REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU
FACULTE DES SCIENCES
DEPARTEMENT DES MATHEMATIQUES

Memoire de Master

Présenté pour l'obtention du diplôme de :

MASTER EN MATHEMATIQUES

Spécialité

Recherche Opérationnelle

Thème

Ensembles indépendants maximaux dans les graphes WCG : Recherche et sécurité

Présenté par :

BERKOUK Karima

BOUTIAB Saida

Devant le jury :

M^r. M. AOUANE	Président
M^r. B. SADI	Rapporteur
M^r. K. KASDI	Examinatreur

TIZI-OUZOU, Juillet 2019

Remerciements

Nous tenons à remercier :

Le bon Dieu de nous avoir donné la patience et la volonté pour accomplir ce travail.

Nos remerciements s'adressent également à :

Notre promoteur M^r SADI pour ses conseils, ses orientations pour nous avoir transmis les renseignements nécessaires à la réalisation de ce travail, et son aide durant l'encadrement.

Nous remercions également :

Les membres de jury, pour l'honneur qu'ils nous font en acceptant de juger de lire et d'évaluer ce mémoire.

Nous tenons également à remercier :

Tous les enseignants de notre département qui nous ont accompagnés au cours de notre formation et à tout le personnel de la bibliothèque de l'université.

Enfin, nous remercions toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

DEDICACES KARIMA

Je dédie ce modeste travail

A

Mes chers parents

Mes frères

Mes sœurs et leurs maris

Ma belle sœur

Mes adorables nièces

Et à

Toute ma famille

Et

Mes ami(e)s

DEDICACES SAIDA

Je dédie ce modeste travail

A

Mes chers parents

Mes frères

Ma sœur et son mari

Mes belles sœurs

Mes adorables neveux et nièce

Et à

Toute ma famille

Et

Mes ami(e)s

Table des matières

Table des matières **3**

Table des figures **6**

Introduction générale **9**

1 Généralités **11**

1.1 Introduction 11

1.2 Notions fondamentales de graphe 11

1.3 Quelques graphes particuliers 15

1.3.1 Graphe biparti 15

1.3.2 Graphe d'intervalles 16

1.3.3 Graphe de comparabilité 17

1.3.4 Graphe triangulé 18

1.3.5 Arbre 19

1.3.6 Cactus 19

1.3.7 Graphe parfait 20

1.4 Complexité algorithmique 20

1.4.1 Notions sur la complexité des algorithmes 21

1.4.2 Notions de la théorie de la complexité 22

1.4.2.1 La théorie de la complexité 22

1.4.2.2 Les classes P et NP 23

1.4.2.3 La classe NP-Complet 24

2 Reconnaissance des well covered graphs 25

2.1 Introduction 25

2.2 Algorithme qui calcule un couplage maximum dans un graphe quelconque . 25

2.2.1 couplage et chaîne alternée 25

2.2.2 Notion de chaîne alternée et théorème de Berge 26

2.2.3 Oprération de contraction 27

2.2.4 Blossom 27

2.2.5 arbre alterné 28

2.3 Algorithme qui calcule un couplage maximum dans les graphes bipartis . . 31

2.4 Couplage et transversal 32

2.5 Algorithme qui calcul un stable maximal 33

2.6 Reconnaissance des graphes WCG 33

3 La sécurisation des indépendants maximums dans les graphes 36

3.1 Introduction 36

3.2 Heuristique de recherche d'un stable maximum dans les graphes parfaits . . 36

3.2.1 Algorithme de recherche d'un stable maximum dans les graphes parfaits 37

3.2.2 Principe de l'algorithme 39

3.3 Sécurisation des indépendants maximums 42

3.3.1 Sécurisation d'un indépendant maximum dans les graphes parfaits . 44

3.4 Application de la sécurisation des indépendants maximums 51

3.4.1 Le problème de l'allocation de fréquences dans un réseau de téléphone mobile 51

3.4.1.1 Modélisation du problème 51

3.4.2 Le problème d'aviateurs pendant la guerre 54

3.4.2.1 Modélisation du problème 54

Conclusion et perspectives 57

Bibliographie **58**

Table des figures

1.1 $C1 = \{1, 2, 4\}; C2 = \{2, 3, 4\}; C3 = \{4, 5\}$ sont les cliques maximales de G . 13
1.2 Un graphe G et sa matrice d'adjacence 14
1.3 $T[i]$ 15
1.4 Graphe biparti 16
1.5 un arbre 19
1.6 Cactus 19
1.7 Représentation des classes de complexité 24

2.1 Exemples de couplages (arêtes en gras) 26
2.2 Exemples de transversaux (sommets en bleu) 32

3.1 calcul du stable maximum dans un graphe biparti 39
3.2 S=$\{1\}$ 40
3.3 S=$\{1, 7\}$ 40
3.4 S=$\{1, 7, 8\}$ 41
3.5 S=$\{1, 7, 8, 3\}$ 41
3.6 S=$\{1, 7, 8, 3, 2\}$ 42
3.7 S est sécurisé 43
3.8 S n'est pas sécurisé 44
3.9 Un réseau d'antennes 52
3.10 Graphe associé au réseau étudié 53
3.11 Un réseau d'antennes 53

3.12 Graphe associé au réseau étudié 54

Introduction générale

La Recherche opérationnelle (**RO**) peut-être considérée comme un ensemble de méthodes utilisables pour élaborer des meilleures décisions. Elle permet de modéliser des problèmes issus du monde réel, identifier les méthodes de résolution et les outils les plus adaptée face à un problème pratique. Elle fait partie de "l'aide à la décision" qui est un ensemble de techniques permettant pour une personne donnée d'opter pour la meilleure prise de décision possible. Comme toute théorie, la recherche opérationnelle ne cesse de se développer pour élargir son champ d'application dans les différents domaines. Certains problèmes de prise de décision imposent la prise en compte de la présence de plusieurs objectifs pour le décideur. Par conséquent, l'homme d'étude doit s'appuyer sur ces objectifs, qui serviront de critères, pour sélectionner la décision réalisable à proposer au décideur.

L'histoire de graphe est relativement récente puisqu'elle n'est apparue formellement qu'au cours du XXe siècle. Mais elle est aujourd'hui devenu indispensable dans nombreux domaines, notamment en informatique fondamentale et appliquée, en optimisation. L'étude des graphes et de leurs applications est donc l'occasion d'aborder des questions très diverses, dont les applications sont nombreuses. C'est ainsi qu'on développera, par exemple, les méthodes d'ordonnancement de tâches à partir des chemins optimaux dans les graphes, ou encore des propriétés de réseaux de communication à propos de la connectivité des graphes. Historiquement, les graphes ont été en fait considérés, bien avant la base de la théorie, avec des problèmes célèbres comme celui de ponts de königsberg.

La théorie des graphes s'est alors développée dans diverses disciplines telles que la chimie, la biologie, les sciences sociales. Depuis le début du XXe siècle, elle constitue une

branche à part entière des mathématiques, grâce aux travaux de Konig, Menger, Cayley puis de Berge et d'Erdös.

Le but de ce travail est de caractériser la sécurisation des indépendants maximums après avoir trouvé ces derniers.

Le chapitre 1 est dédié à la présentation des notions essentielles à la bonne compréhension de ce mémoire. Nous introduisons tout d'abord des notions propres aux graphes, ainsi que des classes de graphes, puis des notions concernant la complexité des problèmes et des algorithmes.

Le deuxième chapitre est consacré à la reconnaissance d'une autre classe de graphes "Well covered graph" qui va nous faciliter d'étudier la sécurisation des indépendants maximums.

Nous nous intéressons dans le chapitre 3 à exposer les résultats trouvés sur la sécurisation des indépendants maximums dans les graphes parfaits, tout en donnant les conditions nécessaires pour avoir la sécurité. En dernier, nous abordons quelques situations réelles de sécurité dans le domaine des télécommunication.

Chapitre 1

Généralités

1.1 Introduction

Dans ce premier chapitre nous allons présenter quelques notions de base sur la théorie des graphes qui seront utiles tout au long de ce mémoire. Nous donnons un ensemble de définitions et de notations générales concernant les graphes, ainsi qu'un ensemble de graphes particuliers qui seront nécessaires par la suite, et à la fin, nous terminons par une introduction sur la théorie de la complexité algorithmique.

1.2 Notions fondamentales de graphe

Nous décrivons dans cette section seulement les graphes non orientés qui seront utiles pour la compréhension des prochains chapitres. Un graphe est un ensemble de points appelés **sommets** reliés entre eux par des **arêtes** (une arête est une paire non ordonnée de sommets distincts) ou **arcs** si le graphe est orienté(un arc est une paire de sommets possèdant une extrémité initiale et une extrémité terminale). Le graphe est un outil puissant de modélisation de nombreux problèmes combinatoires et de n'importe quel problème comportant des objets avec des relations entre ces objets (ordonnancement de tâches, carte géographique en coloriant les villes...).

Un **graphe non orienté** est défini par deux ensembles finis V et E, V est l'ensemble des sommets du graphe avec $V \neq \emptyset$ et E son ensemble d'arêtes ; il est noté $G=(V,E)$ où $|V|$=n est l'ordre du graphe, $|E|$=m est le nombre d'arêtes de G. Il est dit simple s'il est sans boucle et entre deux sommets quelconques, il existe, au plus, une arête. Un **sous-graphe** de G est un graphe de la forme $H=(A,F)$ où $A \subset V$ et $F \subset E$ sont tels que toute arête de F a ses extrémités dans A. Un graphe d'ordre n est **complet** si deux sommets quelconques sont reliés par une arête. On le note K_n. Deux sommets $x, y \in V$ sont **voisins** ou **adjacents** s'ils sont reliés par une arête. Le **voisinage** d'un sommet x est l'ensemble $\Gamma(x)=\{y \in V, xy \in E\}$ c'est aussi l'ensemble des adjacents de x noté $Adj(x)$. Deux arêtes e_1 et e_2 sont adjacentes si elles ont un sommet en commun. Le nombre d'arêtes adjacentes à un sommet x est appelé **degré** de x et noté $d(x)$. Un sommet $x \in V$ tel que $d(x)=0$ est dit isolé. Il est pendant si $d(x)=1$. Un graphe $G=(V,E)$ est dit **régulier** si tout ses sommts ont même degré ; il est **k-régulier** si tous sommet est de degré k. Une suite finie d'arêtes adjacentes est une **chaîne** reliant x à y. La longueur de la chaîne est le nombre d'arêtes qu'elles contient. Elle est fermée si et seulement si les deux extrémités sont confondues ; dans ce cas la chaîne est appelée **cycle**, il est **élémentaires** s'ils n'utilisent pas plus d'une fois le même sommet. Un cycle est **hamiltonien** s'il passe exactement une fois par tout sommet de G ; il est **eulérien** s'il passe exactement une fois par chaque arête de G. Un graphe G est **acyclique** s'il ne contient pas de cycle. Il est **connexe** s'il existe une chaîne entre toute paire de sommets. Si G n'est pas connexe, il contient, au moins, deux sous-graphes connexes, maximaux au sens de l'inclusion, appelés **composantes connexes**.[2]

Le graphe **complémentaire** d'un graphe $G=(V,E)$ est le graphe $\overline{G}=(V,\overline{E})$ ayant les mêmes sommets que G et tel que deux sommets distincts de $\overline{G}$ sont adjacents si et seulement s'ils ne le sont pas dans G.

