République Algérienne Démocratique et Populaire

Université Saad Dahlab Blida

Faculté des sciences

Département des Mathématiques

Domaine Mathématique

Mémoire de Master

Option :

Recherche Opérationnelle

Thème :

Problèmes du stable maximum et stable de poids maximum : Complexité et résolution

Présenté par :

KHEDDAOUI Leila

Devant le jury composé de :

Pr. BOUDHAR Mourad	**Président**	**USTHB**
Dr. AMROUCHE Karim	**Examinateur**	**Alger 3**
Dr. BENDRAOUCHE Mohamed	**Promoteur**	**USDB1**

2016/2017

Remerciements

Grâce à Allah, nous avons abouti à la concrétisation de ce travail.

En préambule à ce mémoire, je souhaite adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

En particulier, à mon promoteur Mr BENDRAOUCHE Mohamed pour avoir accepté de m'encadrer tout au long de ce travail, pour son amabilité, sa disponibilité, son aide, ses conseils et suggestions et le temps qu'il a bien voulu me consacrer malgré ses charges académiques et professionnelles.

Je remercie également les membres du Jury d'avoir accepté d'examiner mon mémoire.

Que tout enseignant m'ayant fait bénéficier de son savoir durant tout mon cursus universitaire, trouve ici l'expression de ma profonde gratitude, en particulier le chef de Département de Maths , ainsi que le Doyen de notre Faculté.

Je n'oublie pas ma famille, et spécialement mes parents pour leur contribution, leur soutien et leur patience.

Enfin, je ne serai oublier dans ces remerciements tous ceux qui m'ont aidé pour mener à bien ce travail dans de bonnes conditions.

Dédicaces

Je dédie ce modeste travail

À mes très chers parents que Dieu les garde

En témoignage de ma profonde gratitude et mon incontestable reconnaissance, pour leurs sacrifices, leur confiance qu'ils m'accordent et tout l'amour dont ils m'entourent.

À mes sœurs « Fadhia » « Safia » « Lobna » et « Siham » et mes frères « Youcef » « Aissa» « Houcin » et « Sidali » mon époux « Rédha ».

À toute ma famille et ma belle-famille.

Et à tous ceux qui me sont chers.

Que Dieu vous garde.

Résumé

Dans ce mémoire, nous avons étudié les problèmes du stable maximum (MIS) et stable de poids maximum (MWIS). Après avoir donné les concepts de base de la théorie des graphes et de la complexité, des définitions et motivations ainsi que des propriétés de ces deux problèmes ont été présenté. Un état de l'art de quelques cas polynomiaux est reporté. Comme dans le cas général, ces problèmes sont NP-difficiles, et par suite nous avons présenté des algorithmes d'approximation et des bornes inférieures et supérieures pour leurs résolutions.

Abstract

In this memory, we have studied the problems of Maximum Independent Set (MIS) and Maximum Weighted Independent Set (MWIS). After providing the basics of graph theory and complexity, some definitions and motivations as well as properties of these problems have been presented. A literature review of some polynomial cases is reported for both problems. As in the general case these two problems are NP-hard, some approximation algorithms as well as some lower and upper bounds are devised for the resolution of these problems.

Table des matières

Liste des figures et des tableaux

Introduction Générale

La théorie des graphes a connu un essor spectaculaire ces dernières années, en partie due aux importances entre les aspects théoriques et algorithmiques du domaine et les nouveaux enjeux scientifiques. Les graphes jouent un rôle prépondérant dans la recherche en sciences et technologies de l'informatique, ainsi qu'en bio-informatique par exemple.

Le problème du stable maximum ou (maximum independent set problem (MIS), en anglais), est un problème d'optimisation qui consiste, étant donné un graphe non orienté G=(V, E) à trouver un stable de cardinal maximum. Autrement dit on cherche un sous-ensemble de sommets du graphe, le plus grand possible, tel que les éléments de ce sous-ensemble ne soient pas voisins.

Prenons l'exemple du graphe de la Figure 1. Celui-ci représente l'organisation d'un dîner où les invités potentiels sont Adam, Mohamed, Réda, Mostapha, Omar et Ali. Les sommets sont les invités potentiels. Mais Adam et Mohamed ne s'apprécient pas, Adam et Réda ne s'apprécient pas, Mohamed et Mostapha ne s'apprécient pas, etc. Il y a une arête entre deux personnes qui ne s'apprécient pas. L'objectif est d'inviter le maximum de personnes qui s'apprécient mutuellement. Il s'agit donc de trouver un stable maximum dans ce graphe.

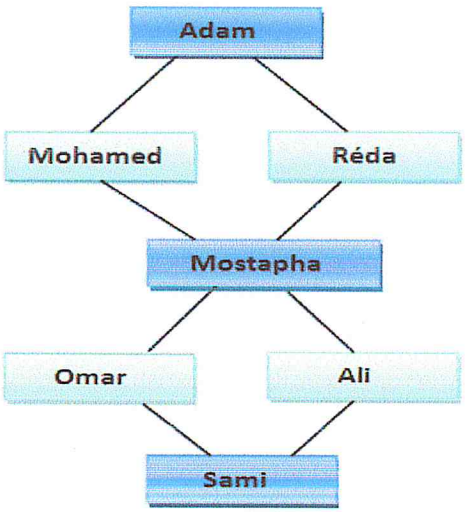


Figure 1 – Exemple d'organisation d'un dîner.

Dans la vie pratique, il existe des situations plus générales que le problème du stable maximum. Dans de tels cas, le graphe G est muni d'une pondération en associant à chaque sommet v de G un entier ω(v) appelé le poids de v. Le poids P(Y) d'un sous-ensemble Y de V est alors égal à la somme des poids des sommets de Y.

Le problème du stable maximum dans un graphe pondéré G , en anglais Maximum Weighted Independent Set (MWIS) consiste à déterminer un stable de poids maximum dans G. Dans la littérature, il existe aussi une piste de recherche, qui s'intéresse à la détermination des nombres $\alpha(G)$ et $\alpha_\omega(G)$, qui sont la cardinalité d'un stable maximum (pour MIS) et le poids d'un stable de poids maximum (pour MWIS).

Parfois, on a la chance de disposer d'algorithmes « efficaces » (c'est-à- dire qui exécutent un nombre d'instructions majoré par un polynôme en la taille du problème, avant de se terminer) pour résoudre un problème donné. Par exemple, le problème du couplage maximum peut être résolu en temps polynomial grâce à l'algorithme d'Edmonds [17].

Mais très souvent, et c'est le cas pour les problèmes MIS et MWIS, la difficulté vient justement de ce qu'un tel algorithme n'existe pas (ou du moins, la communauté s'accorde désormais à penser qu'il ne peut exister). On parle de problèmes NP-difficiles, et on se tourne alors vers d'autres solutions, comme les algorithmes d'approximation.

Ce mémoire comporte cinque chapitres et est organisé comme suit. Dans le premier chapitre, nous rappelons toutes les définitions de théorie des graphes nécessaires à une bonne compréhension de la suite du mémoire, en particulier nous redonnons les notions d'ensemble stable maximum, et certaines classes de graphes que nous rencontrerons souvent dans les chapitres suivants. Le second chapitre est un bref aperçu de la théorie de la complexité des algorithmes en insistant sur les notions fondamentales des classes de problèmes P et NP. Les trois chapitres suivants concernent les problèmes du stable maximum (MIS) et du stable de poids maximum (MWIS), rapports de performance respectivement.
Comme nous l'avons mentionné ci-dessus, ces problèmes ayant de très nombreuses applications pratiques, mais ils sont NP-difficiles en général. Pour cela, nous avons commencé par donner un état de l'art pour ces problèmes, où nous avons cité quelques cas polynomiaux. Dans les cas généraux, des méthodes d'approximations ainsi que des bornes inférieures et supérieures des deux paramètres cités ont été proposées. La conclusion termine ce mémoire.

Chapitre 1

Notions de base en théorie des graphes

Théorie des graphes

1. Concepts de base

Premières Définitions

Intuitivement un graphe (graph) est un ensemble de points ou sommets (vertices) (que nous supposerons toujours non vide et fini dans la suite) dont certaines paires sont reliées, formant ainsi les extrémités (endpoints) d'une arête (edge). Plus formellement :

Définition 1.1 Un graphe G est défini par la donnée de deux ensembles :
- $V = \{v_1, v_2, \dots, v_n\}$ est l'ensemble des sommets de G
- $E = \{e_1, e_2, \dots, e_m\}$ est l'ensemble d'arêtes de G tel que chaque arête est une paire de sommets distincts de G. Le graphe G est alors noté $G = (V, E)$.

Le nombre de sommets $|V|$, est généralement noté n, le nombre d'arêtes est quant à lui noté m.

Si e = {u, v}, u et v sont appelés extrémités de e. Deux sommets u et v sont dit adjacents s'il existe une arête ayant pour extrémités u et v. Nous disons également que deux arêtes sont adjacentes si elles possèdent une extrémité commune.

Notons que la définition 1.1 laisse la possibilité pour une arête d'avoir ses deux extrémités identiques ; une telle arête est appelée boucle (loop).
De plus, pour de nombreuses applications il peut être utile d'avoir plusieurs arêtes ayant mêmes extrémités ; on parle alors d'arêtes multiples (multiple edges).
Un graphe ayant des arêtes multiples est un multi-graphe (multi-graph). Un graphe ne contenant ni boucle ni arêtes multiples est qualifié de simple (simple graph).

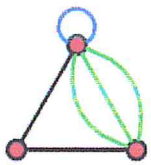


Figure 1.1 – Un exemple de multi-graphe avec des arêtes multiples (en vert) et une boucle (en bleu).

Il est également courant de rencontrer des graphes pour lesquels on distingue l'extrémité initiale de l'extrémité terminale des arêtes, c'est-à- dire que l'on ne considère plus des paires mais des couples de sommets, on parle alors d'arcs (arcs) et non plus d'arêtes. Un tel graphe est qualifié d'orienté (directed graph ou digraph) (figure 1.2). Dans la suite, sauf mention contraire explicite, tous les graphes que nous considérerons seront simples et non orientés.

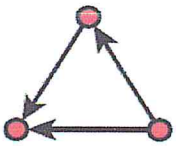


Figure 1.2 – Un exemple de graphe orienté.

De plus, il est courant d'attribuer un nom ou un numéro aux sommets d'un graphe. Un tel procédé est appelé étiquetage (labeling). Ainsi, nous parlerons souvent «du sommet u » ou «du sommet v » d'un graphe. Ceci nous autorise donc à parler de l'arête $\{u, v\}$ pour évoquer, si elle existe, l'arête ayant pour extrémités les sommets u et v, afin d'alléger les notations, nous nous autoriserons même à écrire plus simplement «l'arête uv».

Enfin, nous serons fréquemment amenés à considérer des graphes pondérés (weighted graphs), c'est-à-dire muni d'une fonction de poids sur les sommets et/ou les arêtes.

Voisinage et degré

Deux sommets v et w formant les extrémités d'une même arête sont dit adjacents (adjacent vertices) ou voisins (neighbors), ce que l'on note $v \sim w$ (et $v \nsim w$ dans le cas contraire). On définit alors le voisinage d'un sommet :

Définition 1.2
-Le voisinage (ouvert) ((open) neighborhood) d'un sommet v, noté $N(v)$, est l'ensemble des sommets qui lui sont adjacents.
- Le voisinage fermé (closed neighborhood) d'un sommet v, noté $N[v]$, est égal à $N(v) \cup \{v\}$.
Un sommet n'ayant aucun voisin est qualifié d'isolé (isolated vertex). On définit également le voisinage d'un ensemble de sommets :
Définition 1.3 Le voisinage (ouvert) d'un ensemble de sommets S, noté $N(S)$, est l'ensemble $\{x \notin S | \exists y \in S, xy \in E\}$.

Définition 1.4 Le degré (degree) d'un sommet v, noté $d(v)$, est égal au nombre$|N(v)|$de ses voisins.

Le plus petit degré d'un graphe G est noté $\delta(G)$ et le plus grand degré est noté $\Delta(G)$, ils sont respectivement égaux au degré d'un sommet ayant le moins et le plus de voisins, $\overline{d_G}$ le degré moyen de G.

Lemme 1.5 ((des poignées de mains) (hand-shaking lemma)) Dans un (pseudo) graphe,

$$\sum_{v \in V} d(v) = 2m$$

Preuve : Il suffit de constater que chaque arête compte pour deux degrés (un pour chacune de ses extrémités).

Connexité

Définition 1.6 Un graphe est connexe (connected) si pour toute paire de sommets il est possible de passer de l'un à l'autre par une suite de sommets adjacents.

La connexité définit une relation d'équivalence sur l'ensemble des sommets, et chacune des classes d'équivalence est appelée composante connexe (connected component) du graphe.

Exemple 1.7
Soit le graphe (G) suivant

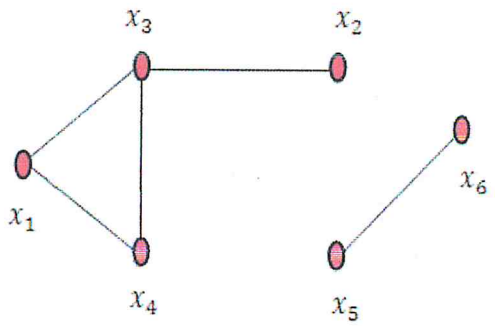


Figure 1.3 – Exemple d'un graphe non connexe.

- l'ensemble $\{x_1, x_2, x_3, x_4\}$ forme la 1ère composante connexe, on la note C_1.

- l'ensemble $\{x_5, x_6\}$ forme la 2ième composante connexe, on la note C_2. On constate que les sommets de C_1 n'ont pas relies avec les sommets de C_2.

Autrement dit, un graphe est connexe si et seulement s'il ne contient qu'une seule composante connexe. Intuitivement, le nombre de composantes connexes correspond au nombre de « morceaux » du graphe quand on le dessine. Un résultat classique énonce que

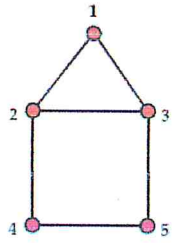
Lemme 1.8 [49] Pour tout graphe G, G est connexe ou $\bar{G}$ est connexe.

Définition 1.9 Un sommet (resp. ensemble) d'articulation (cutvertex (resp. cutset)) est un sommet (resp. ensemble de sommets) tel que sa suppression rend le graphe non connexe.

2. Représentations des graphes en machine

Matrice d'adjacence

Définition 1.10 La matrice d'adjacence (adjacency matrix) d'un graphe d'ordre n est la matrice carrée A de taille $n * n$ telle que $a_{ij} = 1$ s'il existe une arête entre les sommets i et j, et $a_{ij} = 0$ sinon.
Notons que la matrice d'un graphe non orienté est par conséquent symétrique. De plus, une matrice d'adjacence dépend de la numérotation des sommets du graphe qu'elle représente. Ainsi la notion de matrice d'adjacence n'a-t-elle de sens que pour un graphe étiqueté, et l'on parle alors bien de la matrice d'adjacence du graphe.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 1.4 – Un graphe (la maison) et sa matrice d'adjacence.

Matrice d'incidence

Définition 1.11 La matrice d'incidence (incidence matrix) d'un graphe d'ordre n est la matrice B de taille n × m (m désignant comme vu plus haut le nombre d'arêtes du graphe) telle que
bij = 1 si le sommet i est une extrémité de l'arête j, 0 sinon.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 1-2 & 1-3 & 2-3 & 2-4 & 3-5 & 4-5 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Figure 1.5 – La matrice d'incidence du graphe de la figure 1.

Listes d'adjacence

Définition 1.12 Une liste d'adjacence (adjacency list) est une structure de données dans laquelle on associe à chaque sommet sa liste de voisins.

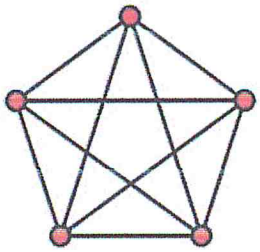
$$\begin{aligned} &1 \to 2 \to 3 \\ &2 \to 1 \to 3 \to 4 \\ &3 \to 1 \to 2 \to 5 \\ &4 \to 2 \to 5 \\ &5 \to 3 \to 4 \end{aligned}$$

Figure 1.6 – Représentation par listes d'adjacence du graphe de la figure 1.

3. Graphes particuliers

Graphes complets.

Définition 1.13 Le graphe complet (complete graph) d'ordre n, noté K_n, est le graphe où chaque paire de sommets sont adjacents deux à deux.



K_5

Figure 1.7 Graphe complet d'ordre 5.

Graphes réguliers.

Définition 1.14 Un graphe est dit régulier (regular graph) si tous les sommets ont le même degré.
Si le degré commun est k, alors le graphe est dit k-régulier.

Graphes cubiques.

Définition 1.15 Un graphe cubique (cubic graph) est un terme équivalent de graphe 3-régulier.

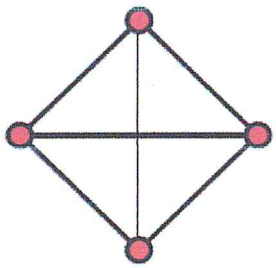


Figure 1.8 – Un exemple de graphe 3-régulier (cubique).

Graphes complémentaires.

Quand on recherche les propriétés d'un graphe, il est parfois plus simple d'étudier son complémentaire :

Définition 1.16 Le complémentaire (complement) d'un graphe $G = (V, E)$ est le graphe noté $\overline{G} = (V, \overline{E})$ tel que $uv \in E_{\bar{G}}$ ssi $uv \notin E_G$

C'est-à-dire : L'ensemble des sommets de $\overline{G}$ est le même que G. Une arête appartient au graphe complémentaire $\bar{G}$ si elle n'appartient pas au graphe initial G.

Définition : Si G=(V, E) et G'=(V, E') le graphe G∪G'=(V, $E \cup E'$).

Conséquence 1.17 G∪$\bar{G}$ = (V, $E \cup \overline{E}$) est un graphe complet.

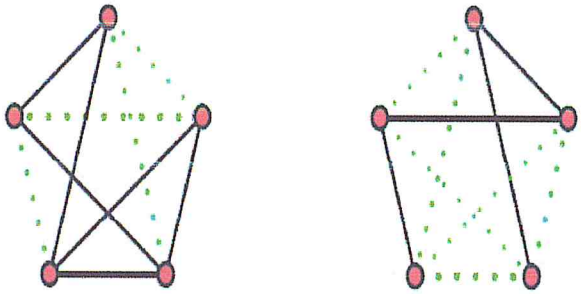


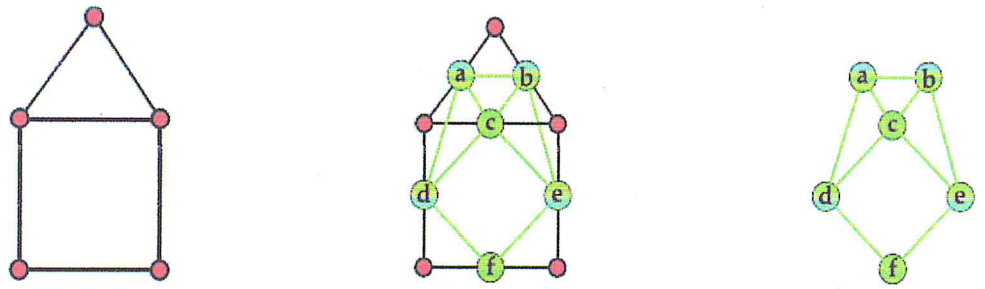
Figure 1.9 Un graphe et son complémentaire.

Graphes adjoints.

Définition 1.18 Le graphe adjoint (line graph) d'un graphe $G = (V, E)$ est le graphe noté $L(G)$ tel que :

- $V_{L(G)}$=E
- deux sommets de $L(G)$ sont adjacents si et seulement si les arêtes correspondantes dans G sont adjacentes.

La figure 1.10 illustre cette construction. Par définition, tout graphe possède un graphe adjoint; mais un graphe n'est pas forcément le graphe adjoint d'un autre graphe. Considérons par exemple le graphe de la figure 1.11, appelé griffe (claw) que nous rencontrerons souvent par la suite. Appelons **a** le sommet central, b, c et d les trois autres sommets, et essayons de reconstruire un graphe G dont la griffe serait le graphe adjoint.



(a) Graphe G (b) On relie les sommets correspondant à des arêtes adjacentes (c) Graphe $L(G)$

Figure 1.10 – Un graphe et la construction de son graphe adjoint.

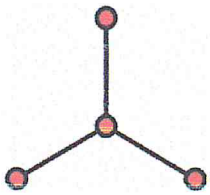


Figure 1.11 – La griffe.

Dans G, a est une arête, dont les extrémités peuvent être notées u et v. Comme a est adjacent à b, on peut supposer que ces arêtes ont pour extrémité commune u dans$G$, de même a et c ont une extrémité commune dans G, mais ce ne peut être u puisque$b$et c n'ont pas d'extrémité commune, c'est donc v. On voit alors qu'il n'est pas possible que l'arête d soit adjacente à l'arête a sans être adjacente ni à l'arête b ni à l'arête c. La griffe ne peut donc pas être le graphe adjoint d'un graphe.

En 1970, Beineke [1] a caractérisé les graphes adjoints, en donnant la liste complète des graphes minimaux qui ne sont pas des graphes adjoints, au nombre de neuf (et qui comprend donc la griffe); autrement dit, tout graphe qui n'est pas le graphe adjoint d'un autre graphe contient un de ces neuf graphes.

