كلية العلوم و التقنيات فاس
ⵜⴰⵖⵙⵓⵍⵜ ⵏ ⵜⵇⴰⵙⵙⵉⵡⵉⵏ ⴷ ⵜⵓⵙⵙⵙⵜⵉⵏ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⴰⵎⵎⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

DEPARTEMENT DES MATHEMATIQUES

Master Mathématique et Application au Calcul Scientifique (MACS)

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du Diplôme de Master Sciences et Techniques (MST)

Graphes Parfaits : Systèmes et Algorithmes

Réalisé par: KACHKACH Sara

Encadré par: HILALI Abdelmajid

Soutenu le jour mois année : 13-07-2021

Devant le jury composé de:

-Pr. HILALI Abdelmajid	FST-FES
- Pr. EL AYADI Rachid	FST-FES
-Pr. EL KHOMSSI Mohammed	FST-FES

Année Universitaire 2020 / 2021

FACULTE DES SCIENCES ET TECHNIQUES FES – SAISS

B.P. 2202 – Route d'Imouzzer – FES

☎ 212 (0)5 35 61 16 86 – Fax: 212 (0)5 35 60 82 14

Site web: http://www.fst-usmba.ac.ma

REMERCIEMENTS

Au terme de mon projet de fin d'études, j'adresse toute ma gratitude et mes vifs remerciements à mon professeur et encadrant ***Mr.HILALI Abdelmajid*** *d'être toujours mis à ma disposition et qui n'a jamais hésité à m'aider,de me répondre à toute question posée, à m'encourager et à me donner de précieux conseils pour réaliser un bon travail.*

je remercie aussi tous les membres de jury et professeurs de m'avoir fait l'honneur de juger mon travail. Veuillez trouver ici le témoignage de mon respect le plus profond.

je tiens à remercier aussi mes proches et principalement mes parents, qui m'ont encouragé le long de la préparation de ce projet, et qui m'ont offert toutes les conditions nécessaires pour un environnement favorable à la recherche, sans eux ce travail n'aurait pas pu être mené à son terme.

Merci infiniment à toutes les personnes qui ont contribué de prés ou de loin à l'élaboration de ce travail.

Table des matières

Table des figures

Liste des algorithmes

INTRODUCTION

Depuis sa découverte, la théorie de graphes réussit à résoudre plusieurs problèmes réels en commençant par la coloration des cartes, le problème de voyageur de commerce, en arrivant à l'affectation de fréquences radio, et bien d'autres problèmes dans des différents domaines.

La façon de résolution par la théorie des graphes consiste à simplifier tout d'abord le problème en le représente sous forme des graphes constituée par une série de sommets reliés par des arêtes, tout dépend des composantes des problèmes, par la suite les graphes obtenues vont être passés par un algorithme convenable afin de trouver une solution que se soit une optimisation du graphe, ou bien de déterminer le plus court chemin ou même de caractériser la structure du graphe.

En 1960, Claude Berge a décidé de créer une classe qui est un peu plus large contenant les graphes ayant une caractéristique spéciale ou bien une belle caractéristique, et il l'a appelé la classe des graphes parfaits et se diffère des autres classes par le fait que le nombre optimale de couleurs pour colorier un graphe est égale à la taille maximum d'une clique dans le même graphe.

Dans notre projet, on a consacré le chapitre 1 et 2 pour donner des rappels qui vont nous aider dans les chapitres qui suivent que se soient dans la notion des graphes, des algorithme et de la coloration des graphes. Le troisième chapitre on a défini les graphes parfaits , les conjectures de Berge et on a donné quelques graphes basiques dans cette classe, et dans le chapitre 4 j'ai répondu à la question : Comment savoir qu'un graphe est parfait ? puis dans le chapitre 5 on a défini une nouvelle notion de contraction des paires d'amis qui va nous aider par la suite à la coloration des graphes parfait (chapitre 6).

CHAPITRE 1

Notions de base

1.1 Notions des graphes

1.1.1 Concepts de base

Définition 1.1

Si S est un ensemble fini quelconque non vide et si k est un entier positif, alors on note $[S]^{(k)}$ l'ensemble des parties de S qui ont exactement k éléments.
Un **graphe simple** est un couple d'ensembles $G = (S, A)$ tel que $A \subseteq [S]^{(2)}$.
Les éléments de S sont appelés les *sommets* de G, et les éléments de A sont appelés les *arêtes* de G.

Si G est un graphe simple, alors S(G) désignera l'ensemble de ses sommets et A(G) l'ensemble de ses arêtes.
Exemple :

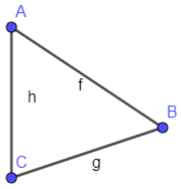


FIGURE 1.1 – Graphe simple.

Le graphe simple particulier $G = (S, \emptyset)$ est appelé graphe discret.

Définition 1.2

Le nombre de sommets d'un graphe simple est appelé son *ordre* , on le note Ord(G) ou |G|. le nombre d'arêtes d'un graphe simple est appelé sa *taille* , on le note tail(G).
On parlera d'un (n,p)-graphe pour désigner un graphe d'ordre n et de taille p.

Définition 1.3

On considère un (n,p)-graphe simple G = (S,A). Etant donné une arête $a = s_i s_j \in A$, on dit que s_i et s_j sont les extrémités de a,que a relie s_i à s_j . Les sommets s_i et s_j sont alors dits *adjacents (ou voisins)*, on dit aussi qu'ils sont *incidents* à l'arête a ou que l'arête a est incidente à s_i et s_j .
Un sommet qui n'as pas de voisins est dit *isolé*.
Deux arêtes sont adjacentes, distinctes si elles sont incidentes au même sommet.
On dit que le graphe simple G d'ordre n est **complet** , et on note K_n , si deux sommets quelconques, distincts de G sont adjacents.

Exemple :

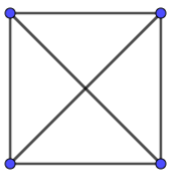


FIGURE 1.2 – K_4.

Définition 1.4

Etant donné un graphe simple G et $s \in S$, on appelle degré de s et on note $d_G(s)$ le nombre d'arêtes de A incidentes à s.

On dira que le sommet s est pendant si son degré est un.
Le degré d'un sommet isolé et nul.

Définition 1.5

Soient G = (S,A) un graphe simple et deux sous-ensembles $S' \subseteq S$, $A' \subseteq A$. On dit que $H = (S', A')$ est un **sous-graphe** de G si pour toute arête a de A' les extrémités de a sont dans S'.

Si en plus $S = S'$, le graphe $H = (S', A')$ est dit alors *sous-graphe partiel* de G.

Définition 1.6

Soit G = (S,A) un graphe simple et $S' \subseteq S$, on appelle sous-graphe engendé par S' le graphe $H = (S', A')$ où A' est l'ensemble des arêtes de G ayant leurs extrémités dans S' . On note $H = G[S']$

On note $G \setminus \{s\}$ le sous-garphe engendré par $S \setminus \{s\}$.

Définition 1.7

Soit G = (S,A) un (n,p)-graphe simple et $K_n = (S, A')$ le graphe complet dont l'ensemble des sommets est S. On appelle **complémentaire** de G, le graphe $\bar{G} = (S, \bar{A})$ où $\bar{A} = A' \setminus A$.

Exemple :

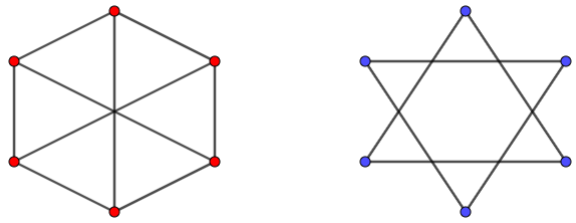


FIGURE 1.3 – Un graphe et son complémentaire.

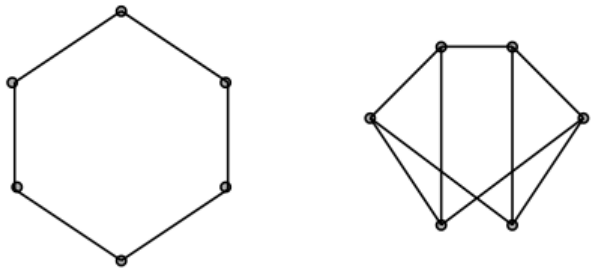
Définition 1.8

— Soient G = (S,A) graphe simple et C une partie de S, On dit que C est une **clique** de G si le graphe G[C] engendré par C est un graphe complet.
— On dira que C est une **clique maximale** de G si C est une clique et si le graphe G[B] n'est pas complet chaque fois que C est strictement contenue dans B.
— Une partie I de S est dite **stable** ou indépendante si le graphe complémentaire du graphe engendré par I est complet.

Chemin, cycle et trou

Définition 1.9

- On appelle **chemin** dans un graphe simple G = (S,A) une suite finie de sommets $(s_1...s_m)$ $(m \geq 2)$ telle que pour tout $1 \leq i \leq m-1$, $s_i s_{i+1} \in A$. On notera C =$[s_1...s_m]$ ou C = $(s_1...s_m)$. L'entier $m-1$ s'appelle la longueur du chemin. Les sommets s_1 ets_m sont appelés les extrémités du chemin.
- Un **cycle** de longueur $m-1$ est un chemin $\mu = (s_1...s_m)$ tel que $s_1 = s_m$
- Un chemin de longueur strictement supérieur à un dont toutes les arêtes sont distinctes est dit *simple*.
- Un chemin de longueur strictement supérieur à un dont tous les sommets sont distincts est dit **élémentaire**.
- Les sommets de l'ensemble $\{s_{\lfloor \frac{k+1}{2} \rfloor}, s_{\lceil \frac{k+1}{2} \rceil}\}$ sont dit milieu(x) du chemin
- Un **cycle élémentaire** est un chemin élémentaire fermé ayant au moins trois arêtes (seuls les sommets d'extrémité sont identiques)
- Les arêtes de G de la forme $s_i s_j$avec $|i-j| > 1$ sont appelées les **cordes** du chemin. Lorsque le cycle n'a pas de corde, on parle de cycle **sans corde**.
- On appelle **trou**, tout graphe T ayant au moins quatre sommets et dont les sommets peuvent être ordonnés de manière à former un cycle sans corde de T.
- Un graphe est un **antitrou** si son graphe complémentaire est un trou.
  On note par C_n (resp.$\bar{C}_n$) un trou (resp., antitrou) avec n sommets (voir figure ci-dessous)
- Un carré est un cycle sans corde avec 4 sommets noté C_4



C_6 $\overline{C_6}$

FIGURE 1.4 – Un trou et antitrou.

1.1.2 Graphes particuliers

Graphes connexes

Définition 1.10

Deux sommets s, s' distincts d'un graphe simple G = (S,A) sont connectés s'ils sont reliés par un chemin, ce que l'on notera $x \sim y$. La relation $\sim$ est une relation d'équivalence sur S. Une classe d' équivalence pour $\sim$ est appelée une composante connexe de G. Le graphe simple G est dit connexe si $X/\sim$ contient une seule classe d'équivalence, i.e., G possède une seule composante connexe
Si $\bar{G}$ est connexe, alors G est dit *co-connexe*

Définition 1.11

Soit G = (S,A) un graphe simple connexe (d'ordre $\geq$2). Un sommet s de G est un **point d'articulation** de G si $G - s$ est non connexe.
Une arête a de G est un **pont** de G si $G - a$ est non connexe.
En général, un **ensemble d'articulation** S de G est un ensemble des sommet tel que $G \setminus S$ contient plus de composante connexe que G

Graphes bipartis

Définition 1.12

Le graphe G = (S,A) est dit **biparti** s'il existe une partition en deux sous-ensembles S_1 et S_2 de S telle que S_1 et S_2 sont des stables.
Si de plus il existe une arête entre tout sommet de S_1 et tout sommet de S_2, alors on dit que G est un graphe *biparti complet*, avec $|S_1|$ = a et $|S_2|$ = b, alors on le note habituellement $K_{a,b}$.

Exemple :

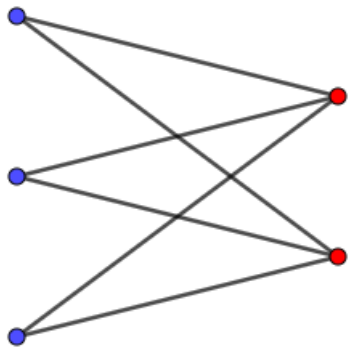


FIGURE 1.5 – Un graphe biparti.

Théorème 1.1

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

Griffe et diamant

1. **Griffe**

Définition 1.13

Le graphe *griffe* est un graphe biparti possédant 4 sommets et 3 arêtes

Exemple :

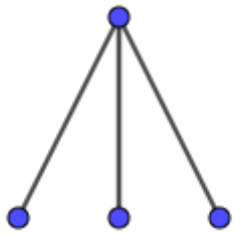


FIGURE 1.6 – La griffe.

2. **Diamant**

Définition 1.14

Le *diamant* est le graphe à quatre sommets dont le complémentaire ne contient qu'une seule arête

Exemple :

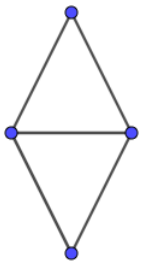


FIGURE 1.7 – Le diamant.