Une **clique** d'un graphe G est un sous-graphe complet et la taille maximum d'une clique est notée $\omega(G)$ [11].

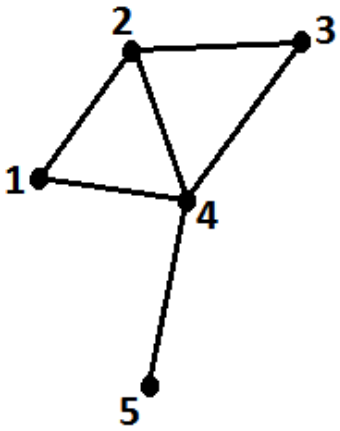


FIG. 1.1: $C1 = \{1, 2, 4\}; C2 = \{2, 3, 4\}; C3 = \{4, 5\}$ sont les cliques maximales de G

Un **Stable** S ou ensemble indépendant est un ensemble de sommets, deux à deux, non adjacents.

- S **maximal** si $S \bigcup \{x\}$; $x \notin S$, n'est pas stable.
- S **maximum** si $|S| \geq |T|$, $\forall T$ stable de G.

Par exemple, pour le graphe G de la figure 1.1. les ensembles $S1$=$\{4\}$; $S2$=$\{1, 3, 5\}$; $S3$=$\{2, 5\}$ sont des stables maximaux de G. On note $\alpha(G)$, la taille maximum d'un stable de G.

On appelle **nombre chromatique** de G le nombre minimum de couleurs, noté $\chi(G)$, nécessaires pour colorier chaque sommet du graphe G de façon que deux sommets adjacents quelconques soient de couleurs différentes.

La représentation d'un graphe en machine se fait à l'aide de sa **matrice d'adjacence** ou **liste d'adjacence**.[6]

Une matrice d'adjacence est une matrice carrée A=$(a_{ij})_{i=\overline{1,n};j=\overline{1,n}}$ telle que :

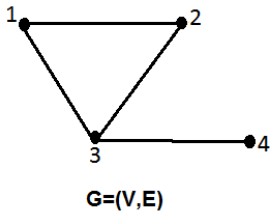
$$a_{ij} = \begin{cases} 1, & \text{si } ij \in E\,; \\ 0, & \text{sinon.} \end{cases} \tag{1.1}$$

i représente les lignes

j représente les colonnes

La matrice associée à un graphe non orienté est symétrique, c'est à dire, $a_{ij} = a_{ji}$.

On donne un exemple de graphe et sa matrice associée :



	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

matrice d'adjacence de G

FIG. 1.2: Un graphe G et sa matrice d'adjacence

Soit le graphe $G = (V, E)$. On suppose que les sommets de V sont numérotés de 1 à n, avec $n = |V|$. La représentation par **liste d'adjacence** de G consiste en un tableau T de n listes, une pour chaque sommet de V. Pour chaque sommet $x \in V$, la liste d'adjacence $T[x]$ est une liste chainée de tous les sommets $y \in V$ tels qu'il existe une arête $xy \in E$. Autrement dit, $T[x]$ contient la liste de tous les sommets adjacents à x. Les sommets de chaque liste d'adjacence sont généralement chainés selon un ordre arbitraire. Dans le cas de graphes non orientés, pour chaque arête $xy \in E$, on aura y qui appartiendra à la liste chainée de $T[x]$, et aussi x qui appartiendra à la liste chainée de $T[y]$.

Pour l'exemple de la figure 1.2 le tableau T correspond aux adjacents des sommets du graphe G :

x	Adj(x)
1	{2,3}
2	{1,3}
3	{1,2,4}
4	{3}

FIG. 1.3: $T[i]$

1.3 Quelques graphes particuliers

Nous présentons, ici, quelques graphes particuliers par exemple :

1.3.1 Graphe biparti

[7] G=(V, E) est biparti si on peut partionner ses sommets en deux sous-ensembles V_1 et V_2 avec aucune arête entre deux sommets de V_1 ou de V_2.On le note G=(V_1, V_2, E). Un graphe biparti complet c'est un graphe biparti G=(V_1, V_2, E), avec $|V_1| = a$, $|V_2| = b$, dans lequel toute paire de sommets avec $x_1 \in V_1$ et $x_2 \in V_2$ est reliée par une arête. Il est noté $K_{a,b}$.

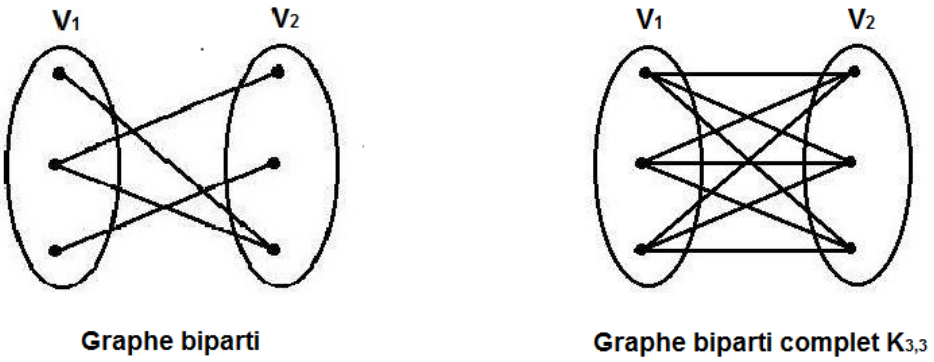


FIG. 1.4: Graphe biparti

1.3.2 Graphe d'intervalles

[7]

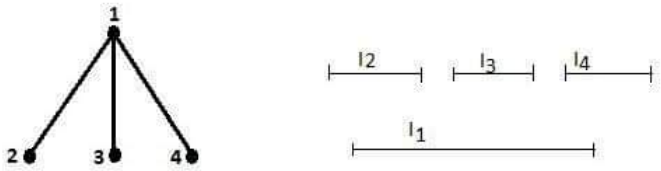
Un graphe **d'intervalles** est le graphe d'intersection d'un ensemble d'intervalles de la droite réelle. Chaque sommet représente un intervalle et une arête relie deux sommets lorsque les intervalles correspondants sont d'intersection non vide.

Soit G=(V, E) un graphe

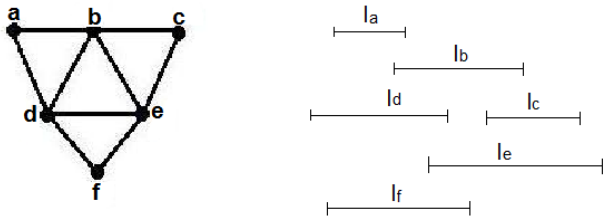
$x \in V \longrightarrow I_x$=intervalle de $\Re$

$y \in V \longrightarrow I_y$=intervalle de $\Re$

$xy \in E \Leftrightarrow I_x \cap I_y \neq \varnothing$



(a) Graphe d'intervalle



(b) Graphe n'est pas d'intervalle

On donne dans la figure ci-dessus deux exemples tels que (a) nous donne un graphe d'intervalle car :

$I_a \bigcap I_b \neq \emptyset$

$I_a \bigcap I_c \neq \emptyset$

$I_a \bigcap I_d \neq \emptyset$

par contre dans (b) le graphe n'est pas d'intervalle comme

$I_f \bigcap I_a \neq \emptyset$ or af $\notin E$

$I_f \bigcap I_b \neq \emptyset$ or bf $\notin E$

$I_f \bigcap I_c \neq \emptyset$ or cf $\notin E$

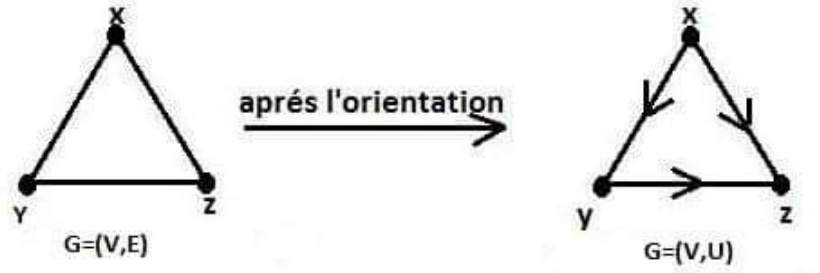
1.3.3 Graphe de comparabilité

G=(V, E) est de **comparabilité** s'il existe une orientation $\cup$ des arêtes de E telle que :

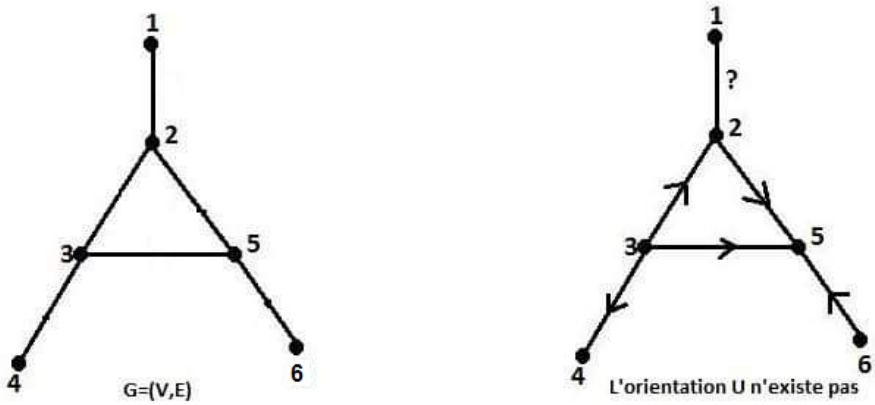
$\forall\ x, y, z \in V\ (x, y) \in \cup$ et $(y, z) \in \cup \Rightarrow (x, z) \in \cup$ [6].