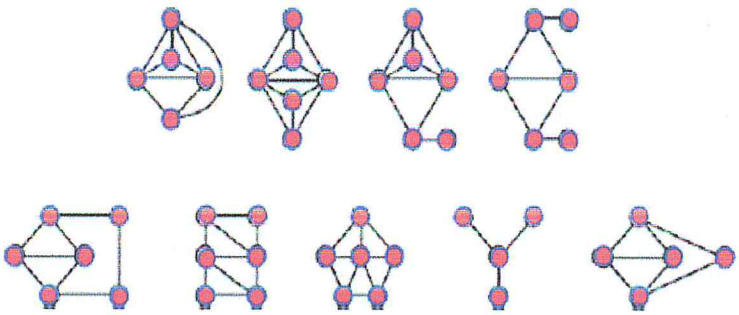


Figure 1-12 Graphes minimaux qui ne sont pas des graphes adjoints.

Il existe également un graphe qui peut être le graphe adjoint de plusieurs graphes : c'est le triangle (un graphe constitué de trois sommets adjacents deux à deux). Il est en effet facile de vérifier que le triangle est à la fois le graphe adjoint de la griffe et son propre graphe adjoint. Un résultat fondamental dû à Whitney en 1932 [2] affirme qu'il s'agit là d'une exception, et que dans tous les autres cas, il est possible de reconstruire un graphe à partir de son graphe adjoint.

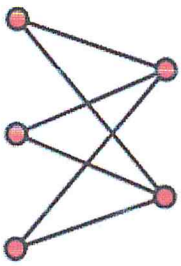
Graphes bipartis.

Définition 1.19 Un graphe biparti G (bipartite graph) est un graphe dont on peut partitionner l'ensemble des sommets en deux ensembles A et B non vides tels que toute arête a une extrémité dans A et l'autre dans B. Autrement dit de sorte que A et B sont des stables de G (sous ensemble de sommets de G deux à deux non adjacents).

On note alors $G = (A, B, E)$.

Définition 1.20 Un graphe biparti complet est un graphe biparti $G = (A, B, E)$ où tout sommet de A est adjacent à tout sommet de B. On le note $K_{m,n}$ où $m = |A|$ et $n = |B|$.

Définition 1.21 Un graphe biparti est dit bi-régulier si tous les sommets de A ont le même degré, et si tous les sommets de B ont le même degré.



$k_{3,2}$

Figure 1.13 – Un exemple de graphe biparti complet (bi-régulier).

Graphes k-partis.

Définition 1.22 Un graphe G est appelé k-parti (k-partite graph) si l'ensemble de ses sommets peut être partitionné en k stables $V_1, V_2, \ldots, V_k$. Les termes graphe biparti et triparti sont utilisés pour décrire les graphes k -partis pour k égal 2 et 3, respectivement. Un graphe k -partis est appelé complet si tout sommet $v \in V_i$ est adjacent à tous les sommets de V_j pour toute paire i , j tel que i $\neq j$. Le symbole $K_{n1,n2,\ldots,nk}$ est utilisé pour décrire le graphe k-partis complet, avec des tailles de partition égales à $|V_i| = n_i$ pour $i = 1,2, \ldots$,k.

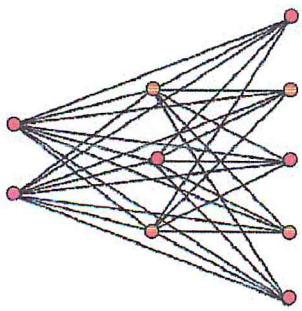
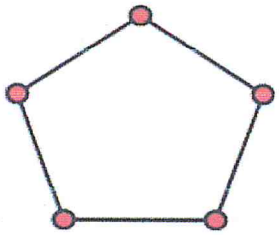


Figure 1.14 – Un graphe k-parti.

Cycles et Chaines :

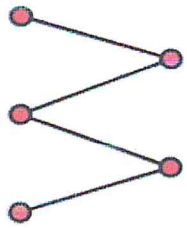
Définition 1.23 Le cycle (cycle) d'ordre n, noté C_n, est le graphe connexe dont tous les sommets sont de degré 2.



C_5

Figure 1.15 – Un exemple de cycle d'ordre 5.

Définition 1.24 La chaîne (path) d'ordre n, noté P_n, est le graphe connexe dont tous les sommets sont de degré 2, sauf les deux extrémités de la chaîne qui sont de degré 1.



P_5

Figure 1.16 – Un exemple de chaîne d'ordre 5.

La longueur d'un cycle ou d'une chaîne est le nombre d'arêtes qu'il/elle contient.

Exemple 1.25

Le graphe ci-dessous contient entre autres les chaînes $(v_1,e_1,v_2,e_2,v_3,e_5,v_5)$ et $(v_4,e_4,v_3,e_2,v_2,e_1,v_1)$.

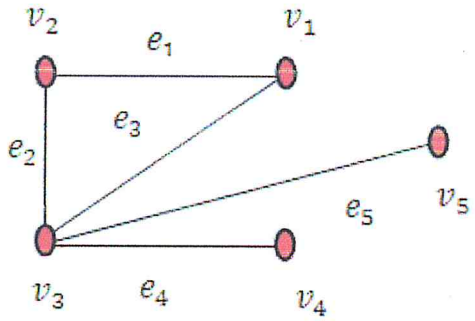


Figure 1.17 – Un graphe contenant des chaines

- Une chaine est élémentaire si chacun de ses sommets y apparait une seule fois.
- Une chaine est simple si chacune de ses arêtes y apparait au plus une seule fois.
- Une chaine dont les sommets de départ et de fin sont les même est appelée chaine fermée.

- Une chaine fermée simple est appelée cycle.

Graphes partiels et sous-graphes [53].

Définition 1.26 Un graphe $G' = (V', E')$ est un sous-graphe (subgraph) de $G = (V, E)$ si $V' \subseteq V$ et $E' \subseteq E$, G est alors un sur-graphe (supergraph) de G'.

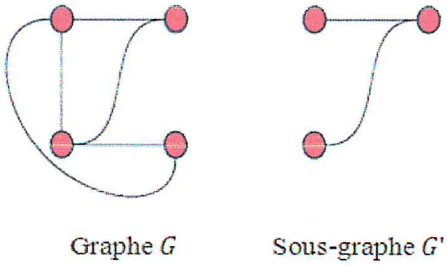


Figure 1.18 – Un graphe et un sous-graphe

Définition 1.27 Un sous-graphe couvrant (ou parfois graphe partiel) (spanning subgraph) de $G = (V, E)$ et un sous-graphe $G'' = (V, E')$ (c'est-à-dire qu'on peut l'obtenir à partir de G en supprimant uniquement des arêtes.

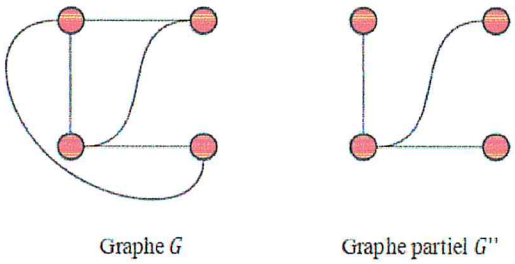


Figure 1.19 – Un graphe et un graphe partiel.

Définition 1.28 Le sous-graphe de $G = (V, E)$ induit par $V' \subseteq V$ (induced subgraph), noté $G[V']$, est le sous-graphe ayant pour ensemble de sommets V', et pour ensemble d'arêtes toutes les arêtes ayant leurs deux extrémités dans V'.

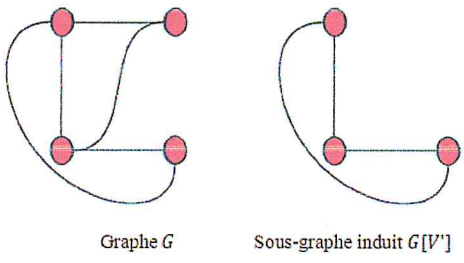


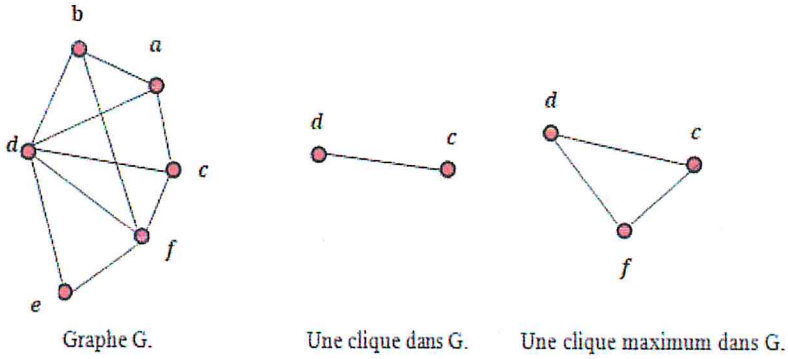
Figure 1.20 – Un graphe et un sous-graphe induit.

Quelques sous-graphes souvent utiles [53].

Quelques sous-graphes, graphes partiels et sous-graphes induits sont souvent recherchés dans un graphe pour trouver une solution à un problème.

Cliques.

Définition 1.29 Une clique (clique) d'un graphe G = (V, E) est un sous ensemble de sommets deux à deux adjacents (parfois c'est le sous graphe induit complet)



Graphe G. Une clique dans G. Une clique maximum dans G.

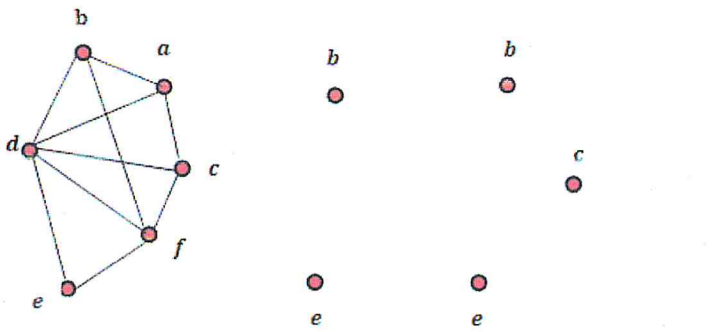
Figure 1.21 – Exemples de cliques.

Application 1.30

Nous nous sommes interrogés plus haut sur l'organisation d'une soirée ou toutes les personnes sont amies sur un réseau social. Le problème en termes de graphe consiste à chercher un sous-graphe induit complet de cardinalité maximum. Un sous-graphe induit complet est appelé une clique, l'objet cherché ici est une clique maximum. La Figure 1.21 montre une clique et une clique maximum de *G*.

A l'inverse, on peut chercher dans ce même réseau social, un ensemble de personnes qui ne sont pas amies. Ce sous ensemble s'appelle un stable.

On peut alors chercher le plus grand nombre de personnes que l'on peut rassembler et qui ne sont pas amies. On parle alors du stable maximum ou stable de cardinalité maximum. La Figure 1.22 montre un stable et un stable maximum de *G*.



Graphe G Un stable dans G. Un stable maximum dans G.

Figure 1.22 – Exemples de stables.

Ainsi une clique dans un graphe est un stable dans son complémentaire et inversement.

Il est facile de voir qu'un ensemble maximum est aussi maximal, mais un ensemble maximal n'est pas nécessairement maximum.

Un stable maximal Un stable maximum

Figure 1.23 – Différence entre un stable maximal et un stable maximum.

Le cardinal d'un plus grand stable est le nombre de stabilité de G, on le note $\alpha(G)$, et celui d'une clique maximum est notée $\omega(G)$. Naturellement, on a $\alpha(G) = \omega(\bar{G})$.

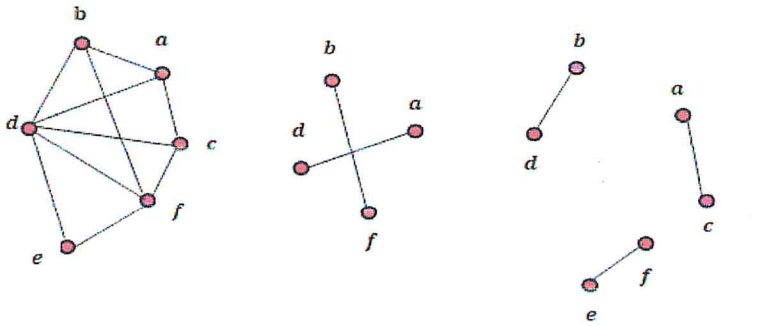
Déterminer un stable maximum ou une clique maximum dans un graphe arbitraire n'est pas du tout trivial. Nous reviendrons plus longuement sur ces problèmes au chapitre 3.

Couplages.

Définition 1.31 Un couplage (matching) d'un graphe G = (V, E) est un sous-graphe composé d'arêtes deux à deux non adjacentes.

Là encore, il est possible de chercher dans un graphe un couplage maximum, c'est à dire celui contenant le plus grand nombre d'arêtes. Si le graphe obtenu est un graphe partiel, i.e. couvrant tous les sommets, le couplage est dit parfait. La Figure 1.24 montre un couplage et un couplage parfait de G.

Définition 1.32 Un couplage parfait (perfect matching) d'un graphe G = (V, E) est un graphe partiel composé d'arêtes deux à deux non adjacentes.



Graphe G. Un couplage dans G. Un couplage parfait dans G.

Figure 1.24 – Exemples de couplages.

Graphes k-dégénéré.

Définition 1.33 Un graphe est k-dégénéré (k-degenerate graph), si tout sous-graphe de G possède un sommet de degré inférieur ou égal à k, et la dégénérescence d'un graphe est le plus petit k tel qu'il est k-dégénéré.

Graphes eulériens.

Définition 1.34 On appelle cycle eulérien d'un graphe G un cycle passant une et une seule fois par chacune des arêtes de G.

Définition 1.35 Un graphe est dit eulérien s'il possède un cycle eulérien.

Définition 1.36 On appelle chaîne eulérienne d'un graphe G une chaîne passant une et une seule fois par chacune des arêtes de G.

Définition 1.37 Un graphe ne possédant que des chaînes eulériennes est semi-eulérien. Plus simplement, on peut dire qu'un graphe est eulérien (ou semi-eulérien) s'il est possible de dessiner le graphe sans lever le crayon et sans passer deux fois sur la même arête.

Graphes hamiltoniens

Définition 1.38 On appelle cycle hamiltonien d'un graphe G un cycle passant une et une seule fois par chacun des sommets de G. Un graphe est dit hamiltonien s'il possède un cycle hamiltonien.

Définition 1.39 On appelle chaine hamiltonienne d'un graphe G une chaine passant une et une seule fois par chacun des sommets de G.

Définition 1.40 Un graphe ne possédant que des chaines hamiltoniennes est semi-hamiltonien.

Contrairement aux graphes eulériens, il n'existe pas de caractérisation simple des graphes (semi-hamiltoniens). On peut énoncer quelques propriétés et conditions suffisantes :

– Un graphe connexe est eulerien si et seulement si tous ses sommets sont de degré pair.

– Un graphe possédant un sommet de degré 1 ne peut pas être hamiltonien.

– Un sommet dans un graphe est de degré 2, alors les deux arêtes incidentes à ce sommet doivent faire partie du cycle hamiltonien.

– les graphes complets K_n sont hamiltoniens.

Graphes planaires [57].

Définition 1.41 On dit qu'un graphe est planaire (planar graph) si on peut le dessiner dans le plan de sorte que ses arêtes ne se croisent pas.

Graphes série-parallèles.

Définition 1.42 Un graphe G=(X, U) est dit série-parallèle (S-P) s'il ne contient pas de sous-graphe partiel homéomorphe à K_4.

(Un homéomorphe à K_4 est obtenu par subdivision des arêtes de la clique à 4 sommets (K_4). Voir Fig. 1.25.1).

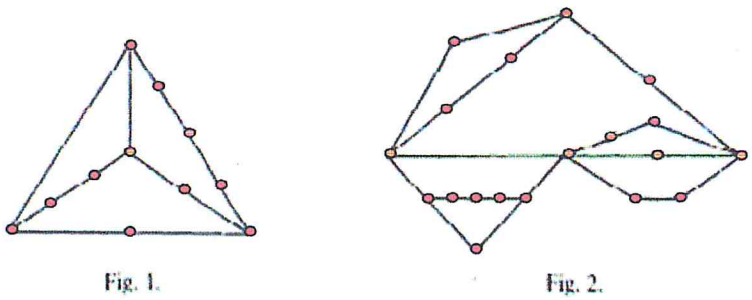


Fig. 1. Fig. 2.

Figure 1.25 – Exemples de graphe série-parallèle.

Un exemple de graphe série-parallèle est donné par la Fig. 1.25.2. Ils peuvent être utilisés pour modéliser des circuits électriques en série et en parallèle.

Propriété 1.43 Un graphe S-P possède un sommet de degré ≤ 2.

Propriété 1.44 Un graphe S-P est planaire (et son dual planaire est S-P).

Propriété 1.45 Un graphe S-P est trois coloriable.

Propriété 1.46 Les graphes S-P vérifiant la conjecture forte des graphes parfaits.

4. Coloration d'un graphe

Définition 1.47 Étant donné un graphe $G = (V, E)$, une coloration des sommets de G est une application $f: V \to C$, où C désigne l'ensemble des couleurs.

Il est toujours possible de colorer un graphe « proprement » ; il suffit d'attribuer une couleur différente à chaque sommet. Mais cette solution n'est clairement pas à privilégier, dans la mesure où pour la majorité des problèmes les couleurs symbolisent des ressources que l'on cherche justement à économiser. Ainsi, le problème de la coloration d'un graphe consiste-t-il à déterminer le nombre minimum de couleurs pour le colorer :

Problème de la coloration de graphe (Vertex Coloring)
Instance : Un graphe $G = (V, E)$
Question : Déterminer une coloration de G utilisant un nombre minimum de couleurs
Ce nombre est appelé nombre chromatique (chromatic number) de G, et est noté $\chi(G)$.

Définition 1.48

- Un graphe est k-colorable s'il est possible de le colorer avec k couleurs.
- Un graphe est k-chromatique si son nombre chromatique est égal à k.

Pour terminer ce paragraphe, notons que les sommets d'une même couleur étant deux à deux non adjacents, ils forment un stable. Autrement dit :

Proposition 1.49 Un graphe est k-colorable si et seulement s'il est possible de partitionner son ensemble de sommets en k stables.

On en déduit qu'un graphe est biparti si et seulement s'il est 2- colorable.

5. Quelques classes de graphes

Nous présentons ici rapidement quelques classes de graphes particulières qui occupent une place majeure en théorie des graphes et que nous rencontrerons souvent par la suite, avec leurs propriétés et les liens entre elles.

Graphes bipartis, arbres et forêts

Nous avons déjà mentionné les graphes bipartis, définis comme étant les graphes dont l'ensemble de sommets peut être partitionné en deux stables (ou, de façon équivalente, les graphes 2-colorables). Il est intéressant de noter que les graphes bipartis peuvent également être caractérisés comme suit :

Proposition 1.50 Un graphe est biparti si et seulement s'il ne contient aucun cycle de longueur impaire.

Définitions 1.51
- Une forêt (forest) est un graphe sans cycles.
- Un arbre (tree) est un graphe connexe sans cycles.

Autrement dit, une forêt est un ensemble d'arbres. De plus, puisqu'une forêt ne contient aucun cycle, elle ne contient à fortiori aucun cycle impair; par conséquent, toute forêt (et donc tout arbre) est un graphe biparti.

Graphes parfaits [57].

Définition 1.52 Un graphe parfait (perfect graph) G est un graphe vérifiant $\chi(H) = \omega(H)$ pour tout sous-graphe induit H de G.

En 1972, Lovász a démontré la conjecture faible des graphes parfaits : le complémentaire d'un graphe parfait est parfait. Dans son étude des graphes parfaits, Berge a été amené à définir une classe de graphes qui porte désormais son nom :

Définition 1.53

- Un trou (hole) est un cycle de longueur au moins 4.
- Un anti-trou (anti-hole) est le complémentaire d'un trou.
- Un graphe de Berge (Berge graph) est un graphe sans trou impair ni anti-trou impair.

Berge avait conjecturé que la classe des graphes parfaits coïncide avec la classe des graphes de Berge : c'était la fameuse conjecture forte des graphes parfaits, démontrée en 2002 :

Théorème 1.54 (Chudnovsky, Robertson, Seymour et Thomas (2002) [3]) Un graphe est parfait si et seulement s'il est de Berge.

Proposition 1.55 Les graphes bipartis sont parfaits.

Théorème des graphes parfaits : G est parfait $\Leftrightarrow$ $\overline{G}$ est parfait.

Graphes cordaux (triangulés) [57].

Définition 1.56 Un graphe est cordal ou triangulé (chordal graph) s'il ne contient pas de trous.

Autrement dit, dans un graphe cordal tout cycle de longueur au moins 4 contient une corde (une arête joignant deux sommets non consécutifs du cycle), i.e. les seuls cycles induits sont des triangles.

Proposition 1.57 Les graphes cordaux sont parfaits.

Graphes scindés.

Définition 1.58 Un graphe scindé (split graph) est un graphe dont l'ensemble des sommets peut être partitionné en deux sous-ensembles, dont l'un est un stable et l'autre est une clique.

Graphes d'intervalles [57].

Définition 1.59 Un graphe d'intervalles (interval graph) est le graphe d'intersection d'un ensemble d'intervalles de la droite réelle, i.e. chaque sommet représente un intervalle et une arête relie deux sommets lorsque les intervalles correspondants s'intersectent.

La figure suivante montre un graphe d'intervalles.

Proposition 1.60 Un graphe est d'intervalles si et seulement s'il est possible d'ordonner toutes ses cliques maximales $M_1, M_2, \ldots, M_k$ de sorte que pour tout sommet $v \in (M_i \cap M_k)$ $(i < k)$, $v \in M_j, \forall i \leq j \leq k$.

Plusieurs algorithmes efficaces ont été proposés pour reconnaître les graphes d'intervalles [4], [5]. Enfin, notons que :

Proposition 1.61 Les graphes d'intervalles sont cordaux, et donc parfaits.

