , et sa matrice d'adjacence ; tout le reste représente les résultats fournis par l'application . Parmi ces derniers , on trouvera l'ensemble indépendant maximal construit ; mais aussi une invitation à une éventuelle vérification de l'existence de la propriété sur un sous-ensemble à préciser . Cependant dans ce premier exemple , nous n'avons pas fait état de ce cas .

[root@localhost root]# ./matrice

Donner l'ordre du graphe:
5
Donnez la ligne numéro 1 de la matrice d'adjacence de G:
0 1 1 1 0
Donnez la ligne numéro 2 de la matrice d'adjacence de G:
1 0 0 0 0
Donnez la ligne numéro 3 de la matrice d'adjacence de G:
1 0 0 1 0
Donnez la ligne numéro 4 de la matrice d'adjacence de G:
1 0 1 0 1
Donnez la ligne numéro 5 de la matrice d'adjacence de G:
0 0 0 1 0

La matrice d'adjacence de G est:

0 1 1 1 0
1 0 0 0 0
1 0 0 1 0
1 0 1 0 1
0 0 0 1 0

Le graphe comporte 5 arcs.

La suite des degrés , ordonnée de façon croissante est:

1 1 2 3 3

Voici les numéros des sommets du graphe, ordonnés par degré et numéro croissants:

1 4 2 0 3
Pour débuter l'algorithme, considérons le sommet de numéro 1 :

Les sommets 1 et 0 sont liés.
Les sommets 2 et 3 sont liés.
Les sommets 4 et 3 sont liés.

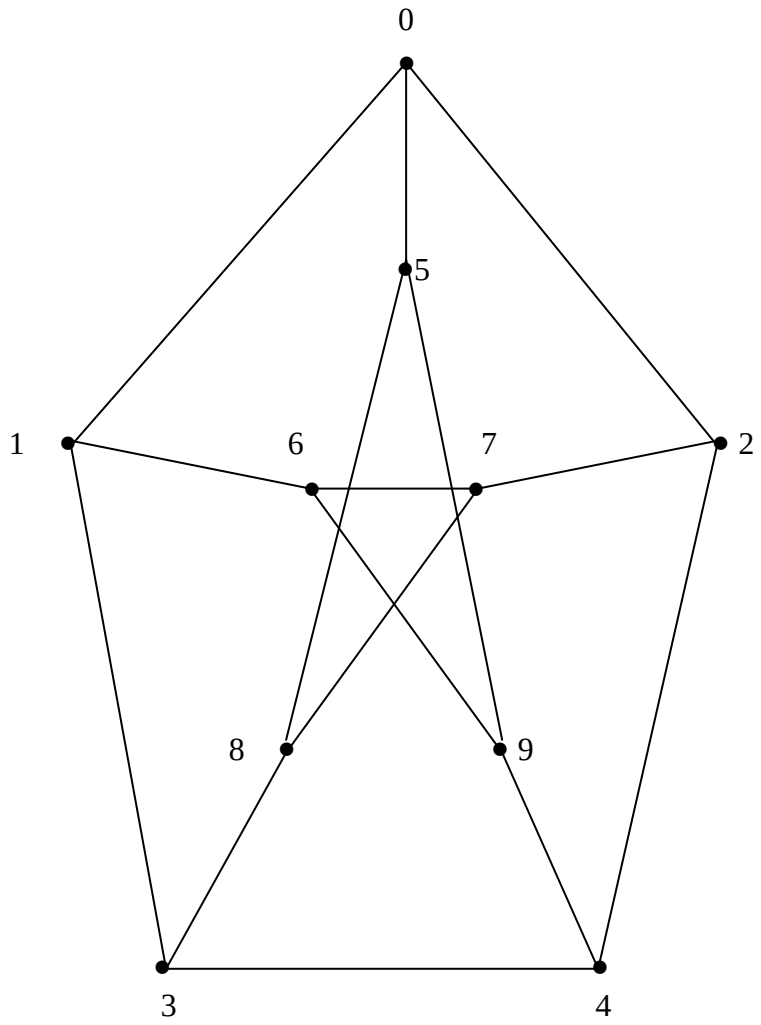
Les 3 sommets suivants forment un ensemble indépendant maximal:

1 2 4

<u>Exemple</u> 4

Considérons le graphe de Petersen .
Il est d'ordre 10 .

L’application de l’algorithme à ce graphe donne les résultats suivants :
En outre , il apparaît clairement que ce graphe est régulier de degré 3 .



[root@localhost root]# ./matrice
Donner l'ordre du graphe:
10
Donnez la ligne numéro 1 de la matrice d'adjacence de G:
0 1 1 0 0 1 0 0 0 0
Donnez la ligne numéro 2 de la matrice d'adjacence de G:
1 0 0 1 0 0 1 0 0 0
Donnez la ligne numéro 3 de la matrice d'adjacence de G:
1 0 0 0 1 0 0 1 0 0
Donnez la ligne numéro 4 de la matrice d'adjacence de G:
0 1 0 0 1 0 0 0 1 0
Donnez la ligne numéro 5 de la matrice d'adjacence de G:
0 0 1 1 0 0 0 0 0 1
Donnez la ligne numéro 6 de la matrice d'adjacence de G:
1 0 0 0 0 0 0 0 1 1
Donnez la ligne numéro 7 de la matrice d'adjacence de G:
0 1 0 0 0 0 0 1 0 1

Donnez la ligne numéro 8 de la matrice d'adjacence de G:
0 0 1 0 0 0 1 0 1 0
Donnez la ligne numéro 9 de la matrice d'adjacence de G:
0 0 0 1 0 1 0 1 0 0
Donnez la ligne numéro 10 de la matrice d'adjacence de G:
0 0 0 0 1 1 1 0 0 0

La matrice d'adjacence de G est:

0 1 1 0 0 1 0 0 0 0
1 0 0 1 0 0 1 0 0 0
1 0 0 0 1 0 0 1 0 0
0 1 0 0 1 0 0 0 1 0
0 0 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 0 1 1
0 1 0 0 0 0 0 1 0 1
0 0 1 0 0 0 1 0 1 0
0 0 0 1 0 1 0 1 0 0
0 0 0 0 1 1 1 0 0 0

Le graphe comporte 15 arcs.

La suite des degrés, ordonnée de façon croissante est:

3 3 3 3 3 3 3 3 3 3

Voici les numéros des sommets du graphe, ordonnés par degré et numéro croissants:
0 1 2 3 4 5 6 7 8 9

Pour débuter l'algorithme, considérons le sommet de numéro 0 :

Les sommets 0 et 1 sont liés.
Les sommets 0 et 2 sont liés.
Les sommets 4 et 3 sont liés.
Les sommets 8 et 3 sont liés.
Les sommets 9 et 4 sont liés.
Les sommets 0 et 5 sont liés.
Les sommets 7 et 6 sont liés.
Les sommets 9 et 6 sont liés.
Les sommets 8 et 7 sont liés.