On a dans la figure ci-dessous deux exemples (c) et (d) tels que le graphe (c) est de comparabilité vu qu'il admet une orientatioin $\cup$ sur toutes les arêtes, par contre (d)

n'est pas de comparabilité car il existe une arête qui n'admet pas d'orientation.



(c) G de comparabilité



(d) G n'est pas de comparabilité

1.3.4 Graphe triangulé

[6]

Un graphe est **triangulé** si tout cycle de longeur ≥ 4 posséde une corde (arête entre deux sommets non-consécutifs du cycle).

1.3.5 Arbre

Un **arbre** est un graphe connexe sans cycle.

Conséquence :

- $m = n - 1$.
- Pour toute paire de sommet $(x, y) \in V^2$, il existe une chaîne unique qui relie x à y.

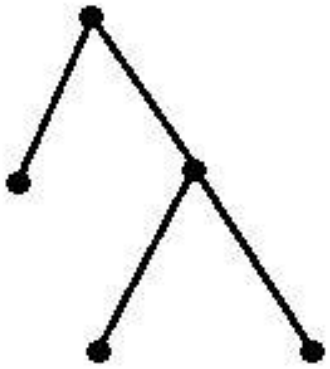


FIG. 1.5: un arbre

1.3.6 Cactus

Un Cactus est un graphe tel que chacune de ses arêtes appartient à au plus un cycle.

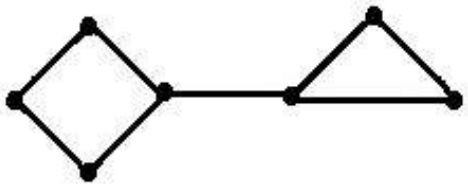


FIG. 1.6: Cactus

1.3.7 Graphe parfait

Au départ nous définissons les notions suivantes : $G = (V, E)$ un graphe simple non orienté

- $\alpha(G)$=la taille maximale d'un stable de G.
- $\chi(G)$=le nombre chromatique de G.
- $\omega(G)$=la taille maximale d'une clique.
- $\theta(G)$=la taille minimale d'une partition en cliques.

Claude Berge a été le premier à introduire la notion des graphes parfaits au début des années 60. Un graphe est **parfait** si est seulement s'il est α-parfait ou γ-parfait et n'admet pas de cycle impair tel que :

1. Un graphe G est α-parfait si $\alpha(G)=\theta(G)$.
2. G est $\gamma(G)$-parfait si $\omega(G)=\chi(G)$.

1.4 Complexité algorithmique

La notion d'algorithme est très ancienne. On considère souvent que le premier algorithme non trivial est l'algorithme d'Euclide qui permet de trouver le $P.G.C.D$ de deux entiers, ou de prouver que ces entiers sont premiers entre eux. Comme autre exemple d'algorithme on peut noter le crible d'Erahostème qui permet de dresser la liste des nombres premiers entre 1 et n en barrant, dans l'ordre naturel, les multiples des coefficients supérieurs à 1 des entiers de 2 à n (non déjà barrés).
L'efficacité des algorithmes a été recherchée de tout temps, en particulier, en ce qui

concerne le problème de savoir si un nombre n est premier. Gauss, déjà, posait la question de l'existence d'un algorithme vraiment efficace. Le développement des ordinateurs a favorisé, sous la poussée des applications, une vision **algorithmique**.

1.4.1 Notions sur la complexité des algorithmes

La première idée qui vient à l'esprit pour évaluer et comparer des algorithmes est de les programmer, puis de mesurer leurs durées d'exécution. En fait, le temps de calcul est de mesure imparfaite car elle dépend trop de la machine, du language de programmation, du compilateur particulier utilisé pour ce language, et des données.
On préfère, en pratique, compter le **nombre d'opérations élémentaires** de l'algorithme à évaluer : ce nombre ne dépend ni de la machine, ni du language, et peut s'évaluer sur le papier. Evalué dans certaines conditions, on l'appelle **complexité** de l'algorithme. Il s'agit d'un terme technique n'ayant rien à voir avec la difficulté de programmation !

Définition 1.1. [11] ***La complexité d'un algorithme*** *A est une fonction $C_A(N)$, donnant le nombre d'instructions élémentaires exécutées par A dans le pire des cas, pour une donnée de taille N.*

Cette définition appelle quelques commentaires. Pour un problème avec une donnée de taille N fixée, la complexité est basée sur le pire cas, c'est-à-dire sur la donnée qui va demander le plus de travail à l'algorithme.
La complexité est une fonction de la taille des données pour prédire la variation du temps de calcul, par exemple quand on passe de petits jeux d'essai à de grands problèmes. Ceci permet aussi d'estimer la taille maximale des problèmes qu'on peut résoudre en pratique.

1. **Taille d'une donnée**
   Une donnée doit être comprise ici dans le sens général. Une donnée d'un algorithme de chemin optimal dans un graphe valué G=(V, U, C) comprendra par exemple : le nombre n de sommets, la liste des sommets de V, le nombre m d'arcs et la liste des arcs de U avec, pour chacun d'eux, la valeur numérique associée. La taille d'une

donnée est la quantité de mémoire nécessaire pour la stocker [8].

2. **Ordre d'une fonction**

   Soit f, g deux fonctions de $\mathbb{R}$ dans $\mathbb{R}$. On dit que f est d'ordre inférieur ou égal à g, ou d'ordre au plus g, si on peut trouver un réel x_0 et un réel positif c tels que : $\forall x \geq x_0, f(x) \leq c.g(x)$.

   En d'autres termes, g devient plus grande que f à partir d'un certain nombre x_0, à un facteur c près. On écrit : f est $\mathcal{O}(g)$, f est en $\mathcal{O}(g)$, f=$\mathcal{O}(g)$ ou $f \in \mathcal{O}(g)$ (en interprétant $\mathcal{O}(g)$ comme l'ensemble des fonctions d'ordre au plus g) et on prononce "grand $\mathcal{O}$ de g" [8].

3. **Bons et mauvais algorithmes**

   Une classification grossière mais reconnue distingue les algorithmes **polynomiaux** (de complexité de l'ordre d'un polynôme), des autres, dits **exponentiels**. Des exemples de complexités polynomiales sont $\log n$, $n^{0,5}$, $n \log n$, n^2 ... etc. Une complexité exponentielle peut être une vraie exponentielle au sens mathématique ($\exp^2, 2^n$), mais aussi des fonctions comme n^{logn}, la fonction factorielle $n!$ et n^n. Un bon algorithme est polynomial [8].

1.4.2 Notions de la théorie de la complexité

1.4.2.1 La théorie de la complexité

Définition 1.2. *Un* ***problème de décision*** *consiste à chercher dans un ensemble fini S s'il existe un élément s vérifiant une certaine propiété P. Les problèmes de décision peuvent toujours être un énoncé et une question à réponse Oui-Non :*

Problème de décision

donnée G=(V, E), $k \in \mathbb{N}$;

question , Existe-t-il un stable de card k ?

Certains problèmes d'optimisation combinatoire disposent depuis longtemps d'algorithmes polynomiaux, tandis que d'autres n'en ont toujours pas.

La théorie de la complexité a été développée vers 1970 pour répondre à cette question, par des logiciens et des spécialistes en informatique théorique. La théorie de la complexité se heurte à d'énormes difficultés théorique et n'a pu obtenir pour l'instant que des résultats partiels. Le principal résultat est que tous les problèmes difficiles sont liés : la découverte d'un algorithme polynomial pour un seul d'entre eux permettrait de déduire des algorithmes polynomiaux pour tous les autres.
A cause des difficultés rencontrées, la théorie de la complexité ne traite que des problèmes de décision, car leur formalisme à réponse Oui-Non, Vrai-Faux, a permis de les étudier avec les outils de la logique mathématique.
Ceci n'est pas trop gênant pour les problèmes d'optimisation. Un algorithme efficace pour un problème d'optimisation p résout évidemment le problème de décision associé q et un algorithme efficace pour q peut être utilisé pour résoudre efficacement p, par dichotomie sur les valeurs possibles de la fonction économique. Un problème d'optimisation est donc au moins aussi difficile que son problème de décision associé [8], [11].

1.4.2.2 Les classes P et NP

La classe P : Un problème de décision est dans P si on peut le résoudre en temps polynomial en fonction de la taille de son entrée. Il admet donc un algorithme le résolvant dont le temps d'exécution est en $\mathcal{O}(n^c)$, n étant la taille de l'entrée et c étant une constante. Quelques problèmes de la classe P : Le plus court chemin dans un graphe, couplage de cardinal maximal, etc...

La classe NP : La classe NP est l'ensemble des problèmes pour lesquels on peut vérifier la validité d'un certificat pour ce problème (c'est-à-dire une solution acceptable de ce problème) en temps polynomial. NP inclut la classe P.

La notation NP ne signifie pas Non Polynomial (NDP : Non Determinist Polynomial)

1.4.2.3 La classe NP-Complet

Il s'agit des problèmes les plus difficiles de NP. Un problème **NP-Complet** est un problème de NP en lequel se transforme polynomialement tout autre problème de NP et ils représentent les plus durs de NP. On montre qu'un problème est NP-Complet par comparaison avec un autre problème connu dans la même classe, en utilisant la transformation suivante : Soient p_1 et p_2 deux problèmes de décision. On notera $D(p_i)$ l'ensemble des données de p_i.

On dira que p_1 se transforme polynomialement en p_2 s'il existe $f : D(p_1) \longrightarrow D(p_2)$ telle que :

1. $\forall$ d$\in D(p_1)$, le temps de calcul de $f(d)$ est borné par un polynôme en la taille de d.
2. d est une donnée à réponse positive de p_1 si et seulement si $f(d)$ est une donnée à réponse positive de p_2.