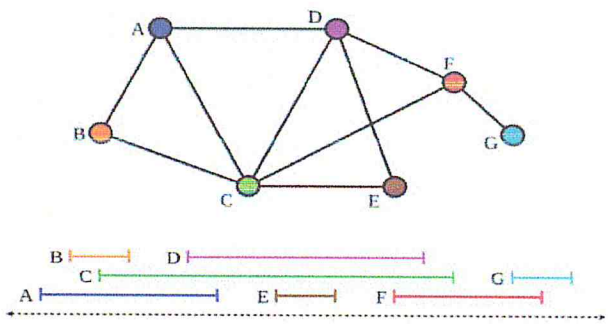


Figure 1.26 – Un graphe d'intervalles.

Graphes de comparabilité [57].

Définition 1.62 Un graphe est de comparabilité si on peut orienter ses arêtes de façon transitive, c'est-à-dire de telle sorte que s'il existe un arc de i vers j et un arc de j vers k, alors il existe également un arc de i vers k.

Théorème 1.63 Les graphes de comparabilité sont parfaits.

Graphes de permutation [57].

Définition 1.64 Un graphe G est de permutation s'il existe une permutation π des sommets telle que x est adjacent à y dans G si et seulement si x<y et π (y)< π (x) ou y<x et π (x)< π (y).

Propriété 1.65 Les graphes de permutation sont de comparabilité.

Théorème 1.66 Un graphe G est de permutation si et seulement si G et $\bar{G}$ sont de comparabilité.

Exemple 1.67 Considérons π tel que π (1)=3, π (2)=2, π (3)=4, π (4)=6, π (5)=1 et π (6)=5.

Le graphe de permutation associé à π est :

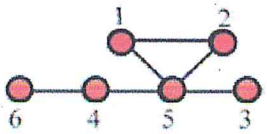


Figure 1.27 – Un graphe de permutation.

Et la figure géométrique dont G est le graphe d'intersection des segments est :

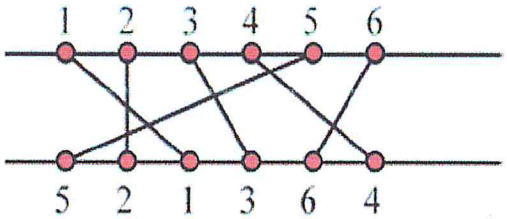


Figure 1.28 – Un graphe d'intersection des segments.

Propriété 1.68 Un graphe d'intervalles est le graphe complémentaire d'un graphe de comparabilité.

Théorème1.69 Un graphe G est d'intervalles si et seulement si G est de comparabilité et G ne contient aucun C_4 comme sous-graphe induit. (Un C_4 est un cycle de longueur 4 sans corde, c'est-à-dire un carré).

Conclusion du chapitre 1 :

Dans ces quelques pages, nous avons rappelé les notions de base de la théorie des graphes Nous disposons à présent de tous les outils nécessaires pour aborder sereinement les prochains chapitres ; en particulier, les chapitres suivants traitent en profondeur des problèmes au cœur de ce mémoire : les problèmes du stable maximum et du stable de poids maximum.

Chapitre 2

Théorie de la complexité

Théorie de la complexité

La théorie de la complexité est une branche des mathématiques et de l'informatique ayant pour cadre l'étude de la difficulté intrinsèque des problèmes algorithmiques, et qui vise à classer ces problèmes en fonction de cette difficulté. Ici, les mots «complexité» et «difficulté» ne se rapportent pas à la mise au point d'un algorithme de résolution, ou aux concepts avancés auxquels il peut faire appel (comme une structure de données élaborée), mais plutôt à la quantité de ressources à utiliser pour résoudre le problème.

1. Concepts de base

Définition 2.1

- Un problème est une question générale possédant des paramètres dont la valeur n'est pas connue.
- Une instance d'un problème est obtenue en affectant une valeur à chacun de ses paramètres.
- La taille d'une instance désigne généralement la quantité de cases mémoires nécessaires pour décrire les paramètres.

Exemple 2.2 Le problème du voyageur de commerce (ou TSP pour Traveling salesman problem) consiste, étant donné un ensemble de villes séparées par des distances connues, à trouver le plus court chemin qui relie toutes les villes, en ne passant qu'une seule fois par chaque ville. Une instance du TSP est donc un ensemble de n points (représentant les villes) définis chacun par un couple de coordonnées et la taille de cette instance est $2n + 1$ (il faut une case mémoire pour chaque coordonnée des n points et une autre pour stocker l'entier n).

Parmi les problèmes qui nous intéressent, il existe généralement deux grands types de problèmes :

Problèmes de décision et d'optimisation :

Problème de décision :

Définition 2.3 Un problème de décision est un problème auquel la réponse est oui ou non.

Il est représenté génériquement par une instance (données) et par une question.

Exemple 2.4 Partition

Donnée : Un ensemble A de n nombres $a_1, a_2, ..., a_n$.
Question : Existe-t-il un sous ensemble $I_1 \subseteq \{1, 2, ..., n\}=I$ tel que : $\sum_{i\in I1} a_i = \sum_{i\in I\setminus I1} a_i$.

Problème d'optimisation :

Définition 2.5 Un problème d'optimisation consiste à déterminer la meilleure solution parmi toutes les solutions réalisables.

Définition 2.6 Un problème d'optimisation combinatoire (POC en abrégé) est défini par la donnée d'un ensemble fini D de solutions réalisables et une fonction objectif $f: D \rightarrow \mathbb{R}$ ou on cherche un élément $x_0 \in D$ tel que $f(x_0) = minf(x)$.

Ce problème s'appelle un problème de minimisation et est noté $\begin{cases} Min f(x) \\ x \in D \end{cases}$

Le problème consistant à chercher un élément maximum au lieu d'un élément minimum est de même nature puisque $Max\ f(x) = -Min(-f(x))$.

Exemple 2.7

Problème de voyageur de commerce : Etant donné un graphe aves des poids associés aux arêtes. Déterminer un cycle fermé qui passe exactement une fois par chaque sommet du graphe de façon à minimiser le poids total des arêtes.

Notation «grand *O*»

La notation grand O, dite aussi symbole de Landau, décrit le comportement asymptotique d'une fonction, exprimé à l'aide d'une autre fonction généralement plus simple. Plus formellement, nous dirons que :

Définition 2.8 (Notation grand O) $f(n) = O(g(n))$ («$f(n)$ est en grand O de $g(n)$ ») quand $n \to \infty$ si et seulement si $\exists M > 0, n_0 \in \mathbb{R}$ tels que $\forall n \geq n_0$, $|f(n)| \leq M|\ g(n)\ |$.

Exemple 2.9 Soit $f(n) = 6n^4 - 2n^3 + 5$. Choisissons $n_0 = 1$. Alors pour tout $n \geq n_0$,

On a $|6n^4 - 2n^3 + 5| \leq 6n^4 + |2n^3| + 5$

$\leq 6n^4 + |2n^4| + 5n^4 = 13n^4 = 13|n^4|$

Ainsi, en prenant $M = 13$, on a $f(n) = O(n^4)$.

Autrement dit, à un facteur constant près, $f(n)$ ne croît pas plus rapidement que n^4.

Il est facile de voir qu'un polynôme $P(n)$ de degré k est toujours en $O(n^k)$.

Complexité des algorithmes :

Un algorithme qui résout un problème prend en entrée une instance I dite une entrée du problème. D'autre part pour implémenter cet algorithme sur un ordinateur, cette instance est codée sous forme de chaines binaires 0,1 (codage binaire). La longueur de cette chaine définit la taille de l'entrée $|I|$, notée n.

La complexité d'un algorithme est une fonction $T(n)$ égale au nombre maximum d'opérations élémentaires requises par l'exécution de l'algorithme sur une entrée de taille n (on considère donc le pire des cas possible).

L'efficacité d'un algorithme pour un problème se caractérise par sa complexité :

On dira qu'un algorithme est polynomial ou efficace si sa complexité est en $O(n^k)$ pour une certaine constante k.

En revanche, un algorithme dont la complexité ne peut pas être bornée polynomialement est qualifié d'exponentiel, par exemple des algorithmes de complexité en $O(2^n)$ et en $O(n!)$.

Complexité des problèmes :

La classification usuelle des problèmes se fait en considérant leurs complexités. Celle-ci correspond à la complexité de l'algorithme le plus efficace pour résoudre ce problème. A cause des difficultés rencontrés, la théorie de la complexité ne traite fondamentalement que les problèmes de décision, car leur formalisme Oui-Non, Vrai-Faux a permis de les étudier avec les outils de la logique mathématique. Cependant on étend la notion de la complexité aux problèmes d'optimisation. En effet il est facile de transformer un problème d'optimisation en un problème de décision. Par exemple chercher à optimiser une certaine valeur v revient à traiter un problème de décision qui consiste à comparer v à une certaine k, puis en générant un certain nombre de valeurs k, on peut déterminer la valeur optimale (exemple par l'utilisation de l'algorithme de dichotomie).

Classes de complexité P et NP :

La classe P (polynomial)

Un problème de décision est dit dans la classe P (polynomial) si, pour chacune de ses instances, dont la taille est notée n, il existe un réel positif k tel qu'il peut être résolu par un algorithme de complexité temporelle $O(n^k)$, c'est-à-dire qu'il peut être décidé en temps polynomial.

Les problèmes de la classe P sont dits faciles. Ce sont ceux que l'on sait résoudre efficacement.

La classe *NP* (Nondeterministic polynomial)

Elle regroupe tous les problèmes de décision qui peuvent être résolus en temps polynomial par des algorithmes non déterministes. Un algorithme est dit non déterministe s'il comporte des instructions de choix. Pour ces algorithmes, si à chaque instruction, le bon choix est effectué, le temps de calcul est polynomial. Si au contraire tous les choix sont énumérés l'algorithme devient déterministe et son temps de calcul devient exponentiel. De façon informelle, un problème de décision appartient à la classe NP si on peut vérifier en un temps polynomial, qu'une solution proposée (ou devinée) permet d'affirmer que la réponse est 'oui'.

Remarque1 Naturellement, si on peut résoudre un problème avec un algorithme polynomial, on peut aussi vérifier en temps polynomial que la solution fournie est bien une solution ; par conséquent $\mathbf{P} \subset \mathbf{NP}$.

2. Réduction polynomiale :

On dit qu'un problème de décision (D) se réduit polynomialement à un problème de décision (D') s'il existe une application f qui transforme chaque instance I de (D) en une instance I' de (D') telle que :

– f est polynomiale c'est à dire la construction de I' est polynomiale par rapport à la taille I.

–La réponse est oui pour I si et seulement si la réponse est oui pour I'.

Si un problème (D) se réduit polynomialement à un problème (D') on note $D \leq_p D'$.

On dit aussi que (D) n'est pas plus difficile que (D').

Par exemple, le problème consistant à élever un nombre au carré se réduit au problème plus général de multiplication de deux nombres (ici, aucune transformation n'est nécessaire).

Une réduction est polynomiale lorsque le processus de transformation peut se faire en temps polynomial.

Exemple 2.10 HC ≤p TSP On doit montrer que le problème de décision de cycle hamiltonien (HC) se réduit polynomialement à un problème de décision de PVC (TSP).

-Soit I une instance du problème HC c'est-à-dire : un graphe G= (V, E), Construisons une instance I' du problème TSP comme suit : G'= (V, E') est le graphe complet de G tel que : d(e)=$\begin{cases} 1 \; si \; e \in E \\ 2 \; si \; e \in E' \setminus E \end{cases}$, un seul k=n .

- Il est claire que la construction de I' set polynomiale par rapport à la taille de I.

- montrons que la réponse par rapport à HC est oui pour I ⇔ la réponse par rapport a TSP est oui pour I'.

C.-à-d. on doit montrer que si ∃ un cycle hamiltonien dans G ⇔ ∃ un tour T dans G' de longueur l(T)≤ n ou l(T) =$\sum_{e \in T} d(e)$;

⇒) Supposons qu'il ∃ un cycle hamiltonien C dans G donc C est un tour qu'on note T dans G' l(T)=$n \leq n$

Inversement : si ∃ un tour T dans G' tel que l(T) ≤ n ⇒ touts les arête du tour sont dans E sinon l(T) serai> n.

Propriété 2.11 La réduction polynomiale est une relation transitoire

Si $D_1 \leq_p D_2$ et $D_2 \leq_p D_3 \Rightarrow D_1 \leq_p D_3$

3. Classe *NP*-Complet :

La classe NP-Complet est composée des problèmes les plus difficiles de NP, On dit qu'un problème est NP-Complet si sa résolution en temps polynomial entraînerait la résolution en temps polynomial de tout problème de NP.

Un problème de décision D est dit NP-Complet s'il vérifie les deux conditions suivantes :

– D appartient à la classe NP,

– Tous les problèmes de la classe NP se réduisent polynomialement à D.

La classe des problèmes de décision NP peut être partitionnée en 3 sous classe : les problèmes de la classe P, les problèmes NP-complet et les autres, voir la figure 1.26

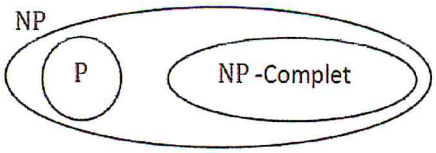


Figure 2.1 – Géographie de la classe NP

Remarque2 : Tout problème d'optimisation combinatoire dont le problème de décision associé est NP-Complet est dit NP-difficile. Le problème du stable maximum est NP-difficile

Le premier problème qui a été prouvé NP-Complet par S.A.Cook en 1970, est le problème de satisfiabilité (SAT).

Le problème SAT :

- x_i est une variable booléenne (qui vaut vrai ou faux)
- $\bar{x}_i$ est sa négation
- x_i et $\bar{x}_i$ sont des littéraux.
- Une clause une disjonction de littéraux, par exemple $x_1 \vee x_2 \vee x_3$
- Une forme normale conjonctive est une conjonction de clauses, par exemple $C_1 \wedge C_2 \wedge C_3 \wedge C_4$

Définition 2.13 Etant donnée une formule φ sous forme normale conjonctive (CNF), existe-t-il un assignement des variables booléennes (vrai ou faux) tel que φ soit vrai (φ est satisfiable)?

Preuve de NP-Complétude :

Pour montrer qu'un problème de décision (D) est NP-complet on procède en 3 étapes:
Etape 1: montrer que le problème $(D) \in$ NP.
Etape 2 : choisir un problème $(D') \in$ NP connu pour être NP-complet
Etape 3: montrer que (D') se réduit polynomialement à (D).

Conjecture fondamentale $P \neq NP$

On a vu dans la remarque 1 que $P \subseteq NP$, mais on ne sait pas si P est égale ou non à NP. S'il s'avérait que $P = NP$, alors on pourrait résoudre tous les problèmes de la classe NP en un temps polynomial. Or, les problèmes NP-Complets sont très fréquents et pour aucun d'entre eux on a réussi à trouver un algorithme polynomial le résolvant.

La communauté scientifique pense que $P \neq NP$, mais ce n'est pas prouvé et que cette conjecture reste un problème ouvert depuis 1971.

Quelques problème NP-Complets de base :

Nous citons six problème NP-Complets dont la NP-Complétude n'a pu être démontrée que grâce au problème de satisfiabilité.

3-SAT (3-satisfiabilité): Etant donné un ensemble U de variables et une collection φ de k clauses sur U, contenant chacune 3 variables. Existe-t-il une affectation en 'vrai' et 'faux' des variables telle que toutes les clauses prennent la valeur 'vrai' ?

Recouvrement minimum (Vertex Cover): Etant donné un graphe $G = (V, E)$et un entier k. Existe-t-il $X \subseteq V$ tel que $|X| \leq k$ et toute arête de G a au moins une de ses extrémités dans X ?

Cycle Hamiltonien (Hamiltonian Cycle) : Etant donné un graphe $G = (V, E)$. Existe-t-il un cycle hamiltonien dans G (c'est-à-dire un cycle passant une et une seule fois par tous les sommets de G) ?

Clique maximum : Etant donné un graphe $G = (V, E)$ et un entier k. Existe-t-il $X \subseteq V$ tel que $|X| \geq k$ et tous les sommets de X sont deux à deux adjacents?

PVC (TSP) : Etant donné un graphe $G = (V, E)$ et d est une fonction définie sur l'ensemble des arêtes e et e $\rightarrow d(e) \in N$,$d(e)$ s'appelle la distance entre les extrémités de e. Existe-t-il un tour (parcours fermé) de longueur inférieur ou égale à k?

Stable (Independent Set) : Etant donné un graphe $G = (V, E)$ et $k \in N^*$. Existe-t-il un stable S dans G de taille (le nombre de sommets) k ?

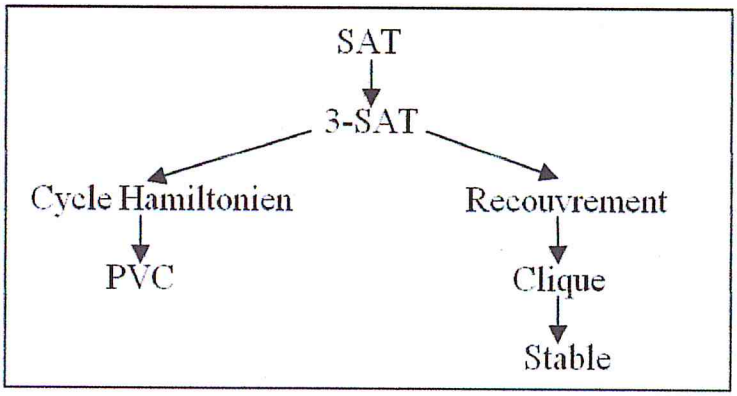


Figure 2.2 –Enchainement des preuves de la NP-Complétude.

Théorème2.14 [52] 3-SAT ≤p Independent set.

Démonstration [52]: Soit φ une instance de 3-SAT. On construit une instance (G, k) de Independent set tel que (G, k) a un independent set de taille k si et seulement si φ est satisfiable.

On construit avec 3 sommets pour chaque clause, chacun d'eux représentant un littéral de la clause. On relit par des arcs ces trois sommets. On relit aussi chaque littéral avec sa négation

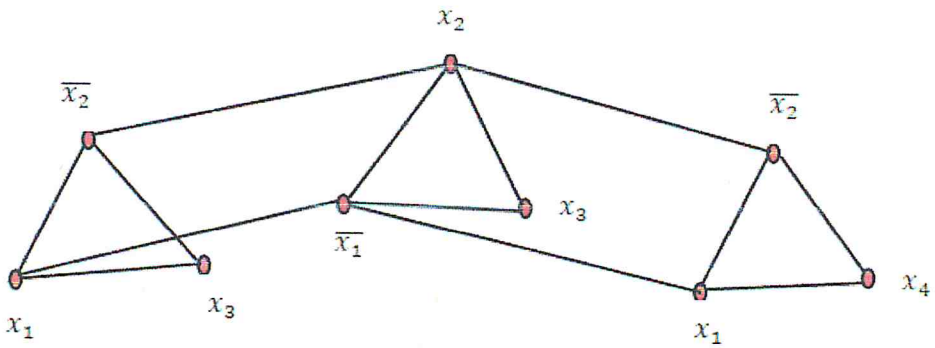


Figure 2.3 –Le graphe associe à l'instance φ de 3-SAT.

$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$

Lemme 2.15 [52] Le graphe G contient un independent set de taille k égal au nombre de clauses de φ si et seulement si φ est satisfiable.

Démonstration [52]: Soit S un independent set de G de taille k. Alors S a un sommet dans chaque triangle. On met ce sommet à vrai. Cet assignement est correct et chaque clause est vrai. Donc φ est vrai.

Réciproquement, on part d'un assignment des variables booléennes qui rend φ vrai. On choisit un littéral vrai dans chaque triangle. C'est un independent set pour G.

Chapitre 3

Problème du stable maximum

1. Problème du stable maximum.

Le problème du stable maximum consiste à déterminer un stable de cardinalité maximale dans un graphe. Ce problème est un problème classique d'optimisation combinatoire, ayant de très nombreuses applications pratiques (il intervient par exemple en vision par ordinateur, en biologie moléculaire, ou encore dans des problématiques d'ordonnancement).

Dans ce chapitre, nous commençons par rappeler les deux formes du problème (décision et optimisation), et nous redonnons la démonstration de sa NP-difficulté ; nous exposerons également un certain nombre de problèmes connexes.

2 Énoncé et applications.

Nous rappelons qu'un stable dans un graphe est un ensemble de sommets deux à deux non adjacents.

Il est très facile de trouver un ensemble stable S dans un graphe G : il suffit de choisir arbitrairement un premier sommet v que l'on ajoute à S, de supprimer dans G v ainsi que tous ses voisins et de recommencer sur le sous-graphe ainsi obtenu.

En exécutant cet algorithme glouton (dont on montre facilement qu'il s'exécute en temps $O(\max(m,n))$, jusqu'à ce que le graphe ne contienne plus de sommets, on obtient un stable maximal, c'est-à-dire qui n'est contenu dans aucun autre ensemble stable :

Algorithme1: Stable maximal [49]

Entré : Un graphe G= (V, E)
Sortie : Un stable maximal dans G

Début

I≠ Ø;i=0 ; $G_i = G$;

Tant que $V(G_i) \neq \emptyset$ faire

Choisir un sommet v_i de G_i ;

I: = IU {vi} ;

$G_{i+1} := G_i [V(G_i) - N_{Gi}[v_i]]$;

i := i+1 ;

fin Tant que ;

Sortie I ;

Fin.

Dans l'algorithme 1 si on choisit v_i de G_i de degré minimum on obtient l'algorithme MIN.

Algorithme MIN [22]

Entré : Un graphe G= (V, E)
Sortie : Un stable maximal dans G

Début

I≠ Ø;i=0; $G_i = G$;

Tant que $V(G_i) \neq \emptyset$ faire

Choisir un sommet v_i de G_i de degré minimum;

I: = IU {vi};

$G_{i+1} := G_i [V(G_i) - N_{Gi}[v_i]]$;

i := i+1 ;

fin Tant que ;

Sortie I ;

Fin.