Prismes, pyramides

1. **Prismes :**
   - On appelle *prisme* tout graphe F dont les sommets se partitionnent en trois ensembles induisant des chemins P_1, P_2 ,P_3 tels que pour i = 1,2,3, P_i est de longueur au moins 1, d'extrémités a_i etb_i , tels que a_1 ,a_2 ,a_3 et b_1 ,b_2 ,b_3 induisent des triangles, et tels qu'il n'y ait aucune arête entre des sommets de P_i et P_j $(1 \leq i < j \leq 3)$ autre que celles des triangles.
     Si F est sous-graphe induit d'un graphe G, alors on dit que les trois cheminsP_1, P_2 et P_3 forment un prisme de G. On dit que les sommets a_i ,b_i , i = 1,2,3 sont les *coins* du prisme.
   - On dit que F est pair (resp. impair) si les trois chemins P_1 ,P_2 ,P_3 sont de longueur paire (resp. impaire).
     Exemples :

     - Un prisme impair :

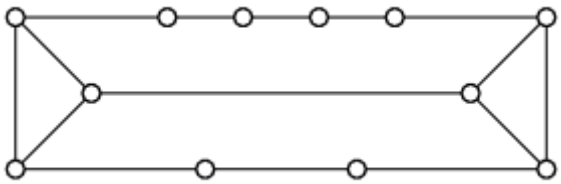


FIGURE 1.8 – Un prisme impair.

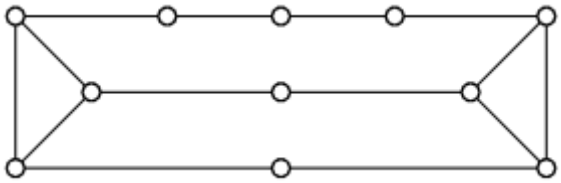


FIGURE 1.9 – Un prisme pair.

— Un prisme pair :

Lemme 1.1

Soit F un prisme ni pair ni impair. Alors F contient un trou impair.

2. **Pyramides :**

Définition 1.15

On appelle *pyramide* tout graphe F dont l'ensemble des sommets est l'union de trois ensembles induisant des chemins P_1 , P_2 , P_3 tels que pour i = 1,2,3, P_i est de longueur au moins 1, d'extrémités a et b_i , et tels que b_1 ,b_2 ,b_3 induit un triangle, tels que a est l'unique sommet commun aux trois chemins, et tels qu'il n'y ait aucune arête entre des sommets de P_i et Pj $(1 \leq i < j \leq 3)$ autre que celles du triangle et celles d'extrémité a. De plus, un seul chemin parmi P_1,P_2 ,P_3 est autorisé à être de longueur 1, les deux autres doivent être de longueur au moins 2.

Exemple :

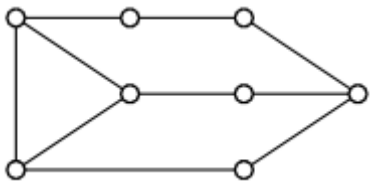


FIGURE 1.10 – Une pyramide.

Lemme 1.2

Soit F une pyramide. Alors F contient un trou impair.

1.2 Les algorithmes

Depuis lontemps, la recherche des algorithmes et qui donne des résultats rapidement enrichit globalement les mathématiques pures, d'ailleurs citons l'exemple de l'algorithme d'Euclide pour le calcul de PGCD qui même sa simplicité permet de prouver des grands théorèmes.
Donc, en 1960, la théorie de la compléxité des algorithmes est venue pour mettre le point sur le problème de la rapidité des algorithmes

1.2.1 Problèmes, algorithmes et compléxité

Définition 1.16

Un **problème** est une donnée d'un objet mathématique qui est représenté par un nombre fini des symboles (instance ou entrée) et d'une question dont la réponse dépend uniquement de l'instance
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.

Problèmes de décision et d'optimisation

Définition 1.17

Un **problème de décision** est un problème auquel la réponse est oui ou non

Définition 1.18

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

Définition 1.19

Un **algorithme** est une description de longueur finie, d'une suite finie d'opération élémentaires (le calcul) ne dépendant que de l'instance fournissant une réponse à la question pour aboutir à une sortie accopagné d'un test d'arrêt

Définition 1.20

La **compléxité** d'un algorithme est le nombre d'opérations élémentaires nécessaires à l'éxécution de l'algorithme pour une instance de taille n dans le pire des cas

Notation « grand O »

Définition 1.21

f(n) = **O**(g(n)) (f(n) est en grand **O** de g(n)) quand $n \to +\infty$ si et seulement si $\exists M > 0, n_0 \in N \quad tels \quad que \quad \forall n \geq n_0$,|f(n)| ≤ M|g(n)|

Définition 1.22

Un algorithme est dit *efficace* s'il est en temps polynomial c'est à dire si sa compléxité est un grand **O** d'un polynôme

Exemple :
Un algorithme dont la compléxité est **O**(2^n) sera donc par définition non éfficace.

Classes de complexité $\mathcal{P}$ et $\mathcal{NP}$

Définition 1.23

Un problème de décision est dans la classe $\mathcal{P}$ 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.

Définition 1.24

Un certificat du oui (resp. du non) pour un problème de décision est un objet représentable par une suite finie de symboles, dépendant seulement de l'instance et qui existe si et seulement si la réponse au problème est "oui" (resp. "non").
Un bon certificat est un certificat dont la taille est bornée par un polynôme en la taille de l'instance, et qui est vérifiable en temps polynomial. C'est-à-dire qu'il doit exister un algorithme en temps polynomial, prenant en entrée l'instance et le certificat, et qui répond "oui" (resp. "non") si et seulement si l'entrée est un certificat du oui (resp. du "non") pour cette instance

Définition 1.25

On dit qu'un problème appartient à la classe NP s'il existe pour ce problème un bon certificat du "oui".

Définition 1.26

La classe des problèmes $\mathcal{NP}$-*complet* est une classe de problèmes pour lesquels s'il est possible de résoudre l'un d'entre eux en temps polynomial, alors il est possible de résoudre tous les autres

CHAPITRE 2

Coloration de graphes

Les premiers résultats de la coloration des graphes ont été consacrés pour colorier les cartes, après elle devienne un domaine très intéressant de la théorie des graphes grâce à ses applications pour plusieurs problèmes comme l'ordonnancement des tâches, l'allocation des fréquences, etc.

2.1 Premières définitions

Définition 2.1

Soit k un entier. On appelle k-coloration du graphe G = (S,A) toute partition de S en k stables $S_1, ..., S_k$ de G.

Remarque :
Une k coloration du graphe $G = (S, A)$ est une application f de S à valeurs dans 1,2...,k telle que $f(s) \neq f(s')$ si s et s' sont adjacents.

Définition 2.2

On appelle **nombre chromatique** de G le plus petit entier k tel que G admet une k-coloration.On note $\chi(G)$ le nombre chromatique de G. Une coloration de G avec $\chi(G)$ couleurs est dite *optimale*

Exemple :

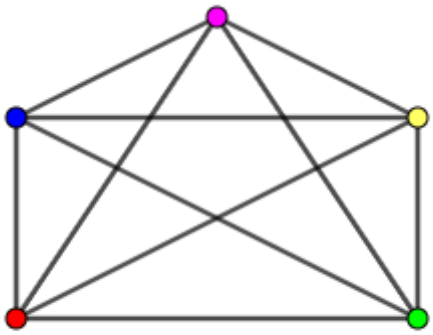


FIGURE 2.1 – $\chi(K_5) = 5$.

En général $\chi(K_n) = n$

2.2 Propriétés du nombre chromatique

— On note $\omega(G)$ la taille d'une clique de G de taille maximum c'est-à-dire :

$$\omega(G) = max\{|C| : C \quad est \quad une \quad clique \quad de \quad G\}$$

— On note $\alpha(G)$ la taille d'un stable de G de taille maximum c'est-à-dire :

$$\alpha(G) = max\{|C| : C \quad est \quad un \quad stable \quad de \quad G\}$$

où $|C|$ désigne le nombre d'éléments de C.

Proposition 2.1

Soit G un graphe simple d'ordre n.

1. $\chi(G) = 1$ si et seulement si G est discret.
2. $\chi(G) \leq n$.
3. Si G a au moins une arête alors, G est biparti si et seulement si $\chi(G) = 2$.
4. $\chi(C_n) = 2$ si n est pair et $\chi(C_n) = 3$ si n est impair.
5. Pour tout sous-graphe H de G on a $\chi(H) \leq \chi(G)$.
6. $\chi(G) \geq \omega(G)$

Preuve :

1. G n'a pas d'arêtes, donc aucun de ses sommets n'est adjacent à un autre et par la suite tous les sommets vont avoir la même couleur. *Exemple : voir figure 2.2*
2. il suffit de donner à chaque sommet une couleur différente.
3. Un graphe biparti est un graphe dont on peut partitionner l'ensemble des sommets en deux stables, d'où le résultat.
4. Par 1), $\chi(C_n) \geq 2$. Supposons que l'ensemble des sommets de C_n sont $s_1, s_2, ..., s_n$ avec $s_i s_{i+1} \in A$ pour tout $1 \leq i \leq n-1$ et $s_1 s_n \in A$. Si n est pair, alors on colorie tous les sommets d'indice pair avec une couleur et les autres avec une autre couleur, donc $\chi(C_n) \leq 2$. Si n est impair, on montre que C_n n'est pas 2-coloriable. Supposons qu'il le soit et qu'on colorie ses sommets en bleu et en rouge. On suppose que s_n est en bleu. Alors s_1 et s_{n-1} doivent être en rouge, s_2 et s_{n-2} doivent être en bleu et de façon générale s_k et s_{n-k} sont de la même couleur pour tout $1 \leq k \leq \frac{n}{2}$. Si n = 2m + 1, alors avec k = m, on voit que $s_m$et s_{m+1} sont de la même couleur, c'est une contradiction, donc $\chi(C_n) \geq 3$.
   Voir *Exemple : figure 2.3*
5. Facile, car tous les sommets adjacents dans H sont adjacents dans G.
6. $\omega(G)$ est le cardinal d'une clique maximum C de G, donc G[C] est complet et par la suite $\omega(G) = \chi(G[C]) \leq \chi(G)$

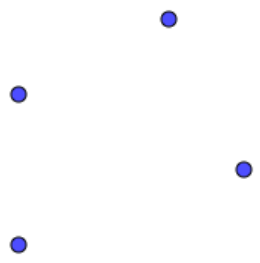


FIGURE 2.2 – Graphe discret

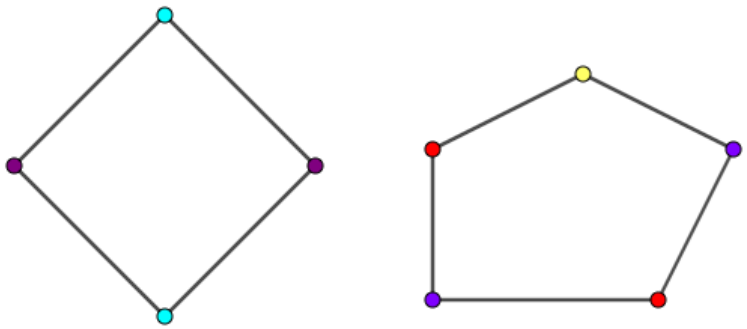


FIGURE 2.3 – $\chi(C_4) = 2, \chi(C_5) = 3$.

Proposition 2.2

Soit G = (S ,A) un graphe simple d'ordre n de degré maximum $\Delta(G)$, alors

$$\chi(G) \leq \Delta(G) + 1$$

Preuve :
Raisonnons par récurrence sur le nombre n de sommets du graphe.
Si n = 1 c'est trivial. Supposons l'assertion vraie pour tout graphe ayant au plus n 1 sommets, $n \geq 1$, et soit G un graphe avec n sommets. Si on supprime un sommet x le graphe obtenu G' a un degré maximum qui est au plus égal à $\Delta(G)$ (i.e. $\Delta(G') \leq \Delta(G)$). Par hypothèse de récurrence, on a $\chi(G') \leq (\chi(G) + 1)$.
Une $(\chi(G) + 1)$-coloration de G est obtenue en coloriant le sommet x avec une couleur différente des couleurs déjà affectées aux sommets adjacents à x (il y en a au plus $\chi(G)$).

Proposition 2.3

Soit G = (S ,A) un graphe simple connexe d'ordre n et de degré maximum $\Delta(G)$. Si G est un cycle élémentaire impair ou un graphe complet, alors

$$\chi(G) = \Delta(G) + 1$$

Preuve :
Si G est un cycle élémentaire impair, alors d'après 5 de la proposition 2.1 on a $\chi(G) = 3$. Dans un cycle élémentaire, tous les sommets sont de degré 2, donc $\Delta(G) = 2$. On en conclut que $\Delta(G) + 1 = 2 + 1 = 3 = \chi(G)$.
Si G est un graphe complet alors tous les sommets sont de degrés $n-1$, donc $\Delta(G) = n - 1$, et d'après 4 de la proposition 2.1 $\chi(G) = n$, on en conclut que $\chi(G) = n = \Delta(G) + 1$.