On notera, alors : $p_1 \prec p_2$.

On dit qu'un problème est NP-difficile pour les problèmes d'optimisation si le problème d'existance associé est NP-Complet. En 1971, Stephen A. Cook est le premier à avoir prouvé l'existence de tels problèmes, par exemples : Stable maximal, transversal minimal, ...etc [4] .

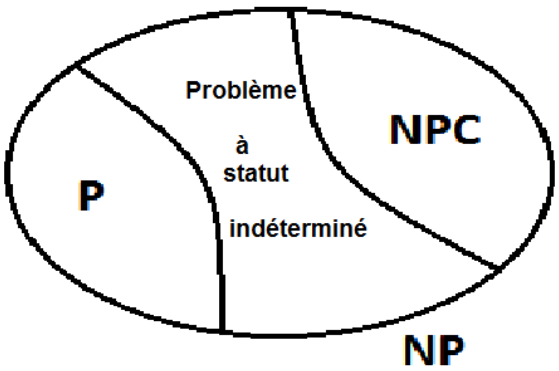


FIG. 1.7: Représentation des classes de complexité

Chapitre 2

Reconnaissance des well covered graphs

2.1 Introduction

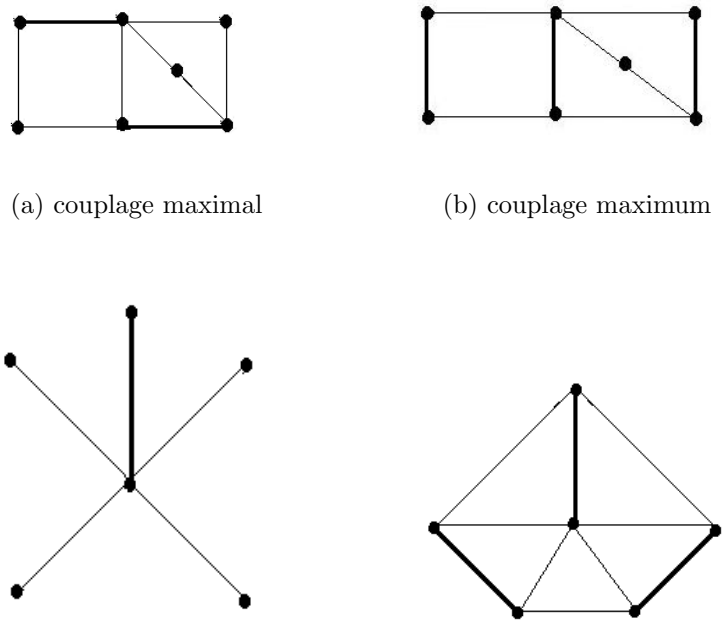
Dans ce deuxième chapitre, nous définissons la classe des graphes **well-covered graphs** (WCG) tout en donnant la définition et les outils de reconnaissance de cette classe.
Le chapitre suivant sera consacré à la sécurisation des indépendants maximums. L'étude de la sécurisation des indépendants de la classe des WCG est la plus facile, car elle se fait sans passer par le calcul du stable maximum qui est un problème NP-Complet.

2.2 Algorithme qui calcule un couplage maximum dans un graphe quelconque

2.2.1 couplage et chaîne alternée

Un couplage d'un graphe G est un ensemble M d'arêtes deux à deux non adjacentes. Un sommet du graphe est saturé par un couplage M, ou M-saturé, s'il est extrémité d'une arête de M. Il est dit insaturé par M, ou M-insaturé, sinon. Si tout sommet de G est M-saturé, alors le couplage M est dit parfait. Un couplage est dit maximal s'il est maximal

pour cette propriété . Un couplage M est dit maximum s'il a le plus grand nombre possible d'arêtes.



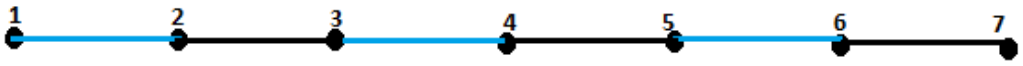
(a) couplage maximal (b) couplage maximum

(c) couplage maximum (d) couplage parfait

FIG. 2.1: Exemples de couplages (arêtes en gras)

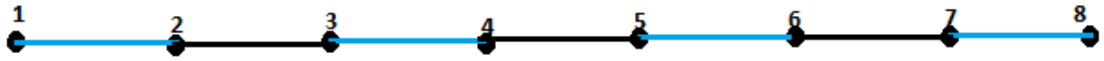
2.2.2 Notion de chaîne alternée et théorème de Berge

Etant donné un graphe $G = (V, E)$ et un couplage M de G, une **chaîne alternée** relativement à M, ou chaîne M**-alternée**, est une chaine élémentaire dont les arêtes sont alternativement dans M et dans E\ M. Une chaîne M-alernée est dite augmentante pour M, ou M-augmentante, si ses extrémités sont M-insaturées.
les arêtes en bleu sont dans M [9] .



- [1 2 3 4 5 6 7] est une chaîne alternée mais non augmentante.

- [1 2 3 4 5 6 7 8] est une chaîne alternée augmentante



- Chaîne augmentée.

Théorème 2.1. [7] [*BERGE*] *Un couplage M d'un graphe G est maximum si et seulement s'il n'existe pas de chaine M-augmentante.*

2.2.3 Oprération de contraction

La contraction de l'arête e se traduit par la fusion de ses deux extrémités.
La contraction d'une chaine est la contraction de toutes les arêtes de la chaîne.

2.2.4 Blossom

Soit M un couplage dans un graphe G.
Un cycle B est un **blossom** si
- il possède $2k+1$ arêtes
- k de ses arêtes sont dans M

Théorème 2.2. [5] [*Edmonds 1965*] *Soit M un couplage dans un graphe G. Supposons que B est un blossom de G.*
Soit G' le graphe obtenu par la contraction de B dans G.
Soit M' le couplage de M projeté dans G'. G' a un chemin augmentant pour M' si et seulement si G a un chemin augmentant pour M.

2.2.5 arbre alterné

Soit M un couplage du graphe G.

Un arbre T dans G est dit **alterné** si

• Il existe un unique sommet v non saturé (Un sommet v est dit saturé par M s'il est incident a une arête de M) par M (par convention : c'est la racine)

• Pour tout sommet v saturé par l'arbre T, le chemin de la racine vers v est un chemin alternant

Par convention :

▷ un sommet v est interne si sa distance à la racine est impaire

▷ un sommet v est externe si sa distance à la racine est paire.

Soient M un couplage du graphe G, et T un arbre alterné.

C_v^w= chemin de v à w

Remarque 2.1. [6] *Soit v un sommet de T. Si v est adjacent à un sommet w dans G de même type (externe, ou interne), alors $C_v^w \cup vw$ forment un blossom.*

Remarque 2.2. *Soit v un sommet de T. Si v est adjacent à un sommet w n'appartenant pas à T dans G et non saturé par M, alors $C_r^v \cup vw$ forment une chaine augmentante.*

Algorithm 1 Construction-chemin (un graphe G, un couplage M)

$F \leftarrow \emptyset$; $\forall v \in V$ non-marqué ; seules les arêtes $\in M$ marquées.
pour tout sommet v non saturé par M **faire**
créer un arbre T_v composé uniquement de v ; $F \leftarrow F \cup T_v$
fin pour
tantque $\exists v \in F$ non-marqué externe **faire**
tantque il existe $e = (v; w)$ non-marqué de F **faire**
si $w \notin F$ **alors**
Soit x le sommet apparié à w. Ajouter $(v; w)$ et $(w; x)$ à l'arbre contenant v
sinon
si w est interne **alors**
si $r(v) \neq r(w)$ **alors**
retourner $\{r(v) \rightarrow v, w \rightarrow r(w)\}$
sinon
$B \leftarrow$ blossom formé par e et par $v \rightarrow w$ dans T
G', $M' \leftarrow$ contracter B dans G et M
$P' \leftarrow$ Construction-chemin(G', M')
retourner P' **projeté dans** G
fin si
fin si
fin si
marquer e ;
fin tantque
marquer v ;
fin tantque
retourner $\emptyset$;

(Après avoir calculé la chaîne alternée on applique l'algorithme suivant pour calculer le couplage maximum)

Algorithm 2 $Couplage - max(G, M)$

Entrée : Un graphe G=(V, E) et un couplage M.

sortie : Un couplage maximum M.

- trouver un chemin P augmentant de M.
- s'il n'existe pas alors retourner M.
- sinon retourner le $couplage - max(G, M)$.

2.3 Algorithme qui calcule un couplage maximum dans les graphes bipartis

Algorithm 3 Couplage maximum dans les graphes bipartis

Entrée $G=(V_1, V_2, E)$ Biparti, E' couplage maximal.

Sortie Calculer un couplage maximum de G.

si E' sature tous les sommets **alors**

il est maximum.

sinon

algo :

•Aucun sommet n'est marqué

1-

si tous les sommets non saturés sont marqués **alors**

Aller en 2-.

si il existe x non saturé et non marqué **alors**

Aller en 3-.

fin si

fin si

2- Marquer de + (respectivement de -) tous les sommets non marqués de V_1(respectivement V_2), **terminé**.

3- Marquer x de + et appliquer respectivement la procédure :

si y est marqué + **alors**

marquer de - tous les voisins de y

fin si

si un sommet non saturé est marqué - **alors**

terminé.

fin si

Marquer de + le voisin dans (V_1, V_2, E) d'un sommet marqué -, **aller** en 1-.

fin si

2.4 Couplage et transversal

Un ensemble **transversal**, ou simplement transversal, d'un graphe G est un ensemble de sommets de ce graphe tel que toute arête a au moins une extrémité dans cet ensemble. Un transversal minimum est un transversal qui a le plus petit nombre de sommets. On définit le transversal d'un graphe G comme le plus petit nombre de sommets d'un transversal de G, c'est-à-dire le cardinal d'un transversal minimum. Ce nombre est noté $\tau(G)$.
On définit de même le nombre de couplage d'un graphe G comme le plus grand nombre d'arêtes dans un couplage de G, c'est-à-dire le cardinal d'un couplage maximum. Ce nombre est noté $\nu(G)$ [6] .