Exemple d'application de l'algorithme MIN :

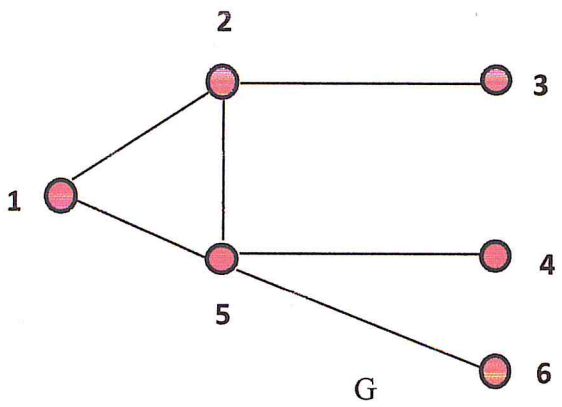
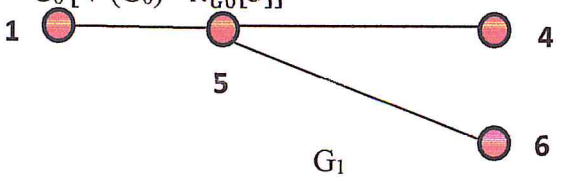


Figure 3.1 – Le Graphe G d'application de l'algorithme MIN.

Iteration 0: I =Ø ; I =0; G_0=G;

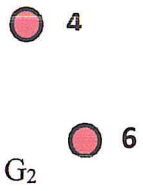
$V(G_0)= V(G) \neq \emptyset$; d(3)=1 est un sommet de degré minimum.

I = {3}; $G_1 = G_0 [V(G_0) - N_{G0}[3]]$



i=0+1=1;

Iteration 1: V (G_1) ≠ Ø ; d (1) = 1; I = {3, 1}; G_2= G_1 [V (G_1) - N_{G1}[1]].



G_2

i=1+1=2;

Iteration 2: V (G_2) ≠ Ø ; d (6) = d (4) = 0; I = {3, 1, 6, 4}; G_3= G_2 [V (G_2) - N_{G2}[4]- N_{G2}[6]].

Iteration 3: V (G_3)= Ø; I=V (G_i) = {3, 1, 6, 4}.

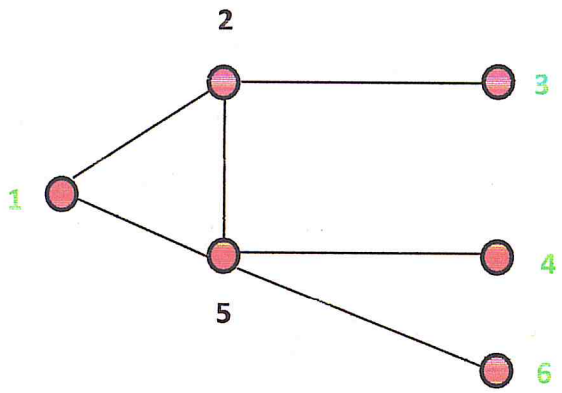


Figure 3.2 – Stable maximal dans G obtenu par l'algorithme MIN.

Algorithme2: Stable maximal [22]

Entré : Un graphe G= (V, E)

Sortie : Un stable maximal dans G

Début

I≠ Ø;i=0 ; G_i=G ;

Tant que E(G_i) ≠ Ø faire

Choisir un sommet v_i de G_i ;

I := I∪{v_i};

G_{i+1}:= G_i [V (G_i) - {v_i}];

i := i+1 ;

fin Tant que ;

I :=V(G_i) ;

Sortie I ;

Fin.

Dans l'algorithme 2 si on choisit v_i de G_i de degré maximum on obtient l'algorithme MAX.

Algorithme MAX [22]

Entré : Un graphe G=(V, E)
Sortie : Un stable maximal dans G

Début
$I \neq \emptyset; i=0$; $G_i=G$;
Tant que $E(G_i) \neq \emptyset$ faire
Choisir un sommet v_i de G_i de degré maximum;
$I := I \cup \{v_i\}$;
$G_{i+1} := G_i [V(G_i) - \{v_i\}]$;
$i := i+1$;
fin Tant que ;
$I := V(G_i)$;
Sortie I ;
Fin.

Exemple d'application de l'algorithme MAX :

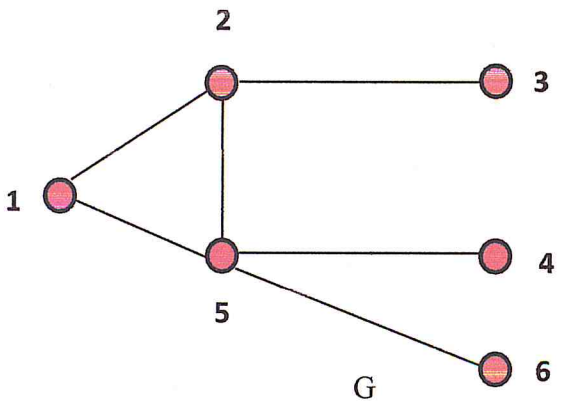
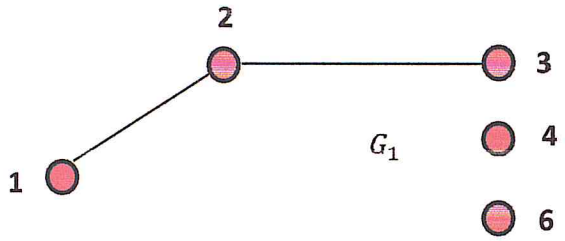


Figure 3.3 – Le Graphe G d'application de l'algorithme MAX.

Itération 0: $E(G_0)=E(G) \neq \emptyset$.

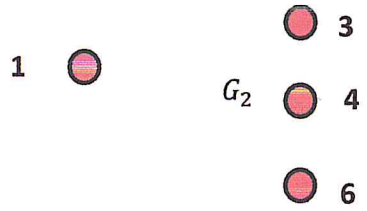
d (5)=4 est un sommet de degré maximum, on le choisit

$I := \{5\}$; $G_1=G [v(G_0) - \{5\}]$.



i=0+1=1;

Iteration 1: E (G_1) ≠ ∅ ; d (2) = 2 ; G_2= G_1[V(G_1)-{2}].



i=1+1=2;

Iteration 2: E (G_2)= ∅ .

I: =V (G_i) = {1, 3, 4, 6}.

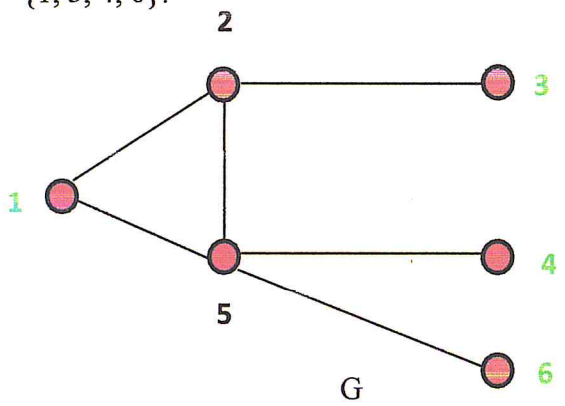


Figure 3.4 – Stable maximal dans G obtenu par l'algorithme MAX.

Puisque l'on sait calculer facilement un stable, une question vient assez naturellement : étant donnés un graphe G et un entier positif k, peut-on déterminer si G contient un stable de cardinalité au moins k?

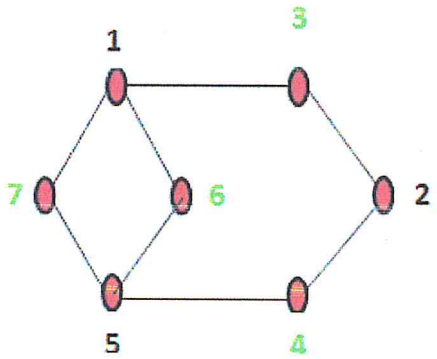
Comme nous l'avons vu au chapitre précédent, il s'agit là d'un problème de décision, auquel la réponse est soit «oui» soit «non».

Le problème d'optimisation correspondant, qui consiste à trouver dans un graphe un stable de cardinalité maximum, est appelé problème du stable maximum (Maximum Independent Set Problem), que nous abrégerons souvent par «MIS» [17] [18] dans la suite :

Problème du stable maximum (MIS) (Maximum Independent Set)

Instance : Un graphe $G = (V, E)$
Objectif : Déterminer un stable maximum de G

Rappelons que la cardinalité d'un stable maximum de G est notée $\alpha(G)$.



S= {3, 4, 6, 7} Stable de cardinalité maximum$\alpha(G) = 4$.

Figure 3.5- Exemple d'un stable maximum.

3. Exemples.

Exemple 1: Soit le graphe suivant :

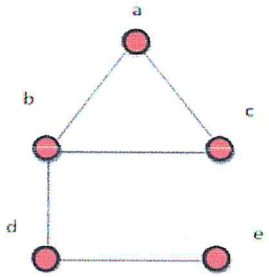


Figure 3.6- Exemple de la détermination du nombre α(G).

Il y a $C_5^3 = 10$ cas possibles (sous-ensembles d'ordre 3).

Les sous-ensembles d'ordre 3 contenant ab :{a, b, c} {a, b, d} {a, b, e}ceux ne sont pas des stables car {a, b}est une arête

Les sous-ensembles d'ordre 3 contenant a, c : {a, c, d} {a, c, e} ceux ne sont pas des stables car {a, c}est une arête

Les sous-ensembles d'ordre 3 contenant b, d : {b, d, e} {b, d, c} ceux ne sont pas des stables car {b, d} est une arête

Les sous-ensembles d'ordre 3 contenant d, e : {d, e, a} {d, e, c} ceux ne sont pas des stables car {d, e} est une arête

Les sous-ensembles d'ordre 3 contenant b, c : {b, c, e} n'est pas stable car {b, c} est une arête $\Rightarrow \alpha(G) = 2$.

Certains auteurs définissent MIS sous une forme plus faible, comme le problème consistant à calculer seulement $\alpha(G)$. Bien que les deux problèmes soient de même difficulté, nous préférons la formulation donnée ci-dessus, porteuse de plus d'informations.

Ce problème a de nombreuses applications pratiques [15] : il intervient en théorie de l'information, en vision par ordinateur, dans l'allocation des registres d'un processeur, en biologie moléculaire, dans l'attribution des fréquences radio ou encore dans les problèmes de planification.

Plus généralement, le problème du stable maximum apparaît dès que l'on cherche à utiliser au mieux un ensemble de ressources sans créer de conflit entre elles.

Exemple et application 2: [49]

Un responsable d'une agence de tourisme se rend pour une semaine dans un centre de vacances proposant diverses activités de loisir et sportives. Afin de satisfaire l'ensemble de la clientèle, certaines activités ont lieu en même temps, et il n'est donc pas possible de participer à toutes. Soucieux de rentabiliser son séjour, le responsable désire participer à un maximum d'activités.
La théorie des graphes est un excellent moyen de modéliser cette situation : on associe à chaque activité le sommet d'un graphe, et on relie deux sommets si les activités correspondantes sont « incompatibles », c'est-à-dire s'il est impossible de participer aux deux, pour des contraintes d'horaires. Il est alors facile de voir qu'un ensemble activités «compatibles » constitue un stable. Ainsi, le problème d'un responsable se ramène à la recherche d'un stable maximum dans un graphe.

4. Formulation (PLNE) du problème de stable maximum MIS [56]:

- Une variable $v_i \in \{0, 1\}$ pour tout sommet i, indiquant si i est choisi (=1) ou non (=0).
- Une contrainte $v_i + v_j \leq 1$ pour toute arête $\{ i, j\}$.
- On maximise la somme des v_i.

$$\alpha(G) = Max\left(\sum_{i=1}^{n} v_i\right)$$

$$v_i + v_j \leq 1 \; \forall \{i, j\} \in E$$

$$v_i \in \{0, 1\}, i = \{1, 2, \dots, n\}.$$

Exemple [55] :

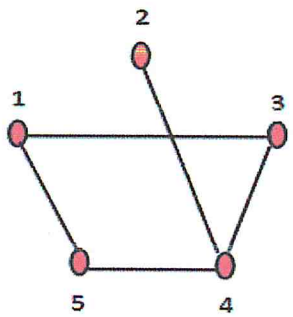


Figure 3.7- Exemple du MIS dans la (PLNE).

$$\alpha(G) = Max\left(\sum_{i=1}^{5} v_i\right) = (v_1 + v_2 + v_3 + v_4 + v_5)$$

$$v_1 + v_3 \leq 1$$
$$v_3 + v_4 \leq 1$$
$$v_2 + v_4 \leq 1$$
$$v_4 + v_5 \leq 1$$
$$v_1 + v_5 \leq 1$$
$$v_i \in \{0, 1\}, i = \{1, 2, \dots, 5\}.$$

5. Problèmes connexes [49].

Il existe un certain nombre de problèmes d'optimisation étroitement liés au problème du stable maximum :

5.1 Le problème de la clique maximum :

Il s'agit ici de déterminer une clique de plus grande taille dans un graphe G :

Problème de la clique maximum (Maximum Clique)
Instance : Un graphe $G = (V, E)$
Objectif : Déterminer une clique maximum de G

Évidemment, comme une clique dans un graphe G est un stable dans son complémentaire $\bar{G}$ et réciproquement, ce problème revient à chercher un stable maximum dans $\bar{G}$.

Rappelons que la cardinalité d'une clique maximum de G est notée $\omega(G)$, et qu'on a donc $\alpha(G) = \omega(\bar{G})$.

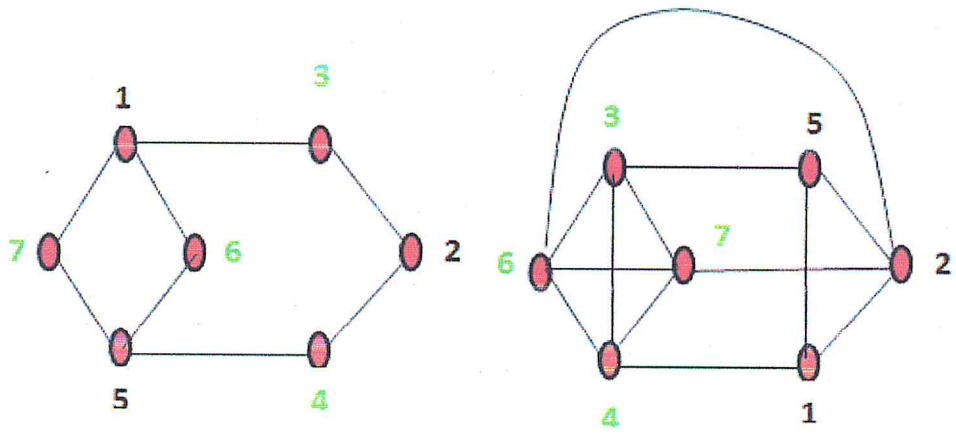


Figure 3.8- Exemple de $\alpha(G) = \omega(\bar{G})$.

Exemple :

Soit le graphe G quelconque suivant :

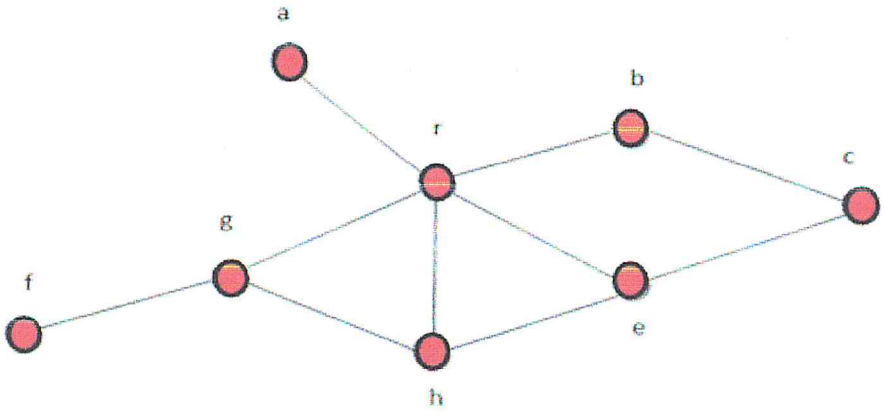


Figure 3.9- la cardinalité d'une clique maximum dans le graphe complémentaire $\bar{G}$.

Soit $\bar{G}$ le graphe complémentaire de G

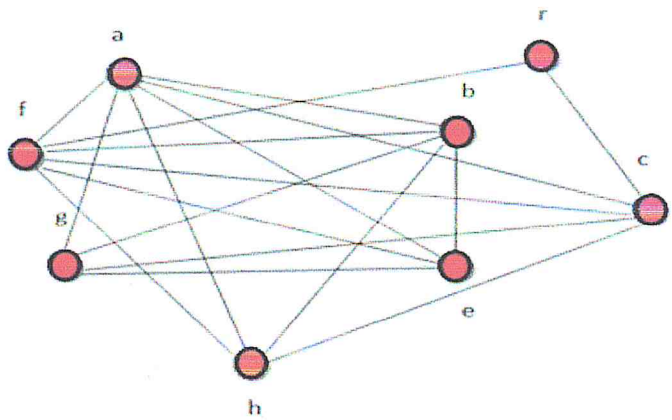


Figure 3.10- le graphe complémentaire $\bar{G}$ de G.

On ne peut pas trouver une clique d'ordre 5 contenant le sommet r car d(r)=2 dans $\bar{G}$.

Le sous graphe $\bar{G} \setminus \{r\} = \bar{G}[X \setminus \{r\}]$

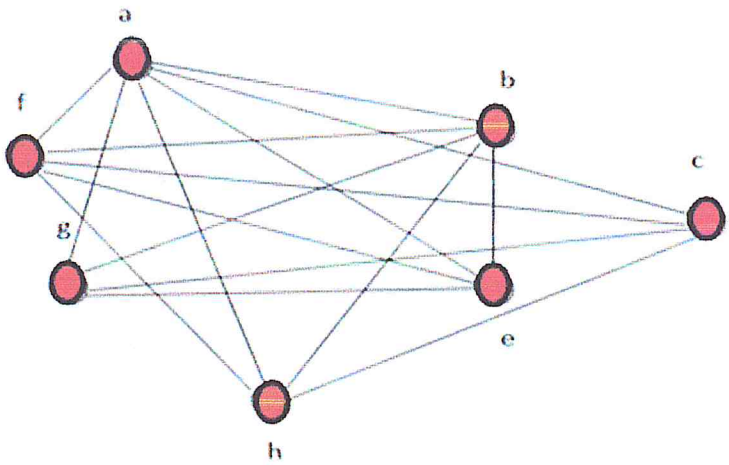


Figure 3.11- le graphe $\bar{G} \setminus \{r\}$.

Clique d'ordre 5 contenant le sommet a : ∄

1er possibilité : Le sommet a avec les sommets g, b, e, h ça ne donne pas une clique car g et h ne sont pas adjacents.

En continuant le même argument, on montrera que le graphe $\bar{G}$ ne contient pas de cliques d'ordre 5. Donc le graphe G ne contient pas de stable d'ordre 5 et par conséquent

$\alpha(G) = \omega(\bar{G}) = 4.$

5.2 Le problème du dominant stable minimum :

Un (ensemble) dominant (dominating set) est un sous-ensemble $D \subseteq V$ tel que tout sommet de $V \setminus D$ ait au moins un voisin dans D. Par exemple, V est un dominant (trivial) ; aussi le problème consiste-t-il à déterminer un plus petit dominant dans un graphe :

Problème du dominant minimum (Minimum Dominating Set)
Instance : Un graphe $G = (V, E)$
Objectif : Déterminer un dominant minimum de G

On désigne par $\gamma(G)$ la taille d'un dominant minimum, appelée nombre de domination (domination number) de G. Le lien avec le problème du stable maximum est illustré par les propositions suivantes :

Proposition [49]: Un stable S est maximal si et seulement s'il est dominant.

Démonstration : Soit S un stable.

⇒ Si S n'est pas dominant, il existe un sommet v qui n'a aucun voisin dans S; donc $S \cup \{v\}$ est un stable, et S n'est donc pas maximal.

⇐ Si S n'est pas maximal, il existe un sommet v tel que $S \cup \{v\}$ est stable ; donc v n'a aucun voisin dans S, et S n'est donc pas dominant.

□

Par conséquent, tout stable maximal est aussi dominant. On peut même montrer que :

Proposition [49]: Tout stable maximal est un dominant minimal.

Démonstration : Soit S un stable maximal. D'après la proposition précédente, S est un ensemble dominant. Supposons qu'il ne soit pas dominant minimal ; il existe donc $V \in S$ tel que $S \setminus \{v\}$ est également dominant. En particulier, v est dominé par $S \setminus \{v\}$, donc adjacent à un sommet de $S \setminus \{v\}$, ce qui contredit l'hypothèse que S est stable. Par conséquent, S est bien dominant minimal.

Bien entendu, la réciproque n'est pas vraie puisqu'un dominant n'est pas forcément stable (par exemple, dans un P_4, les deux sommets centraux constituent un dominant minimal mais pas un stable).

□

On note à présent i(G) la cardinalité minimum d'un stable maximal. D'après ce qui précède, on a :

Proposition [49]: Pour tout graphe G, $\gamma(G) \leq i(G) \leq \alpha(G)$.

Ainsi, alors que le problème du stable maximum consiste à déterminer le plus grand des stables maximaux, le problème du dominant stable minimum consiste à trouver le plus petit (ce qui explique qu'il soit aussi connu dans la littérature comme Minimum Maximal Independent Set), que nous noterons MIDS (ou MWIDS dans sa version pondérée) :

Problème du dominant stable minimum (Minimum Independent Dominating Set)
Instance : Un graphe G = (V, E)
Objectif : Déterminer un dominant stable minimum de G

Pour plus de précisions sur la domination dans les graphes, on pourra se reporter à [6] et [7].

5.3 Le problème du transversal minimum :

Un transversal (vertex cover) est un ensemble de sommets contenant au moins une extrémité de chaque arête. Une nouvelle fois, V est un exemple trivial de transversal.