Proposition 2.4

Soit G = (S ,A) un graphe simple d'ordre n, alors

$$\lceil \frac{n}{\alpha(G)} \rceil \leq \chi(G) \qquad (*)$$

et

$$\chi(G) + \alpha(G) \leq n + 1 \qquad (**)$$

Preuve :

— Nous savons qu'une coloration des sommets est une partition $\{S_1, S_2, ..., S_{\chi(G)}\}$ de S . De plus pour tout $1 \leq i \leq \chi(G)$, S_i est un stable et nous avons $|S_i| \leq \alpha(G)$. Donc

$$n = |V_G| = |S_1 \cup S_2 \cup ... \cup S_{\chi(G)}| = \sum_{1 \leq i \leq \chi(G)} |S_i| \leq \chi(G)\alpha(G)$$

D'où $(*)$

— Si on colorie un ensemble stable maximum S_m avec une couleur, on utilisera au plus $|S|\alpha(G)$ couleurs pour colorier $S \setminus S_m$. Ainsi on a $\chi(G) \leq (|S|\alpha(G)) + 1 = n\alpha(G) + 1$. D'où $(**)$.

De les proposition 2.2 et 2.4, on obtient un encadrement du nombre chromatique comme le corollaire suivant l'indique :

Corollaire 2.1

Soit G = (S ,A) un graphe simple d'ordre n, alors

$$\lceil \frac{n}{\alpha(G)} \rceil \leq \chi(G) \leq \Delta(G) + 1$$

Proposition 2.5

Soit G = (S ,A) un graphe simple d'ordre n. Alors

$$\chi(G) \leq 1+max\{\delta(H), H \quad sous-graphe \quad induit \quad de \quad G\} \leq 1+\Delta(G)$$

Où $\delta(H)$ est le degré minimum des sommets de H.

Preuve :
Posons $k = max\{\delta(H), H - graphe \quad induit \quad de \quad G\}$.
Le graphe G contient un sommet x_n tel que $d_G(x_n) = \delta(G) \leq k$, car G est également un sous-graphe induit de G.
Si on supprime ce sommet on obtient le graphe $G_{n-1} = G[S \setminus \{x_n\}]$. Soit x_{n-1} un sommet de G_{n-1} tel que $d_{Gn-1}(x_{n-1}) = \delta(G_{n-1}) \leq k$.
On continue le processus jusqu'à l'obtention du graphe $G_1 = G[S\setminus\{x_n, x_{n-1}, ..., x_2\}]$ qui est constitué du seul sommet x_1 . Ainsi nous avons ordonné les sommets $x_1, x_2, ..., x_n$ de telle sorte que chaque $x_i, 1 \leq i \leq n$, a au plus k voisins dans G_i , car $\delta(G_i) \leq k$ est le degré de x_i dans G_i . Ainsi on a besoin d'au plus 1 + k couleurs pour colorier le graphe G (i.e.$\chi(G) \leq k + 1$).

Pour la deuxième inégalité il suffit de remarquer que pour tout sous-graphe induit H de G on a $\delta(H) \leq \Delta(H) \leq \Delta(G)$.

Corollaire 2.2

Soit G = (S ,A) un graphe simple connexe qui n'est pas régulier. On a

$$\chi(G) \leq \Delta(G)$$

Preuve :
Supposons que $\chi(G) > \Delta(G)$, c'est à dire $\chi(G) = \Delta(G) + 1$. D'après la proposition précédente il existe un sous graphe induit H tel que $\delta(H) = \Delta(G)$. Ainsi H est $\Delta(G)$-régulier. Ce graphe ne peut pas être connecté à un autre sommet de G car tous les sommets de H sont de degré $\Delta(G)$. Comme G est connexe, on a G = H donc G est régulier. Absurde.

Théorème 2.1 (Brooks 1941)

Soit G un graphe simple connexe. Si G n'est ni un cycle d'ordre impair, ni un graphe complet, alors

$$\chi(G) \leq \Delta(G)$$

.

2.3 Algorithmes de coloration

2.3.1 Algorithme glouton

On considère ici un coloriage comme une fonction définie de l'ensemble des sommets dans l'ensemble des entiers. L'algorithme glouton nous donne facilement un coloriage du graphe, le principe consiste à prendre les sommets les uns après les autres et pour chaque sommet s d'affecter la couleur minimale qui n'apparait pas dans les voisins coloriés de s.

Algorithme 1 Algorithme glouton de coloriage d'un graphe

Entrées: : Un graphe G=(S,A)
Sorties: : Une coloration $\phi : S \longrightarrow \mathbf{N}^*$ de G
  Pour $s \in S$ **faire**
    $\phi(s) \longleftarrow$ couleur non utilisée par les voisins de s
  Fin pour

L'algorithme s'arrête une fois que tous les sommets sont colorés.

Complexité :
On entre $|S|$ fois dans la boucle, chaque fois que l'on passe dans la boucle on regarde tous les voisins du sommet considéré, on a au plus $\Delta(G)$ voisins à regarder. Dans le pire des cas, on une complexité $\mathcal{O}(\Delta(G)|S|)$.
Remarque :
L'algorithme glouton ne donne pas toujours des resultats optimals, ça dépend de l'ordre dans lequel on choisit les sommets. *Exemple :*

FIGURE 2.4 – 2 colorations possibles.

2.3.2 Algorithme de Welsh-Powell

Le principe de cet algorithme est de clorier tout d'abord qui imposent le plus des contraintes (sommet de plus haut degré) et en utilisant la couleur qu'on vient d'utiliser là où celà est possible.
Pour certaines classes de graphes cet algorithme donne le coloriage optimal.

Algorithme 2 Algorithme de Welsh-Powell pour colorier un graphe

ENTRÉES: : Un graphe G=(S,A) ;
SORTIES: : Une coloration $\phi : S \longrightarrow \mathbf{N}$ de G
L $\longleftarrow$ liste des sommets ordonnés par dgré décroissant ;
couleur-courante $\longleftarrow$ 0 ;
Tantque $L \neq \emptyset$ **faire**
couleur-courante $\longleftarrow$ couleur-courante +1 ;
Colorier s le premier sommet de L avec couleur-courante ;
Eliminer s de L ;
V $\longleftarrow$ voisins de s ;
Pour $x \in L$ **faire**
Si $x \neq V$ **alors**
Colorier x avec la couleur-courante ;
Eliminer x de L ;
Ajouter les voisins de x à V ;
Finsi
Fin pour
Fin tantque

2.3.3 Algorithme COLOR

Etant donné un ordre sur les sommets du graphe, cet algorithme parcourt les sommets selon cet ordre et donne à chaque sommet la plus petite couleur non attribuée à ses voisins.
Remarque : L'algorithme COLOR peut ne pas donner une coloration optimale, prenons par exemple P_4 :

Algorithme 3 Algorithme COLOR

ENTRÉE : Un graphe G avec n sommets et un ordre σ sur sess sommets ;
SORTIE : Une coloration des sommets de G ;
CALCUL :
;
Pour i=1,...,n :
— choisir le sommet non colorié x qui est minimum pour σ
— colorier x avec la plus petite couleur qui n'est pas dans son voisinage
COMPLÉXITÉ : $\mathcal{O}(n+p)$

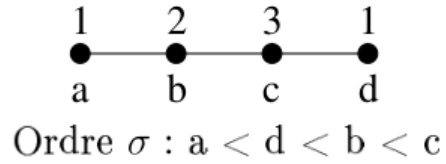


FIGURE 2.5 – Exécution de l'algorithme COLOR.

Définition 2.3

Soient G un graphe et σ un ordre sur les sommets de G. L'ordre σ est dit *Parfait* si, pour tout sous graphe induit H de G, la coloration obtenue par l'algorithme COLOR sur le graphe H et l'ordre σ restreint à H est optimale.
Un graphe G est dit *parfaitement ordonnable* s'il possède un ordre parfait

2.3.4 L'algorithme LexBFS-COLOR

Premièrement, l'algorithme LexBFS (Lexicographic Breadth-First Search) de Rose, Tarjan et Lucker. Cet algorithme parcourt les sommets d'un graphe et les numérote un par un de n à 1 ; lors de l'étape générale, chaque sommet non numéroté a une étiquette, correspondant à l'ensemble des numéros de ses voisins déjà numérotés : l'étiquette L(a) est strictement plus grande que l'étiquette L(b) s'il existe un entier $i \in L(a) \setminus L(b)$ tel que $\forall j > i, j \in L(a) \cap L(b)$ ou $j \notin L(a) \cup L(b)$. Le prochain sommet qui sera numéroté est le sommet dont l'étiquette est maximale selon l'ordre lexicographique
Alors l'algorithme LexBFS-COLOR est obtenu en appliquant l'algorithme COLOR à l'ordre inverse de celui donné par l'algorithme LexBFS

Algorithme 4 Algorithme LexBFS

ENTRÉE : Un graphe G
SORTIE : Un ordre σ sur les sommets de G.
CALCUL :
— Pour tout sommet a de G, soit $L(a) = \emptyset$
— Pour i=n,...,1 :
  — Choisir un sommet a d'étiquette maximale et poser $\sigma(a) = i$;
  — Pour chaque voisin non numéroté v de a, ajouter i à L(v)
COMPLEXITÉ : $\mathcal{O}(n + p)$
— choisir le sommet non colorié x qui est minimum pour σ
— colorier x avec la plus petite couleur qui n'est pas dans son voisinage
COMPLÉXITÉ : $\mathcal{O}(n + p)$

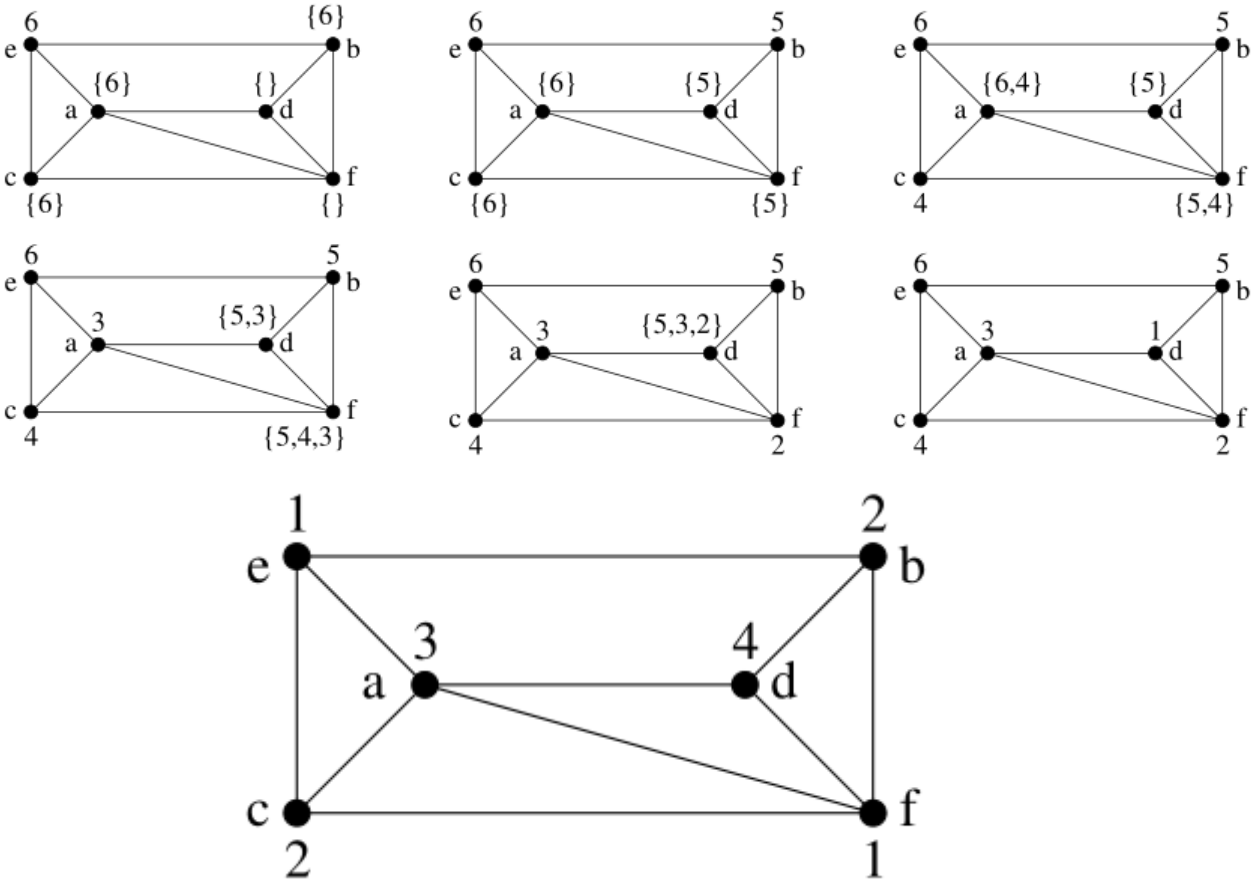


FIGURE 2.6 – Exécution de l'algorithme LexBFS-COLOR.