(a) transversal (b) transversal minimum

FIG. 2.2: Exemples de transversaux (sommets en bleu)

Proposition 2.1. *$G=(V,E)$ un graphe*
Pour tout couplage M et tout transversal L de G, $|M| \leq |L|$.
φ est une applications injective
$\varphi : M \mapsto L$
$e \mapsto \varphi(e)$
Si $\nu(G)=\tau(G)$ alors le couplage M attient son maximum et le transversal L atteint son minimum.

Mais on n'a pas toujours l'égalité, comme le montre l'exemple du graphe de la figure 2.2, graphe pour lequel $\nu(G)=3$ et $\tau(G)=4$.

2.5 Algorithme qui calcul un stable maximal

Algorithm 4 Stable Maximal

Entrée un graphe G=(V, E), non orienté

Sortie un stable maximal S de G

S : stable maximal

$\Gamma(x)$: voisin de x

pour chaque sommet x de G **faire**

marque[x]←FAUX

fin pour

S←$\emptyset$

tantque il existe un sommet x non marqué **faire**

S← $S \bigcup \{x\}$

marque[x] ←VRAI

pour chaque z∈ $\Gamma(x)$ **faire**

marque[z] ←VRAI

fin pour

fin tantque

retourner S

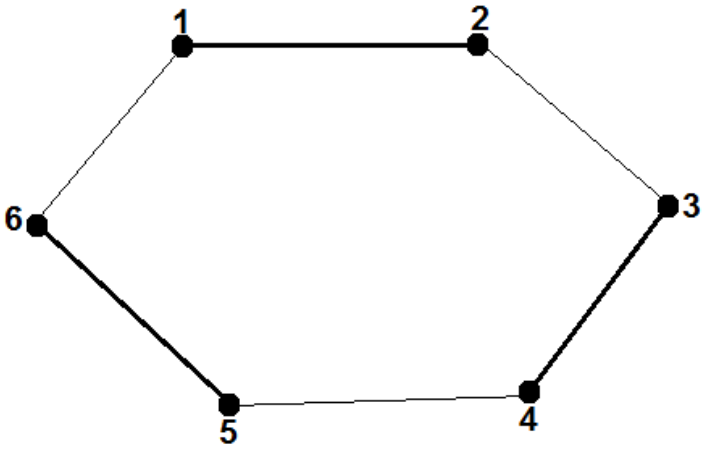
2.6 Reconnaissance des graphes WCG

Définition 2.1. *Un graphe G est well-covered graph si chaque ensemble indépendant maximal est maximum.*

Proposition 2.2. *Soit G=(V, E) un graphe. G est un graphe WCG si $|V \backslash L|$=$|S|$, comme le transversal est un problème NP-Complet donc on passera au couplage alors G est WCG si $|V \backslash M|$=$|S|$.*

Exemple 2.1. *Soit un graphe G=(V, E), M couplage maximum calculé par l'algorithme 2 et S stable maximal calculé par l'algorithme 4.*

1. *$S1$, $S2$, $S3$, $S4$, $S5$, $S6$ stables maximaux.*



$V = \{1, 2, 3, 4, 5, 6\}$

$S1$=$\{1, 3, 5\}$

$S2$=$\{6, 3\}$

$S3$=$\{1, 4\}$

$S4$=$\{2, 5\}$

$S5$=$\{2, 4, 6\}$

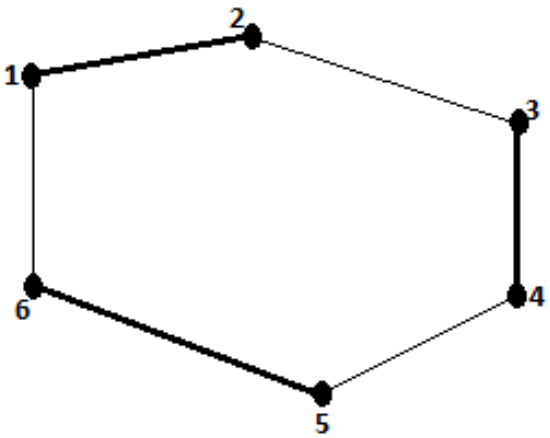
$|M|$=*3* $|V \backslash M|$=*3*

$|V \backslash M| \neq |S|$

$S2$ $S3$ et $S4$ sont de cardinalité 2 et $|V \backslash M|$=3 donc le graphe G n'est pas un WCG.

2. $S1$, $S2$, $S3$, $S4$, $S5$, $S6$ stables maximaux.

$V = \{1, 2, 3, 4, 5, 6\}$



$S1$={1, 3, 5}

$S2$={2, 4, 6}

$S3$={3, 6}

$S4$={1, 4}

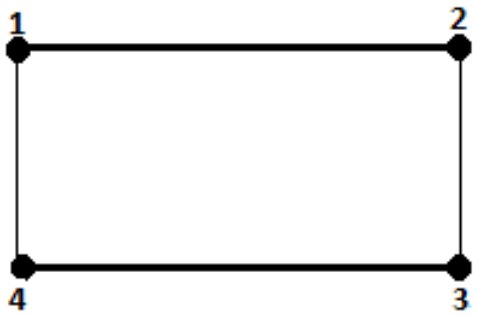
$S5$={2, 5}

M=*3* $|V \backslash M|$=*3*

$|V \backslash M|$=$|S1|$=$|S2|$=*3 mais* $|V \backslash M| \neq |S3|$ *alors G n'est pas un WCG.*

3. $S1$, $S2$ *stables maximaux.*

$V = \{1, 2, 3, 4\}$



$S1$={1, 3}

$S2$={2, 4}

$|M|$=*2*

: !

$|V \backslash M|$=*2*

$|V \backslash M|$=$|S1|$=$|S2|$

Dans ce cas G est WCG.

Chapitre 3

La sécurisation des indépendants maximums dans les graphes

3.1 Introduction

Dans ce troisième chapitre, nous essayons de trouver les différentes propriétés concernant la sécurité des indépendants maximums dans les graphes parfaits qu'on a déjà cités au premier chapitre, tout en faisant appel à des définitions et des méthodes qui vont nous aider à trouver des conditions nécessaires et suffisantes pour construire la securité dans les indépendants maximums. Et pour bien comprendre le travail, nous allons donner quelques exemples pratiques.

3.2 Heuristique de recherche d'un stable maximum dans les graphes parfaits

La recherche d'un stable de taille maximum dans un graphe est un problème classique de la théorie de la complexité. Il est NP-Complet.
La notion du stable maximum, joue un rôle important dans la résolution de plusieurs problèmes dans différents domaines. Nous citons quelques domaines d'application :

- Le domaine de l'informatique.
- Le domaine des télécommunications.
- Le domaine de la chimie.

3.2.1 Algorithme de recherche d'un stable maximum dans les graphes parfaits

Les méthodes approchées ont pour but de trouver une solution admissible en un temps raisonnable, mais sans garantie d'optimalité de cette solution. L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, elles englobent deux classes : heuristiques et métaheuristiques. On appelle **heuristique** une méthode de résolution qui ne permet pas d'affirmer que le résultat obtenu est toujours optimal. **Une métaheuristique** est une heuristique générique qu'il faut adapter à chaque problème. Comme le problème du stable maximum estNP-Complet. Pour cela, nous écrivons une heuristique.

Théorème 3.1. *L'algorithme suivant calcule un stable maximum de n'importe quel graphe parfait.*

Algorithm 5 Stable Maximum

Entrée un graphe G=(V, E), non orienté

Sortie un stable maximum S de G

S : stable maximum

i : degré d'un sommet

$\Gamma(x)$: voisin de x

$\Gamma^+(x)$: voisin marqué

1.

pour chaque sommet x de G **faire**

marque[x]←FAUX

fin pour

S←$\emptyset$

2.

pour i=0 **à** n-1 **faire**

tantque il existe un sommet x, d(x)=i sommet non marqué **faire**

si $|\Gamma^+(x)| \geq |\Gamma^+(y)|$ **alors**

S← $S \bigcup \{x\}$

marque[x] ←VRAI

3.

pour chaque z$\in \Gamma(x)$ **faire**

marque[z] ←VRAI

fin pour

sinon

S← S $\bigcup$ {y}

marque[y] ←VRAI

pour chaque z$\in \Gamma(y)$ **faire**

marque[z] ←VRAI

fin pour

fin si

fin tantque

retourner S

fin pour

3.2.2 Principe de l'algorithme

1. Ordonner les sommets par ordre croissant de leurs degrés.
2. Marquer un sommet de degé minimum et le rajouter à l'ensemble S.
3. Marquer tous les sommets adjacents à ce dernier.
4. Si tous les sommets sont marqués, stop.
   sinon, aller à l'étape 2.

Exemple 3.1. *Soit $G(V_1, V_2, E)$, un graphe biparti.*
On utilise (Algorithme 5) pour calculer le stable maximum de G :

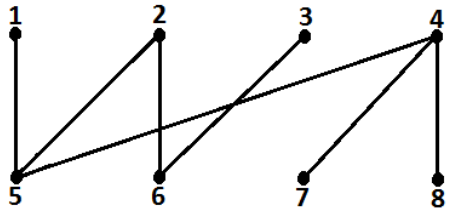


FIG. 3.1: calcul du stable maximum dans un graphe biparti

i=1

d(1)=d(3)=d(7)=d(8)=1
marquons 1 comme premier sommet et le rajouter à S. Par la suite, marquons son adjacent qui est le sommet 5.

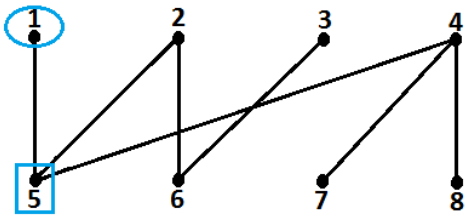


FIG. 3.2: S={1}

d(3)=d(7)=d(8)=1 comme il ne y'a pas d'adjacent déjà marqué, on prend n'importe quel sommet qu'on rajoute à S.Puis,on marque son adajcent.

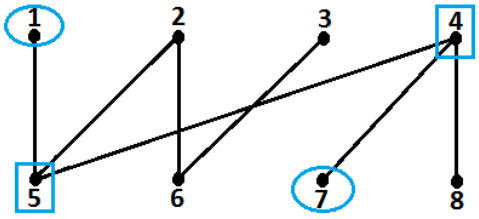


FIG. 3.3: S={1, 7}

d(3)=d(8) comme l'adjacent du sommet 8 est déjà marqué, alors on le prend en priorité et on le rajoute dans S.