Le problème consistant à trouver un transversal minimum est un problème d'optimisation classique, apparaissant dans les 21 problèmes de Karp [8] :

Problème du transversal minimum (Vertex Cover)
Instance : Un graphe G = (V, E)
Objectif : Déterminer un transversal minimum de G

On désigne par $\tau(G)$ la cardinalité d'un transversal minimum de G.

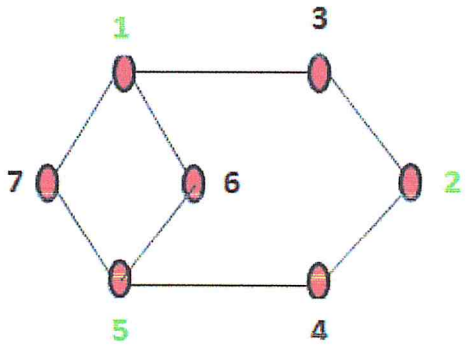


Figure 3.12 Exemple de transversal de cardinalité minimum $\tau(G)= 3$.

Par définition, si T ⊆ V est un transversal, V\T est un stable, et réciproquement. On en déduit immédiatement la relation suivante :

Théorème : (Gallai (1959) [9]) Pour tout graphe G d'ordre n, **$\alpha(G) + \tau(G) = n$**.

Ainsi, si l'on est en mesure de déterminer un transversal minimum de G, on obtient simultanément un stable maximum de ce graphe.

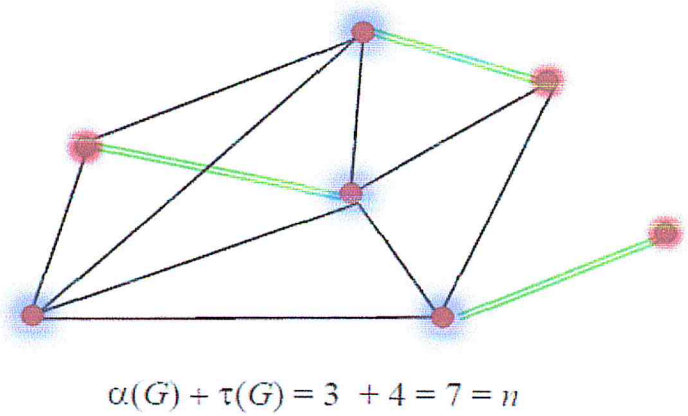


Figure 3.13- Exemple d'un transversal minimum et un stable maximum dans G.

5.4 Le problème du couplage maximum [54] :

Un couplage (matching) est un ensemble d'arêtes deux à deux non adjacentes ; autrement dit, un couplage est aux arêtes d'un graphe ce qu'un stable est aux sommets.

Le problème consiste à trouver un couplage maximum :

Problème du couplage maximum (Maximum Matching)
Instance : Un graphe G = (V, E)
Objectif : Déterminer un couplage maximum de G

Les sommets incidents à une arête du couplage sont dits saturés, et les autres insaturés.

Un couplage est parfait (perfectmatching) si tous les sommets sont saturés.

On note généralement ν(G) la taille d'un couplage maximum de G.

Par définition du graphe adjoint L(G), un couplage dans G est un stable dans L(G), et on a donc l'égalité suivante :

$$\nu(G) = \alpha\,(L(G)).$$

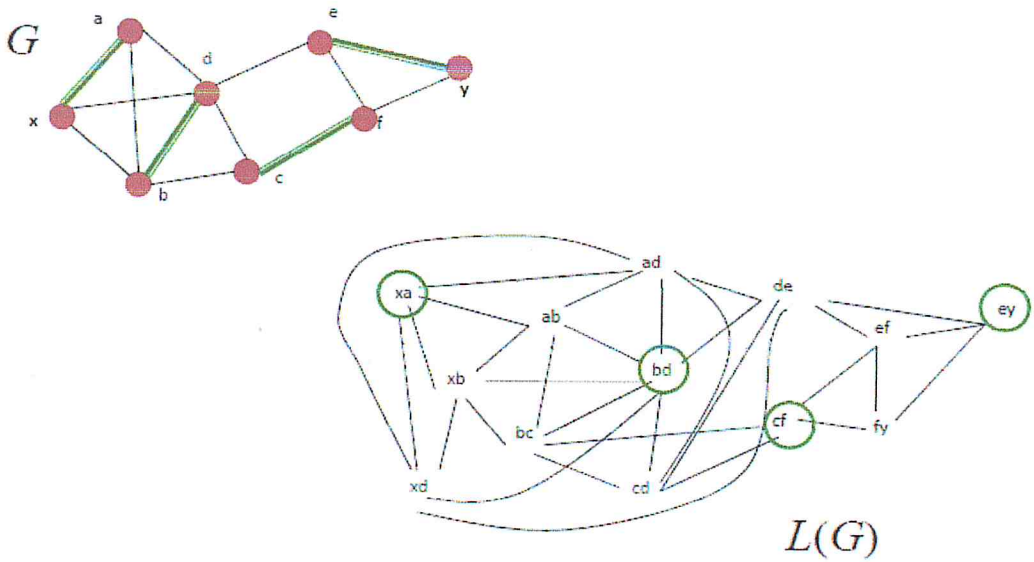
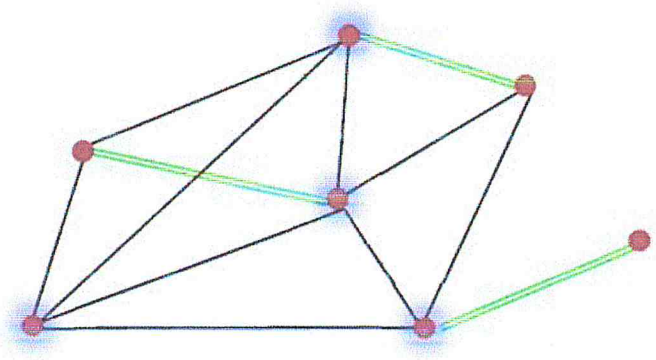


Figure 3.14 Exemple d'un couplage dans G est un stable dans L(G).

De plus, comme un sommet ne peut couvrir plus d'une arête d'un couplage, on a la relation de dualité faible suivante : $\tau(G) \geq \nu(G)$.



$\tau(G) = 4 \geq \nu(G) = 3$

Figure 3.15- Exemple de $\tau(G) \geq \nu(G)$.

Cette inégalité peut être stricte (il suffit de considérer le triangle K_3), mais Konig [10] a démontré en 1931 qu'on a toujours l'égalité quand on considère des graphes bipartis. Ce classique résultat min-max, fut généralisé indépendamment la même année par Egerváry [11] aux graphes spondérés, si bien qu'il est désormais d'usage de l'appeler théorème de Konig-Egerváry :

Théorème : (Konig (1931), Egerváry (1931)) [51] Pour tout graphe biparti, la cardinalité d'un couplage maximum est égale à la cardinalité d'un transversal (couverture) minimum :

$$\nu(G) = \tau(G).$$

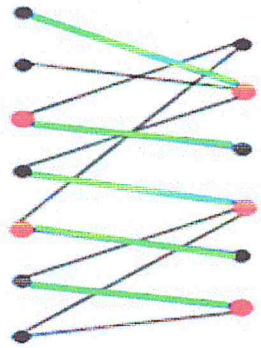


Figure 3.16 Couplage et Couverture dans un graphe biparti.

Théorème : (Frobenius (1917) [12]) Un graphe biparti G = (V, E) a un couplage parfait si et seulement si tout transversal est de cardinalité au moins $\frac{1}{2}|V|$.

Théorème : (Hall (1935)) Un graphe biparti G = (V, E), où V = S ∪ T (et |S| = |T|), a un couplage parfait si et seulement si $|X| \leq |N(X)|$ pour tout $X \subseteq S$.

6. Complexité.

Revenons à présent à notre question initiale : peut-on calculer efficacement un stable maximum dans un graphe ?

Un algorithme simple, puisqu'un stable maximum est aussi maximal, consiste à énumérer tous les stables maximaux du graphe et à en retenir un qui soit de cardinalité maximum. Le problème est qu'on est alors confronté à un phénomène d'explosion combinatoire, car un graphe peut avoir un nombre exponentiel (en fonction de son ordre) de stables maximaux : Moon et Moser [13] ont en effet montré qu'un graphe d'ordre n peut contenir jusqu'à $3^{n/3}$ cliques maximales, et par conséquent (en passant au complémentaire), qu'un graphe d'ordre n peut contenir jusqu'à $3^{n/3}$ stables maximaux.

Théorème : (Karp (1972) [8]) Le problème du stable maximum est NP- difficile.

Démonstration : Pour montrer que MIS est NP-difficile, il suffit de prouver que le problème de décision associé est NP-complet. Un moyen simple d'y parvenir est d'effectuer une réduction à partir du problème 3-SAT : on associe à chaque littéral de la formule booléenne un sommet d'un graphe (en le dupliquant éventuellement autant de fois qu'il apparaît dans ladite formule) et on relie ensuite entre deux sommets s'ils correspondent respectivement à un littéral et à sa négation, ou s'ils apparaissent dans la même clause. Par exemple, la figure 3.16 donne le graphe associé à la formule $\varphi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee x_4)$

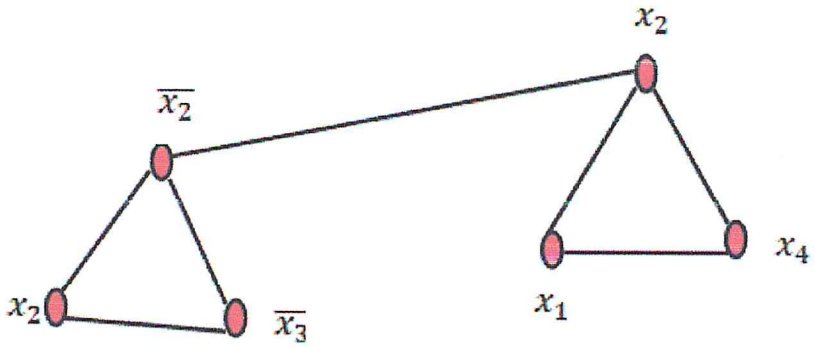


Figure 3.17- Le graphe associé à la formule φ.

Le graphe ainsi construit contient un ensemble stable de taille au moins k (k étant le nombre de clauses) si et seulement si la formule booléenne est satisfaisable. En effet :

⇐ Si l'on connaît une affectation de valeurs aux variables telle que la formule est vraie, on choisit dans chaque clause un littéral rendu vrai par cette affectation (et on peut le faire, puisque la formule est vraie).

L'ensemble constitué des sommets correspondants dans le graphe est bien stable (puisqu'il ne contient qu'un sommet de chaque clause et qu'aucune affectation ne rend vrais à la fois un littéral et sa négation) et est de taille k car la formule contient k clauses.

⇒ réciproquement, supposons que nous ayons un stable S de taille k ou plus. Pour les raisons que l'on vient d'évoquer, il ne peut contenir deux sommets correspondant à des littéraux opposés ou appartenant à une même clause, et comme la formule contient k clauses, chaque clause doit contenir un littéral correspondant à un sommet de S. Par conséquent, en donnant la valeur vraie aux littéraux correspondants aux sommets de S, la formule est satisfaite.

Il est facile de voir que cette réduction peut être effectuée en temps polynomial. Par conséquent, le problème du stable maximum est NP- difficile.

□

Corollaire : Les problèmes de la clique maximum, du stable de poids maximum, du dominant stable minimum et du transversal minimum sont eux aussi NP-difficiles.

7. Etat de l'art sur le nombre de stabilité

7.1 Cas particuliers pour le nombre de stabilité α(G) [50]

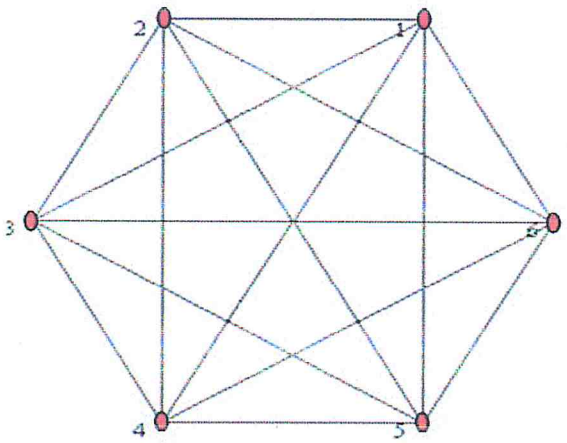
Le problème du stable maximum est NP-complet dans le cas général. Par contre si on se restreint à certaines classes de graphes, on connaît des algorithmes polynomiaux permettant de le résoudre.

Graphes complets[50]

Théorème : Pour le graphe complet K_n , On a :

$$\alpha(K_n) = 1$$

Exemple : Soit le graphe complet K_6 suivant :



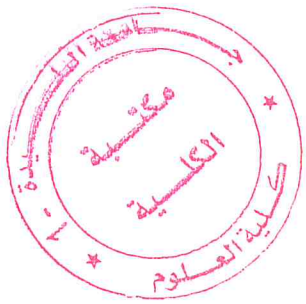


Figure 3.18- Graphe complet K_6.

Par définition, les sommets d'un graphe complet sont tous voisins deux à deux. La cardinalité des ensembles stables maximums possibles ne dépassera donc pas 1.

Pour ce graphe les différents ensembles stables maximums seront donc : {1}, {2}, {3}, {4}, {5}, {6}.

Il est donc aussi trivial de trouver les ensembles stables maximums sur une clique, celle-ci étant par définition un sous-graphe complet.

Graphes bipartis complet [50]:

Théorème : Soit $K_{m,n}$ un graphe biparti complet, on a :

$$\alpha(K_{m,n}) = max(m, n)\ .$$

Un graphe G est biparti s'il existe une partition de V(G) en deux classes V_1 et V_2 telle que toute arête relie un sommet de V_1 à un sommet de V_2.

Dans ce cas, V_1 et V_2 sont deux ensembles stables. Pour obtenir l'ensemble stable maximum, il suffit alors de prendre l'ensemble stable ayant la cardinalité maximum.

Exemple :

Soit G= (V, E) le graphe biparti complet suivant :

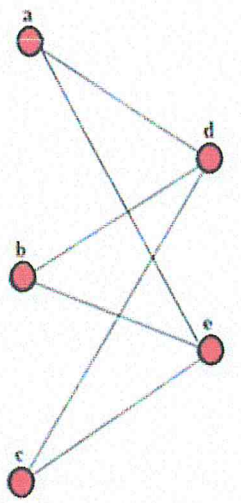


Figure 3.19- Graphe biparti complet $K_{3,2}$.

Ce graphe étant biparti, on en déduit les 2 ensembles stables suivants V_1= {a, b, c} | V_1 |=3

et V_2= {d, e} |V_2 |=2.

L'ensemble stable maximum est donc celui dont la cardinalité est maximum soit V_1.

Les graphes k-partis [50]:

Théorème : Soit $K_{n1,n2,\dots,nk}$ un graphe k-partite, on a :

$$\alpha(K_{n1,n2,\dots,nk}) = \max(n_1, n_2, \dots, n_k).$$

7.2 Quelques exemples de classes MIS-faciles (polynomiaux) [49]:

Malgré la NP-difficulté du problème dans le cas général et dans les cas évoqués ci-dessus, il existe de nombreuses classes de graphes pour lesquelles on sait trouver un stable maximum de manière exacte en temps polynomial.

7.2.1 Graphes bipartis

Les résultats montrent que dans un graphe biparti, le problème du stable maximum se ramène à celui du couplage maximum, puisqu'on a alors

$\nu(G) + \alpha(G) = n$.

L'obtention d'un couplage dans un biparti a été décrite par Konig dans la démonstration du théorème de Konig-Egerváry et repose sur le concept de chaîne augmentant, introduit 40 ans plus tôt par Petersen :

Définition : (Chaîne augmentante) Etant donné un couplage M d'un graphe G = (V, E),

- une chaîne M-alternée (alternatingpath) est une chaîne (non nécessairement induite) dont les arêtes appartiennent alternativement à M et à E\M
- une chaîne M-augmentante (augmentingpath) est une chaîne alternée dont les extrémités sont M-insaturées.

Lemme : (Petersen (1891), Konig (1931), Berge (1957), Norman et Rabin (1959) [16]) Un couplage est maximum si et seulement s'il n'existe aucune chaîne augmentante.

Principe des algorithmes efficaces:
• Construction de couplages successifs
• Etape $i \rightarrow M_i$
Rechercher l'ensemble le plus grand possible (t) de chaînes augmentantes de longueur minimale et sommets-disjointes : $\mu_1, \mu_2, \ldots, \mu_t$.
$M_{i+1} = M_i \Delta \mu_1 \Delta \mu_2 \Delta \ldots \Delta \mu_t$.

Algorithme glouton du couplage maximum dans un graphe biparti [10]:

(In G=(X, Y, A) biparti; Out M: couplage maximum):

Début
Déterminer un couplage maximal M ;
Si X ou Y saturé alors FIN: M est maximum.
Tant que Non FIN faire
marquer + tout sommet de X insaturé ;
Tant que marquage possible et pas de chaîne augmentante trouvée faire
- marquer +x tout sommet de Y relié à un sommet x déjà marqué par une arête insaturée;
- marquer -y tout sommet de X relié à un sommet y déjà marqué par une arête saturée;
- si on a marqué un sommet y* de Y non saturé, alors on a trouvé une chaîne augmentante.

Si pas de chaîne augmentante trouvée alors FIN: M est maximum
Sinon soit μ une chaîne augmentante.
M := couplage obtenu par un transfert dans M le long de μ.
Effacer tous les marquages.
Fin si ;
Fin.

Déroulement de l'algorithme : soit le graphe G suivant :

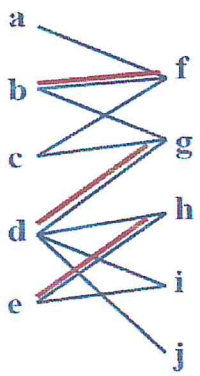


Figure 3.20- M_1 couplage maximal dans G.

$\mu_1 = [a, f, b, g, d, i]$ Chaine augmentante, $M_2 = M_1 \Delta \mu_1 = (M_1 \cup \mu_1) \setminus (M_1 \cap \mu_1)$ couplage maximum.

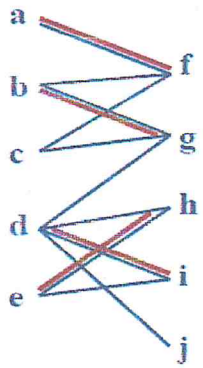


Figure 3.21- M_2 couplage maximum dans G.

Théorème :

Dans un graphe biparti, on peut calculer un stable maximum en temps $O(\sqrt{n}m)$.

Corollaire (König)[49] : Tout graphe k-régulier biparti admet k couplages parfaits disjoints.

7.2.2 Graphes adjoints.

Il est facile de déterminer un couplage maximum dans un graphe biparti. On va maintenant montrer comment trouver en temps polynomial un couplage maximum dans un graphe quelconque.

Ceci implique l'existence d'un algorithme polynomial pour le problème du stable maximum dans la classe des graphes adjoints ; en effet, comme nous l'avons vu précédemment, un stable dans un graphe adjoint L(G) correspond à un couplage dans le graphe original G.
En appliquant l'algorithme d'Edmonds au graphe G, on obtient simultanément un stable maximum de L(G).

Notation :
Soit G= (V, E) un graphe et soit W un sous-ensemble de sommets. On notera G/W le graphe contracté ayant pour ensemble de sommets V-W+ {w} (on remplace W par un sommet w) et où deux sommets x et y sont reliés par une arête si $w \neq x, y$ et il existait une arête entre x et y dans G, ou si x=w et y était relié à au moins un sommet de W dans G.
θ(G) le plus petit nombre de cliques nécessaires pour recouvrir les sommets de G.

Propriété :
Soit W l'ensemble des sommets d'un cycle élémentaire impair dans G= (V, E), soit C^- un couplage dans G/W. Si w est insaturé dans C^- alors posons $C^+ = C^-$; sinon, soit C^+ le couplage de G obtenu à partir de C^- en remplaçant l'arête incidente à w par une arête incidente à un

sommet de W. Il existe alors un couplage maximum C^W dans G [W] tel que $C=C^+ \cup C^W$ est un couplage dans G.

Définition:

Soit G = (V, E) un graphe, soit C un couplage dans G et soit x un sommet insaturé. Un arbre alterné de racine x par rapport à un couplage C, est un sous-graphe partiel T = (V', E') connexe sans cycle de G tel que.

- x est le seul sommet insaturé de l'arbre
- Pour tout y dans T, l'unique chaîne reliant x à y est une chaîne alternée
- Pour tout sommet pendant y, l'unique arête incidente à y fait partie de C.

Si le nombre d'arêtes reliant x à un sommet y de T est pair, on dira que y est pair, sinon y est impair.

Remarque : dans un arbre alterné, le degré de chaque sommet impair est 2.

On va maintenant construire un arbre alterné. Nous noterons V^{pair} les sommets pairs de cet arbre. Au début, la construction de l'arbre se fera sur G. Mais après un certain nombre d'étapes, la construction pourra se poursuivre sur un graphe G' obtenu à partir de G à la suite de contractions sur divers cycles élémentaires impairs. Nous noterons $L_{y,z}$ l'unique chaîne de l'arbre alterné reliant y à z.