2.3.5 L'algoritgme COSINE

L'algorithme COSINE est de hertz, il sert à construire itérativement les classes de couleurs.
Notons c une classe de couleur, voilà l'écriture formelle de notre algorithme

Algorithme 5 Algorithme COSINE

ENTRÉE : Un graphe G
SORTIE :Une coloration des sommets de G
CALCUL :
— soit c=1
— Tant qu'il existe des sommets non coloriés
  — s'il existe un sommet non colorié qui n'a pas de voisins coloriés c :
    — Soit A l'ensemble des sommets non coloriés qui ont un voisin colorié c
    — Choisir un sommet non colorié v qui n'a pas de voisin dans A
    — colorier v avec la couleur c
  — Sinon, poser c = c+1
COMPLEXITÉ : $\mathcal{O}(np)$

2.3.6 L'algorithme LexCOLOR

L'algorithme LexCOLOR est de Roussel et Rusu [2], il associe des étiquettes $label_x$ de longueur n à chaque sommet x de G. Pour chaque couleur $1 \leq c \leq$, si x n'a pas de voisin colorié c, alors $label_x(c)$ est égal à 0 ; si x a un voisin colorié c, alors $label_x(c)$ est égal à l'entier i tel que le premier voisin de x colorié c est le (n-i)-ième sommet colorié du graphe. On définit l'ordre lexicographique sur les étiquettes de la manière suivante : $label_x <_L exlabel_y$si et seulement si $label_x(c) < label_y(c)$ et $\forall c' > c \quad label_x(c') = label_y(c')$.
A chaque itération, l'algorithme sélectionne un sommet non colorié dont l'étiquette est maximale pour l'ordre lexicographique, pour lui affecter de la plus petite couleur non présente dans son voisinage. cet algorithme s'arrête quand tous les sommets sont coloriés. Pour bien comprendre voilà l'algorithme sous sa forme usuelle :

Algorithme 6 Algorithme LexCOLOR

ENTRÉE : Un graphe G à n sommets
SORTIE :Une coloration des sommets de G
CALCUL :
— Pour tout sommet x et toute couleur c, soit $label_x(c) = 0$
— Pour i =1,...n :
  — Choisir un sommet non colorié x qui maximise $label_x$ pour $<_{Lex}$
  — colorier x avec la plus petite couleur c qui n'est pas présente dans son voisinage
— Pour tout voisin non colorié y de x si $label_y(c) = 0$, poser $label_y(c) = n - i$
COMPLEXITÉ : $\mathcal{O}(n^2)$

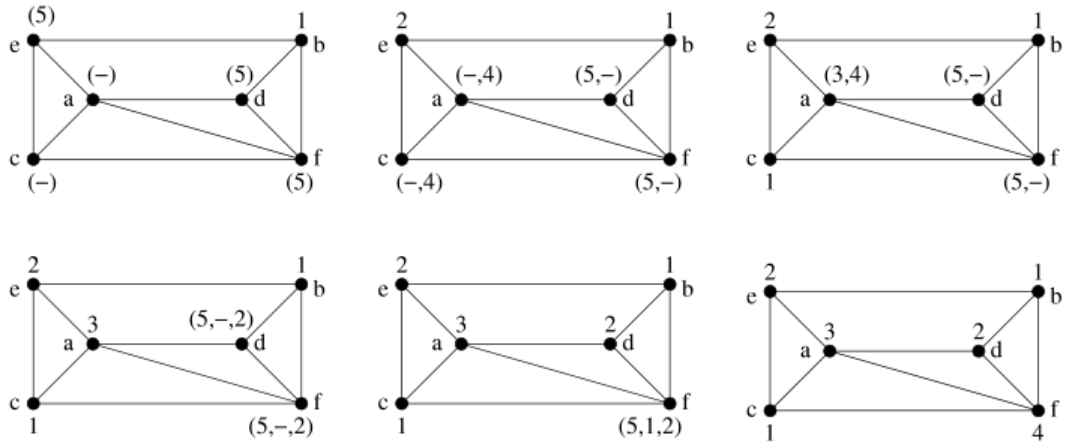


FIGURE 2.7 – Exécution de l'algorithme LexCOLOR.

CHAPITRE 3

Les graphes parfaits

Les graphes parfaits ont été introduits par Claude Berge au début des années 1960 à la suite d'un cheminement assez complexe qui sont par la suite devenus comme classe générale grâce à laquelle les problèmes de coloration pouvaient être résolus efficacement

3.1 Graphes α-parfaits et χ-parfaits

Pour un graphe simple G, désignons comme précédemment par
α(G) le nombre de stabilité,
θ(G) le nombre minimum de cliques qui partitionnent S,
χ(G) le nombre chromatique, ω(G) le nombre maximum de sommets qui forment une clique.
On sait que $\alpha(G) \leq \theta(G)$, puisque un ensemble stable ne peut avoir que 0 ou 1 sommet dans chaque clique de la partition ; de même on a $\omega(G) \leq \chi(G)$

Définition 3.1

[5] Un graphe G sera dit α-parfait si l'on a

$$\alpha(G_A) = \theta(G_A) \quad (A \subset S)$$

Un graphe G sera dit χ-parfait si l'on a

$$\chi(G_A) = \omega(G_A) \quad (A \subset S)$$

Où G_A est la restriction de G sur un ensemble de sommets A

Conjecture 3.1

Un graphe est α-parfait si et seulement si il est χ-parfait.

Théorème 3.1

[5] Un graphe G est α-parfait si et seulement si son complémentaire $\bar{G}$ est χ-parfait.

Preuve :
En effet, on sait que :

$$\alpha(G_A) = \omega(\bar{G}_A)$$

et

$$\theta(G_A) = \chi(\bar{G}_A)$$

Donc $\alpha(G_A) = \theta(G_A)$ est équivalent à $\omega(\bar{G}_A) = \chi(\bar{G}_A)$

Corollaire 3.1

Si un graphe G ou son complémentaire $\bar{G}$ contient un cycle élémentaire impair de longueur > 3 et sans cordes, G n'est ni α-parfait, ni χ-parfait.

Preuve :
Supposons que G qui contient un cycle élémentaire impair de longueur $2k + 1 > 3$ et sans cordes, soit donc A l'ensemble des sommets de ce cycle. Donc G_A n'est pas α-parfait car $\alpha(G_A) = k$ et $\theta(G_A) = k + 1$ et de même G_A n'est pas χ-parfait car $\chi(G_A) = 3$ et $\omega(G_A) = 2$.
Donc on peut conclure que G n'est ni α-parfait, ni χ-parfait

De même, si le graphe complémentaire $\bar{G}$ possède un tel cycle, il n'est ni α-parfait ni χ-parfait, donc d'après le théorème 3.1, le graphe G n'est ni α-parfait, ni χ-parfait.

Conjecture 3.2

[5] Pour un graphe G, les conditions suivantes sont équivalentes :

1. G est α-parfait
2. G est χ-parfait
3. G ne contient pas un ensemble A tel que G_A ou $\bar{G}_A$ soit un cycle élémentaire impair de longueur > 3 sans cordes

3.2 Graphes parfaits

Définition 3.2 (Berge)

Un graphe G est dit *parfait* si et seulement si pour chacun de ses sous-graphes induits G' on a

$$\chi(G') = \omega(G')$$

Définition 3.3

Un graphe G est dit de *Berge* si et seulement s'il ne contient ni trou impair ni antitrou impair.

Proposition 3.1

Soit G un graphe. Tout sous-graphe induit H de G contient un stable A tel que $\omega(H \setminus A) < \omega(H)$ si et seulement si G est parfait.

Théorème 3.2

Soit G un graphe. Le graphe G est parfait si et seulement si pour tout sous-graphe induit H de G, l'inégalité

$$|H| \leq \alpha(H)\omega(H)$$

est vérifiée.

Claude Berge a énoncé deux conjectures célèbres :

1. *Conjecture faible des graphes parfaits* qui se résume sous le théorème suivant :

   Théorème 3.3 (Lovász)

   Un graphe G est parfait si et seulement si $\bar{G}$ est parfait.

   Preuve :
   Il suffit de montrer que si G est parfait, alors $\bar{G}$ est parfait. Supposons donc G est parfait. Soit H un sous-graphe induit de $\bar{G}$. Alors $\bar{H}$ est un sous-graphe induit de G donc $|H| = |\bar{H}| \leq \alpha(\bar{H})\omega(\bar{H}) = \omega(H)\alpha(H)$. Donc $\bar{G}$ est parfait d'après le théorème précédent.

2. *Conjecture forte des graphes parfaits* qui se résume sous le théorème suivant :

Théorème 3.4

[1] Un graphe est parfait si et seulement s'il est de **Berge**.

Nous allons voir un peu plus loin (Les théorèmes de décomposition) des propositions concernant la démonstration de ce théorèmes

3.2.1 Les classes basiques

On trouve qu'on a 4 classes basiques des graphes parfaits :

1. **Les graphes bipartis**
   Leur perfection vienne du fait que :
   tout sous-graphe induit non trivial d'un graphe biparti a 2 pour nombre chromatique et pour nombre de clique. Voir figure 1.5.
2. **Complémentaire d'un graphe biparti**
   La perfection du complémentaire d'un graphe biparti se déduit de la conjecture faible des graphes parfaits.
   Exemple :
   On voit bien que le complémentaire du graphe biparti ci-dessus est

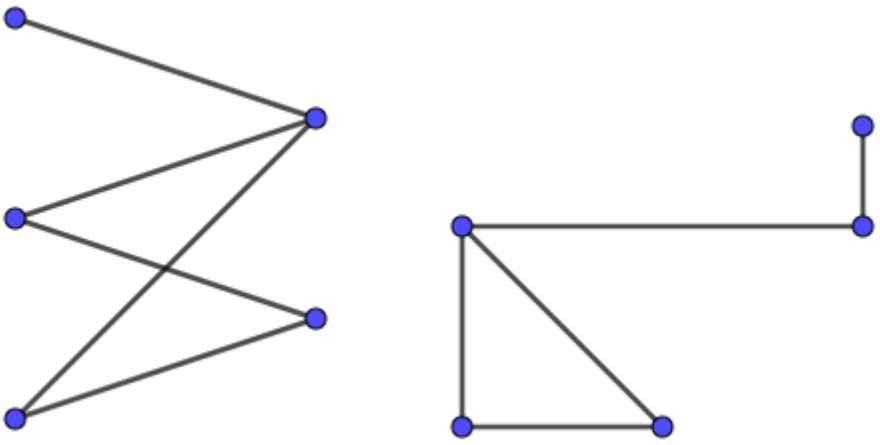


FIGURE 3.1 – Graphe biparti et son complémentaire.

parfait car $\omega(\bar{G}) = \chi(\bar{G}) = 3$. On prend comme une coloration de ce graphe la figure suivante :

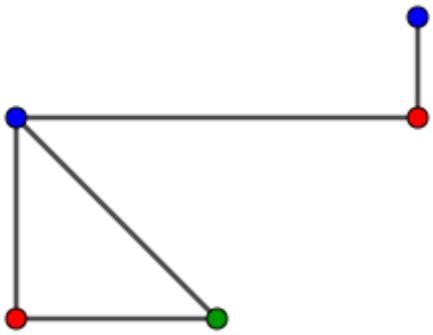


FIGURE 3.2 – $\chi(\bar{G}) = 3$.

3. **Line-graphes de bipartis (graphes adjoints)**

Définition 3.4

Le line graph L(G) d'un graphe non orienté G, est un graphe qui représente la relation d'adjacence entre les arêtes de G.
Autrement dit, le line-graphe L(G) est le graphe défini de la façon suivante :

— Chaque sommet de L(G) représente une arête de G ;
— Deux sommets de L(G) sont adjacents si et seulement si les arêtes correspondantes partagent une extrémité commune dans G (on dit alors qu'elles sont adjacentes).

.

Exemple de construction :

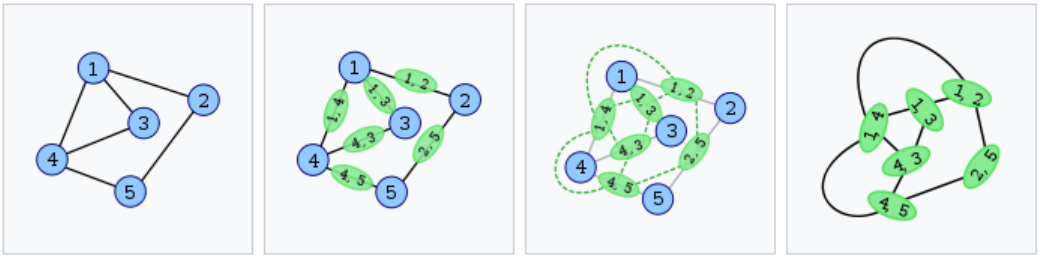


FIGURE 3.3 – Line-graphe.

Exemple d'un graphe adjoint d'un graphe biparti

En appliquant la définition précédente en se basant sur l'exemple d'application. Voilà un exemple d'un graphe biparti et son graphe adjoint :

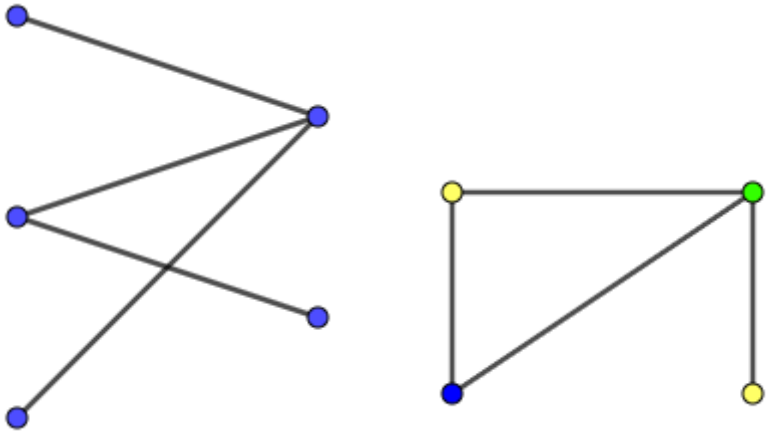


FIGURE 3.4 – Graphe biparti et son line-graphe.