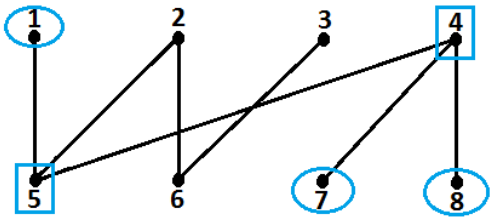


FIG. 3.4: S=$\{1, 7, 8\}$

Marquons le dernier sommet de i=1, d(3)=1, marquons son adjacent

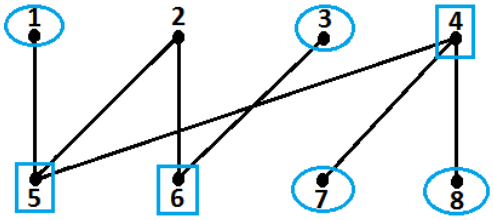


FIG. 3.5: S=$\{1, 7, 8, 3\}$

i=2

Marquons le dernier sommet du graphe.

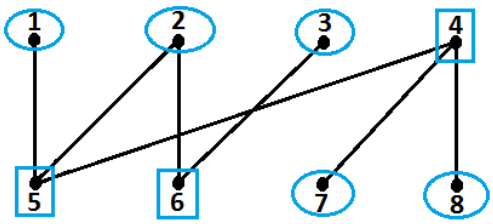


FIG. 3.6: S=$\{1, 7, 8, 3, 2\}$

Tous les sommets sont marqués alors stop.

On obtient à la fin un stable maximum qui est S=$\{1, 7, 8, 3, 2\}$ de cardinalité $|S|$=5.

3.3 Sécurisation des indépendants maximums

Dans cette section, nous étudierons la sécurité des indépendants maximums, tout en donnant la définition de la sécurité dans un ensemble stable et des exemples de graphe dans le but d'avoir les conditions nécessaires et suffisantes pour que l'ensemble stable soit sécurisé.

Définition 3.1. *Soit $x \in S$, on dit que x est sécurisé s'il existe un remplaçant dans $V - S$ du tel sorte S reste toujours stable.*

Définition 3.2. *Soit G=(V, E) un graphe et $S \subset V$ un stable maximum. On dit que S est* ***sécurisé*** *si $\forall x \in S$, $\exists y \in V - S$ tel que $S - \{x\} \bigcup \{y\}$ reste stable.*

Pour avoir la sécurité dans tout S à chaque fois qu'on enlève un élément de S il existe un remplaçant dans $V - S$ tel que S reste stable. Cela n'est pas toujours vérifié.

Proposition 3.1. *$\forall x \in S$ maximal, x ne peut être remplacé que par son voisin dans un stable.*

Preuve 3.1. – *Soit S stable on remplace $x \in S$ par un sommet dans $V - S - \Gamma(x)$, tous ces sommets sont adjacents aux sommets de S au moins une fois. Comme S est maximum, alors $\forall\ y \in V - S - \Gamma(x)$, l'ensemble $S - \{x\} \bigcup \{y\}$ ne reste pas stable.*

Proposition 3.2. *$a \in S$ sécurisé* $\Leftrightarrow \Gamma(a) - \Gamma(S - \{a\}) \neq \emptyset$

Preuve 3.2. *1. $a \in S$ sécurisé* $\Rightarrow \Gamma(a) - \Gamma(S - \{a\}) \neq \emptyset$.

$\Gamma(a) - \Gamma(S - \{a\}) = \emptyset \Rightarrow$ *a n'est pas sécurisé.*

$\Gamma(a) - \Gamma(S - \{a\}) \neq \emptyset$ *c'est à dire que* $\Gamma(a) \subset \Gamma(S - \{a\})$ *donc* $\nexists y \in V - S$ *tel que* $S - \{a\} \bigcup \{y\}$ *ne reste stable donc a n'est pas sécurisé.*

2. $\Gamma(a) - \Gamma(S - \{a\}) \neq \emptyset \Rightarrow$ *a sécurisé.*

$\Gamma(a) - \Gamma(S - \{a\}) \neq \emptyset$ *c'est à dire il existe au moin un voisinage de* $\{a\}$ *qui est adjacent seulement à lui,* $y \in \Gamma(a)$, *et* $y \notin \Gamma(S - \{a\})$ *donc* $S - \{a\} \bigcup \{y\}$ *est stable* $\Rightarrow$ *a est sécurisé.*

Dans le cas général : S **sécurisé** $\Leftrightarrow \forall a \in S$, $\Gamma(a) - \Gamma(S - \{a\}) \neq \emptyset$

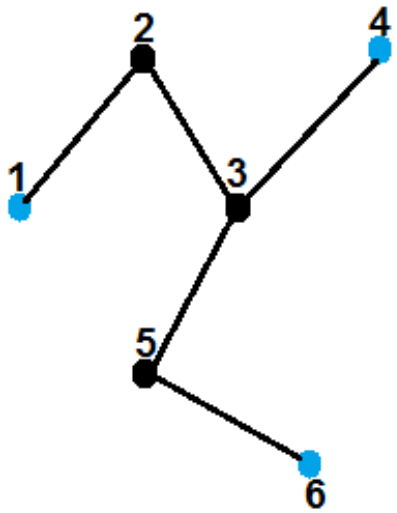


FIG. 3.7: S est sécurisé

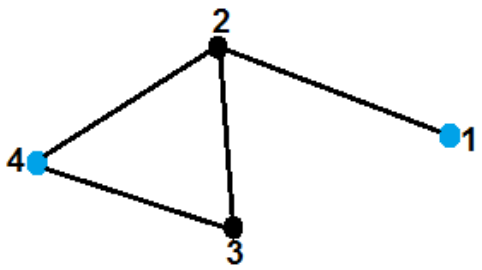


FIG. 3.8: S n'est pas sécurisé

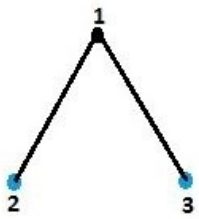
3.3.1 Sécurisation d'un indépendant maximum dans les graphes parfaits

• **Dans les arbres**

Dans cette première partie, on va donner quelques exemples de la sécurisation de S dans un arbre.

Pour que S soit sécurisé il faut que tout sommet $x \in S$, x ne peut être remplacé que par son voisin et $\forall y, z \in S$, $Adj(y) \bigcap Adj(z) = \emptyset$

– soit $G = (V, E)$, $V = \{1, 2, 3\}$

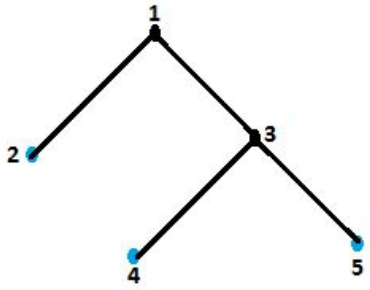


On as $S = \{2, 3\}$ et $V - S = \{1\}$

On remplace 2 par 1 on trouve $\{1, 3\}$ n'est pas stable car 1 est adjacent à 3,donc il

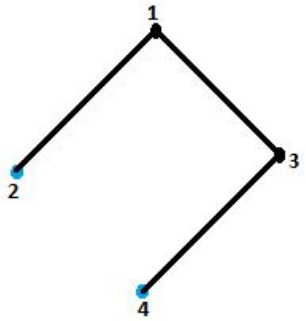
existe un sommet dans S qui n'a pas de remplacants dans $V - S$ alors S n'est pas securisé

– soit $G = (V, E)$, $V = \{1, 2, 3, 4, 5\}$



On a $S = \{2, 4, 5\}$, $V - S = \{1, 3\}$
-On commence par remplacer 2 par 1 on trouve que $\{1, 4, 5\}$ est un stable on passe au sommet 4 on le remplace par 1 on voit que l'ensemble {2,1,5} n'est pas stable car 1 est adjacent à 2 et à 5 or $2, 5 \in S$ donc il existe un sommet dans S qui n'a pas de remplaçant dans $V - S$ alors S n'est pas sécurisé.

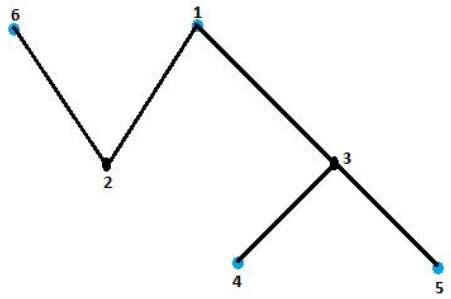
– Soit $G = (V, E)$ est un arbre et $V = \{1, 2, 3, 4\}$



On a $S = (2, 4)$,$V - S = \{1, 3\}$.

On remplace le sommet 2 par 1 on trouve que l'ensemble {1,4} forme un stable par contre si on le remplace par 3, l'ensemble {3,4} n'est pas un stable car 3 est adjacent à 4 même chose pour le sommet 4 on le remplace par 1,{2,1}ne forme pas un stable car 1 est adjacent à 2 mais si on le remplace par 3, l'ensemble {2,3} est un stable . On conclut que 2 et 4 ont des remplaçants dans $V - S$ alors S est sécurisé.

Soit $G = (V, E)$ est un arbre et $V = \{1, 2, 3, 4, 5, 6\}$



On a $S = \{4, 5, 6, 1\}$,$V - S = \{2, 3\}$

On remplace 4 par 2 ; on trouve que l'ensemble {2,5,6,1} n'est pas stable car $Adj(2)$=$\{1, 6\}$ même chose si on le remplace par 3, on trouve que l'ensemble {3,5,6,1} n'est pas stable car $Adj(3) = \{1, 4, 5\}$ et $\{1, 5\} \in S$ alors S n'est pas sécurisé.

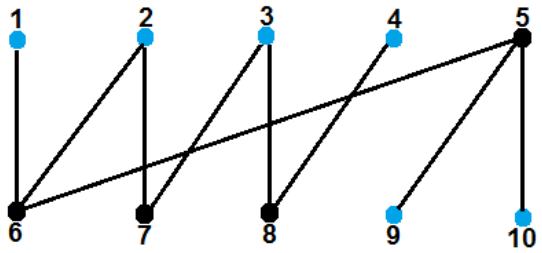
• Dans les autres graphes parfaits

Dans cette deuxième partie, on donne plusieurs exemples de la sécurisation des indépendants maximums dans les graphes parfaits .