Algorithme de construction d'un arbre alterné ou détection d'une chaîne augmentante dans G :

1) V' :=V^{pair} := {x}; E' :=∅; G' :=G; Considérer chaque sommet de G comme non marqué;
2) S'il existe un sommet y∈V^{pair} et un sommet z insaturé non marqué tel que z est relié à y dans G', Alors STOP : en rajoutant l'arête [y, z] à $L_{x,y}$, on obtient une chaîne alternée augmentante.
3) S'il existe un sommet y∈V^{pair} et un sommet z saturé non marqué tel que z est relié à y dans G', Alors soit [z, w] l'arête du couplage incidente à z : poser V' :=V'∪ {z,w}; V^{pair}:= V^{pair} ∪ {w}, E' :=E'∪{{y, z},{z, w}}, marquer z et w et aller à 2)
4) S'il existe deux sommets y et z ∈V^{pair} tel que y est relié à z dans G' alors considérons l'ensemble W des sommets du cycle élémentaire impair L_{yz} ∪ {[y, z]}. Soit r le sommet où les chaînes L_{xy} et L_{xz} se rejoignent : on dira que W est une orbite de racine r.
Soit T = (V', E'∪ {[y, z]}) : poser T:=T/W, G':=G'/W, mettre le sommet remplaçant W dans V^{pair} et aller à 2)
5) STOP.

Algorithme de recherche d'un couplage maximum dans un graphe G quelconque
1) Choisir un couplage initial C.
2) S'il y a moins de deux sommets insaturés, alors STOP : le couplage C est maximum
Sinon, soit x un sommet insaturé : construire un arbre alterné de racine x
3) Si l'algorithme de construction d'un arbre alterné détecte une chaîne augmentante dans G', alors poser C' égal au couplage obtenu par transfert le long de cette chaîne augmentante, reconstituer le couplage correspondant dans G en décontractant les cycles impairs, poser C égal à ce nouveau couplage et retourner à 2)

4) Sinon supprimer de G tous les sommets marqués. Et retourner à 2)

À l'étape 3, lorsqu'il s'agit de reconstituer un couplage dans G à partir d'un couplage C' dans G', on s'y prend comme suit. Soit W une orbite de racine r qui a été transformée en un sommet w. Soit C^W l'ensemble des arêtes de C qui sont dans G [W]. Le couplage C^W sature tous les sommets de G [W] sauf la racine r. Si C' ne passe pas par w alors on peut décontracter le cycle et rajouter C^W à C'. Sinon, C' contient une arête [z, w]. Si en décontractant le cycle, le sommet z est relié à r, alors on peut remplacer [z, w] par [z, r] et rajouter C^W à C'. Sinon, on peut remplacer [z, w] par une arête [z, y] où y est un voisin de z dans W, et on effectue un transfert de C^W le long de l'unique chaîne paire qui mène de y à r dans G[W] (en d'autres termes, on remplace le couplage maximum C^W dans G[W] par un couplage maximum laissant y insaturé au lieu de r). Les différents cas cités ci-dessus sont illustrés ci-dessous.

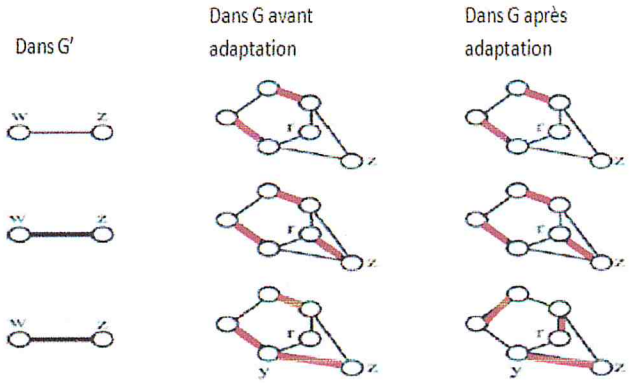


Figure 3.22- Couplage maximum dans un graphe G quelconque.

Théorème [49] : Dans un graphe adjoint, on peut trouver un stable de cardinalité maximum en temps $O(\sqrt{n}m)$.

7.2.3 Graphes sans griffe

En 1980, Minty [23] et Sbihi [24,25] ont, indépendamment, montré que l'on peut calculer un stable maximum en temps polynomial dans la classe des graphes sans griffe.

L'algorithme de Sbihi (de complexité $O(n^3)$) est une extension de la technique de contraction des blossoms utilisée dans l'algorithme d'Edmonds pour trouver un couplage maximum; l'algorithme de Minty réduit le problème à celui du couplage maximum dans un graphe auxiliaire appelé graphe d'Edmonds (l'un pouvant être obtenu à partir de l'autre en temps polynomial). Dans les deux cas, le principe consiste à déterminer des chaînes augmentantes de sommets, une idée qui sera par la suite étendue au concept de graphe augmentant.

Théorème : Dans un graphe sans griffe, on peut trouver un stable de cardinalité maximum en temps $O(n^3)$.

7.2.4 Graphes parfaits

De nombreuses autres classes, historiquement très étudiées, disposent d'algorithmes polynomiaux, voire linéaires, permettant de résoudre le problème du stable maximum :

- les graphes cordaux (triangulés) (qui contiennent notamment les graphes à seuil (threshold graphs), introduits par Chvátal et Hammer [26,27]) : un algorithme pour MIS a été donné par Gavril [28].
- les graphes de comparabilité (comparability graphs) (qui contiennent les bipartis, les graphes de permutation, les cographes (graphes sans P_4) et également les graphes à seuil) (voir par exemple [29]).
- les graphes de co-comparabilité (complémentaires des graphes de comparabilité, et qui contiennent **les graphes d'intervalles** et les cographes) [30].

Tous ces graphes sont des cas particuliers de graphes parfaits, évoqués au chapitre 1. En 1975, Chvátal [31] a décrit le polytope des stables des graphes parfaits. Puis, en 1981, Grötchel, Lovász et Schrijver [32, 33] ont montré les conséquences en optimisation combinatoire de la méthode de l'ellipsoïde. En particulier, ils ont introduit un algorithme qui résout en temps polynomial le problème de séparation associé au polytope des stables, ce qui a pour conséquence immédiate le théorème suivant :

Théorème : Dans un graphe parfait, on peut trouver un stable de cardinalité maximum en temps polynomial.

Déterminations des stables maximums dans les graphes d'intervalle :

- Un ensemble stable (ou stable) dans un graphe d'intervalle peut donc être vu comme un ensemble d'intervalles deux à deux disjoints

Donc chercher un stable maximum dans un tel graphe = choisir un ensemble de cardinalité maximum d'intervalles deux à deux disjoints.

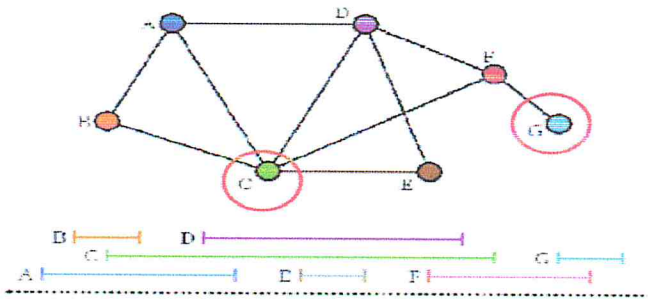


Figure 3.23- Stable de cardinale 2 dans GI.

- Dans cet exemple, {C, G} = stable de cardinal 2, et {A, E, F}, {B, D, G} = deux stables maximums

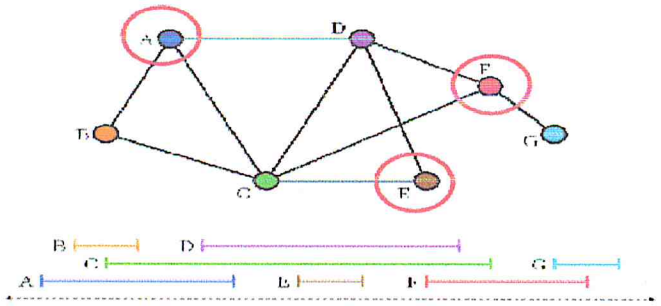


Figure 3.24- Stable maximum {A, E, F} dans GI.

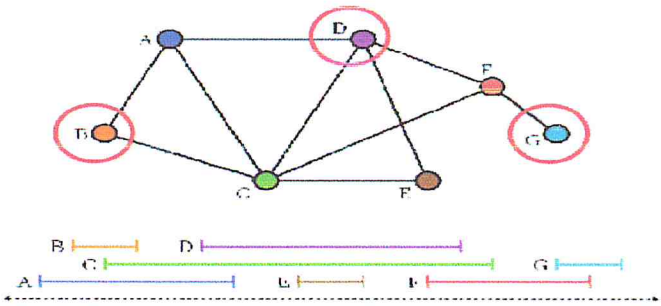


Figure 3.25- Stable maximum {B, D, G} dans GI.

Dans le cas général, comment déterminer un stable maximum dans de tels graphes ?

Partie 1: Dualité en PL et matrices totalement unimodulaires :

Totale unimodularité

Définition: une matrice à valeurs réelles est dite totalement unimodulaire (TU) si et seulement si le déterminant de toute sous-matrice carrée qui en est extraite vaut 0, 1 ou -1.

Propriété1 : toute sous-matrice d'une matrice TU est TU, et en particulier tout élément d'une. matrice TU vaut 0, 1 ou -1 (sous- matrice carrée extraite ayant une ligne et une colonne)

Propriété2: si une matrice est TU, alors la juxtaposition de cette matrice avec la matrice identité est aussi une matrice TU.

Propriété3: une matrice est TU si et seulement si sa transposée l'est.

Caractérisation de Ghouila-Houri: une matrice est TU ssi tout sous- ensemble de colonnes peut être partitionné en deux parties telles que, sur chaque ligne, les sommes des éléments sur chacune de ces deux parties diffèrent d'au plus 1.

Corollaire de la Propriété 3: ce théorème est aussi vrai en échangeant «colonne» et «ligne».

Conséquence de la totale unimodularité

Propriété4: si un PL a une matrice des contraintes TU et des seconds membres entiers, alors toute solution de base de ce PL est entière (et, en particulier, toute solution de base optimale l'est).

Corollaire de la Propriété 4:

Résoudre un PLNE ayant une matrice des contraintes TU et des seconds membres entiers est équivalent à résoudre sa relaxation continue (= un PL), et les valeurs optimales sont les mêmes.

Partie 2: Totale unimodularité et algorithmes combinatoires: les approches primales-duales :

Formuler le problème du stable maximum dans les graphes d'intervalle

Bonne formulation pour ce problème ?

Formulation «classique» pour le stable maximum :

- Une variable 0-1 x_i pour tout sommet (ou intervalle) i, indiquant si i est choisi (=1) ou non (=0).
- Une contrainte $x_i+x_j \leq 1$ pour toute arête (i, j).
- On maximise la somme des x_i.

Mais la matrice des contraintes de cette formulation n'est pas toujours TU (exemple du cycle à 3 sommets, comme ABC).

Formulation «alternative» pour le stable maximum :

- Notons $[d_i, f_i] = i^{ème}$ intervalle, et $e_j = j^{ème}$ plus petit élément de l'union des ensembles $\{d_i, f_i\}$.
- Découpage de l'horizon de temps en 2n-1 fenêtres (ou moins), chacune associée à la période $[e_j, e_{j+1}]$ pour un j.
- Mêmes variables.
- En revanche, il y a désormais une contrainte par fenêtre.

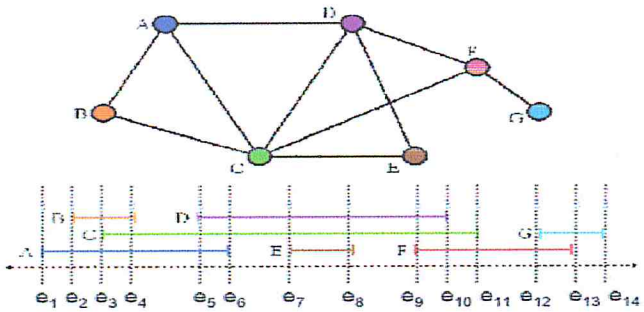


Figure 3.26- Découpage de l'horizon de temps en 2n-1 fenêtres de GI.

Propriétés de cette formulation

- Tout intervalle contient des fenêtres consécutives.
- Sur chaque colonne, les «1» sont donc consécutifs.
- Propriété des 1 consécutifs → matrice d'intervalle Or, toute matrice d'intervalle est TU

- Le problème est donc polynomial par la PL.
- Le problème dual entier consiste à choisir un nombre minimum de fenêtres intersectant tous les intervalles.

Algorithme primal-dual pour le problème du stable maximum dans les graphes d'intervalle :

Faire mieux que la PL ?

- En réalité, dans ce cas, on peut même concevoir un algorithme combinatoire glouton (rapide) pour ce problème
- On va en fait décrire une paire d'algorithmes gloutons (les deux peuvent être implémentés en O(n), où n=nombre d'intervalles)
- Ces algorithmes construisent une solution «primale» et une solution «duale» en se basant sur les relations d'exclusion du 2ème PL et de son dual
- On appelle ces approches «primales-duales», et c'est une méthode de conception d'algorithmes particulièrement utile

Pseudo-code de l'algo.glouton pour le stable max. dans un GI :

- Calculer une représentation par intervalles du GI
- Trier et numéroter les intervalles par ordre croissant de leurs dates de fin
- Intervalle courant := premier intervalle
- Tant que non STOP faire
- Sélectionner l'intervalle courant
- S'il n'existe plus d'intervalle ayant une intersection vide avec les intervalles déjà sélectionnés, alors STOP
- Sinon, intervalle courant := premier intervalle n'ayant aucune intersection avec les intervalles déjà sélectionnés

Dans ce premier algorithme, les numéros des intervalles sélectionnés au cours de l'exécution sont donc croissants

Pseudo-code de l'algorithme glouton pour le PL dual :

- Exécuter l'algorithme précédent (à la fin de ce dernier, une fenêtre est dite «active» si elle est contenue dans l'un des intervalles sélectionnés)
- Pour chaque intervalle sélectionné par l'algorithme précédent
- Sélectionner la fenêtre la plus à droite contenue dans cet intervalle

Exemple de déroulement de l'algorithme primal-dual :

Solutions renvoyées par l'algorithme sur l'exemple :

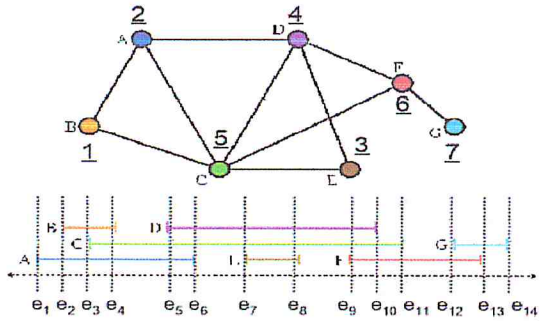


Figure 3.27- Numérotation des intervalles par ordre croissant de leurs dates de fin dans GI.

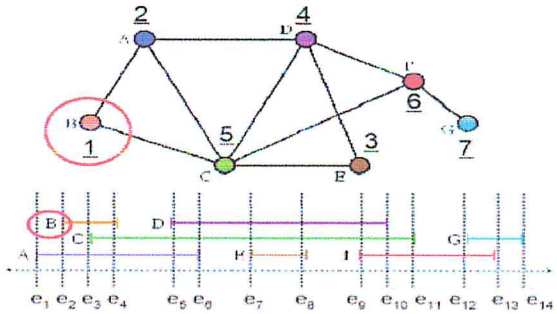


Figure 3.28- Sélectionnaient le premier intervalle courant.

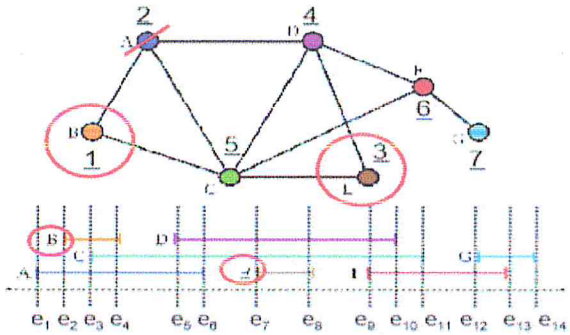


Figure 3.29- Sélectionnaient le deuxième intervalle courant.

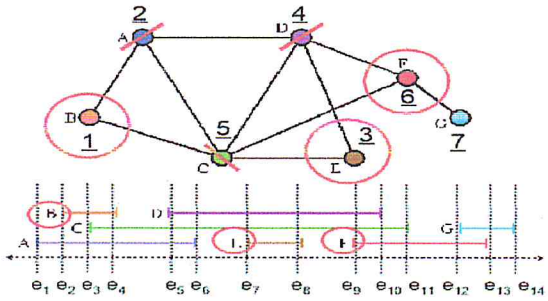


Figure 3.30- sélectionnaient le troisième intervalle courant.

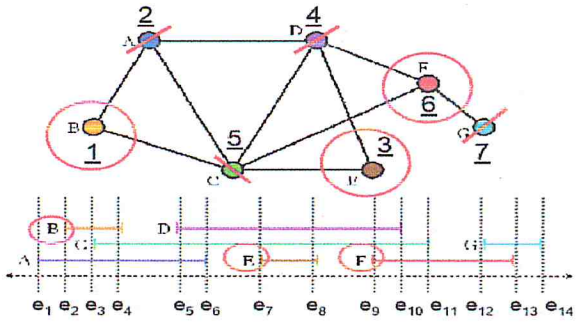


Figure 3.31- Intervalle courant ∄.

D'où le stable maximum est {B, E, F}.

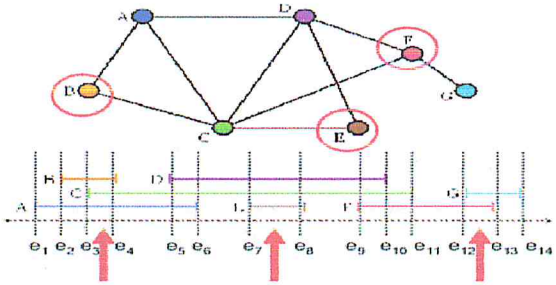


Figure 3.32- Stable maximum dans un GI.

Bilan sur les approches primales-duales

- Les propriétés d'une bonne formulation peuvent parfois permettre d'établir la polynomialité du problème étudié.
- Souvent, cela n'est qu'une première étape dans la résolution pratique du problème, et signifie en fait qu'il existe un algorithme combinatoire efficace pour résoudre ce problème sans passer par la PL.
- Un algorithme primal-dual est un bon candidat dans ce cas-là, puisque ses «choix» sont guidés par les conditions des écarts complémentaires (relations d'exclusion) du PL associé.
- Dans certains cas, cela ne marche pas si bien :
- **Stables de poids maximum dans un graphe d'intervalle (dans notre exemple, si C a un poids de 3, et tous les autres 1, alors la valeur optimale est 4)** (chapitre 4).
- Enfin, ce genre d'approches peut parfois être utilisé également pour obtenir des solutions approchées à des problèmes difficiles.

7.2.5 Graphes série-parallèles

En 1978, M-BOULALA et JP-UHRY [42] ont, indépendamment, montré que l'on peut calculer un stable maximum en temps polynomial dans la classe des graphes série-parallèles.

Théorème : Dans les graphes série-parallèles, on peut trouver un stable de cardinalité maximum en temps polynomial.

7.2.6 Graphes sans P_3

Un graphe sans P_3 est un graphe pour lequel chaque composante connexe est une clique. Ici encore, on résout donc MIS en temps polynomial de manière triviale, en choisissant un sommet dans chaque clique.

7.2.7 Graphes sans chaise

Théorème : (Alekseev (1999), Lozin et Milanic (2006)) Dans un graphe sans chaise, on peut calculer un stable maximum en temps $O(n^7)$.

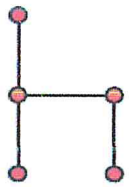


Figure 3.33- La chaise.

7.2.8 Graphes planaires

Théorème : Dans un graphe planaire, on peut calculer un stable maximum en temps polynomial.

7.2.9 Graphes pseudo-scindés

Théorème : (Hertz [43]) Dans un graphe pseudo-scindé, on peut calculer un stable maximum en temps $O(n^2)$.

7.2.10 Graphes sans P_5

Théorème : (Mosca (1997)) Dans un graphe G d'ordre n sans P_5, on peut calculer un stable maximum en temps $O(n^4)$.

7.2.11 Graphes sans P_5 biparti

Proposition : On peut calculer un stable maximum dans un graphe sans P_5 biparti en temps $O(n)$.

Famille de Graphes	Le temps de calcul d'un stable maximum (MIS)
Graphes bipartis	$O(\sqrt{n}m)$ [49]
Graphes adjoints	$O(\sqrt{n}m)$ [49]
Graphes sans griffe (sans étoile)	$O(n^3)$ [49]
Graphes parfaits	Polynomial [49]
Graphes série-parallèles	Polynomial [49]
Graphes sans P_3	Polynomial [49]
Graphes planaires	Polynomial [49]
Graphes sans chaise	$O(n^7)$ [49]
Graphes pseudo-scindés	$O(n^2)$ [49]
Graphes sans P_5	$O(n^4)$ [49]
Graphes sans P_5 biparti	$O(n)$ [49]

Tableau 3.34 Graphes dont le stable maximum se détermine en temps polynomial.

8. Encadrement du nombre de stabilité α(G).

8.1 Bornes supérieures :

Proposition : Soit G= (V, E) un graphe non pondéré, alors

$$\alpha(G) \leq |V(G)| + 1 - \chi(G).$$

Preuve : Considérons S un stable de V de cardinalité α(G). Une coloration possible des sommets consiste à colorer les sommets de S d'une même couleur et les |V(G)|- α(G) autre sommets de couleurs toutes différentes. On en déduit que : $\alpha(G) \leq 1 + (|V(G)| - \chi(G))$.

□

8.2 Bornes inférieures :

Borne inférieure 1 :

Proposition : Soit G= (V, E) un graphe non pondéré, alors

$$\alpha(G) \geq |V(G)| / \chi(G).$$

Preuve : Nous savons qu'une coloration des sommets est une partition $\{V_1, V_2, \ldots, V_k\}$ de V. De plus pour tout i, $1 \leq i \leq \chi(G)$, V_i est un ensemble stable et nous avons $|V_i| \leq \alpha(G)$. Donc

$|V| = |V_1 \cup V_2 \cup \ldots \cup V_K| = \sum |V_i| \leq \chi(G)\, \alpha(G)$, $1 \leq i \leq \chi(G)$. D'où l'inégalité.