Proposition 3.2

Le line-graphe d'un graphe biparti est parfait.

Preuve :
L'indice chromatique $\chi'(G)$ d'un graphe G est le nombre minimum de couleurs nécessaires pour colorier ses arêtes avec k couleurs distinctes de façon que deux arêtes ayant un sommet en commun ne soient pas de la même couleur. König a démontré que l'indice chromatique d'un graphe biparti est égal au degré maximum $\Delta(G)$ (le degré d'un sommet u est le nombre d'arêtes incidentes à u). Si L est le line-graphe d'un graphe biparti G, on a $\chi(L) = \chi'(G)$ et $\omega(L) = \Delta(G)$. Le théorème de König implique donc que $\chi(L) = \omega(L)$. La proposition découle maintenant du fait que les sous-graphes induits de L sont aussi des line-graphes de graphes bipartis.

Théorème 3.5 (Harary et Holzmann)

Soit G un graphe. G est un line-graphe de biparti si et seulement si G est sans griffe, sans diamant et sans trou impair.

4. **Complémentaire du line-graphe du graphe biparti**
   Le complémentaire du line-graphe d'un graphe biparti est parfait d'après la conjecture faible des graphes parfaits.
   Exemple :
   On prend comme exemple le line-graphe de la figure **??**. Donc son complémentaire est aussi parfait (voir la figure ci-dessous) :

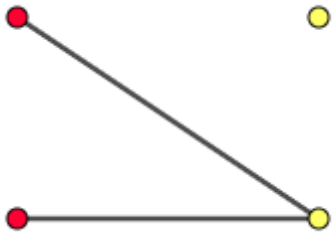


FIGURE 3.5 – Le complémentaire du line-graphe.

Remarque :
$\overline{L(G)}$ est parfait car $\chi(\overline{L(G)}) = \omega(\overline{L(G)}) = 2$.

3.3 Les théorèmes de décomposition

L'utilisation de théorèmes de décomposition pour l'étude de classes de graphes parfaits a commencé avant même que Berge ait formulé clairement sa célèbre conjecture.

3.3.1 Décomposition des graphes triangulés

Définition 3.5

- On appelle graphe *triangulé* si tout cycle de longueur p>3 admet une corde
- On appelle *clique d'articulation* d'un graphe tout ensemble d'articulation K de G, qui induit une clique de G. On appelle alors pièces de G les graphes $G[G_1 \cup K]$, ..., $G[G_k \cup K]$ où $G_1, ..., G_k$ sont les composantes connexes de $G \setminus K$.

Théorème 3.6 (Dirac)

Soit G un graphe triangulé. Alors ou bien G est une clique, ou bien G possède une clique d'articulation.

Théorème 3.7 (Gallai)

Soit G un graphe possédant une clique d'articulation. Si les pièces de G sont parfaites, alors G est parfait.

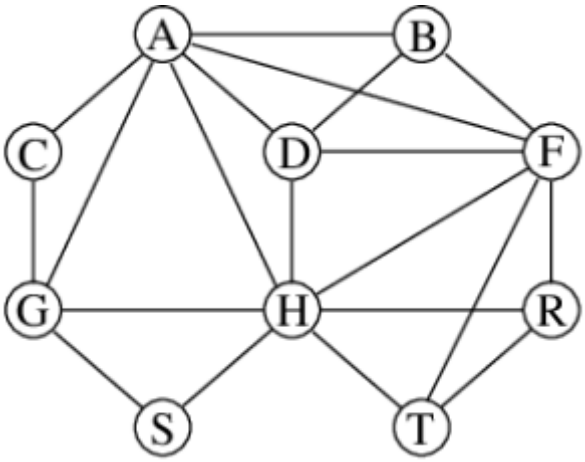


FIGURE 3.6 – Un graphe triangulé.

Théorème 3.8

Un graphe G est triangulé, alors G est parfait.

Preuve :
On peut prouver par une récurrence évidente : les cliques sont des graphes parfaits. Si un graphe triangulé n'est pas une clique, il a une clique d'articulation par le théorème 3.6, ses pièces sont parfaites par hypothèse de récurrence, et il est parfait par le théorème 3.3.1.

3.3.2 Décomposition par l'étoile d'articulation

Définition 3.6

- On appelle **étoile** tout graphe qui possède un sommet voyant tous les autres sommets du graphe.
- Soient G un graphe et $\alpha, \omega \geq 1$ des entiers . On dit que G est (α, ω)-*partitionnable* si et seulement si pour tout sommet s de G, le graphe $G \setminus s$ peut être partitionné en α cliques de taille ω et en ω stables de taille α.
- On appelle graphe *minimalement imparfait* tout graphe qui n'est pas parfait et tel que chacun de ses sous-graphes induits est parfait

Le théorème fort des graphes parfaits peut reformulé de la manière suivante :

Théorème 3.9 (Chudnovsky)

Les seuls graphes minimalement imparfaits sont les trous impairs et leurs compléments.

Théorème 3.10 (Lovasz)

Soit G un graphe minimalement imparfait . Alors G est partitionnable.

Théorème 3.11 (Chvátal)

Dans un graphe partitionnable (et donc dans un graphe minimalement imparfait) il n'y a pas d'étoile d'articulation.

3.3.3 Partition antisymétrique

Définition 3.7

On appelle *partition antisymétrique* d'un graphe G toute partition des sommets de G en deux ensembles A et B tels que G[A] n'est pas connexe et G[B] n'est pas anticonnexe.

Remarque :
Notons que pour les graphes ayant au moins cinq sommets et au moins une arête, S'ils contiennent une étoile d'articulation, alors ils peuvent avoir une partition antisymétrique

Chvátal a introduit une autre décomposition en se basant sur la notion de la partition antisymétrique et il a donc annoncé la conjecture ci-dessous :

Conjecture 3.3

Soit G un graphe minimalement imparfait. Alors G ne possède pas de partitition antisymétrique.

3.3.4 Décomposistion par 2-joint

Définition 3.8

Un graphe G a un 2-joint si ses sommets peuvent se partitionner en deux ensembles S_1 , S_2 , chacun de cardinalité au moins trois, contenant des sous-ensembles non vides disjoints $A_1, B_1 \subseteq S_1$et $A_2, B_2 \subseteq S_2$, tels que tous les sommets de A_1 soient adjacents à tous les sommets de A_2 , tous les sommets de B_1 soient adjacents à tous les sommets de B_2 et ces adjacences soient les seules entre S_1 et S_2

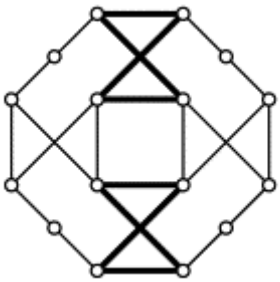


FIGURE 3.7 – Une décomposition par 2-joint.

Lorsqu'un graphe G a un 2-joint, on peut décomposer G en deux blocs G_1 et G_2. La construction de ces deux blocs se fait de la manière suivante :
Si A_2 et B_2 sont dans des composantes connexes différentes de G(S_2), définir le bloc G_1 comme étant G($S_1 \cup \{p_1, q_1\}$), où $p_1 \in A_2$ et $q_1 \in B_2$. Sinon, soit P_1 un plus court chemin avec une extrémité dans A_2 et l'autre dans B_2 , et définir le bloc G_1 comme étant G($S_1 \cup P_1$). Le bloc G_2 est défini de façon similaire.

Théorème 3.12 (Théorème de Décomposition par 2-Joint)

Soit G un graphe qui a un 2-joint. Le graphe G est parfait si et seulement si ses blocs G_1 et G_2 sont parfaits.

Corollaire 3.2

Soit G un graphe minimalement imparfait. Si G possède un 2-joint, alors G est un trou impair.

3.3.5 Paire homogène

La notion de paire homogène a été introduite en 1987 par Chvátal et Sbihi. Un graphe G a une paire homogène si S (G) peut être partitionné en sous-

ensembles A_1 , A_2 et B tels que :
– $|A_1| + |A_2| \geq 3$ et $|B| \geq 2$.
– Si un sommet de B est adjacent à un sommet de A_i , alors il est adjacent à tous les sommets de A_i , pour $i \in 1, 2$.

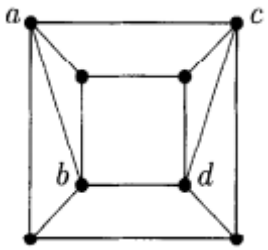


FIGURE 3.8 – Une décomposition par paire homogène.

Théorème 3.13 (Théorème de la Paire Homogène (Chvátal et Sbihi))

Un graphe minimalement imparfait ne peut pas avoir de paire homogène.

3.4 Décomposition des graphes de Berge

Théorème 3.14 (Conforti)

Soit G de Berge sans C_4 . Alors ou bien G est basique, ou bien G possède un 2-joint ou bien G possède une étoile d'articulation. En conséquence, G est parfait.

Définition 3.9

On appelle *étoile double* tout graphe G réduit à un sommet ou possédant une arête e telle que chaque sommet de G est adjacent à au moins l'une des extrémités de e

Théorème 3.15 (Conforti)

Soit G sans trou impair. Alors ou bien G est basique, ou bien G possède un 2-joint ou bien G possède une étoile double d'articulation.

Conjecture 3.4

Soit G un graphe minimalement imparfait. Alors l'un de G et $\bar{G}$ ne possède pas d'étoile double d'articulation.

On peut montrer le théorème fort des graphes parfaits en admettant la conjecture ci-dessus : soit G un contre-exemple minimal au théorème fort des graphes parfaits. Notons que G est de Berge. Donc G et $\bar{G}$ sont sans trou impair. D'après la conjecture faible, $\bar{G}$ est aussi un contre-exemple minimal. Appliquons le théorème 3.15 à G et $\bar{G}$. Si l'un de G, $\bar{G}$ est basique, on contredit la perfection des graphes basiques ; si l'un de G, $\bar{G}$ possède un 2-joint, on contredit le corollaire 3.2 ; donc G et $\bar{G}$ possèdent une étoile double d'articulation, ce qui contredit la conjecture 3.4.

Conjecture 3.5

Soit G un graphe de Berge. Alors ou bien G est basique, ou bien l'un de G, G possède un 2-joint, ou bien G possède une partition antisymétrique.

Théorème 3.16 (Théorème de Décomposition (Chudnovsky)

Tout graphe de Berge G est un graphe parfait élémentaire ou a une partition antisymétrique, ou a une paire homogène, ou G ou $\bar{G}$ a un 2-joint.

On peut montrer le théorème fort des graphes parfait en se basant du théorème et de conjecture précédents ; supposons que le théorème de décomposition soit vrai et qu'il existe un graphe G minimalement imparfait distinct d'un trou impair ou de son complémentaire. G ne peut pas avoir de partition antisymétrique par la conjecture 3.3. G ne peut pas avoir de paire homogène par le théorème 3.13. Ni G ni $\bar{G}$ ne peuvent avoir de 2-joint par le corollaire 3.2. Puisque G est un graphe de Berge, le Théorème de Décomposition implique que G est un graphe parfait élémentaire, ce qui contredit l'hypothèse que G est minimalement imparfait.

CHAPITRE 4

Problèmes de reconnaissance

4.1 Reconnaissance des graphes faiblement triangulés

Définition 4.1

On rappelle qu'un graphe est faiblement triangulé s'il ne contient ni trou long (c'est-à-dire de longueur au moins 5) ni antitrou long

Lemme 4.1

Soit G un graphe et H un long trou de G de taille minimale. Soit $a-u-b$ un chemin de H. Soit P un plus court chemin de a vers b ne contenant ni voisin de u ni voisin commun de a et b. Alors $\{u\} \cup S(P)$ induit un long trou de G de taille minimale.

Algorithme 7

INSTANCE : Un graphe G.
SORTIE : Si G possède un long trou, alors l'algorithme retourne un long trou de G de taille minimale. Sinon, il retourne "Pas de long trou".
COMPLEXITÉ : $\mathcal{O}(n^5)$.

L'algorithme ci-dessus est utilisé pour plusieures raisons. Pour notre cas des graphes faiblement triangulés ; il suffit de l'exécuter sur G puis sur $\bar{G}$.

4.2 Reconnaissance des graphes de Berge

La reconnaissance des graphes de Berge peut se faire par deux algorithmes mais le point commun entre eux est la phase de nettoyage.