Les 3 sommets suivants forment un ensemble indépendant maximal:

0 8 9

Voulez-vous tester si un ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
6
Donner le sommet suivant:
7

Les sommets 6 et 7 sont liés.
Donner le sommet suivant:
9
Les sommets 6 et 9 sont liés.
Donner le sommet suivant:
8
Donner le sommet suivant:
3
Les sommets 8 et 3 sont liés.
Donner le sommet suivant:
4
Donner le sommet suivant:
5
Les sommets 8 et 5 sont liés.
Donner le sommet suivant:
1
Les sommets 6 et 1 sont liés.
Donner le sommet suivant:
2
Les sommets 4 et 2 sont liés.
Donner le sommet suivant:
0
Les 4 sommets suivants forment un ensemble indépendant maximal:
6 8 4 0
Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
9
Donner le sommet suivant:
8
Donner le sommet suivant:
7
Les sommets 8 et 7 sont liés.
Donner le sommet suivant:
6
Les sommets 9 et 6 sont liés.
Donner le sommet suivant:
5
Les sommets 9 et 5 sont liés.
Donner le sommet suivant:
4
Les sommets 9 et 4 sont liés.
Donner le sommet suivant:
3
Les sommets 8 et 3 sont liés.
Donner le sommet suivant:
2
Donner le sommet suivant:
1
Donner le sommet suivant:
0

Les sommets 2 et 0 sont liés.
Les sommets 1 et 0 sont liés.
Les 4 sommets suivants forment un ensemble indépendant maximal:
9 8 2 1
Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
7
Donner le sommet suivant:
8
Les sommets 7 et 8 sont liés.
Donner le sommet suivant:
9
Donner le sommet suivant:
1
Donner le sommet suivant:
2
Les sommets 7 et 2 sont liés.
Donner le sommet suivant:
3
Les sommets 1 et 3 sont liés.
Donner le sommet suivant:
4
Les sommets 9 et 4 sont liés.
Donner le sommet suivant:
5
Les sommets 9 et 5 sont liés.
Donner le sommet suivant:
6
Les sommets 7 et 6 sont liés.
Donner le sommet suivant:
0
Les sommets 1 et 0 sont liés.
Les 3 sommets suivants forment un ensemble indépendant maximal:
7 9 1
Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
7
Donner le sommet suivant:
4
Donner le sommet suivant:
1
Donner le sommet suivant :
8
Les sommets 7 et 8 sont liés.
Donner le sommet suivant:
5
Donner le sommet suivant:
2
Les sommets 7 et 2 sont liés.
Donner le sommet suivant:

9
Les sommets 4 et 9 sont liés.
Les sommets 5 et 9 sont liés.
Donner le sommet suivant:
6
Les sommets 7 et 6 sont liés.
Donner le sommet suivant:
3
Les sommets 4 et 3 sont liés.
Les sommets 1 et 3 sont liés.
Donner le sommet suivant:
0
Les sommets 1 et 0 sont liés.
Les sommets 5 et 0 sont liés.
Les 4 sommets suivants forment un ensemble indépendant maximal:
7 4 1 5
Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
5
Donner le sommet suivant:
6
Donner le sommet suivant:
4
Donner le sommet suivant:
8
Les sommets 5 et 8 sont liés.
Donner le sommet suivant:
9
Les sommets 5 et 9 sont liés.
Donner le sommet suivant:
7
Les sommets 6 et 7 sont liés.
Donner le sommet suivant:
2
Les sommets 4 et 2 sont liés.
Donner le sommet suivant:
1
Les sommets 6 et 1 sont liés.
Donner le sommet suivant:
3
Les sommets 4 et 3 sont liés.
Donner le sommet suivant:
0
Les sommets 5 et 0 sont liés.
Les 3 sommets suivants forment un ensemble indépendant maximal:
5 6 4

IV-3 Ensemble indépendant maximum d'un graphe

Parmi les sous-ensembles indépendants maximaux de G , il en existe au moins un dont le cardinal est le plus grand (car il n'est pas nécessairement unique) .
Supposons que r soit le plus grand degré de G .
S'il existe une clique de taille m , alors , on montre que $E(n / (r+1)) \leq \alpha (G) \leq n-m+1$; où E(x) désigne la partie entière de x .
Dans les chapitres II et III , on avait montré les relations suivantes :

$$\gamma (G) \leq r + 1 \quad \text{et} \quad n \leq \alpha (G).\gamma (G)$$

L'inégalité $n \leq \alpha (G).\gamma (G)$ donne $\alpha (G) \geq n / \gamma (G)$.
De même $\gamma (G) \leq r + 1$ entraîne $1/ \gamma (G) \geq 1/ (r+1)$.
Ainsi , en multipliant par n , on a : $n / (r+1) \leq \alpha (G)$. Ceci donne la preuve de la première inégalité puisque $E(x) \leq x$ pour tout réel $x \geq 0$.
L'inégalité $\alpha (G) \leq n-m+1$ relève du domaine de l'évidence puisque au plus un élément de la clique identifiée peut appartenir à un indépendant maximum .

Recherche d'un maximum global

$I \leftarrow I_m$;

Pour tout i , tel que $card (I_m) +1 \leq i \leq n-m+1$;

S'il existe un sous-ensemble indépendant maximal I_{mi} de taille i , alors faire :

$I \leftarrow I_{mi}$;

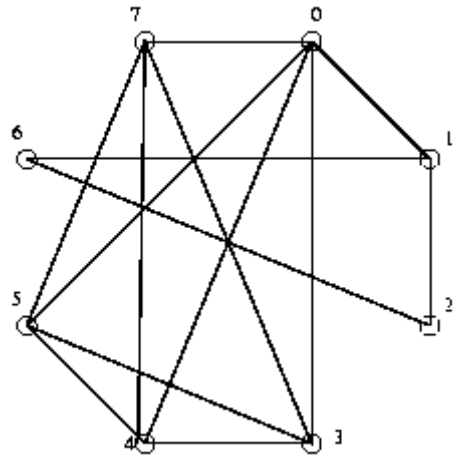
Sinon ;

Retourner I .

FIN .

Exemple 5

S = { 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 }



Dans l'exemple 2 , l'algorithme nous avait permis de construire l'ensemble indépendant maximal $I_m = \{ 2 , 3 \}$.
L'ensemble { 0 , 3 , 4 , 5 , 7 } est une clique de taille 5 .
Considérons alors le sous-graphe constitué des sommets 3 , 1 , 2 , et 6 . Son ordre est 4 .
Ainsi , { 1 , 2 , 6 } est une clique de ce sous-graphe . Sa taille est 3 .
Alors existe t-il un indépendant maximal de taille i tel que : $card (I_m) + 1 \leq i \leq n-m+1$?
Ceci donnerait : $2+1 = 3 \leq i \leq 4 -3+1 = 2$; ce qui est impossible . Donc , on ne peut pas trouver meilleur que I_m du point de vu cardinal .
D'où , on en déduit que $\alpha (G) = 2$ et que I_m est un indépendant maximum de G .

<u>Remarque</u> 4
Il est possible de regrouper les deux moutures qui représentent respectivement la détermination d'un ensemble indépendant maximal (qui est un maximum local) , et celle d'un indépendant maximum (maximum global) . Dans le cas où le graphe est très dense (beaucoup d'arcs) , l'algorithme global peut converger et aboutir à la solution avec un nombre assez réduit d'opérations . En effet , on procède par sous-graphe comme dans l'exemple précédent . Ainsi , à chaque étape , on considère le sous-graphe constitué des sommets de G dont on a soustrait les sommets des cliques trouvées j'usqu'à cette étape .

IV-4 <u>Cas particuliers</u>

Dans cette section , nous allons voir quelques classes de graphe où il est relativement facile de déterminer un stable maximum et son cardinal .

<u>Cas du graphe nul</u>
Le graphe G ne possède pas d'arc .
Alors tous les sommets de G sont indépendants .
Dans ce cas , S est l'ensemble indépendant maximum de G .

<u>Cas du graphe complet</u>
Ici , deux sommets quelconque de G sont liés par au moins un arc .
Ainsi , chaque sommet de ce graphe constitue à lui seul un ensemble indépendant maximum .

<u>Cas d'un graphe composé de k-cliques</u>

Il suffit de prendre un sommet par clique et on obtient alors un ensemble indépendant maximum ayant k éléments .

Cas d'un cycle

Si n est pair alors $\alpha (G) = n / 2$.

Si n est impair alors $\alpha (G) = (n-1) / 2$.

Il suffit de choisir les sommets qui vont constituer l'ensemble indépendant maximum .