Pour avoir la sécurité dans S, il faut que tout sommet $x \in S$, x ne peut être remplacé que par son voisin et $\forall y, z \in S$, il existe, au moins, un sommet $t \in V/S$ si $t \in Adj(y)$ $t \notin Adj(z)$, par contre dans un arbre touts les adjacents des sommets de S leurs intersections est vide.

Biparti

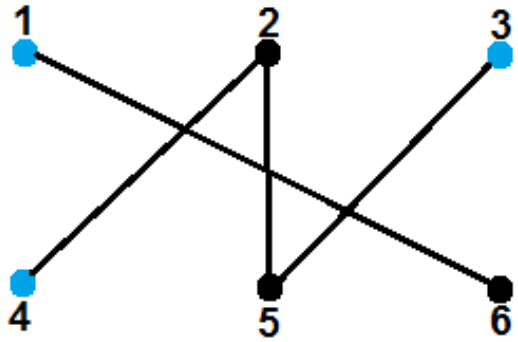
1. Soit G=(V, E) V=$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$



On a S=$\{1, 4, 9, 10, 2, 3\}$ et $V - S$=$\{5, 6, 7, 8\}$

– Si on remplace 1 par 5, on trouve un ensemble qui n'est pas stable car $Adj(5) = \{6, 9, 10\}$ or $\{9,10\}\in S$, sinon si on le remplace par 6 ou 7 ou 8, on trouve la même chose car $Adj(6) = \{1, 2, 5\}$ or $2\in S$, $Adj(7) = \{2, 3\}$; $\{2,3\}\in S$, $Adj(8) = \{3, 4\}$; $\{3, 4\} \in S$ alors S n'est pas sécurisé.

2. Soit G=(V, E) V=$\{1, 2, 3, 4, 5, 6\}$

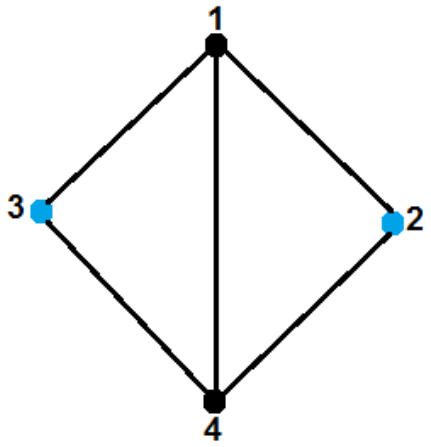


On a S=$\{1, 3, 4\}$ et $V - S$=$\{2, 5, 6\}$

– Dans ce cas on remplace 1 par 6 et l'ensemble S reste stable. Par contre, si on le remplace par 2 ou 5,S n'est plus stable car $Adj(2) = \{4, 5\}$; $4 \in S$ et $Adj(5) = \{2, 3\}$; $3 \in S$. On remplace 3 par 5 et non pas par 2 ou 6 car leurs adjacents sont dans S. On remplace 4 par 2 et non pas par 5 ou 6 car leurs adjacents sont dans S . Comme tous les sommets de S ont des remplaçants dans $V - S$, alors S est sécurisé.

Graphe d'intervalles

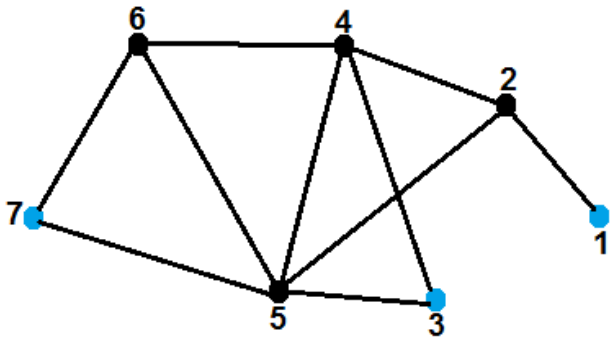
1. Soit G=(V, E) V=$\{1, 2, 3, 4\}$



On a S=$\{2, 3\}$ et $V - S$=$\{1, 4\}$

– Si on remplace 2 par 1, on trouve un ensemble qui n'est pas stable car $Adj(1) = \{2, 3\}$ Or 3$\in S$, sinon si on le remplace par 4, on trouve la même chose $Adj(4) = \{2, 3\}$; 3$\in S$ alors S n'est pas sécurisé.

2. Soit G=(V, E) V=$\{1, 2, 3, 4, 5, 6, 7\}$



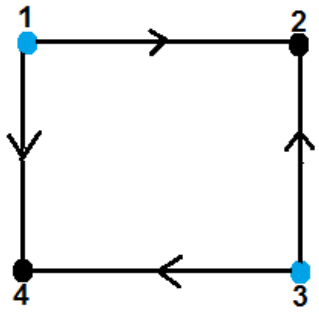
.

On a S=$\{1, 3, 7\}$ et $V - S$=$\{2, 4, 5, 6\}$

– Dans ce cas, $\forall x \in S$, $\exists y \in V - S$ tel que, à chaque fois qu'on remplace x par y, S reste stable : $S1$=$\{2, 3, 7\}$, $S2$=$\{1, 4, 7\}$, $S3$=$\{1, 3, 6\}$ alors S est sécurisé.

Graphe de comparabilité

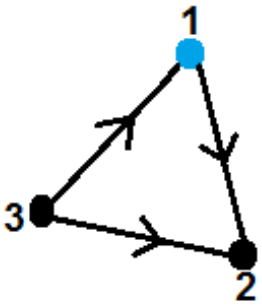
1. Soit G=(V, E) V=$\{1, 2, 3, 4\}$



On a S=$\{2, 3\}$ et $V - S$=$\{1, 4\}$

– Si on remplace 2 par 1, on trouve un ensemble qui n'est pas stable car $Adj(1) = \{2, 3\}$ Or 3$\in S$, sinon si on le remplace par 4, on trouve la même chose $Adj(4) = \{2, 3\}$; 3$\in S$ alors S n'est pas sécurisé.

2. Soit G=(V, E) V=$\{1, 2, 3\}$

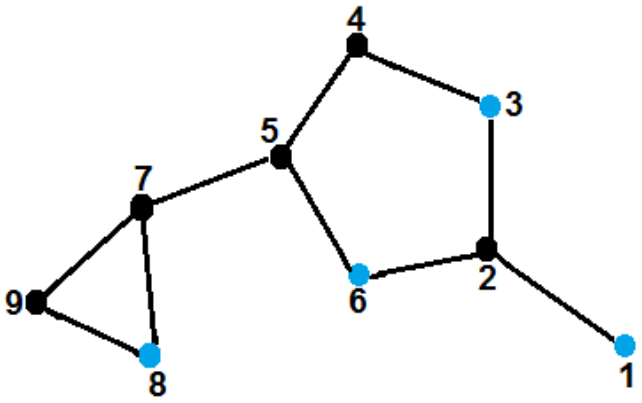


On a S=$\{1\}$ et $V - S$=$\{2, 3\}$

– Dans ce cas, $\forall x \in S$, $\exists y \in V - S$ tel que, à chaque fois qu'on remplace x par y, S reste stable : $S1$=$\{2\}$, $S2$=$\{3\}$ alors S est sécurisé.

Cactus

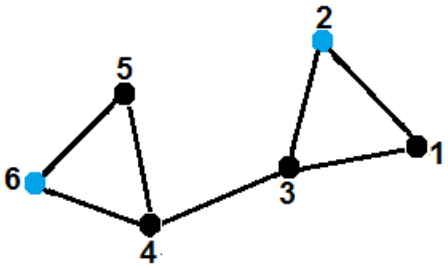
1. Soit G=(V, E) V=$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$



On a S=$\{1, 3, 6, 8\}$ et $V - S$=$\{2, 4, 5, 7, 9\}$

- Si on remplace 1 par 2, on trouve un ensemble qui n'est pas stable car $Adj(2) = \{1, 3, 6\}$ Or {3,6}$\in S$, sinon si on le remplace par les autres sommets de $V-S-\{2\}$, on trouve la même chose car tous leurs adjacents sont dans S alors S n'est pas sécurisé.

2. Soit G=(V, E) V=$\{1, 2, 3, 4, 5, 6\}$



On a S=$\{2, 6\}$ et $V - S$=$\{1, 3, 4, 5\}$

– Dans cet exemple, $\forall x \in S$, $\exists y \in V - S$ tel que, à chaque fois qu'on remplace x par y, S reste stable : $S1$=$\{1, 6\}$ ou $S1$=$\{3, 6\}$, $S2$=$\{2, 5\}$ ou $S2$=$\{2, 4\}$ alors S est sécurisé.

3.4 Application de la sécurisation des indépendants maximums

Dans cette section, nous allons présenter quelques situations concrètes pour les quelles nous allons modéliser notre problème sous forme d'un graphe afin d'étudier la sécurité en utilisant les propositions cités précédemment.

3.4.1 Le problème de l'allocation de fréquences dans un réseau de téléphone mobile

Définition 3.3. *Les télécommunications au sens large comprennent l'ensemble des moyens techniques nécessaires à l'acheminement aussi fidèle et fiable que possible d'informations entre deux points a priori quelconques, à une distance quelconque, avec des coûts raisonnables.*

Selon les données qu'on a récupérées de l'Internet concernant un réseau de télécommunication, il y a des émetteurs émettant chacun sur une fréquence particulière. Pour éviter les interférences sur le réseau, il ne faut pas allouer la même fréquence à deux émetteurs trop proches l'un de l'autre. De plus, à cause des contraintes financières, nous nous intéressons à remplacer un émetteur en cas d'une panne.

3.4.1.1 Modélisation du problème

Le problème considéré peut être résolu par l'étude de la sécurisation d'un indépendant maximum dans un graphe. En effet, il suffit de mettre un sommet pour chaque émetteur et une arête entre deux sommets, si et seulement si les deux émetteurs correspondants sont trop proches. Par conséquent, remplacer un emetteur par un autre en cas d'une panne

revient à la sécurisation d'un sommet dans le graphe correspondant.