Définition 4.2

— Soient G un graphe et C un trou de G. On dit qu'un sommet s de G est *C-majeur* si l'ensemble de ses voisins sur C n'est inclu dans aucun P_3 de C.
— On dit que C est *nettoyé* si G ne contient aucun C-majeur.
— Soient C est le plus petit trou impair de G, et X un sous-ensemble de sommets de G. On dit que X est nettoyeur de C si $X \cap S(C) = \emptyset$ et X contient tous les C-majeurs. Si X est nettoyeur de C, alors C est nettoyé dans $G \setminus X$
— On dit qu'un sous-ensemble de sommets X est *quasi-nettoyant* si X contient tous les C-majeurs et $X \cap S(C)$ est un sous-ensemble de sommets des P_3 de C.
—

4.2.1 Le premier algorithme

Définition 4.3

— un sous-ensemble de sommets d'un graphe G, X, est dit *anticonnexe* si le sous-graphe induit de $\bar{G}$ sur X est connexe
— Soit X un sous-ensemble de sommets de G, on dit qu'un sommet s est *X-complet* si $s \in S(G) \setminus X$ est adjacent à tout les sommets dans X.
— On dit qu'une arête a de G est *X-complète* si ses extrémités sont X-complets

Définition 4.4

On dit qu'un trou est **amenable** si :

1. C est un plus petit trou impair de G de longueur d'au moins 7
2. Pour tout ensemble anticonnexe de C-majeurs, elle existe une arête X-complète dans C

Cet algorithme est basé sur 3 grandes parties :

Détection des plus petits trous impairs dans un graphe

Cette partie consiste à donner un algorithme polynomial en considérant comme entrée un graphe G et un sous ensemble X de S(G) tel que si X est

un quasi-nettoyant pour un plus petit trou impair dans G, alors l'algorithme va découvrir un trou impair dans G. Avant de donner l'algorithme final pour cette partie, il faut éliminer les pyramides ou bien les diamants de notre graphe d'entrée.

Détection des pyramides

Définition 4.5

— Soit K est une pyramide, formé par les 3 chemins P_1, P_2 et P_3 reliant a par b_1, b_2 et b_3 respectivement, on définit le *cadre* de K le 10-uplet :

$$a, b_1, b_2b_3, s_1, s_2, s_3, m_1, m_2, m_3$$

tels que :

+ Pour i=1,2,3, s_i est le voisin de a dans P_i
+ Pour i=1,2,3, $m_i \in S(P_i)$ satisfait $d_{P_i}(a, m_i) - d_{P_i}(m_i, b_i) \in \{0, 1\}$ avec $d_{P_i}(a, m_i)$ est le plus petit chemin reliant a et m_i dans P_i, de même pour $d_{P_i}(m_i, b_i)$.

— Une pyramide K de Gest dit *optimal* s'il existe aucune pyramide K' de G tel que $|S(K')| < |S(K)|$

Lemme 4.2

Soit K une pyramide optimale, son cadre $a, b_1, b_2, b_3, s_1, s_2, s_3, m_1, m_2, m_3$, S_1 et T_1 sont les sous-chemins de P_1 d'éxtrémités m_1, s_1 et m_1, b_1 respectivement. Soit F l'ensemble de sommets non adjacents à s_2, s_3, b_2, b_3

1. Soit Q un chemin entre s_1 et m_1 avec des points intérieurs qui appartiennent à F de longueur minimale. Alors $a - s_1 - Q - m_1 - T_1 - b_1$ est un chemin noté $P_1^{'}$, et $P_1^{'}$, P_2 et P_3 forment une pyramide optimale
2. Soit Q un chemin entre m_1 et b_1 avec des points intérieurs qui appartiennent à F de longueur minimale. Alors $a - s_1 - S_1 - m_1 - T_1 - Q - b_1$ est un chemin noté $P_1^{'}$, et $P_1^{'}$, P_2 et P_3 forment une pyramide optimale

Algorithme 8 Algorithme de Chudnovsky et Seymour

ENTRÉE : Un graphe G.
SORTIE : Si G est sans pyramide, l'algorithme retourne "Pas de pyramide". Sinon, il retourne une pyramide de G de taille minimale.
COMPLEXITÉ : $\mathcal{O}(n^9)$.

Détection des jewels

Définition 4.6

On appelle une partie $s_1, ...s_5$, P est un *jewel* dans G, si $s_1, ...s_5$ sont des sommets distincts de G tels que $s_1s_2, s_2s_3, s_3s_4, s_4s_5, s_5s_1$ sont des arêtes, s_1s_3, s_2s_4, s_1s_4 ne sont pas des arêtes, et P est un chemin de G reliant $s_1$et s_4 qui ne contient aucun voisins de s_2, s_3 et s_5

Remarque :
Si un graphe G contient un jewel, alors G contient un trou impair.

Voici un algorithme qui détecte si un graphe contient un jewel ou non :

Algorithme 9 Algorithme de détection du jewel

ENTRÉE : Un graphe G.
SORTIE : décide si le graphe G contient un jewel ou pas
COMPLEXITÉ : $\mathcal{O}(n^6)$.

L'implémentation se fait comme suit :
Enumérer tous les 3-uplets s_2, s_3, s_5 de sommets distincts tels que s_2s_3 est une arête, pour tout choix de ce 3-uplet, trouver un ensemble F de tous les sommets non adjacents à chacun des sommets s_2, s_3, s_5 et trouver aussi ses composantes. Trouver X_1 l'ensemble de sommets adjacents à s_2, s_5 et non adjacents à s_3, et pour $s_1 \in X_1$ et pour toute composante F' de F, on cherche si F' contient des voisins de s_1. On fait la même démarche pour X_2 l'ensemble des sommets adjacents à s_3, s_5 et non adjacents à s_2.Puis tester s'il existe $s_1 \in X_1$ et $s_4 \in X_2$ et une comosante F' de F telle que s_1, s_4 son non adjacents et les deux ont des voisins dans F'. Si on a garanti l'existence alors on va obtenir comme sortie que G contient un jewel, sinon on obtient que G n'a pas de jewels.

Détection du plus petit trou impair nettoyé dans un graphe G

Théorème 4.1

Soit G un graphe qui ne contient pas une pyramide ou un jewel, C un plus petit trou impair nettoyé de G . Soient u,v des sommets de C distinct et non adjacents , L_1 et L_2 sont deux sous-chemins reliant u , v tels que $|A(l_1)| < |A(l_2)|$, alors :

- L_1 est un plus court chemin reliant u et v dans G
- Pour tout plus court chemin P reliant u, v dans G, $P \cup L_2$ est un plus petit trou impair dans G et il est nettoyé

Algorithme 10

ENTRÉE : Un graphe G qui ne contient pas une pyramide ou un jewel
SORTIE : Retourner soit "G contient un trou impair", sinon "il n'existe aucun plus petit trou impair nettoyé dans G
COMPLEXITÉ : $\mathcal{O}(n^4)$.

Voici l'algorithme final pour cette partie

Algorithme 11

ENTRÉE : Un graphe G qui ne contient pas une pyramide ou un jewel, X est un sous ensemble de S(G)
SORTIE : Retourner soit "G a un trou impair", sinon "il n'existe aucun plus petit trou impair nettoyé C dans G tel que X est un quasi-nettoyant de C
COMPLEXITÉ : $\mathcal{O}(n^4)$.

Test d'aptitude (The amenability test)

Cette partie consiste à donner un algorithme polynomial tel que si un graphe G contient un trou impair plus petit qui n'est pas amenable alors l'algorithme va découvrir que G n'est pas de Berge. afin d'aboutir un algorithme efficace, on doit encore une fois éliminer quelques configurations :

1. **Configuration de type $\mathcal{T}_1$** dans G est un trou de longueur 5.
   Il est clair que si G contient cette configuration, alors G n'est pas de Berge. Pour la détection de telle configuration dans un graphe, il existe un algorithme en complexité de $\mathcal{O}(n^5)$
2. **Configuration de type $\mathcal{T}_2$** dans G est une séquence v_1, v_2, v_3, v_4, P, X telle que :
   - $v_1 - v_2 - v_3 - v_4$ est un chemin dans G
   - X est une anti-composante de l'ensemble $\{v_1, v_2, v_4\}$-complet

— P est un chemin dans $G \setminus (X \cup \{v_2, v_3\})$ reliant v_1 et v_4, et il n'existe aucun point intérieur de P qui est X-complet ou qui est adjacent à v_2 ou à v_3.

On peut prouver que si G contient une telle configuration alors il n'est pas de Berge et pour chercher que cette configuration existe dans un graphe, il existe un algorithme de complexité $\mathcal{O}(n^6)$

3. **Configuration de type $\mathcal{T}_3$** dans G est une séquence $v_1, v_2, v_3, v_4, v_5, v_6$, P, X telle que :
   — $v_1, v_2, v_3, v_4, v_5, v_6$ sont des sommets distincts de G
   — v_1v_2, v_3v_4, v_1v_4, v_3v_5, v_4v_6 sont des arêtes et v_1v_3, v_2v_4, v_1v_6, v_2v_6, v_4v_5 ne sont pas des arêtes
   — X est une anti-composante de l'ensemble $\{v_1, v_2, v_5\}$-complet, et v_3, v_4 ne sont pas X-complets
   — P est un chemin de $G \setminus (X \cup \{v_1, v_2, v_3, v_4\}$ reliant $v_5$et v_6, et aucun sommet intérieur de P n'est X-complet ou adjacent à v_1 ou à v_2
   — Si v_5v_6 est une arête, alors v_6 est X-complet

   De même, si G contient une telle configuration, alors il n'est pas de Berge, et il existe un algorithme de complexité $\mathcal{O}(n^6)$ qui peut chercher cette configuration dans un graphe

Pour terminer cette partie, voilà un algorithme :

Algorithme 12

ENTRÉE : Un graphe G
SORTIE : Si G ou $\bar{G}$ contiennent un jewel, une pyramide, ou une configuration de type $\mathcal{T}_1, \mathcal{T}_2$, ou $\mathcal{T}_3$, retourner "G n'est pas de Berge"
COMPLEXITÉ : $\mathcal{O}(n^9)$.

Proposition 4.1

Soit G un graphe qui ne contient ni pyramides, ni configuration de type $\mathcal{T}_1, \mathcal{T}_2$, ou $\mathcal{T}_3$ et G, $\bar{G}$ ne contiennent pas de jewels, alors tout plus petit trou impair est amenable

La phase de nettoyage

Cette partie consiste à donner un algorithme qui a pour sortie des sous-ensembles de S(G) tels que si C est un trou amenable de G, alors l'un de ces sous-ensembles est un quasi-nettoyant pour C

Méthode et algorithme

En rassemblant tous les résultats précédents, on obtient l'algorithme suivant :

Algorithme 13

ENTRÉE : Un graphe G.
SORTIE : $\mathcal{O}(n^5)$ des sous-ensembles de S(G) tels que si C est un trou amenable de G, alors l'un de ces sous-ensembles est un quasi-nettoyant pour C
COMPLEXITÉ : $\mathcal{O}(n^5)$.

Algorithme 14 Algorithme de Chudnovsky, Seymour Cornuéjols, Liu, Vušković

ENTRÉE : Un graphe G.
SORTIE : Si G est de Berge, l'algorithme retourne Berge. Sinon, il retourne Non Berge.
COMPLEXITÉ : $\mathcal{O}(n^9)$.

Méthode :

Premièrement, on fait tourner l'algorithme 12 pour tester si G , $\bar{G}$ contiennent a jewel, une pyramide ou une configuration parmi les 3 types, si oui, alors on retourne que G n'est pas de Berge. Sinon par 4.1, tout plus petit petit trou impair est amenable, on fait tourner 13 on obtient $\mathcal{O}(n^5)$ sous-ensembles. Pour tout sous-ensemble X, on fait tourner l'algorithme 11 pour la paire G et X, si on trouve que G a un trou impair, on sort avec ce résultat, sinon on fait la même démarche pour $\bar{G}$, si on ne peut pas encore détecter un trou impair dans $\bar{G}$. Alors on peut conclure que G est de Berge.

4.2.2 Le deuxième algorithme

[8] L'idée de cet algorithme est de décomposer le graphe G à des graphes plus simples $G_1, ..., G_m$ tels que les deux propriétés suivantes sont satisfaites :

1. G est sans trou impair si et seulement si G_i est sans trou impair pour tout i=1,...,m
2. Pour tout i=1,...,m, il est facile de vérifier si G_i est sans trou impair

Les étapes principales de l'algorithme

1. La phase de nettoyage (Construction du graphe nettoyé)
2. Décomposition par double-étoile, en éliminant quelques configurations
3. Décomposition par 2-joint (le graphe n'a pas une double-étoile d'articulation)
4. Algorithme de reconnaissance des graphes nettoyé sans trou impair

Remarque :

Pour tester si un graphe G est parfait, il suffit d'appliquer l'algorithme de construction d'un graphe nettoyé 2 fois pour G et pour $\bar{G}$ et par la suite appliquer l'algorithme de reconnaissance des graphes sans trou impair pour les deux aussi.

4.3 Détection des sous-graphes

4.3.1 Prisme ou Pyramide

Nous donnons ici deux algorithmes de détection des prismes ou des pyramides. Appliqués à des graphes sans pyramide.

Définition 4.7

On dit qu'un sous-graphe de G est *optimal* si son nombre de sommets est inférieur ou égal au nombre de sommets de tout prisme de G, et inférieur ou égal au nombre de sommets de toute pyramide de G.