A) Stable maximum dans un arbre

Détermination d'un ensemble stable de cardinalité maximale dans un arbre

Voici un exemple particulièrement efficace d'utilisation de la programmation dynamique .
Nous considérons un graphe G qui est un arbre pour lequel nous désirons calculer le nombre de stabilité $\alpha(G)$.
Bien que le nombre de stabilité soit généralement difficile à calculer pour un graphe quelconque , nous allons montrer que dans le cas d'un arbre (rappelons qu'un arbre est un graphe d'au moins 2 sommets , connexe sans cycle) , ce nombre peut être calculé rapidement avec un algorithme de programmation dynamique .
En choisissant arbitrairement un sommet r de l'arbre , nous définissons de manière unique une orientation des arêtes de manière à déduire de cet arbre une arborescence de racine r .
Par souci de simplification , nous appellerons aussi G cet arborescence et confondrons les arêtes de l'arbre avec les arcs de l'arborescence. Nous noterons G_i le sous-arborescence extrait de l'arborescence G de racine i et par extension , le sous-graphe correspondant .
L'algorithme est fondé sur la propriété évidente suivante :
Si S* est un ensemble indépendant pour G_i , alors soit $i \in S^*$, soit $i \notin S^*$.
Pour chaque G_i , nous allons calculer deux valeurs : R_i la taille d'un stable maximal (au sens de l'inclusion) contenant i et $R_i^{\wedge}$ la taille d'un ensemble stable maximal ne contenant pas i .
Ainsi , il est aisé de constater que $\alpha(G_i) = \max (R_i , R_i^{\wedge})$.
Pour chaque sommet i , on note $\Gamma(i)$ l'ensemble des successeurs de i .
Nous allons maintenant montrer comment obtenir les deux valeurs R_i et $R_i^{\wedge}$ à partir des valeurs R_j et $R_j^{\wedge}$ des successeurs j de i .
Premièrement , si G_i ne compte que le seul sommet i (qui est alors une feuille ou un sommet pendant de l'arborescence G) nous avons immédiatement $R_i = 1$ et $R_i^{\wedge} = 0$.
Maintenant si i a des successeurs et appartient à un sous- ensemble stable de G_i alors aucun de ses successeurs j ne peut appartenir à ce stable .
Donc

$$R_i = 1 + \sum_{j \in \Gamma(i)} R_j^{\wedge} \quad ;$$

si i n'appartient pas à un stable de G_i , ses successeurs j peuvent ou non appartenir à ce stable. Donc

$$R_i^{\wedge} = \sum_{j \in \Gamma(i)} \alpha(G_j)$$

La formule récursive suivante , appelée à partir d'un sommet quelconque d'une arborescence G calcule $\alpha(G_i)$.
Si $\Gamma(i) = \varnothing$ $R_i = 1$, $R_i^{\wedge} = 0$, $\alpha(G_i) = 1$
Sinon

$$R_i = 1 + \sum_{j \in \Gamma(i)} R_j^{\wedge} \quad ;$$

$$R_i^{\wedge} = \sum_{j \in \Gamma(i)} \alpha(G_j)$$

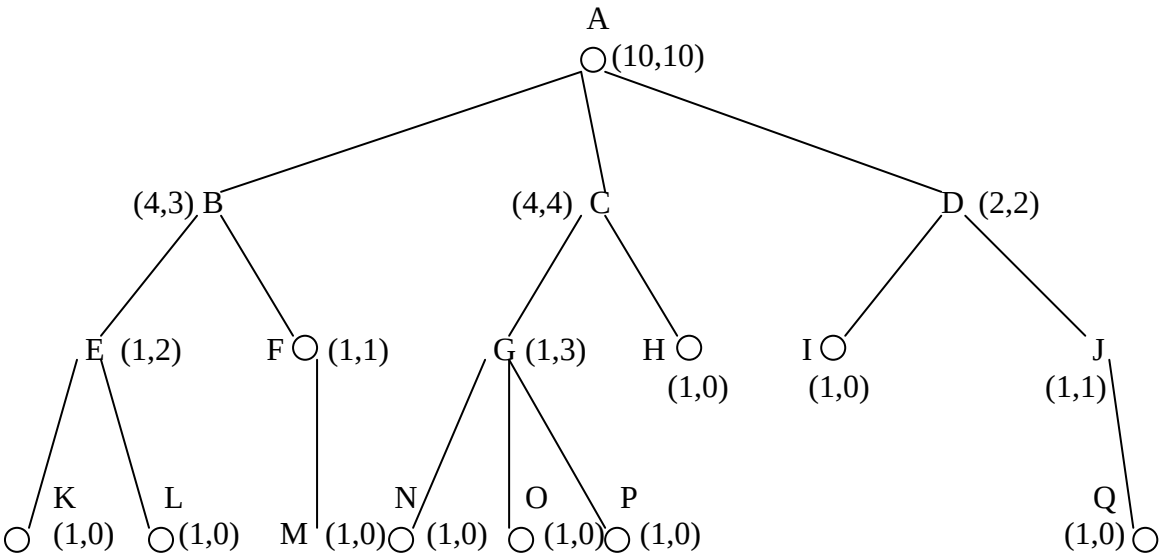
$$\alpha(G_i) = \max_{j \in \Gamma(i)} (R_i , R_i^{\wedge}) .$$

Remarque 5

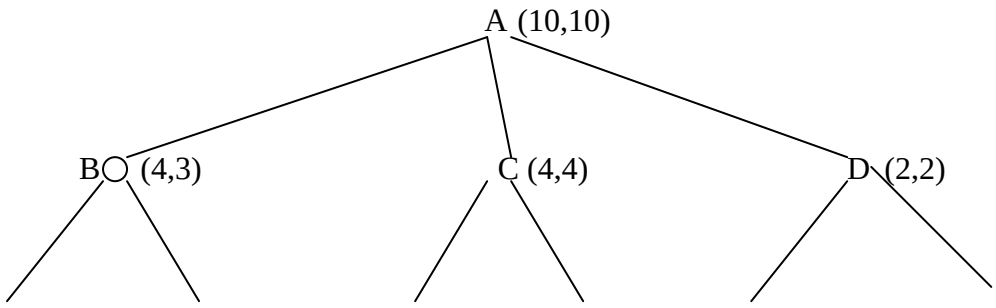
Ce résultat applicable particulièrement pour les arbres nous pousse à réfléchir sur le problème suivant : Ayant un graphe quelconque , est ce qu'on peut l'assimiler à un arbre ? Si oui , le problème de la détermination d'un ensemble indépendant maximum est résolu
Cependant , la réponse est non ; mais on a le résultat suivant :
On prouve que tout graphe connexe admet un graphe partiel qui est un arbre .
Cette propriété permet de construire un arbre à partir d'un graphe connexe .
Un graphe biparti ainsi qu'un cycle pair peuvent être transformés en arbre .

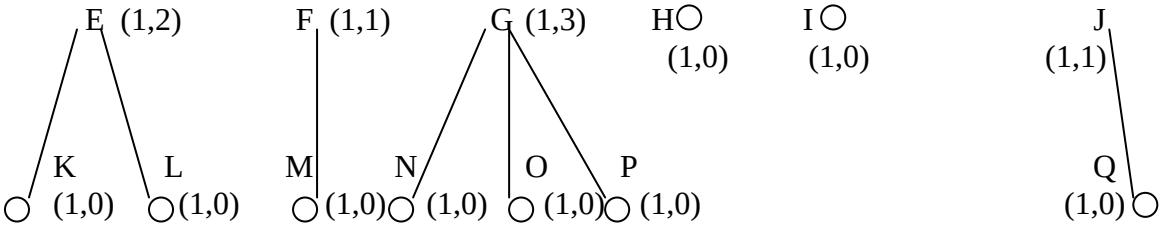
Exemple 6

L'exemple illustre l'application de la formule précédente . Les premières valeurs sont calculées à partir des feuilles de l'arbre puis remontées de proche en proche j'usqu'à la racine. Pour tout sommet i , sont notées les deux valeurs $(R_i , R_i^{\wedge})$, on en déduit aisément
$\alpha(G_i) = \max (R_i , R_i^{\wedge})$. Nous avons indiqué les dix sommets formant un ensemble stable par un cercle. Notons que l'ensemble stable obtenu n'est unique .



On peut prendre S* = { A , F , H , I , K , L , N , O , P , Q } comme ensemble stable de cardinalité maximale .





Un deuxième ensemble stable de cardinal maximal est : { B,H,I,K,L,M,N,O,P,Q }
L'ensemble stable construit dans la première figure a été obtenu de la manière suivante :
Puisque $R_A = R_A^{\wedge} = 10$, le sommet A peut soit appartenir , soit ne pas appartenir à un ensemble stable de 10 éléments : Choisissons la solution qui contient A (remarquons que la deuxième solution ne contient pas A) .
Les sommets B,C,D successeurs de A dans l'arborescence n'appartiennent donc pas au stable .