□

Borne inférieure 2[22]:

Le théorème suivant est connu sous le nom de théorème de Turan [19].

Théorème : Pour tout graphe non pondéré G= (V, E), On a :

$$\alpha(G) \geq \sum_{v \in V} \frac{n}{d_G + 1}$$

Erdos a montré que pour les graphes non pondérés l'algorithme MIN atteint la limite ci-dessus [20]. L'extension suivante du théorème 3.1 a été prouvée d'abord par Wei et plus tard par Alon et Spencer dans une manière différente de Wei.

Borne inférieure 3[22]:

Théorème : Pour tout graphe non pondéré G= (V, E), On a :

$$\alpha(G) \geq \sum_{v \in V} \frac{1}{d(v)+1}$$

Conclusion du chapitre 3:

Dans ce chapitre, nous avons rappelé le problème du stable maximum et les problèmes connexes à ce problème, nous avons vu la complexité et quelque exemple de MIS-faciles (polynomiaux).

Chapitre 4

Problème du stable de poids maximum

1. Problème du stable de poids maximum :

Dans ce chapitre nous étudions le problème du stable de poids maximum (Maximum Weighted Independent Set (MWIS) en abrégé) qui est une généralisation du problème du stable maximum (Maximum Independent Set(MIS)).

Avant de définir le problème MWIS, on donne les définitions suivantes :

Soit G= (V, E) un graphe, on note (G, ω) le graphe pondéré où ω : V→ $\mathbb{N}^*$est une fonction poids des sommets.

Pour un sous ensemble S de V, on définit le poids de S par : ω(S)=$\sum_{v\in S}$ ω(v). La valeur d'un stable de poids maximum dans (G, ω) sera noté $\alpha_\omega(G)$ et parfois $\alpha(G,\ \omega)$.

Le problème MWIS est définit comme suit :

Problème du stable de poids maximum (MWIS)
Instance : Un graphe G = (V, E) et une fonction poids ω : V → $\mathbb{N}^*$
Objectif : Déterminer un stable de poids maximum dans G

2. Exemple et application [49] : Dans l'application du chapitre 3 Supposons à présent que le responsable d'une agence de tourisme émette des préférences sur certaines activités; ceci se traduit sur le graphe en affectant des poids aux sommets. La solution au problème correspond alors à un stable de poids maximum.

3. Formulation (PLNE) du problème MWIS [49] :

$$\text{Maximiser } \alpha_\omega(G) = \sum_{i=1}^{n} \omega_i v_i$$

$$v_i + v_j \leq 1 \ \forall \{i,j\} \in E$$

$$v_i \in \{0,1\}\}, i = \{1,2,\dots,n\}.$$

Où ω est le vecteur des poids des sommets (bien sûr, si toutes ses composantes sont égales, on retombe dans le cas du stable de cardinalité maximum), v est le vecteur caractéristique d'un stable (sa i-^{ème} composante v_i vaut 1 si le sommet v_i appartient au stable considéré, 0 sinon) et la première contrainte exprime le fait qu'au plus une extrémité de chaque arête appartient au stable.

Remarque : Si ω(v) = 1 pour tout v ∈ V, le problème MWIS est réduit au problème MIS.

Le problème MIS étant un sous problème NP-diffcile [8] de MWIS. Ceci implique que le problème MWIS est aussi NP-difficile, et pour cela nous proposons dans ce qui suit, trois algorithmes approchés de type glouton, pour la résolution du problème MWIS.

4. Algorithmes de résolution approchée [22] :

Premier Algorithme :

Algorithme WStableMax1

Entré : Un graphe pondéré (G, ω)
Sortie : Un stable maximal dans G

Début
I≠ Ø;i=0; G_i=G ;
Tant que V(G_i) ≠ Ø faire
Choisir un sommet v_i de G_i ;
I := IU {v_i} ;
G_{i+1} := G_i[V(G_i)-N_{Gi}[v_i]] ;
i := i+1 ;
fin Tant que ;
Sortie I ;
Fin.

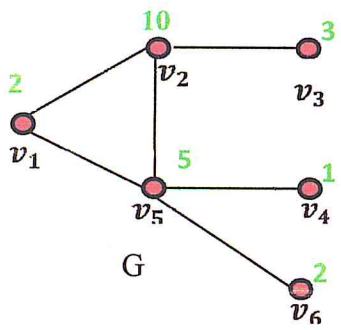
L'algorithme W Stable Max1 peut être vu comme une classe d'algorithmes approchés basés sur le choix des $v_i \in G_i$.

Par exemple, si on choisit un sommet v_i de G_i de degré minimum, on retrouvera l'algorithme MIN (du chapitre 3).

Par contre si on choisit v_i de poids maximum dans G_i, on obtient un autre algorithme.

Illustration de l'Algorithme WStableMax1 :

Version1 : On choisit v_i dans G_i de poids maximum (les poids des sommets sont représenté à côté des sommets).



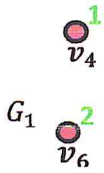
G

Itération 0 : I=Ø ; i=0 ; G_0=G ;

V(G_0)= V(G) ≠ Ø ; On choisit le sommet 2.

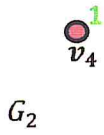
I={2} ;$N_{G0}(2)$={1, 5, 3}

il reste le graphe G_1 suivant :



i=1 ;

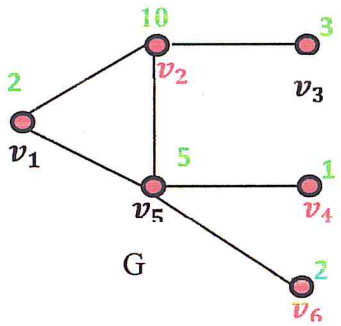
itération 1 : $V(G_1)$= {4,6}≠ ∅;on choisit le sommet 6 ;; G_2= $G_1[V(G_1)-\{6\}]=G_1[4]$;



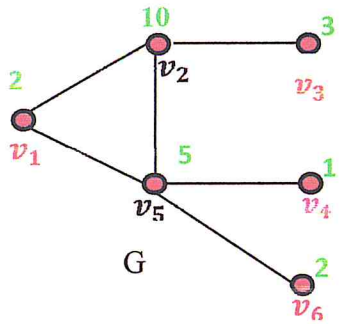
i=2;

Iteration 2: V (G_2) = {4} ≠ ∅; on choisit v_4; I= {2, 6, 4}.

Le stable obtenue par la Version 1 de l'algorithme WStableMax1 est I_1={2, 6, 4}, ω(I_1)=13.



Version 2 de l'algorithme WStableMax1 : On choisit v_i dans G_i de degré minimum (qui est l'algorithme MIN du Chapitre 3)



L'algorithme MIN a été déjà appliqué à cet exemple au chapitre 3 est a produit le stable suivant : I_2= {3, 1, 6, 4}. ω (I_2)=8.

On remarque que la variante 1 donne le meilleur résultat pour cet exemple.

Deuxième Algorithme :

Algorithme WStableMax2

Entré : Un graphe pondéré (G, ω)
Sortie : Un stable maximal dans G

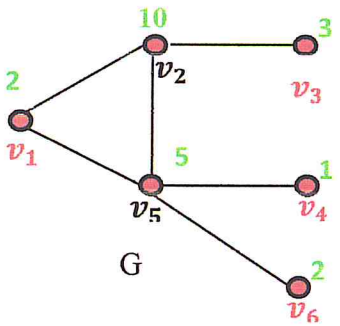
Début
I≠ ∅; i=0 ; G_i=G ;
Tant que E(G_i) ≠ ∅ faire
Choisir un sommet v_i de G_i ;
I := I∪{v_i};
G_{i+1}: = G_i[V(G_i)- {v_i}] ;
i: = i+1 ;
fin Tant que ;
I :=V(G_i) ;
Sortie I ;
Fin.

L'algorithme W Stable Max2 peut être vu comme une classe d'algorithmes approchés basés sur le choix des $v_i \in G_i$.

Si, dans cet algorithme on choisit v_i de degré maximum dans G_i, on retrouvera l'algorithme MAX du chapitre 3.

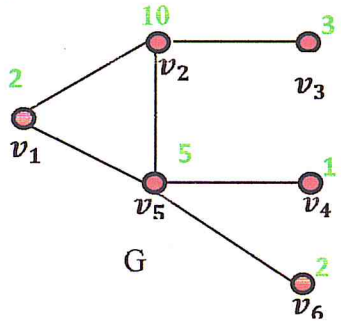
Illustration de l'Algorithme WStableMax2 :

Version 1 de l'algorithme WStableMax2 : On choisit v_i dans G_i de degré maximum (qui est l'algorithme MAX du Chapitre 3)



L'algorithme MAX a été déjà appliqué à cet exemple au chapitre 3 est a produit le stable suivant : I_2= {3, 1, 6, 4}. ω (I_2)=8.

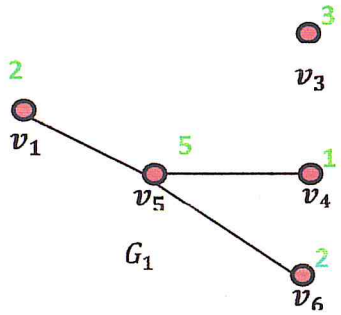
Version 2 de l'algorithme WStableMax2 : On choisit v_i dans G_i de poids maximum



Itération 0 : I=Ø ; i=0 ; G_0=G ;

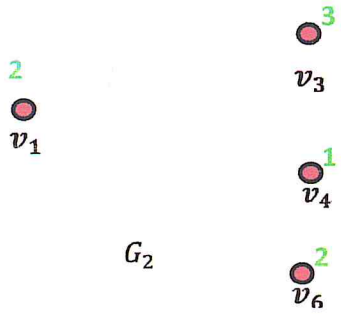
E(G_0)=E(G)≠ Ø; ω(2)=10, le sommet 2 est de degré maximum on le choisit.

G_1=G_0[V(G_0)-{2}]



i=0+1=1 ;

Itération1 : E(G_1)≠ Ø; ω(5)=5, le sommet 5 est de degré maximum on le choisit.
G_2=G_1[V(G_1)-{5}



i=1+1=2 ;

Iteration 2: E (G_2) =Ø.
I=V(G_2)={1, 3, 4, 6} ; ω(I)= ω(1)+ ω(3) + ω(4) + ω(6)=2+3+1+2=8.

On remarque que la variante 1 et la variante 2 donne le même résultat pour cet exemple.

Troisième Algorithme : L'algorithme glouton suivant est basé sur WStableMax1.

Algorithme WStableMax3

Entré : Un graphe pondéré (G, ω)
Sortie : Un stable maximal dans G

Début
I≠ ∅;i=0 ; G_i=G ;
Tant que V(G_i) ≠ ∅ faire
Choisir un sommet v_i de G_i maximisant $\frac{\omega(u)}{\sum_{w\in N_{Gi}[u]}\omega(w)}$;
I := IU {v_i} ;
G_{i+1}:= G_i [V (G_i) - $N_{Gi}[v_i]$] ;
i := i+1 ;
fin Tant que ;
Sortie I ;
Fin.

5. Bornes inférieures du poids d'un stable maximum α_ω(G) d'un graphe G [22] :

Borne inférieure 1[22]:

Théorème : Dans l'algorithme WStableMax1, si chaque v_i ($0\leq i \leq |I|$) vérifie $\omega(v_i)\geq \sum_{u\in N_{Gi}[vi]} \frac{\omega(u)}{d_{Gi(u)+1}}$ (un tel sommet v_i existe pour tout graphe G_i)

Alors WStableMax1 produit un stable de poids au moins égal à $\sum_{v\in V} \frac{\omega(v)}{d_G(v)+1}$.

Corollaire : Pour tout graphe pondéré (G, ω) [21], on a

$$\alpha_\omega(G) \geq \sum_{v\in V} \frac{\omega(v)}{d_G(v)+1}.$$

Preuve :

$$\omega(I)=\sum_{i=1}^{|I|}\omega(v_i) \geq \sum_{i=1}^{|I|}(\sum_{u\in N_{Gi}[vi]} \frac{\omega(u)}{d_{Gi(u)+1}})$$

$$\geq \sum_{i=1}^{|I|}(\sum_{u\in N_G[vi]} \frac{\omega(u)}{d_{G(u)+1}})$$

$$\geq \sum_{v\in V} \frac{\omega(v)}{d_G(v)+1} \quad .$$

□

Si dans l'algorithme WStableMax1 on choisit v_i maximisant $\frac{\omega(u)}{d_{Gi(u)+1}}$ sur l'ensemble des sommets de V(G_i) à chaque itération, l'algorithme obtenu est appelé GWMIN.

Corollaire : GWMIN produit un stable de poids au moins égal à $\sum_{v\in V} \frac{\omega(v)}{d_G(v)+1}$.

Cette borne peut aussi être obtenue par l'algorithme WStableMax2.

Théorème 2 : Dans l'algorithme WStableMax2, si chaque v_i $(0 \leq i < |V(G) - I|)$ vérifie $\sum_{u\in N_{Gi}(vi)} \frac{\omega(u)}{d_{Gi(u)}(d_{Gi(u)+1})} \geq \frac{\omega(v)}{d_{Gi}(v)+1}$ Et $d_{Gi}(v_i) \neq \emptyset$ (un tel sommet v_i existe pour tout graphe G avec $E(G) \neq \emptyset$), Alors WStableMax2 produit un stable de poids au moins égal à $\sum_{v\in V} \frac{\omega(v)}{d_G(v)+1}$.

Preuve :

Pour tout $(0 \leq i \leq |V(G) - I|)$ l'inégalité suivante tient :

$\sum_{u\in V(Gi+1)} \frac{\omega(u)}{d_{Gi+1}(u)+1} = \sum_{u\in V(Gi)} \frac{\omega(u)}{d_G(u)+1} - \frac{\omega(vi)}{d_{Gi}(vi)+1} + \sum_{u\in N_{Gi}(v)} \frac{\omega(u)}{d_{Gi(u)}(d_{Gi(u)+1})} \geq \sum_{u\in V(Gi)} \frac{\omega(u)}{d_G(u)+1}$.

Ainsi, $\omega(I) \geq \sum_{u\in V(G)} \frac{\omega(u)}{d_G(u)+1}$.

□

Si dans l'algorithme WStableMax2 on choisit v_i maximisant $\frac{\omega(u)}{d_{Gi(u)}(d_{Gi(u)+1})}$ sur l'ensemble des sommets $u \in V(G_i)$ à chaque itération, l'algorithme obtenu est appelé GWMAX.

Corollaire : GWMAX produit un stable de poids au moins égal à $\sum_{v\in V} \frac{\omega(v)}{d_G(v)+1}$

Borne inférieure 2[22]:

Théorème : L'algorithme WStablaMax3 produit un stable de poids au moins égal à $\sum_{v\in V(G)} \frac{\omega(v)^2}{\sum_{u\in N_{G[v]}} \omega(u)}$.

Corollaire : Pour tout graphe pondéré (G, ω), on a

$$\alpha_\omega(G) \geq \sum_{v\in V(G)} \frac{\omega(v)^2}{\sum_{u\in N_{G[v]}} \omega(u)}.$$

Preuve : Soit $I = \{v_1, v_2, \ldots, v_t\}$ le stable obtenu par l'algorithme, $f_G(v) = \frac{\omega(v)}{\sum_{u\in N_{Gi[v]}} \omega(u)}$

$$\sum_{i=1}^{t} \omega(v_i) = \sum_{i=1}^{t} \left(f_{Gi}(v_i) \times \sum_{u\in N_{Gi[vi]}} \omega(u) \right)$$

$\geq \sum_{i=1}^{t} \left(\sum_{u \in N_{Gi[vi]}} f_{Gi}(v_i)\ \omega(u) \right)$ (Pour $f_{Gi}(v_i) \geq f_{Gi}(u)\ \forall u \in V(G_i)$)

$\geq \sum_{v \in V(G)} f_G(v)\ \omega(v)$ (Pour $f_{Gi}(u) \geq f_G(u)\ \forall u \in V(G)$)

$$\geq \sum_{v \in V(G)} \frac{\omega(v)^2}{\sum_{u \in N_{G[v]}} \omega(u)}.$$

□

Si ω(v)= 1 pour tout v ∈V (G) (c'est-à-dire le graphe non pondéré), alors $\sum_{v \in V(G)} \frac{\omega(v)^2}{\sum_{u \in N_{G[v]}} \omega(u)}$ est égal à $\sum_{v \in V} \frac{1}{d(v)+1}$ qui est la limite du théorème de Turan (du chapitre 3).

Corollaire : Pour tout graphe pondéré (G, ω), on a

$$\alpha_\omega(G) \geq max\left(\sum_{v \in V} \frac{\omega(v)}{d_G(v)+1},\ \sum_{v \in V(G)} \frac{\omega(v)^2}{\sum_{u \in N_{G[v]}} \omega(u)} \right).$$

6. Quelques cas polynomiaux du problème MWIS :

Graphe biparti :

Dans un graphe biparti pondéré, la recherche d'un stable de poids maximum peut être menée à bien en utilisant un algorithme de flot maximum, par exemple l'algorithme d'Edmonds-Karp (de complexité $O(nm^2)$) [34], surpassé depuis par des algorithmes de complexité moindre que $O(n^3)$ ([35]).

Théorème : Dans un graphe biparti, On peut déterminer un stable de poids maximum en temps $O(n^3)$.

Graphes adjoints :

Un algorithme en $O(n^3)$ pour le couplage pondéré a également été développé par Gabow [36,37]; un algorithme de même complexité est donné dans les ouvrages de Lawler [239] et de Schrijver [35]. Ces complexités ont été plusieurs fois améliorées depuis, et nous invitons le lecteur à se reporter à cette dernière référence pour plus de précisions sur les algorithmes de couplages.

Théorème : Dans un graphe adjoint, on peut trouver un stable de poids maximum en temps $O(n^3)$.

Graphes sans griffe :

L'algorithme de Sbihi (de complexité $O(n^3)$) permet de calculer un stable maximum en temps polynomial dans la classe des graphes sans griffe se limite au cas de graphes non pondérés, celui de Minty (de complexité $O(n^7)$) permet de calculer un stable de poids maximum (bien que la version originale contienne une erreur qui n'a été corrigée qu'en 2001 par Nakamura et Tamura [39]). Pietropaoli et Stauffer [40] ont présenté en 2008 un nouvel algorithme pour le calcul d'un stable de poids maximum dans les graphes sans griffe, basé sur des décompositions et de complexité $O(n^6)$.

Théorème : Dans un graphe sans griffe, on peut trouver un stable de poids maximum en temps $O(n^6)$.

Graphes parfaits :

L'algorithme de Frank permet de trouver un stable de poids maximum dans les graphes cordaux [41] c'est un cas particuliers de graphes parfaits.

Théorème : Dans un graphe parfait, on peut trouver un stable de poids maximum en temps polynomial.

Graphes série-parallèles [42]

Théorème : Dans les graphes série-parallèles, on peut trouver un stable de poids maximum en temps polynomial.

Graphes sans chaise

Théorème : (Alekseev (1999), Lozin et Milanic (2006)) Dans un graphe sans chaise, on peut calculer un stable de poids maximum en temps $O(n^7)$.

Graphes planaires

Théorème : Dans un graphe planaire, on peut calculer un stable de poids maximum en temps polynomial.

Graphes scindés

Les graphes scindés ont été introduits par Földes et Hammer [44, 45] ; ce sont les graphes dont on peut partitionner les sommets en un stable et une clique

Théorème : Dans un graphe scindé, on peut résoudre MWIS en temps linéaire.

Graphes sans P_5

Théorème : (Mosca (2004)) on sait calculer un stable de poids maximum dans un graphe G sans P_5 pondéré d'ordre n en temps $O(n^3)$.

Graphes sans P_5 biparti

Proposition : On peut calculer un stable de poids maximum dans un graphe sans P_5 biparti en temps $O(n)$.

Graphes sans P_5 k-colorable

Théorème : (Maffray (2009)) Pour $k \geq 1$ fixé, il existe un algorithme de complexité $O(n^k)$ qui trouve un stable de poids maximum dans tout graphe sans P_5 k-colorable.

Famille de Graphes	Complexité du problème MWIS
Graphes bipartis	$O(n^3)$
Graphes adjoints	$O(n^3)$
Graphes sans griffe (sans étoile)	$O(n^6)$
Graphes parfaits	Polynomiale
Graphes série-parallèles	Polynomiale
Graphes planaires	Polynomiale
Graphes sans chaise	$O(n^7)$
Graphes scindés	Polynomiale
Graphes sans P_5	$O(n^3)$
Graphes sans P_5 k-colorable	$O(n^k)$
Graphes sans P_5 biparti	$O(n)$

Tableau 4.1 Graphes dont le stable de poids maximum peut être obtenus polynomiallement.

Chapitre 5

Rapport de performances

1. Rapports de performance du l'algorithme GWMIN

Donnons d'abord quelques définitions utiles :

Définition 1.1 Soit (G, ω) un graphe pondéré et A(G, ω) le poids du stable de poids obtenu par l'algorithme A.

Soit $\alpha(G,\ \omega)$ le poids d'un stable de poids maximum dans le graphe G.

Pour simplicité on omettra la lettre ω et on notera (G, ω), A(G, ω), α(G, ω) par G, A(G), α(G) respectivement.

Définition 1.2 Soit G un graphe pondéré et A un algorithme donnant un stable de poids maximum dans G.

Le rapport de performance de l'algorithme A est : $\rho_A = inf_G\{\frac{A(G)}{\alpha(G)}\}$.

2. Historique

Halldorsson et Radhakrishnan ont montré que pour les graphes à degrés bornés par Δ, l'algorithme MIN produit toujours un stable de cardinalité au moins $\frac{3}{\Delta+2}\times\alpha(G)$ où α(G) est la cardinalité d'un stable maximum dans G.