Lemme 4.3

Soit G un graphe quelconque. Soient P_1 , P_2 , P_3, trois chemins de G formant une pyramide optimale F de triangle b_1 ,b_2 ,b_3 . On note a_1 le coin de F et on suppose que pour i = 1,2,3, P_i a pour extrémités a_1 et b_i . Soit $P_1^{'}$ un plus court chemin de a_1 vers b_1 dont les sommets intérieurs manquent b_2 et b_3 . Alors ou bien :

1. L'ensemble S($P_1^{'}$) ∪ S(P_2) ∪ S(P_3) induit un prisme optimal de coin a_1 et de triangle b_1 ,b_2 ,b_3
2. Les chemins $P_1^{'}$, P_2 , P_3 forment une pyramide optimale de coin a_1 et de triangle b_1 ,b_2 ,b_3 .

De même pour P_2 et P_3

Lemme 4.4

Soit G un graphe quelconque. Soient P_1, P_2 , P_3 , trois chemins de G formant un prisme optimal F de triangles a_1 ,a_2 ,a_3 et b_1 ,b_2 ,b_3 . On suppose pour i = 1,2,3 que P_i a pour extrémités a_i et b_i. Soient :

- $P_1^{'}$ un plus court chemin de a_1 vers b_1 dont les sommets intérieurs manquent b_2 et b_3 .
- $P_2^{'}$ un plus court chemin de a_1 vers b_2 dont les sommets intérieurs manquent b_1 et b_3 .
- $P_3^{'}$ un plus court chemin de a_1 vers b_3 dont les sommets intérieurs manquent b_1 et b_2.

Alors :

(a) Les chemins $P_1^{'}$, P_2 , P_3 forment un prisme optimal de G de coin a_1 et de triangle b_1 ,b_2 ,b_3 .

(b) Ou bien V(P_1)$\cup$ V($P_2^{'}$)$\cup$ V(P_3) induit un prisme optimal de coin a_1 et de triangle b_1 ,b_2 ,b_3 , ou bien P_1 ,$P_2^{'}$, P_3 forment une pyramide optimale de coin a_1 et de triangle b_1 ,b_2 ,b_3 .

(c) Ou bien V(P_1)$\cup$ V(P_2)$\cup$V($P_3^{'}$) induit un prisme optimal de coin a_1 et de triangle b_1 ,b_2 ,b_3 , ou bien P_1 , P_2 , $P_3^{'}$ forment une pyramide optimale de G de coin a_1 et de triangle b_1 ,b_2 ,b_3 .

Lemme 4.5

Soit G un graphe sans pyramide. Soient P_1, P_2 , P_3 , trois chemins de G formant un prisme long minimal F de triangles a_1,a_2 ,a_3 et b_1 ,b_2 ,b_3 . On suppose pour i = 1,2,3 que P_i a pour extrémités a_i et b_i . Soit i 1,2,3. Soit$P_i^{'}$ un plus court chemin de a_1 vers b_i dont les sommets intérieurs manquent b_{i+1} et b_{i+2} (l'addition des indices s'entend modulo 3). Alors, l'ensemble V ($P_i^{'}$) $\cup$ V($P_{i+1}^{'}$) $\cup$ V ($P_{i+1}^{'}$) induit un prisme long optimal de G de coin a_1 et de triangle b_1 ,b_2 ,b_3 .

Algorithme 15

ENTRÉE : Un graphe G=(S,A) ;
SORTIE : Si le graphe est sans pyramide et sans prisme : "Ni prisme ni pyramide". Sinon, "Il y a un prisme ou une pyramide".
COMPLEXITÉ : $\mathcal{O}(n^3(n+p))$.

4.3.2 Prismes pairs

Définition 4.8

Soit P un chemin de longueur paire et d'extrémités a et b. On appelle *milieu* de P l'unique sommet m de P vérifiant lg(a-P-m) = lg(m-P-b). Soit F un prisme pair formé par les chemins P_1 , P_2 , P_3 , et de triangles a_1 ,a_2 ,a_3, b_1 ,b_2 ,b_3 , de sorte que pour i = 1,2,3, le chemin P_i soit d'extrémités a_i et b_i . Soit m_i le sommet au milieu du chemin P_i .
On dit alors que le 9-uplet (a_1 ,a_2 ,a_3 ,b_1 ,b_2 ,b_3 ,m_1 ,m_2 ,m_3) est la *trame* de F.

Lemme 4.6

Soit G un graphe sans trou impair et soit F un prisme pair de taille minimale de G. Supposons que F soit de trame (a_1 ,a_2 ,a_3 ,b_1 ,b_2 ,b_3 ,m_1 ,m_2 ,m_3) et formé par les chemins P_1 , P_2 , P_3 avec $a_i, m_i, b_i \in V(P_i)$ (i = 1,2,3). Soit R un chemin de G d'extrémités a_1 ,m_1 , dont les sommets intérieurs manquent a_2 ,a_3,b_2 et b_3 , et de longueur minimale avec ces propriétés.
Alors, $a_1 - R - m_1 - P_1 - b_1$ est un chemin de G que l'on note R_1 , et R_1 , P_2 , P_3 forment un prisme pair de G de taille minimale.

Algorithme 16

ENTRÉE : Un graphe G sans trou impair ;
SORTIE : Si le graphe est sans prisme pair : "Pas de prisme pair". Sinon : un prisme pair minimal de G.
COMPLEXITÉ : $\mathcal{O}(n^{11})$.

Remarque :
Cet algorithme est peu intéressant pour deux raisons : les prismes pairs jouent un rôle important dans la preuve de la conjecture forte des graphes parfaits, en plus, savoir détecter les seuls sous-graphes à la fois autorisés dans les graphes d'Artémis pairs et interdits dans les graphes d'Artémis.

4.3.3 Line-graphes de subdivisions biparties de K_4

Définition 4.9

La subdivision d'une arête uv conduit à un graphe contenant un nouveau sommet w et où l'on a remplacé l'arête uv par deux nouvelles arêtes, uw et wv .
Une subdivision d'un graphe G est le graphe résultant de la subdivision d'arêtes de G.

Le plus petit graphe biparti qui est une subdivision de K_4 est le graphe biparti $K_{3,3} \setminus a$ représenté dans la figure suivante :

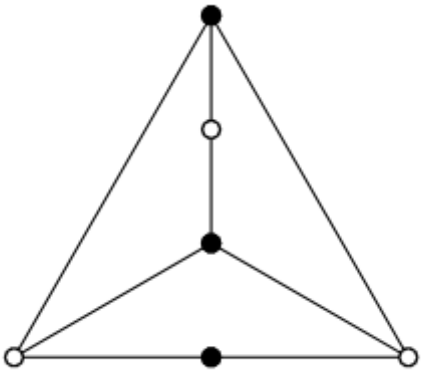


FIGURE 4.1 – $K_{3,3} \setminus a$.

Donc le line-graphe de cette subdivision est le suivant :

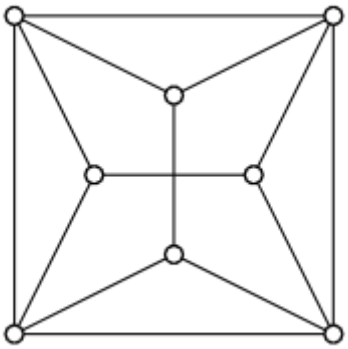


FIGURE 4.2 – $L(K_{3,3} \setminus a)$.

Lemme 4.7

Soit G le line-graphe d'une subdivision bipartie de K_4 . Alors G contient un prisme impair.

Preuve :
On suppose sans perte de généralité que a et b sont du même côté de la bipartition du K_4 . Donc l'arête ab du K_4 est subdivisée en un chemin de longueur paire. Les arêtes incidente à a et b donnent, dans le line-graphe G, les deux triangles d'un prisme, et le chemin de a vers b dans la subdivision de K_4 donne, dans le line-graphe, un chemin de longueur impaire. Comme G est de Berge, on sait que F est un prisme impair.

Lemme 4.8

Soit R une subdivision quelconque deK_4 . Alors on a l'un et l'un seulement des cas suivants : – R est triviale, c'est-à-dire R = K_4 .
– L(R) contient un trou impair.
– R est une subdivision bipartie de K_4 .

Lemme 4.9

Soit R une subdivision quelconque de K_4 . Alors ou bien : – R est triviale, c'est-à-dire R = K_4.
– L(R) contient un trou impair.
– L(R) contient un prisme impair.

Définition 4.10

Soit F le line-graphe d'une subdivision de K_4 avec les notations ci-dessus. Soient m_{ab}, m_{ac},m_{ad}, m_{bc}, m_{bd}et m_{cd} six sommets de F. Si pour tout i < j, le sommet m_{ij} est dans P_{ij} et vérifie $d_{Pij}(v_{ij}, m_{ij})d_{Pij}(v_{ji}, m_{ij}) \in 1, 0, 1$, alors on dit que le 18-uplet (v_{ab} ,v_{ac} ,...,v_{cd},m_{ab} ,...,m_{cd}) $\in V^1 8$ est une trame de F. Les six sommets m_{ab}, m_{ac},m_{ad}, m_{bc}, m_{bd}et m_{cd} s'interprètent comme des sommets "proches des milieux des chemins" P_{ab}, P_{ac},P_{ad}, P_{bc}, P_{bd}et P_{cd}.

Lemme 4.10

Soit G un graphe sans pyramide. Soit F un sous-graphe induit de G qui est une subdivision non triviale de K_4 . On suppose F de taille minimale et de trame (v_{ab} ,v_{ac} ,...,v_{cd},m_{ab} ,...,m_{cd}). Soit P un plus court chemin de v_{ab} vers m_{ab} dont les sommets intérieurs manquent tous les v_{ij} autres que v_{ab} . Alors : L'ensemble $(F \setminus v_{ab}P_{ab}m_{ab}) \cup P$ induit un line-graphe d'une subdivision non triviale de K_4 de taille minimale.

Algorithme 17

ENTRÉE : Un graphe G sans pyramide.
SORTIE : Si le graphe est sans line-graphe de subdivision non triviale de K_4 : "Pas de LGSNTK4". Sinon, l'algorithme retourne un line-graphe de subdivision non triviale de K_4 .
COMPLEXITÉ : $\mathcal{O}(n^{20})$.

L'algorithme suivant consacré pour les graphes sans pyramides On donne maintenant un autre algorithme de détéction des line-graphes d'une subdivision non triviale de K_4 ms dans des graphe sans trou impair

Algorithme 18

ENTRÉE : Un graphe G sans trou impair.
SORTIE : Si le graphe est sans line-graphe de subdivision bipartie de K_4 : "Pas de LGSBK4". Sinon, l'algorithme retourne un line-graphe de subdivision bipartie de K_4 de taille minimale.
COMPLEXITÉ : $\mathcal{O}(n^20)$.

4.3.4 Prismes impairs

Lemme 4.11

Soit G un graphe sans trou impair et sans line-graphe de subdivision bipartie de K_4 . Soit F un prisme de triangles a_1 ,a_2,a_3 et b_1 ,b_2,b_3 et formé par trois chemins P_i (i = 1,2,3) ayant pour extrémités a_i et b_i. Soit P un chemin de a_1 vers b_1 dont les sommets intérieurs manquent a_2,a_3, b_2 et b_3.
Alors : P_1, P_2 , P_3 forment un prisme de G de même parité que F.

Algorithme 19

ENTRÉE : Un graphe G sans trou impair.
SORTIE : Si G est sans prisme impair : "Pas de prisme impair". Sinon : "Il y a un prisme impair"
COMPLEXITÉ : $\mathcal{O}(n^{20})$.

CHAPITRE 5

Paires d'amis

Définition 5.1

Une paire d'amis(resp une paire P_4-libre) est une paire x, y de sommets non adjacents de G telle qu'il n'existe pas de chemin sans corde les reliant, de longueur impaire(resp de longueur 3)

Remarque :
Remarquons qu'une paire d'amis est une paire P_4-libre

5.1 Graphes de quasi-parité

Définition 5.2

- On dit qu'un graphe G est de *quasi-parité* si tout sous-graphe induit H de G non réduit à un sommet contient une paire d'amis de H ou une paire d'amis de $\bar{H}$.
- On dit qu'un graphe est de *quasi-parité stricte* si tout sous-graphe induit H de G différent d'une clique contient une paire d'amis de H.
- Tout graphe de quasi-parité stricte est un graphe de quasi-parité.

Théorème 5.1 (Meyniel)

Un graphe minimalement imparfait n'a pas de paire d'amis.

Remarque :
La perfection des graphes de quasi-parité se déduit de théorème de Meyniel et le théorème fort des graphes parfaits.
Nous allons voir maintenant quelques graphes parfaits de quasi-parité :

5.1.1 Graphes sans taureau :

Définition 5.3

le taureau est un graphe particulier à cinq sommets représenté ci-dessous :

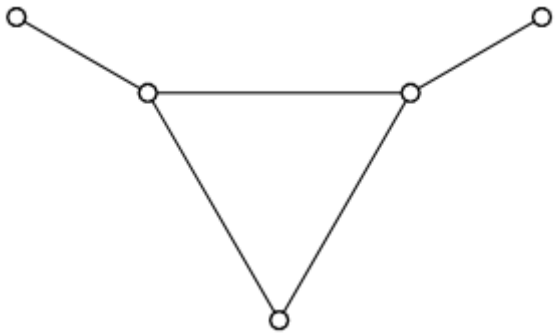


FIGURE 5.1 – Le taureau.