Construction d'un arbre à partir d'un graphe connexe

En effet , il suffit de considérer une arête u_0 puis de rechercher une arête u_1 qui ne forme pas de cycle avec u_0 , de considérer ensuite $\{ u_0 , u_1 \}$ et de rechercher une arête u_2 qui ne forme pas de cycle avec l'une des précédentes ou leur ensemble , et ainsi de suite , ……..
G est sans cycle et l'on crée un cycle et un seul en ajoutant une arête entre deux sommets non adjacents ; on obtiendra un arbre quand cette procédure ne permet plus d'ajouter une arête à l'ensemble obtenu (le résultat n'est pas unique) .

Cas d'un graphe k-partite

Un graphe G est dit k-partite s'il existe une partition de S en k sous-ensembles S_1 , ……., S_k tel que pour tout $u = (i , j) \in A$; il existe des entiers p et q distincts vérifiant :

$1 \leq p \leq k$ et $1 \leq q \leq k$ tel que :

1) $i \in S_p \Rightarrow j \in S_q$
2) $i \in S_q \Rightarrow j \in S_p$

pour k = 2 alors on retrouve le graphe biparti .

Résultat Pour un graphe k-partite , complet (i. e $\forall$ s , p $\in$ { 1 , …. ;k } avec $s \neq p$,) ; $\forall (i , j) \in S_s * S_p$ alors $(i , j) \in A$) ; chacune des partitions S_i sera un ensemble indépendant maximal et celui ayant le plus grand cardinal sera le sous-ensemble indépendant maximum .

Cas d'un Couplage

Dans les problèmes relatifs au couplage d'un graphe , on recherche souvent le couplage maximal , c'est à dire celui qui comporte le maximum d'arcs .
Le problème du couplage nous intéresse tout aussi particulièrement car , ayant obtenu un couplage maximal d'un graphe , on en déduit facilement un ensemble indépendant .
En effet , il suffit de prendre un sommet quelconque par arc du couplage . Puisque les arcs du couplage ne sont pas adjacents , alors deux sommets extrémités de deux arcs distincts ne seront pas liés par un arc , donc ils seront indépendants et on continue ce procédé j'usqu'à parcourir tous les éléments du couplage et on obtient ainsi le résultat .
Son cardinal sera bien évidemment le même que celui du couplage .
On peut obtenir ce résultat en utilisant les graphes lignes .
En effet , les sommets du graphe ligne représentent les arcs du graphe initial . De plus deux sommets du graphe ligne sont adjacents si les arcs correspondant dans le graphe initial sont incidents .

Le graphe ligne transforme un couplage en sous-ensemble indépendant .
En effet , on prouve que si le couplage est maximal , alors le sous-ensemble indépendant correspondant dans le graphe ligne est lui aussi maximal et réciproquement .
Le problème de trouver un couplage maximal dans un graphe biparti trouve de nombreuses applications pratiques . Par exemple on pourrait coupler un ensemble L de machines avec un ensemble B de tâches à effectuer simultanément . On considère la présence d'une arête (u , v) dans A comme signifiant qu'une machine particulière $u \in L$ est capable d'effectuer une tâche particulière $v \in B$. Un couplage maximal permet de faire travailler plus de machines .

Exemple 7

Dans le problème d'affectation suivant , il s'agit étant donné des ouvriers constituant l'ensemble $X = \{ x_1 ; x_2 ; x_3 ; \ldots\ldots ; x_n \}$ et n machines constituant l'ensemble
$Y = \{ y_1 ; y_2 ; y_3 ; \ldots\ldots ; y_n \}$; d'affecter un ouvrier par machine de manière à obtenir un rendement maximal . Bien entendu pour ce faire , on connaît les rendements respectifs (et différents) de chaque ouvrier sur chaque machine .
A priori , il s'agit d'abord de la recherche d'un couplage dans le graphe G . On suppose que chaque ouvrier peut conduire chaque machine .
Toute application bijective de X sur Y est une solution admissible et constitue un couplage de X sur Y . Il n'est évidemment pas question de dresser la liste de tous les couplages pour calculer le rendement correspondant (ce serait n !) .
Par exemple , si n = 11 , en admettant qu'on puisse examiner une solution par seconde , l'exploration de 11 ! durerait plus d'un an et celle de 20 ! 8 milliards d'années .
Le problème se résout en recherche opérationnelle au moyen de l'algorithme Hongrois .

IV-5 Recherche d'une clique maximum

La recherche d'une clique maximale du graphe G' , complémentaire de G peut se ramener à celle d'un ensemble indépendant maximal dans le graphe G .
L' algorithme est inspiré des indications trouvées sur le site : http://www.tcs.mu-lueberk.de/
La recherche se fait en deux fois . On cherche d'abord une clique maximale , puis on cherche s'il une clique plus grande . A chaque étape , on élimine les nœuds dont le degré n'est pas suffisant . Globalement , l'algorithme est le suivant :

Recherche d'une clique maximale C_m (maximum local)

(sous-graphe complet tel que si l'on ajoute un nœud , on perd la complétude)

Trier l'ensemble des nœuds selon leur degré

$C_m \leftarrow$ { nœud de plus haut degré }

Pour chaque nœud N , par degré décroissant

S'il existe une arête reliant N à tous les nœuds de C_m

$C_m \leftarrow C_m \cup \{ N \}$

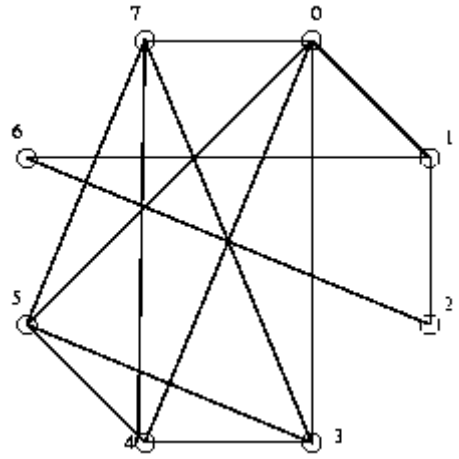
Recherche d'une clique maximum C (maximum global)

Posons d_m le plus haut degré du graphe .

$C \leftarrow C_m$

Pour i de card(C_m) +1 à d_m + 1
S'il existe une clique de taille i
C ← une clique de taille i .
Sinon
Retourner C .

Exemple 8



La suite des degrés de G , ordonnée de façon croissante est : 2 , 2 , 3 , 4 , 4 , 4 , 4 , 5 .
La suite des sommets du graphe , rangée par ordre croissant de leur degré et de leur numéro est : 2 , 6 , 1 , 3 , 4 , 5 , 7 , 0 .
Seul le sommet 0 répond aux critères de l'algorithme . Donc C_m = { 0 } .
Dans le même ordre d'idée , c'est 3 qui correspond au bon choix . Ainsi , C_m = { 0 , 3 } .
On continue le processus j'usqu'à trouver la clique : { 0 , 3 , 4 , 5 , 7 }
En effet , il suffit de chercher s'il existe une clique de taille i vérifiant :
card(C_m) +1 ≤ i ≤ d_m + 1 ;
c'est à dire , 6 ≤ i ≤ 6 .
Ceci donne i = 6 . Il n'existe pas une clique de taille 6 puisqu'on peut trouver au moins deux sommets dont les degrés sont inférieurs strictement à 5 .
La clique maximum trouvée est alors : { 0 , 3 , 4 , 5 , 7 } .