Premier exemple étudié

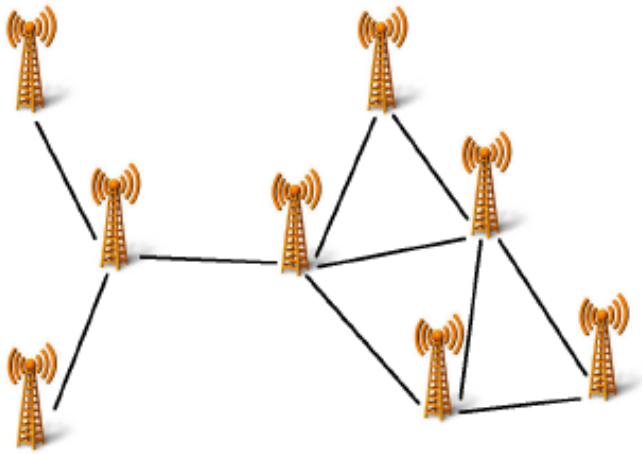


FIG. 3.9: Un réseau d'antennes

Modélisation du réseau

On peut modéliser le réseau sous forme d'un graphe non orienté $G=\{V, E\}$. V est constitué des émetteurs du réseau et E est défini comme suit : Deux émetteurs a et b sont reliés entre eux s'ils sont jugés proches l'un de l'autres.

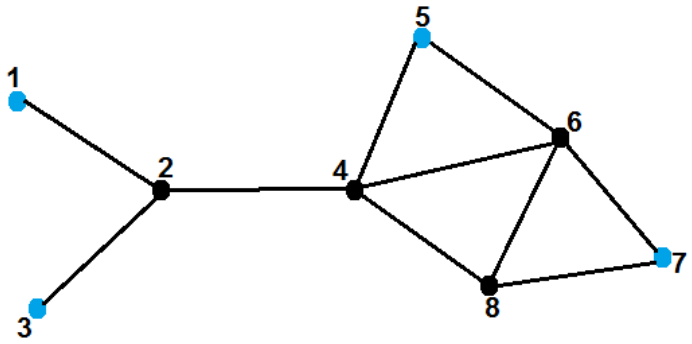


FIG. 3.10: Graphe associé au réseau étudié

V=$\{1, 2, 3, 4, 5, 6, 7, 8\}$

S=$\{1, 3, 5, 7\}$,stable maximum calculé par l'algorithme 5

$V - S$=$\{2, 4, 6, 8\}$

L'ensemble S n'est pas sécurisé car il n'existe pas de remplaçant pour les sommets $\{1, 3\}$. Si on remplace 1 par 2, 2 et 3 sont adjacents, 3 par 2, 1 et 2 sont adjacents. Et dans le réseau, deux sommets adjacents correspondent à deux émetteurs trop proche l'un de l'autre.

deuxième exemple étudié

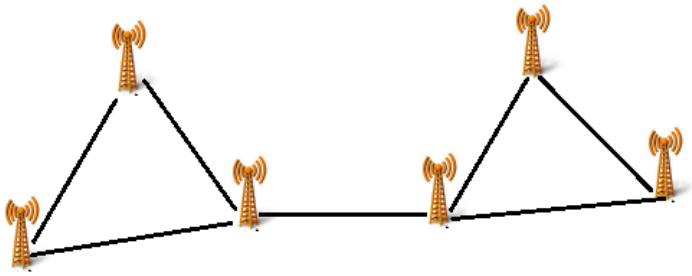


FIG. 3.11: Un réseau d'antennes

Modélisation du réseau

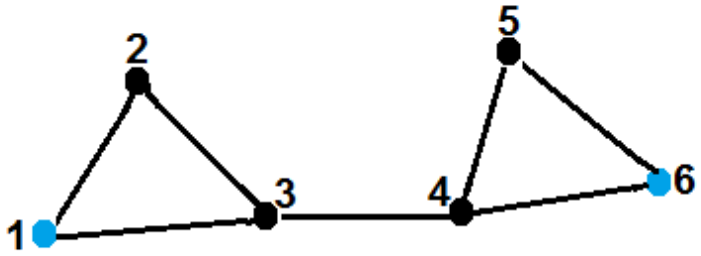


FIG. 3.12: Graphe associé au réseau étudié

V=$\{1,2,3,4,5,6\}$ S=$\{1,6\}$ stable maximum calculer par l'algorithme 5
$V-S$=$\{2,3,4,5\}$
L'ensemble S est sécurisé car il existe des remplaçants pour tous les sommets de S. Si on remplace 1 par 2, S reste stable, 6 par 5, S reste stable. Et dans le réseau un indépendant sécurisé veut dire que si un émetteur tombe en panne, il existe un autre émetteur qui peut le remplacer on lui émettant la même fréquence sur place.

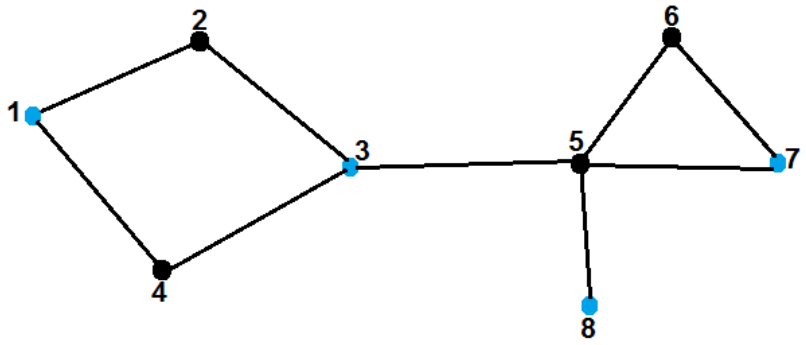
3.4.2 Le problème d'aviateurs pendant la guerre

Il y a des aviateurs qui viennent de partout dans le monde. Pour éviter les catastrophes, il ne faut pas avoir deux aviateurs qui ne se comprennent pas, dans le même avion.

3.4.2.1 Modélisation du problème

Le modèle associé au problème est un graphe G=(V,E) où V est l'ensemble des aviateurs et deux éléments de V sont en relation s'ils représentent deux aviateurs qui ne se comprennent pas. Avoir le maximum d'aviateurs à engager dans la guerre, c'est trouver un indépendant de cardinalité maximum dans le graphe. Il faudra ensuite, sécuriser le stable trouvé.

Premier exemple étudié



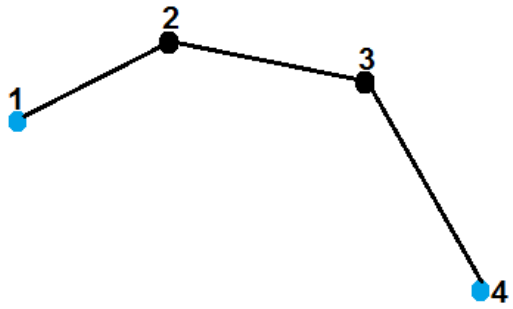
V=$\{1, 2, 3, 4, 5, 6, 7, 8\}$

S=$\{8, 7, 1, 3\}$,stable maximum calculé par l'algorithme 5

$V - S$=$\{2, 4, 5, 6\}$

L'ensemble S n'est pas sécurisé car il n'existe pas de remplaçant pour les sommets $\{1, 3, 8\}$. Et dans la guerre, deux sommets adjacents correspondent à deux aviateurs qui ne se comprennent pas.

Deuxième exemple étudié



V=$\{1, 2, 3, 4\}$

Comme le graphe correspondant est WCG, on a directement le stable maximum S=$\{1, 4\}$

$V - S$=$\{2, 3\}$

L'ensemble S est sécurisé car il existe des remplaçants pour tous les sommets de S. Et dans la guerre, un indépendant sécurisé veut dire que si un aviateur est tué, alors il existe

un autre aviateur qui peut le remplacer.

Conclusion générale

Ce travail présente une vision globale sur la théorie des graphes qui englobe un sujet très intéressant qui est la recherche de la sécurité dans les paramètres de graphes.

L'importance de la théorie des graphes vient aussi du fait qu'elle fournit un cadre conceptuel adéquat pour l'analyse et la résolution de nombreux problèmes. Elle constitue l'un des instruments les plus courants et les plus efficaces pour résoudre des problèmes discrets posés en Recherche Opérationnel(RO).
Le travail réalisé dans le cadre de ce mémoire peut se résumer en deux points essentiels.Le premier consiste en la recherche des indépendants maximums dans les graphes parfaits tout en suivant un ensemble d'instructions d'un algorithme qui est déjà donné. Le dernier point est consacré aux résultats trouvés sur la sécurité des indépendants maximums dans les graphes.

Nous avons donc, dans le premier chapitre, dressé un aperçu général sur les notions de la théorie des graphes, dont on avait besoin tout au long de ce mémoire. Le second chapitre est consacré à l'étude d'une autre classe de graphes "Well covered graphs" dans le but de faciliter la recherche des indépendants maximums. Dans le dernier chapitre, qui représente l'essentiel du travail élaboré, nous avons pu trouver les conditions nécessaires et suffisantes pour construire la sécurité dans les graphes. Nous avons également donné des exemples pratiques pour mieux comprendre les résultats trouvés.

Bibliographie

[1] A. Bondy, U.S.R Murty, Graph theory, 2008, Springer.

[2] A. Bretto, A. Faisan, F. Hennecart, Éléments de théorie des graphes, 2012, Springer.

[3] A. Brandstadt, J. P. Spinrad, Graph Classes, 1999, Siam.

[4] B. Sadi, Théorie des graphes : Complexité algorithmique, 1993, l'OPU.

[5] C. Berge, Graphes et hypergraphes, 1973, Dunod.

[6] C. Berge, Graphes, 1983, Gauthier-villars.

[7] C. Prins, M. Sevaux, P. Lacomme, Algorithmes de graphes, 2003, Eyrolles.

[8] H. Topart, Étude d'une nouvelle classe de graphes les graphes hypotriangulés, Thèse, 2011.

[9] J. C. Fournier, Théorie des graphes et applications, 2006, Antony Row L. td.

[10] J. F. Maurras, Programmation linéaire, Complexité : séparation et Optimisation, 2002, Springer-Verlag, Berlin Heidelberg.

[11] M. C. Golumbic, Algorithmic graph theory and perfect graphs, 2004, Elsevier.