Ils ont aussi montré que le rapport de performance de cet algorithme est fin, c'est-à-dire il est atteint [46] .Griggs [47] et Chvatal et C.McDiarmind [48] ont établi que l'algorithme MAX (problème MIS) produit un stable de cardinalité au moins $\sum_{v\in V}\frac{1}{d_G(v)+1}$ pour tout graphe G. Ceci implique que l'algorithme MAX produit toujours un stable de cardinalité au moins $\frac{1}{\Delta+1}\times\alpha(G)$ pour les graphes à degrés bornés par Δ. Halldorsson et Radhakrishnan ont démontré que le rapports de performance est au plus $\frac{2}{\Delta+1}$ [46].

On rappelle que si dans l'algorithme WStableMax1, on choisit v_i maximisant la quantité $\frac{\omega(u)}{d_{Gi(u)+1}}$ sur V(G_i) à chaque itération, l'algorithme obtenu est appelé GWMIN. Le résultat suivant donne le rapport de performance de GWMIN.

Théorème 2.1 [22] $\rho_{GWMIN}=\frac{1}{\Delta}$.

Preuve : Montrons d'abord que $\rho_{GWMIN}\geq\frac{1}{\Delta}$ **...(0)**

Ceci revient à montrer que $\forall G$: $\frac{A(G)}{\alpha(G)}\geq\frac{1}{\Delta}$ ...(1)

On procède par induction .Soit G un graphe et I le résultat obtenu par l'algorithme GWMIN pour G.

Vérifions d'abord que la propriété (1) est vrai pour les graphes G tel que $|V(G)|\leq 2$.

On a 3 possibilités :

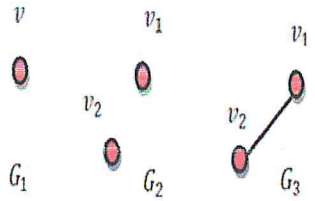


Figure 0 : Cas possibles pour $|V(G)| \leq 2$.

- pour le graphe G_1 : I=$\{v\}$,$A(G_1) = \omega(v), \alpha(G_1) = \omega(v)$. $\frac{A(G_1)}{\alpha(G_1)} = \frac{\omega(v)}{\omega(v)} = 1 \geq \frac{1}{\Delta}$.

- pour le graphe G_2 : $I = \{v_1, v_2\}$,$A(G_2) = \omega(v_1) + \omega(v_2), \alpha(G_2) = \omega(v_1) + \omega(v_2)$.$\frac{A(G_2)}{\alpha(G_2)} = 1 \geq \frac{1}{\Delta}$.

- pour le graphe G_3 : Supposons que v_1 est tel que : $\frac{\omega(v_1)}{d_{G3}(v_1)+1} \geq \frac{\omega(v_2)}{d_{G3}(v_2)+1} \Rightarrow I = \{v_1\}$

$\frac{\omega(v_1)}{1+1} \geq \frac{\omega(v_2)}{1+1} \Leftrightarrow \omega(v_1) \geq \omega(v_2)\ldots(2), A(G_3) = \omega(I) = \omega(v_1), \alpha(G_3) = \omega(v_1) \Longrightarrow \frac{A(G_3)}{\alpha(G_3)} = 1 \geq \frac{1}{\Delta}$.

Supposons maintenant que la propriété (1) est vraie pour les graphes G tel que

$|V(G)| \leq n-1$ et considérons un graphe G tel que $|V(G)| = n$.

Soit v un sommet que l'algorithme GWMIN a choisit, c'est-à-dire vérifiant la condition :

$\forall\, u \in V : \frac{\omega(u)}{d_G(u)+1} \leq \frac{\omega(v)}{k+1}\ldots(3)$ où $k = d_G(v)$.

Soient $V_1 = N_G[v]$ et $V_2 = V(G) - V_1$. On a besoin du Lemme suivant :

Lemme 2.2 [22] $\alpha(G[V_1]/\Delta) \leq \omega(v)$.

Preuve : On a deux possibilités :

1er cas : $\alpha(G[V_1]) \leq \omega(v) \Longrightarrow \frac{\alpha(G[V_1])}{\Delta} \leq \frac{\omega(v)}{\Delta} \leq \omega(v)$ le Lemme est vrai.

2ième cas : $\omega(v) < \alpha(G[V_1])$

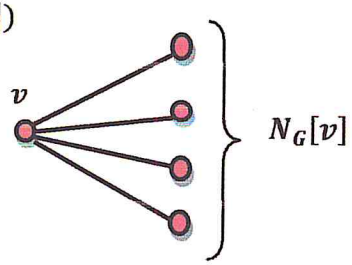


Figure 1 : Illustration du 2ième cas

Soit S^* un stable de poids maximum de G[V_1], donc on a soit $S^*= \{v\}$ ou $S^* \subseteq V_1 \setminus \{v\}$ voir figure 1. D'après la condition donnée du 2ième cas $S^* \neq \{v\}$, alors $S^* \subseteq V_1 \setminus \{v\} \Longrightarrow \omega(S^*) \leq \sum_{u \in V_{1-\{v\}}} \omega(u) \leq \sum_{u \in V_{1-\{v\}}} \frac{\omega(v)(d_G(u)+1)}{k+1}$ (d'après la formule 3).

Comme $k = d(v)$, d'après la figure 1 on a $k = |V_1 - \{v\}|$

$\Longrightarrow \alpha(G[V_1]) \leq (\Delta + 1)\omega(v) \sum_{u \in V_{1-\{v\}}} \frac{1}{k+1}$

$\Longrightarrow \alpha(G[V_1]) \leq \frac{k(\Delta+1)\omega(v)}{k+1} \Rightarrow \omega(v) \geq \frac{(k+1)\alpha(G[V_1])}{k(\Delta+1)}$.

$\frac{k+1}{k(\Delta+1)} = \left(1+\frac{1}{k}\right)\frac{1}{\Delta+1} \geq \left(1+\frac{1}{\Delta}\right)\frac{1}{\Delta+1} = \frac{1}{\Delta} \Longrightarrow \omega(v) \geq \frac{1}{\Delta}\alpha(G[V_1])$, ce qui termine la preuve du Lemme 2.2. □

Suite de la preuve du théorème 2.1 : notons S^* un stable de poids maximum de G

Posons $S_1^* = V_1 \cap S^*$, $S_2^* = V_2 \cap S^*$, on a $S^* = S_1^* \cup S_2^*$.

S_1^* est un stable dans $G[V_1]$, donc $\omega(S_1^*) \leq \alpha(G[V_1])$, de même $\omega(S_2^*) \leq \alpha(G[V_2])$

$\alpha(G) = \omega(S^*) = \omega(S_1^*) + \omega(S_2^*) \leq \alpha(G[V_1]) + \alpha(G[V_2])$. D'après le Lemme2.2 et l'hypothèse de récurrence on déduit : $\alpha(G)/\Delta \leq \alpha(G[V_1])/\Delta + \alpha(G[V_2])/\Delta \leq \omega(v) + \omega(I - \{v\}) = \omega(I) = A(G). \Rightarrow \frac{A(G)}{\alpha(G)} \geq \frac{1}{\Delta}$. ce qui termine la vérité de la condition (0). □

Montrons maintenant que $\rho_{GWMIN} \leq \frac{1}{\Delta}$. Pour cela, considérons le graphe biparti complet G_0 suivant :

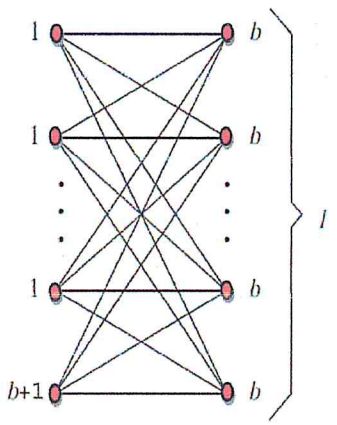


Figure 2 : Graphe G_0

Dans ce graphe on a $\Delta(G_0) = l = \Delta, b \gg l$. Les poids des sommets sont indiqués sur la figure.

$\alpha(G_0) = lb$, après le déroulement de l'algorithme GWMIN sur le graphe G_0 , on obtient le stable I_0 composé des sommets situés dont la partie gauche.

$$A(G_0) = (l + 1) + (b + 1) = l + b.$$

$\frac{A(G_0)}{\alpha(G_0)} = \frac{l+b}{lb}$ $\forall\, b \gg l$ on a $\frac{l+b}{lb} \geq \rho_{GWMIN}$ par passage à la limite

quand $b \longrightarrow \infty$, $\frac{l+b}{lb} \longrightarrow \frac{1}{l} = \frac{1}{\Delta}$.

D'où $\frac{1}{\Delta} \geq \rho_{GWMIN}$, ce qui achève la preuve de Théorème2.1. □

3. Rapport de performance de l'algorithme GWMAX

On rappelle que dans l'algorithme WStableMax2 si on choisit v_i maximisant $\frac{\omega(u)}{d_{Gi(u)}(d_{Gi(u)+1})}$ sur $V(G_i)$ à chaque itération, l'algorithme obtenu est appelé GWMAX.

Le résultat suivant est un encadrement du rapport de performance de l'algorithme GWMAX.

Théorème 3.1 [22] $\frac{1}{\Delta+1} \leq \rho_{GWMAX} \leq \frac{1}{\Delta}$.

Preuve : Il clair que $\frac{1}{\Delta+1} \leq \rho_{GWMAX}$ car $\sum_{v\in V} \frac{\omega(v)}{d_G(v)+1} \geq \sum_{v\in V} \frac{\omega(v)}{\Delta+1} \geq \frac{\alpha(G)}{\Delta+1}$. Considérons le graphe biparti complet G', donné sur la figure 3.

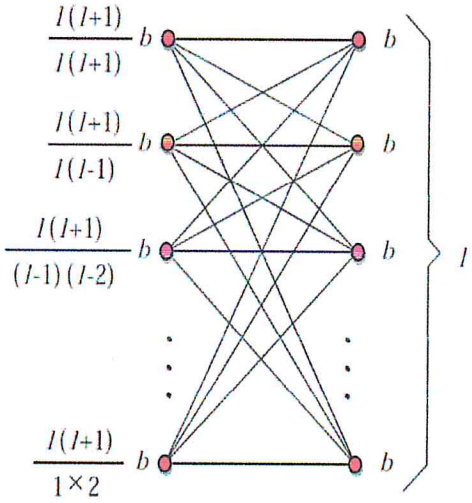


Figure 3- Le graphe G'

Dans ce graphe, on a $\Delta(G') = \Delta = l$. On peut vérifier que ce graphe donné à gauche représente bien un stable de poids maximum dans ce graphe.

D'autre part $\alpha(G') = \frac{l(l+1)}{1\times2}b + \frac{l(l+1)}{2\times2}b + \cdots + \frac{l(l+1)}{l(l+1)}b = l(l+1)b\left\{\left(\frac{1}{1}-\frac{1}{2}\right) + \left(\frac{1}{2}-\frac{1}{3}\right) + \cdots + \left(\frac{1}{l}-\frac{1}{l+1}\right)\right\} = l(l+1)b\left(1-\frac{1}{l+1}\right) = bl^2$.

De plus, GWMAX produit le stable à droite. Par conséquent, $\frac{A(G')}{\alpha(G')} = \frac{bl}{bl^2} = \frac{1}{l} = \frac{1}{\Delta} \geq \rho_{GWMAX}$.

□

Conclusion et perspectives

Dans ce mémoire nous avons étudié deux problèmes : Problèmes du stable maximum et stable de poids maximum.

Des définitions ainsi que des exemples d'application ont été donnés pour bien éclaircir ces problèmes, qui sont importants dans le domaine d'optimisation combinatoire. Comme ces deux problèmes sont NP-difficiles, nous avons cité quelques cas polynomiaux connus dans la littérature. Ensuite des algorithmes d'approximation ont été présentés pour leur résolution, dans le cas général. Des bornes inférieures et supérieures ont aussi été incluses.

Ce travail ouvre beaucoup de pistes de recherche pour des nouveaux étudiants ou étudiantes en Master de recherche opérationnelle, en particulier comparer empiriquement les méthodes approchées proposées dans ce mémoire et considérer des méthodes exactes comme Branch and Bound ainsi que la programmation dynamique.

Références

[1] . L.W. Beineke : Characterizations of derived graphs. Journal of Com- binatorial Theory, 9(2):129–135, 1970(Cité page 12).

[2] . P.G.H. Lehot : An optimal algorithm to detect a line graph and output its root graph. Journal of the ACM (JACM), 21(4):569–575, 1974. (Cité page 47.)

[3] . M. Chudnovsky, N. Robertson, P. Seymour et R. Thomas : The strong perfect graph theorem. Annals of Math., 164:51–229, 2006. (Cité pages 22 et 181.)

[4] . K.S. Booth et G.S. Lueker : Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algo- rithms. Journal of Computer and System Sciences, 13(3):335–379, 1976. (Cité page 24.)

[5] . M. Habib, R. McConnell, C. Paul et L. Viennot : Lex-BFS and par- tition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. Theoretical Computer Science, 234(1-2):59–84, 2000. (Cité page 24.)

[6] . T.W. Haynes, S.T. Hedetniemi et P.J. Slater : Domination in graphs. Advanced topics. Marcel Decker Inc., New York, 1998. (Cité page 40.)

[7]. T.W. Haynes, S.T. Hedetniemi et P.J. Slater : Fundamentals of domi- nation in graphs. Marcel Decker Inc., New York, 1998. (Cité page 40.)

[8]. R.M. Karp : Reducibility among combinatorial problems. Com- plexity of computer computations, 1972. (Cité pages 40, 43, 111 et 183.)

[9]. T. Gallai : Über extreme Punkt-und Kantenmengen. Ann. Univ. Sci. Budapest, Eotvos Sect. Math, 2:133–138, 1959. (Cité page 41.)

[10]. D. König : Graphok és matrixok (Graphes et matrices). Matematikai és Fizikai Lapok, 38:116–119, 1931. (Cité pages 41 et 46.)

[11] J. Egerváry : Matrixok kombinatorius tulajdonságairól [Hungarian, with German summary]. Matematikai és Fizikai Lapok, 38:16–28, 1931. (Cité page 41.)

[12]. G. Frobenius : Über zerlegbare Determinanten, Sitzungsber König. Preuss. Akad. Wiss, 18:274–277, 1917. (Cité page 42.)

[13]. J.W. Moon et L. Moser : On cliques in graphs. Israel journal of Mathematics, 3(1):23–28, 1965. (Cité page 42.)

[15] I. Bomze, M. Budinich, P. Pardalos et M. Pelillo : Handbook of Combinatorial Optimization, volume 4, chapitre The maximum clique problem, pages 1–74. Kluwer Academic Publishers, Boston, MA, 1999. (Cité page 38.)

[16]. R.Z. Norman et M.O. Rabin : An algorithm for a minimum cover of a graph. Proceedings of the American Mathematical Society, 10(2):315– 319, 1959. ISSN 0002-9939. (Cité page 46.)

[17]. L.Lovasz, stable set and polynomials, Discrete Mathematics 124(1994), 137-153.

[18] P. M. Pardalos and Jue Xue, The Maximum Clique Problem, J. Global Optimization 4 (1994), 301-328.

[19] C. Berge, Graphes et Hypergraphes, Dunod, Paris (1970).

[20] P. Erd• os, On the graph theorem of Tur_an (in Hungarian), Mat. Lapok 21 (1970), 249-251.

[21] S.M. Selkow, A probabilistic lower bound on the independence number of graphs, Discrete Mathematics (1994), 363-365.

[22] A note on greedy algorithms for maximum weighted independent set problem Version 2 Shuichi Sakai, Mitsunori Togasaki, Koichi Yamazaki_Department of Computer Science Gunma University 1-5-1 Tenjin-cho, Kiryu zip: 376-8515, Gunma, Japan shuichi,togasaki,koichi@comp.cs.gunma-u.ac.jp.

[23] S. Mellin : Polynomielle Färbungsalgorithmen für Pk-freie Graphen. Rapport technique, Diplomarbeit am Institut für Informatik, Universität zu Köln, 2002. (Cité page 112.)

[24] N. Sbihi : Étude des stables dans les graphes sans étoile. Thèse de doctorat, Université Scientifique et Médicale de Grenoble (Mathématiques Appliquées), Grenoble, 1978. (Cité page 47.)

[25] N. Sbihi : Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. Discrete Mathematics, 29(1):53–76, 1980. (Cité pages 47 et 59.)

[26] V. Chvátal et P.L. Hammer: Set-packing and threshold graphs. Rapport technique, Univ. Waterloo Res. Report, CORR, 1973. (Cité pages 48 et 102.)

[27] V. Chvátal et P.L. Hammer: Aggregation of Inequalities in Integer Programming. Annals of Discrete Mathematics, 1:145–162, 1977. (Cité page 48.)

[28] F. Gavril: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM Journal on Computing, 1:180–187, 1972. (Cité pages 48 et 111.)

[29] M.C. Golumbic : Algorithmic graph theory and perfect graphs. North Holland, 2004. (Cité page 49.)

[30] R.M. McConnell et J.P. Spinrad : Modular decomposition and transitive orientation. Discrete Mathematics, 201(1-3):189–241, 1999. (Cité pages 49 et 72.)

[31] V. Chvátal: On certain polytopes associated with graphs. Journal of Combinatorial Theory, Series B, 18(2):138–154, 1975. (Cité page 49.)

[32] M. Grötschel, L. Lovász et A. Schrijver : The ellipsoid method and its consequences in combinatorial optimization. Combinatorica, 1(2):169–197, 1981. (Cité pages 32, 49, 111 et 181.)

[33] M. Grötschel, L. Lovász et A. Schrijver : Polynomial algorithms for perfect graphs. Topics on perfect graphs, Ann. Disc. Math., 21:325–356, 1984. (Cité pages 49 et 111.)

[34] J. Edmonds et R.M. Karp: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM), 19 (2):248–264, 1972. (Cité page 46.)

[35] A. Schrijver : Combinatorial optimization : polyhedra and efficiency. Springer-Verlag, 2003. (Cité pages 46, 47 et 182.)
[36] H.N. Gabow : An Efficient Implement at ion of Edmonds' Maximum-Matching Algorithm.Rapport technique, Department of Computer Science, Stanford University, Stanford, California, 1972. (Cité page 47.)
[37] H.N. Gabow : An efficient implementation of Edmonds' algorithm for maximum matching on graphs. Journal of the ACM (JACM), 23 (2):221–234, 1976. (Cité page 47.)
[38] E.L. Lawler: Combinatorial optimization : networks and matroids. Holt, Rinehart and Winston, NY, USA, 1976. (Cité page 47.)
[39] D. Nakamura et Tamura A. : A revision on Minty's algorithm for finding a maximum weight stable set of a claw-free graph. J. Oper. Res. Soc. Japan, 44(2):194–204, 2001. (Cité page 48.)
[40] G. Oriolo, U. Pietropaoli et G. Stauffer: A new algorithm for the maximum weighted stable set problem in claw-free graphs. Integer Programming and Combinatorial Optimization, pages 77–96, 2008. (Cité pages 48 et 51.)
[41] A. Frank : Some polynomial algorithms for certain graphs and hypergraphs. In Proceedings of the Fifth British Combinatorial Conference, University of Aberdeen, Aberdeen, July *14-18, 1975*, pages 211–226. Utilitas Mathematica Pub., 1976. (Cité page 48.)
[42] Mouloud BOULALA and Jean-PierreUHRY : POLYTOPE DES INDEPENDANTS D'UN GRAPHE SERIE-PARALLELE. Discrete Mathematics 27(1979)225-243.
[43] A. Hertz : On the use of Boolean methods for the computation of the stability number. Discrete Applied Mathematics, 76(1-3):183–203, 1997. (Cité page 96.)
[44] S.W. Földes et P.L. Hammer : Split graphs. In Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory, and Computing, Louisiana State University, Baton Rouge, February *28*-March *3, 1977*, pages 311–315. Utilitas Mathematica, 1977. (Cité page 102.)
[45] S.W. Földes et P.L. Hammer : Split graphs having Dilworth number two. Canad. J. Math, 29(3):666–672, 1977. (Cité page 102.)
[46] Magn_us M. Halld_orsson and Jaikumar Radhakrishnan, Greed is Good: Approximating Independent Sets in Sparse and Bounded-Degree Graphs, Algorithmica 18 (1) (1997), 145-163.
[47] J. R. Griggs, Lower bounds on the independence number in terms of the degrees , J. Combin. Theory Ser. B 34 (1983), 22-39.
[48] V. Chv_atal and C. McDiarmid, Small transversals in hypergraphs, Combinatorica 12 (1) (1992), 19-26.
[49] Gregory Morel. Stabilite et coloration des graphes sans P5. Mathematiques generales [math.GM]. Universite de Grenoble, 2011. Francais. <NNT : 2011GRENM042>. <tel-00651941v3>
[50] - Mathématiques pour l'Informatique - LE PROBLEME DU STABLE MAXIMUM Nicolas & Matthieu.
[51] Graphes bipartis Frédéric Meunier 2 novembre 2016
[52] Marie-Pierre Bal Université Paris-Est. L3 Informatique
[53] Graphes : algorithmes et modélisation Olivier Briant, Hadrien Cambazard, Nicolas Catusse et Bernard Penz.
[54] Optimisation dans les Graphes 2014-2015 MPRO – OG Marie-Christine Costa, Cédric Bentz, Christophe Picouleau.

[55] Maximum Independent Set Problem CS 230A UCSB TG.
[56] Solving the Maximum Independent Set Problem in Graphs with Large Independence Number, Sergiy Butenko, Svyatoslav Trukhanov Industrial and Systems Engineering Texas A&M University College Station, TX Yalta 2006.
[57] Quelques classes de graphes, les graphes parfaits.