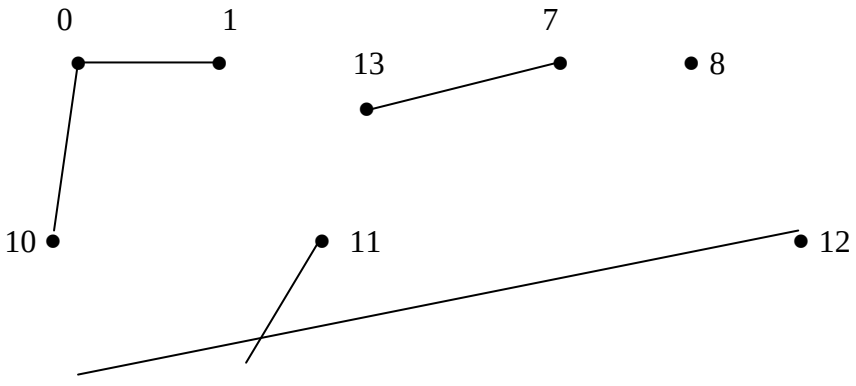
Chapitre V Applications

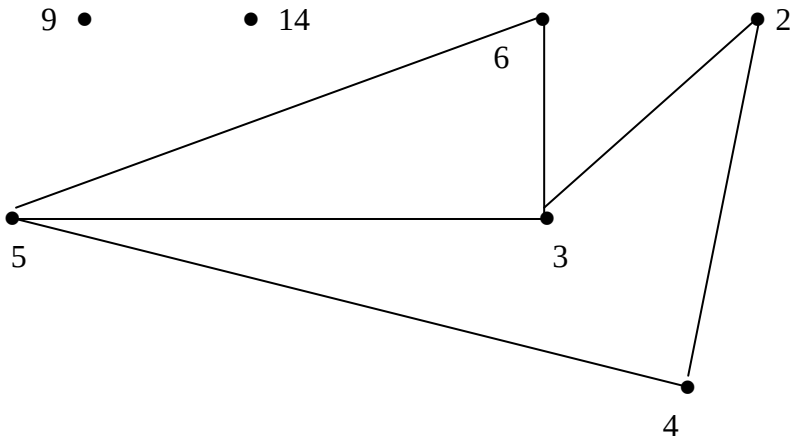
Il serait déconcertant de traiter ce sujet , dont la portée est immense sans mettre en exergue certaines applications pratiques , fussent elles les plus simples . C'est ainsi que nous avons proposé quelques exemples souvent récréatifs , mais qui pourront tout de même être modélisés sous un angle beaucoup plus général .

V-1 Application à la localisation

Une grande entreprise commerciale de la place souhaite étendre son réseau en installant de nouvelles unités dans une métropole donnée .
Pour éviter la concurrence entre les implantations , ainsi qu'une durée de stockage assez longue des produits dont certains risqueraient de périmer ; mais aussi optimiser le personnel ; les dirigeants de l'entreprise après une étude commandée dans un cabinet ont reçu comme principale recommandation de ne pas implanter deux unités à moins de 400m l'une de l'autre.
La question est alors pour ces hommes d'affaires de savoir les lieux à retenir parmi les 15 qui sont déjà ciblés dans la ville .

Considérons le graphe dont les sommets représentent l'ensemble des cibles (numérotés de 0 à 14).
Il existe un arc entre deux sommets si et seulement si les unités correspondantes sont situées à moins de 400m .
Tout ensemble indépendant maximal est alors une solution potentielle du problème .
D'ou l'application de l'algorithme , qui fournit tous les cas possibles . A nos hommes d'affaires de faire alors leurs choix .





Déroulement et mise en oeuvre de l'algorithme:

[root@localhost root]# vi matrice.c

[root@localhost root]# make matrice
cc matrice.c -o matrice

[root@localhost root]# ./matrice

Donner l'ordre du graphe:
15
Donnez la ligne numéro 1 de la matrice d'adjacence de G:
0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
Donnez la ligne numéro 2 de la matrice d'adjacence de G:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Donnez la ligne numéro 3 de la matrice d'adjacence de G:
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
Donnez la ligne numéro 4 de la matrice d'adjacence de G:
0 0 1 0 0 1 1 0 0 0 0 0 0 0 0
Donnez la ligne numéro 5 de la matrice d'adjacence de G:
0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
Donnez la ligne numéro 6 de la matrice d'adjacence de G:
0 0 0 1 1 0 1 0 0 0 0 0 0 0 0
Donnez la ligne numéro 7 de la matrice d'adjacence de G:
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
Donnez la ligne numéro 8 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
Donnez la ligne numéro 9 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Donnez la ligne numéro 10 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
Donnez la ligne numéro 11 de la matrice d'adjacence de G:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Donnez la ligne numéro 12 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Donnez la ligne numéro 13 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
Donnez la ligne numéro 14 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
Donnez la ligne numéro 15 de la matrice d'adjacence de G:
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

La matrice d'adjacence de G est:

0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 1 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

Le graphe comporte 11 arcs.

La suite des degrés, ordonnée de façon croissante est:

0 1 1 1 1 1 1 1 1 2 2 2 2 3 3

Voici les numéros des sommets du graphe, ordonnés par degré et numéro croissants:

8 1 7 9 10 11 12 13 14 0 2 4 6 3 5

Pour débuter l'algorithme, considérons le sommet de numéro 8 :

Les sommets 9 et 12 sont liés.
Les sommets 7 et 13 sont liés.
Les sommets 11 et 14 sont liés.
Les sommets 1 et 0 sont liés.
Les sommets 10 et 0 sont liés.

Les sommets 2 et 4 sont liés.
Les sommets 2 et 3 sont liés.
Les sommets 6 et 3 sont liés.
Les sommets 6 et 5 sont liés.

Les 8 sommets suivants forment un ensemble indépendant maximal:

1 2 6 7 8 9 10 11

Voulez-vous tester si un ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
0
Donner le sommet suivant:
1
Les sommets 0 et 1 sont liés.
Donner le sommet suivant:
2
Donner le sommet suivant:
3
Les sommets 2 et 3 sont liés.
Donner le sommet suivant:
4
Les sommets 2 et 4 sont liés.
Donner le sommet suivant:
5
Donner le sommet suivant:
6
Les sommets 5 et 6 sont liés.
Donner le sommet suivant:
7
Donner le sommet suivant:
8
Donner le sommet suivant:
9
Donner le sommet suivant:
10
Les sommets 0 et 10 sont liés.
Donner le sommet suivant:
11
Donner le sommet suivant:
12
Les sommets 9 et 12 sont liés.
Donner le sommet suivant:
13
Les sommets 7 et 13 sont liés.
Donner le sommet suivant:
14
Les sommets 11 et 14 sont liés.
Les 7 sommets suivants forment un ensemble indépendant maximal:
0 2 5 7 8 9 11

Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
14
Donner le sommet suivant:
13
Donner le sommet suivant:
12
Donner le sommet suivant:
11
Les sommets 14 et 11 sont liés.
Donner le sommet suivant:
10
Donner le sommet suivant:
0
Les sommets 10 et 0 sont liés.
Donner le sommet suivant:
1
Donner le sommet suivant:
2
Donner le sommet suivant:
3
Les sommets 2 et 3 sont liés.
Donner le sommet suivant:
9
Les sommets 12 et 9 sont liés.
Donner le sommet suivant:
8
Donner le sommet suivant:
7
Les sommets 13 et 7 sont liés.
Donner le sommet suivant:
4
Les sommets 2 et 4 sont liés.
Donner le sommet suivant:
6
Donner le sommet suivant:
5
Les sommets 6 et 5 sont liés.
Les 8 sommets suivants forment un ensemble indépendant maximal:
14 13 12 10 1 2 8 6
Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:
Si oui, donner le premier sommet:
8
Donner le sommet suivant:
5
Donner le sommet suivant:
2
Donner le sommet suivant:
3
Les sommets 5 et 3 sont liés.

Les sommets 2 et 3 sont liés.
Donner le sommet suivant:
6
Les sommets 5 et 6 sont liés.
Donner le sommet suivant:
9
Donner le sommet suivant:
7
Donner le sommet suivant:
1
Donner le sommet suivant:
4
Les sommets 5 et 4 sont liés.
Les sommets 2 et 4 sont liés.
Donner le sommet suivant:
11
Donner le sommet suivant:
14
Les sommets 11 et 14 sont liés.
Donner le sommet suivant:
10
Donner le sommet suivant:
13
Les sommets 7 et 13 sont liés.
Donner le sommet suivant:
12
Les sommets 9 et 12 sont liés.
Donner le sommet suivant:
0
Les sommets 1 et 0 sont liés.
Les sommets 10 et 0 sont liés.
Les 8 sommets suivants forment un ensemble indépendant maximal:
8 5 2 9 7 1 11 10

Par exemple , une solution possible est de retenir les lieux 1 2 6 7 8 9 10 11 .

<u>Remarque</u> 1
Les hommes d'affaires ont alors plusieurs choix .
Devant la console , ils peuvent tester toutes les combinaisons qui les intéresseraient .
A partir de ce moment , peuvent alors intervenir d'autres critères de choix comme la densité du milieu (naturellement le plus dense serait plus intéressant) , la sécurité de l'endroit ainsi que l'estimation du bénéfice qui y serait réalisé .

IV-2 <u>Application à la recherche opérationnelle</u>

Considérons le problème d'optimisation combinatoire suivant :

$\max Z = x_1 + x_2 + x_3 + x_4 + x_5$

$x_1 + x_2 \leq 1$

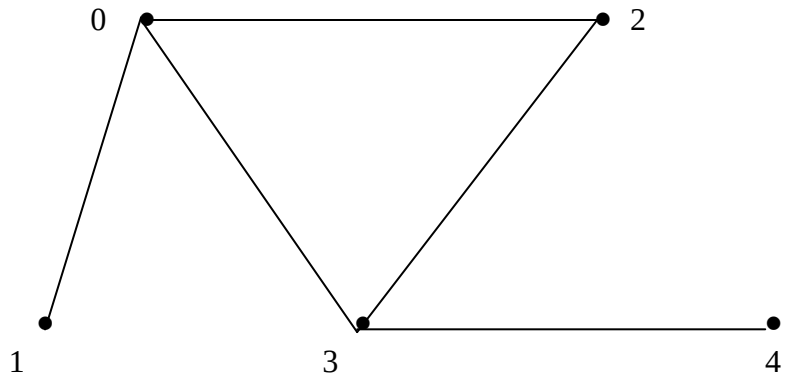
$x_1 + x_3 + x_4 \leq 1$

$x_4 + x_5 \leq 1$

où x_1 ; x_2 ; x_3 ; x_4 et x_5 sont des variables bivalentes .
Déterminons une solution de ce système :

Pour cela , on posera le problème comme celui de la recherche d'un ensemble indépendant de cardinal maximal dans un graphe que l'on déterminera .

Soit G le graphe définit de la façon suivante :
S = { 0 , 1 , 2 , 3 , 4 } représente l'ensemble des sommets .
A = { x_{01} ; x_{02} ; x_{03} x_{23} ; x_{34} } l'ensemble des arêtes du graphe .
A chaque variable x_i correspond un sommet i-1 de G .
Une arête relie deux sommets i et j (ou j et i) si et seulement si les deux variables x_i et x_j appartiennent à une même équation .
Représentation planaire du graphe défini ci-dessus :



Rechercher une solution maximisant Z revient à chercher alors un ensemble stable de cardinal maximal dans G .

Appliquons l'algorithme :

[root@localhost root]# ./matrice
Donner l'ordre du graphe:
5
Donnez la ligne numéro 1 de la matrice d'adjacence de G:
0 1 1 1 0
Donnez la ligne numéro 2 de la matrice d'adjacence de G:
1 0 0 0 0
Donnez la ligne numéro 3 de la matrice d'adjacence de G:
1 0 0 1 0
Donnez la ligne numéro 4 de la matrice d'adjacence de G:
1 0 1 0 1

Donnez la ligne numéro 5 de la matrice d'adjacence de G:
0 0 0 1 0

La matrice d'adjacence de G est:

0 1 1 1 0
1 0 0 0 0
1 0 0 1 0
1 0 1 0 1
0 0 0 1 0
Le graphe comporte 5 arcs.

La suite des degrés, ordonnée de façon croissante est:

1 1 2 3 3

Voici les numéros des sommets du graphe, ordonnés par degré et numéro croissants:

1 4 2 0 3
Pour débuter l'algorithme, considérons le sommet de numéro 1 :

Les sommets 1 et 0 sont liés.
Les sommets 2 et 3 sont liés.
Les sommets 4 et 3 sont liés.

Les 3 sommets suivants forment un ensemble indépendant maximal:

1 2 4

On trouve $S_m = \{ 1 , 2 , 4 \}$; soit $x_1 = 0$, $x_2 = 1 = x_3 = x_5$ et $x_4 = 0$.
D'où Z = 3 .

IV-3 Ensemble indépendant de coût maximum

Etant donné un graphe G; supposons qu'à chaque sommet de G , on associe un valeur réelle positive qu'on appelle coût . On définit alors une application :

$$C : S \longrightarrow IR+$$
$$i \longrightarrow c_i$$

La recherche du stable de coût maximum sera la recherche d'une sous-famille I de S stable et qui maximise le coût total $\sum_{i \in I} c_i$.

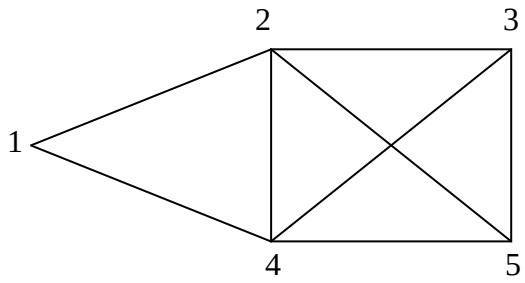
En terme de recherche d'applications entre ensembles finis discrets , le problème du stable de G se pose comme la recherche d'une application non nécessairement partout définie ,

$$f : S \longrightarrow S \quad \text{et qui maximise} \sum_{i \in S} c(f(i)) \text{ où}$$

f(i) est l'ensemble des sommets non adjacents au sommet i .
Soit A=(a_{ij}) la matrice d'incidence arêtes-sommets (deux 1 par ligne) du graphe.
Si I $\subset$ S , son vecteur caractéristique est défini par $x = (x_1,\ldots\ldots x_n)$ tel que :
$x_j = 1$ si $x_j \in I$ et $x_j = 0$ sinon .
Alors , le problème du stable de coût maximum dans G se formalisera comme la recherche de la solution du programme linéaire en nombres entiers :

$$\text{Max } z = \sum_{j=1}^{n} c_j x_j \quad \text{avec} \quad \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad ; \; i \text{ allant de } 1 \text{ à } m .$$

Exemple 1



Un ensemble stable de coût maximum du graphe ci-dessus sera la solution du programme linéaire en nombres entiers :

$$\text{Max } z = \sum_{j=1}^{n} c_j x_j$$

avec :

$x_1 + x_2 \leq 1$	$x_2 + x_5 \leq 1$		
$x_1 + x_4 \leq 1$	$x_3 + x_4 \leq 1$	où $x_j = 0$ ou 1	
$x_2 + x_4 \leq 1$	$x_3 + x_5 \leq 1$		
$x_2 + x_3 \leq 1$	$x_4 + x_5 \leq 1$		

Remarque 2
Soit C = (C_i) l'ensemble des cliques maximales de G .
A tout ensemble stable maximal du graphe , correspond une clique maximale du graphe complémentaire .
En considérant les cliques maximales de G : E={1,2,4} et F={2,3,4,5} , un ensemble stable maximum de G sera la solution du programme linéaire suivant :

$$n$$

$$\text{Max } z = \sum_{j=1} c_j x_j$$

avec : $x_1 + x_2 + x_4 \leq 1$ où $x_j = 0$ ou 1

$x_3 + x_3 + x_4 + x_5 \leq 1$

Si le polyèdre des contraintes a tous ses sommets entiers , le problème peut se résoudre par l'algorithme du simplexe .
Les matrices ayant cette propriété seront dites parfaites .
Il existe des théorèmes (notamment Padberg 1947 et Berge 1972) qui donnent des caractérisations de ces matrices . Nous n'allons pas en dire plus ; cependant la personne intéressée pourra consulter les auteurs mentionnés ci-dessus .

<u>Exemple</u> 2

Une chaîne de magasins à grande surface envisage l'implantation de nouvelles unités dans une région où elle n'en possède encore aucune .
Cette société a recensé l'ensemble des localités L = { l_1 ,……, l_p } où il serait possible d'implanter un nouveau magasin . Pour chaque localité , elle a estimé le bénéfice b(l_1) que ferait annuellement un magasin situé dans cette localité . Pour éviter la concurrence , entre deux magasins de la même chaîne, la société a décidé de ne pas implanter deux unités à moins de 40 kilomètres l'une de l'autre et les estimations des bénéfices ont été réalisées sous cette hypothèse .
On peut poser le problème du choix des localités qui maximisent le bénéfice total annuel comme la recherche d'un ensemble stable de cardinal maximal dans un graphe G défini comme suit:
S = L ; A est l'ensemble des arêtes du graphe . Pour tout couple (l_i , l_j) de L*L ; (l_i , l_j) est une arête de G si et seulement si les deux localités l_i et l_j sont situées à moins de 40km l'une de l'autre . Les localités à retenir pour implanter des magasins sont celles qui forment un ensemble stable de poids maximal . On rappelle que le poids d'un sous-ensemble de G est la somme des poids des sommets le constituant . Dans notre cas , le poids du sommet l_i est égal au bénéfice b(l_i) .

Conclusion

Ce travail revêt une importance capitale dans la résolution de certains problèmes concrets issus de la modélisation de certaines situations .
De façon générale , la détermination d'un ensemble indépendant maximal dans un graphe quelconque ne pose pas de problème .
Dans certains cas , la solution des problèmes passe par la détermination du nombre minimal de stables maximaux , aussi appelé nombre chromatique . Ce dernier peut être trouvé avec un algorithme de coloration , mais cette approche se limite aux problèmes de taille très faibles .
Ainsi , un intérêt particulier a été porté sur le plus grand stable , pour ses multiples applications . En effet , pour certaines classes de graphe comme les arbres , les graphes bipartis , et complets , des méthodes particulièrement efficaces existent pour déterminer ce sous-ensemble et son cardinal .
En outre , l'importance de la notion d'ensemble indépendant maximal , est telle , qu'on a cherché à l'implémenter dans le langage C , pour que désormais , la détermination d'un stable maximal se fasse de façon automatique .
Une autre dimension importante du problème , et qui est prise en compte par le programme est de pouvoir vérifier l'existence de la propriété sur un sous-ensemble à préciser .
Ce thème , tel que donc prouvé par ses applications se trouve au carrefour de la théorie des graphes , de l'optimisation combinatoire et de la localisation .
Cela veut dire que les frontières entre les branches de l'optimisation ne sont pas hermétiques , mais elles expriment plutôt des orientations méthodologiques différentes .
Ce travail nous a conduit à réfléchir sur un autre problème qui est la détermination d'un ensemble stable de coût maximum qui sera lui résolu par la recherche opérationnelle , grâce à la programmation linéaire , suite à l'introduction de la notion d'hypergraphe .
En fait , ses applications pratiques sont très diverses : Optimisation des réseaux de transport , (transport routier ou transport d'information) ; conception de réseaux électriques , de réseaux de communication , ordonnancement des travaux etc…..
Le champ de la recherche en théorie des graphes est immense et est motivé par des questions d'ordre très pratiques et de véritables sujets de thèses peuvent être posés sous forme de «jeux Mathématiques » .

Annexe

Présentation du programme en C de la détermination d'un ensemble indépendant maximal :

La structure de données que l'on va utiliser sera la matrice d'adjacence sommets-sommets du graphe G .
On suppose que G est un 1-graphe d'ordre n .
On saisit la matrice M d'ordre n (par souci de simplicité et pour être conforme à la notation du langage C concernant les matrices , les sommets de G sont supposés numérotés de 0 à n-1) , c'est à dire n lignes et n colonnes .
Si les sommets i et j sont reliés par un arc , c'est à dire si (i , j) est dans A, alors on met la valeur 1 à l'intersection de la ligne i et de la colonne j de la matrice M
Sinon , on met la valeur 0 . On saisit M , ligne par ligne puisque telle est la façon dont le compilateur comprend une matrice à plusieurs dimensions .
Le symbole /*……*/ correspond à des commentaires en langage C et que n'est donc pas interprété par le compilateur . Il permet simplement au lecteur de bien comprendre ce que font les instructions .
Le degré d'un sommet est le nombre d'arcs qui lui sont incidents.
C'est donc le nombre d'arcs dont le sommet en question constitue une extrémité
La suite des degrés pourra être vue comme une matrice ligne formée des degrés des sommets : (d[0],d[1], ….,d[n-1]) ; d[i] (i=0 à n-1) est le degré du sommet i . Donc pour chaque sommet i , son degré d[i] est le nombre de valeurs non nulles (ou le nombre de 1) se trouvant dans la matrice M , en parcourant la ligne i .
On crée ainsi la matrice ligne (d[0],d[1], …..,d[n-1]) .
On suppose qu'on dispose déjà de la matrice d'adjacence du graphe .
On parcourt la ligne i de la matrice d'adjacence colonne par colonne et à chaque fois qu'on rencontre la valeur 1 , on incrémente la valeur qui se trouve à l'emplacement d[i]
(correspondant au degré du sommet i) .
Maintenant , on range la suite des degrés de façon croissante .
Avant cela on , crée une nouvelle ligne D de même taille que d ; dans laquelle on garde les valeurs originales de la suite d .
Ensuite, on crée un tableau I d'ordre n contenant les numéros des sommets indépendants .

Programme global

```
#define ord_max 100        /*On travaille avec des graphes ayant au plus 100
                                                                    sommets*/
main()
{
/*Déclaration des variables*/
int i,j,n,nbre_arcs,temp,p,r,m,k,s,t,q ;
/*Déclaration des tableaux*/
int M[ord_max][ord_max],d[ord_max],D[ord_max],I[ord_max],E[ord_max],
F[ord_max],G[ord_max],H[ord_max],L[ord_max],N[ord_max];
/*Définition de la matrice d'adjacence de G*/
printf(''Donner l\'ordre du graphe:\n'') ;
          scanf(''%d'',&n) ;
/*Initialisation de la matrice d'adjacence M*/
       for(i=0 ;i<ord_max ;i++)
            for(j=0;j<ord_max;j++)
                 M[i][j]=-1;
/*Saisie de la matrice*/
        for(i=0 ;i<ord_max ;j++)
             {
printf(''Donner la ligne numéro %d de la matrice d\'adjacence :\n'',i+1) ;
              for(j=0 ;j<n ;j++)
                    scanf(''%d'',&M[i][j]);
              }
/*Affichage de la matrice*/
printf(''La matrice d\'adjacence de G est:\n'') ;
printf(''\n'');   /*Saut de ligne*/
       for(i=0 ;i<ord_max ;i++)
            for(j=0;j<ord_max;j++)
                 printf(''%d'',M[i][j]);
printf(''\n'');
/*Calcul du nombre d'arcs dans le graphe*/
   nbre_arcs=0 ;
         for(i=0 ;i<ord_max ;i++)
              for(j=0;j<ord_max;j++)
                   nbre_arcs=nbre_arcs+M[i][j];
/*Affichage du nombre d'arcs.*/
printf(''Le graphe comporte %d arcs.\n'',nbre_arcs/2) ;
printf(''\n'') ;
/*Initialisation des tableaux*/
          for(i=0 ;i<ord_max ;i++)
             {
                 d[i]=0;
                 D[i]=0;
                 I[i]=-1;
                 E[i]=-1;
                 F[i]=-1;
                 G[i]=-1;
                 H[i]=-1;
```

```
                L[i]=-1;
                N[i]=-1;
            }
/*Définition de la suite des degrés.*/
        for(i=0 ;i<n ;i++)
            for(j=0;j<n;j++)
                d[i]=d[i]+M[i][j];
/*Conserver les éléments du tableau d dans le tableau D*/
        for(i=0 ;i<n ;i++)
            D[i]=d[i];
/*Affichage de la suite des degrés.*/
printf(''La suite des degré de G est:\n'') ;
         for(i=0 ;i<n ;i++)
            printf('' %d '',d[i]);
printf(''\n'');
/*On range la suite de façon croissante.*/
         for(i=0 ;i<n ;i++)
            for(j=i+1;j<n;j++)
                if(d[i]>d[j])
                    {
                        temp=d[i];
                        d[i]=d[j];
                        d[j]=temp;
                     }
/*Affichage de la suite ordonnée*/
printf(''La suite des degrés , ordonnée de façon croissante est : \n'') ;
printf(''\n'') ;
          for(i=0 ;i<n ;i++)
              printf('' %d '',d[i]);
printf(''\n'');
/*Détermination des sommets dont le degré est le plus petit*/
           for(i=0 ;i<n ;i++)
               if(D[i]==d[0])
                   I[i]=i;
/*Affichage des sommets de plus faibles degrés*/
printf(''Le(s) sommet(s) suivant est(sont) de plus faible(s) degré(s):\n'') ;
printf(''\n'') ;

                for(i=0 ;i<n ;i++)
                    if(I[i]!= -1)
                        printf('' %d '',I[i]);
 printf(''\n'');
/*On ordonne de façon croissante le(s) sommet(s) de plus faible(s) degré(s)*/
                for(i=0 ;i<n ;i++)
                    for(j=i+1;j<n;j++)
                        if(I[i]>I[j])
                    {
                        temp=I[i];
                        I[i]=I[j];
                        I[j]=temp;
```

```
                        }
/*Pour trouver le nombre de sommets de plus faibles degrés*/
    p=0 ;
          for(i=0 ;i<n ;i++)
              if(I[i]>=0)
                  p=p+1;
/*Affichage des numéros de sommets de plus faibles degrés , ordonnés de façon croissante par numéro de sommet*/
printf(‘’Voici les sommets de plus faibles degrés , ordonnés de façon croissante par numéro de sommet : Ils sont au nombre de %d\n’’,p) ;
printf(‘’\n’’) ;
              for(i=0 ;i<n ;i++)
                   if(I[i]!=-1)
                        printf(‘’ %d ’’,I[i]);
printf(‘’\n’’);
/*Affichage des numéros de sommets par ordre de degré croissant*/
/*On supprime les redondances dans la suite d*/
               for(i=0 ;i<n ;i++)
                    if(d[i]==d[i+1])
                         d[i]=-3;
printf(‘’Voici les numéros de sommet du graphe , ordonnés par degré croissant:\n’’);

               for(i=0 ;i<n ;i++)
                         for(j=0 ;j<n ;j++)
                              if(D[j]==d[i])
                                   printf(‘’ %d ‘’,j);
printf(‘’\n’’);
/*Affichage du sommet de plus petit numéro parmi les sommets de plus faibles degré*\
printf(‘’Considérons le sommet de numéro %d:\n’’,I[n-p]) ;
printf(‘’\n’’) ;
/*On range de nouveau les sommets de plus faibles degrés dans le tableau L*/
F[0]=I[n-p] ;
              for(i=n-p;i<n;i++)
                   L[i+p-n]=I[i];
/*printf(‘’Les sommets de plus faibles degré par numéro croissant:\n’’);
               for(i=0;i<p;i++)
                    printf(‘’ %d ‘’,L[i]);
printf(‘’\n’’); */
/*On détermine les sommets indépendants*/
               for(i=1;i<p;i++)
                    if(M[L[i]][L[0]]= = 0 )
                         F[i]=L[i];
                else
                   F[i]=-1;
/*Affichage
               for(i=1;i<p;i++)
                    if(F[i]>=0)
                         printf(‘’ %d ‘’,F[i]);
printf(‘’\n’’); */
/*Recopie dans N des sommets indépendants parmi ceux de plus faibles degré*/
```

```
                    for(i=0;i<p;i++)
                        if(F[i]>=0)
                            N[F[i]]=F[i];
/*On veut appliquer l'algorithme pour déterminer un ensemble indépendant maximal*/
  for(i=0;i<n;i++)
       for(j=0;j<n;j++)
         if((D[j]= = d[i])&&(j!=L[0]))
          {     if(M[L[0]][j]= = 0)
              {   N[j]=j;
                    for(q=0;q<n;q++)
                       if((M[N[q]][j]= =1)&&(N[q]>=0]))
                         {  printf(''Les sommets %d et %d sont liés.\n'',N[q],j) ;
                            N[j]=-2 ;
                         }
                }
 else
  printf(''Les sommets %d et %d sont liés.\n'',L[0],j) ;
           }
/*Affichage de l'ensemble indépendant maximal construit. On calcule le nombre de sommets
concernés.*/

      for(i=0 ;i<n ;i++)
         if(N[i]>=0)
            m=m+1;
printf(''Les %d sommets suivant forment un ensemble indépendant maximal.\n'',m) ;
printf(''\n'') ;
        for(i=0 ;i<n ;i++)
           if(N[i]>=0)
               printf('' %d '',N[i]);
printf(''\n'');
/*On veut tester si un ensemble de sommet est indépendant maximal.*/
printf(''Voulez-vous tester si un ensemble de sommets est indépendant maximal.\n'') ;
for(k=0 ;k<n ;k++)
{    printf(''Si oui , donner le premier sommet:\n'');
           scanf(''%d'',&E[0]) ;
              G[0]=E[0];
    for(i=0;i<n;i++)
   {     printf(''Donner le sommet suivant:\n'');
         if(M[G[0]][E[i]]= = 0)
        {       G[i]=E[i];
                  for(j=0;j<n;j++)
                     if((G[j]>=0)&&(M[G[j]][E[i]]= = 1))
                       {  G[i]=-2;
                          printf(''Les sommets %d et %d sont liés:\n'',G[j],E[i]);
                        }
         }
           else
              printf(''Les sommets %d et %d sont liés:\n'',G[0],E[i]);
    }
/*Affichage du résultat du test . On calcule d'abord le nombre de sommets concernés.* /
```

```
          s=0 ;
           for(q=0;q<n;q++)
             if(G[q]>=0)
                s= s+1;
printf(‘’Les %d sommets suivant forment un ensemble indépendant maximal.\n’’,s) ;
           for(t=0;t<n;t++)
             if(G[t]>=0)
                printf(‘’ %d ‘’,G[t]);
printf(‘’Voulez-vous tester si un autre ensemble de sommets est indépendant maximal:\n’’);
/*Initialiser les suites G et E avant le prochain tour de boucle*/
   for(r=0;r<n;r++)
        {  G[r]= -7*n;
           H[r]= -5*n;
        }
printf(‘’\n’’);
}
```

Bibliographie

[1] GONDRAN , Michel et MINOUX , Michel Graphes et Algorithmes éd. Eyrolles , Paris (1978) .

[2] Labelle , Jacques , théorie des graphes éd.Modulo , Québec (1981).

[3] ROY , Bernard . Algèbre moderne et Théorie des graphes orientés vers les sciences économiques et sociales . Tome1 : Notions et résultats fondamentaux ed.Dunod , Paris (1969)

[4] ROY , Bernard . Algèbre moderne et Théorie des graphes orientés vers les sciences économiques et sociales . Tome2 : Applications et problèmes spécifiques ed.Dunod , Paris (1970)

[5] (barnier , brisset)@recherche.enac.fr

[6] http://www.laas.fr/~lopez/cours/GRAPHES/graphes.html

[7] http://www.labri.fr/~courcell/SafeMail.php

[8] Claude Berge , Graphes et hypergraphes , Bordas 1973 (300 pages).

[9] Nguyen Huy , Mathématiques discrètes et informatique , Masson , 1997 .

[10] Aimé Sache , la Théorie des graphes , Que Sais-je ? , 1974 , réédition prévue en 2004 chez Cassini .

[11] M.Kaufmann , Des points et des flèches , la théorie des graphes , Dunod Sciences-poche , épuisé .

[12] Alan Gibbons , Algorithmic graph theory ,Cambridge University Press 1985

[13] J.Korner , B.Larose , C.Malvenuto , Maximal stable sets in multinomial Kneser-type

[14] http://www.tcs.mu-lueberk.de/