Vašek Chvátal et Najiba Sbihi ont démontré que les graphes de Berge sans taureau sont parfaits. Par la suite, Celina de Figueiredo, Frédéric Maffray et Oscar Porto ont montré que les graphes de Berge sans taureau sont des graphes de quasi-parité

5.1.2 Les conjectures de Hougardy

Théorème 5.2 (Hougardy)

Soit R un graphe biparti 3-connexe. Alors L(R) et $L(\bar{R})$ n'ont pas de paire d'amis.

Conjecture 5.1

Tout graphe qui n'est pas de quasi-parité stricte, et qui est minimal pour l'inclusion avec cette propriété, est ou bien un trou impair, ou bien un antitrou long, ou bien le line-graphe d'un graphe biparti.

Conjecture 5.2

Soit G un graphe qui n'est pas de quasi-parité et qui est minimal pour l'inclusion avec cette propriété. Alors l'un de G, $\bar{G}$ est un trou impair ou le line-graphe d'un graphe biparti.

5.1.3 Les graphes bipartisans

Définition 5.4

On appelle graphe bipartisan tout graphe de Berge sans prisme long, sans complémentaire de prisme long, sans double-diamant et sans $L(K_{3,3} \setminus s)$.

Conjecture 5.3

Soit G un graphe bipartisan. Alors G est de quasi-parité.

5.1.4 Détection des paires d'amis dans les line-graphes

On va donner premièrement un algorithme qui décide si le graphe donné en entrée est un line-graphe :

Algorithme 20 Algorithme de Lehot et Roussopoulos

ENTRÉE : Un graphe G=(S,A) ;
SORTIE : Un graphe R tel que G = L(R). Si un tel graphe n'existe pas, l'algorithme retourne "G n'est pas un line-graphe".
COMPLEXITÉ : $\mathcal{O}(n+m)$.

Ajoutons à cet algorithme un autre algorithme de Jack Edmonds :

Algorithme 21 Algorithme de Edmonds

INSTANCE : Un graphe G et deux sommets a et b de G.
SORTIE : Une chaîne de G, de longueur paire, reliant a et b et de longueur minimale — si une telle chaîne existe. Sinon, "pas de chaîne de longueur paire".
COMPLEXITÉ : n^3.

De plus on a besoin du lemme suivant :

Lemme 5.1

Soit R un graphe et G = L(R). Soit F un ensemble d'arêtes de R. Alors F est l'ensemble des arêtes d'une chaîne de R si et seulement si F induit un chemin de G.

Alors voilà notre algorithme qui sert à détécter les paire d'amis dans un line-graphe :

Algorithme 22

INSTANCE :Un line-graphe G et deux sommets a et b de G.
SORTIE : Un chemin de longueur impaire reliant a et b, s'il existe un tel chemin. Sinon, "a, b est une paire d'amis de G".
COMPLEXITÉ : n^3.

5.2 Les graphes parfaitement contractiles

Définition 5.5 (La contraction)

Soit un graphe G=(S,A), on définit l'opération de contraction comme suit : on enlève tout d'abord deux sommets x et y de G, puis on ajoute un nouveau sommet noté xy relié à chaque sommet de $N(x) \cup N(y)$ .On note G/xy le graphe obtenu.
On dit que xy représente x dans G/xy (notons que xy représente aussi y)

Théorème 5.3

Soit G un graphe et x,y une paire d'amis de G. Alors :

1. Il existe une coloration optimale de G qui donne la même couleur à x et y.
2. $\chi(G) = \chi(G/xy)$
3. $\omega(G) = \omega(G/xy)$
4. Si G est parfait, alors G/xy est parfait.

Remarque :
La contraction d'une paire d'amis peut transformer un graphe imparfait en un graphe parfait

Définition 5.6

- Un graphe G est dit *contractile*(resp P_4-*libre-contractile* si G est une clique ou s'il contient une paire d'amis (resp.une paire P_4-libre) dont la contraction donne un graphe contractile (resp.P_4-libre-contractile)
- Le graphe G est dit*parfaitement contractile*(resp.parfaitement P_4-libre-contractile) si tout sous-graphe induit de G est contractile (resp.P_4-libre-contractile)

Remarque :
Remarquons qu'un graphe contractile est P_4-libre-contractile

Conjecture 5.4

Soit G un graphe parfaitement contractile différent d'une clique. Alors G possède une paire d'amis dont la contraction redonne un graphe parfaitement contractile.

Lemme 5.2

Soit G un graphe P_4-libre-contractile, alors $\chi(G) = \omega(G)$

Lemme 5.3

Un graphe parfaitement P_4-libre-contractile est parfait

Définition 5.7

On appelle *graphe d'Artémis pair* tout graphe sans trou impair, sans antitrou long et sans prisme impair.On peut l'appeler aussi un graphe de Grenoble.
En général, un *graphe d'Artémis* est un graphe sans trou impair, sans antitrou long et sans prisme

Conjecture 5.5

Un graphe est parfaitement contractile si et seulement s'il est un graphe d'Artémis pair.

Conjecture 5.6

Si un graphe est d'Artémis, alors il est parfaitement contractile.

Maintenant, on donne quelques exemples des graphes parfaitement contractiles et qui sont d'Artémis.

1. Les graphes de Meyniel

Définition 5.8

Un graphe est dit de Meyniel si tous ses cycles impairs de longueur au moins 5 possèdent au moins deux cordes.

Exemple :

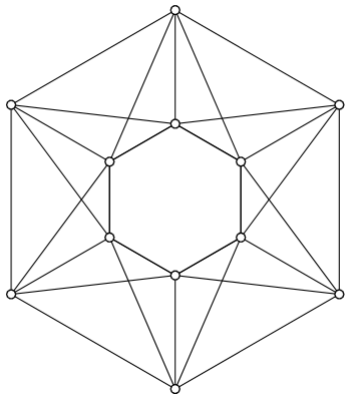


FIGURE 5.2 – Un graphe de Meyniel.

2. Les graphes faiblement triangulés

Définition 5.9

Un graphe est *faiblement triangulé* si et seulement s'il ne contient ni trou long (c'est-à-dire de longueur au moins 5) ni antitrou long.

5.3 Ordre de contraction

Pour un graphe G, un ordre de contraction de G est une notation de ses sommets sous la forme $S(G) = \{x_1^1, ..., x_1^{k_1}, x_2^1, ..., x_2^{k_2}, ..., x_l^1, ..., x_l^{k_l}\}$tels que, pour $1 \leq c \leq l$, les ensembles $V_c = \{x_1^1, ..., x_c^{k_c}\}$ sont des stables.
La coloration qui, à chaque sommet x_c^i associe la couleur c, est appelée coloration associée à cet ordre.
La suite de contractions associée à cet ordre est la suite de graphes G_c^i et les sommets w_c^i $(1 \leq c \leq l,\ 1 \leq i \leq k_c)$ définis de la manière suivante :
Soit $G_1^1 = G$. Pour $2 \leq c \leq l$, soit $G_c^1 = G_{c-1}^{k_c-1}$. Pour $1 \leq c \leq l$, soit $w_c^1 = x_c^1$. Pour $1 \leq c \leq l$ et $2 \leq i \leq k_c$, soit G_c^i le graphe obtenu à partir de G_c^{i-1} en contractant les sommets w_c^{i-1} et x_c^i en un nouveau sommet w_c^i .

Pour $1 \leq c \leq l$, le sommet $w_c^{k_c}$ est noté tout simplement w_c
Un ordre de contraction est dit *amical*(resp P_4-libre) si la suite de contraction associée vérifie, pour tout $1 \leq c \leq l$ et $2 \leq i \leq k_c$:

1. Les sommets w_c^{i-1} et x_c^i sont une paire d'amis(resp.une paire P_4-libre) de G_c^{i-1}
2. Le sommet w_c est adjacent à tous les sommets de $G_c^{k_c}$

Voici un algorithme qui nous donne un ordre de contraction amical à partir d'un ordre parfait :

Algorithme 23 Algorithme OrdreContraction

ENTRÉE : Un graphe G et un ordre σ sur ses sommets ;
SORTIE : Un ordre de contraction $\{x_1^1, ..., x_1^{k_1}, x_2^1, ..., x_2^{k_2}, ..., x_l^1, ..., x_l^{k_l}\}$ et la coloration associée
CALCUL :
— Soient c=1 et i=1 ;
— Tant qu'il existe des sommets non coloriés :
  — S'il existe un sommet non colorié qui n'a pas de voisin colorié c :
    — Choisir un tel sommet v minimum pur σ ;
    — Colorier v avec la couleur c, poser $x_c^i = v$ et i=i+1 ;
  — Sinon, poser c=c+1 et i=1
COMPLEXITÉ : $\mathcal{O}(n+p)$.

On a aussi un algorithme qui retourne un ordre de contraction :

Algorithme 24 Algorithme OrdreCosine

ENTRÉE : Un graphe G
SORTIE : Un ordre de contraction $\{x_1^1, ..., x_1^{k_1}, x_2^1, ..., x_2^{k_2}, ..., x_l^1, ..., x_l^{k_l}\}$ et la coloration associée
CALCUL :
— Soient c=1 et i=1 ;
— Tant qu'il existe des sommets non coloriés :
  — S'il existe un sommet non colorié qui n'a pas de voisin colorié c :
    A l'ensemble des sommets non coloriés qui ont un voisin colorié ;Choisir un sommet non colorié v qui n'a pas de voisin colorié c et qui a le nombre maximum de voisins dans A Colorier v avec la couleur c, poser $x_c^i = v$ et i=i+1 ;
  — Sinon, poser c=c+1 et i=1
COMPLEXITÉ : $\mathcal{O}(np)$.

CHAPITRE 6

Coloration des graphes parfaits

Premiers résultats

I. Pour colorier un graphe parfait quelconque, on dispose juste d'un algorithme polynomial([5]) ci-dessous :

Algorithme 25 Gröstchel, Lovász, Schrijver

ENTRÉE : Un graphe parfait G.
SORTIE : Une coloration optimale de G
COMPLEXITÉ : Polynomiale.

Cet algorithme utilise la méthode des ellipsoïdes qui a la réputation d'être très difficile à implémenter en raison de problèmes d'instabilité numérique.
C'est pour cela, dans ce chapitre on se restreint juste sur la coloration de quelques graphes parfaits.

II. Si G est un graphe parfait possédant une paire d'amis. L'algorithme fonctionne comme suit :
Tant qu'on trouve dans le graphe G une paire d'amis, on la contracte, on peut donc suivre la trace de tout sommet v de G au cours des contractions successives et chaque sommet de G est représenté par un unique sommet dans le graphe résultant. On colorie le graphe résultant (qui ne possède pas de paire d'amis), et l'on en déduit une coloration du graphe de départ en donnant à chaque sommet la couleur de son représentant dans le graphe résultant.

6.1 Graphes sans taureau

Les graphes sans taureau sont les graphes qui ne contiennent ni trou impair, ni antitrou, ni taureau. On note la classe de ces graphes $\mathcal{B}$.
Dans cette partie on va donner un algorithme qui sert à trouver pour tout graphe de cette classe, une coloration optimale.

Définition 6.1

- Un chemin sans corde $v_1 - v_2 - ... - v_k$ est dit *simplicial* s'il ne s'étend pas à un chemin sans corde $v_0 - v_1 - v_2 - ... - v_k - v_{k+1}$
- Un sommet P_k-simplicial est un sommet v tel que tout chemin sans corde sur k sommets dont v est un milieu est un chemin simplicial
- Un ordre d'élimination P_k-simplicial est un ordre (v_1,...,v_n) des sommets de G tel que, pour $1 \leq i \leq n$, le sommet v_i est P_k-simplicial dans le graphe $G[v_1, ..., v_i]$
- Un sommet est dit $\bar{P}_k$-simplicial, s'il est P_k-simplicial dans le graphe complémentaire
- Un ordre d'élimination $\bar{P}_k$-simplicial est un ordre (v_1,...,v_n) dans des sommets de G tel que pour $1 \leq i \leq n$, le sommet v_i est $\bar{P}_k$-simplicial dans le graphe $G[v_1, ..., v_i]$

6.1.1 L'algorithme LexBFS*

Ce qui nous interesse par la suite est la notion de P_3-simplicialité, car un sommet P_3-simplicial est un sommet qui n'est pas le milieu d'un P_5. Malheureusement l'algorithme LexBFS ne peut pas donner un ordre d'élimination P_3-simplicial. Donc l'algorithme LexBFS* est venu comme un cas particulier de l'algorithme LexBFS pour départager les cas d'égalité de ce dernier afin d'obtenir l'ordre d'élimination voulu. Notons $\mathcal{C}$ la classe des graphes , ne

Algorithme 26 Algorithme LexBFS*

ENTRÉE : Un graphe G.
SORTIE : Un ordre σ sur les sommets de G
CALCUL :

1. Pour tout sommet a, soit $L(a) = \emptyset$;
2. Pour i=n,...,1 :
   - 2.1 Soit A l'ensemble des sommets non numérotés ayant une étiquette maximum, et soit U les autres sommets non numérotés ;
   - 2.2 Tant que $|U| > 0$;
     - 2.2.1 Choisir un sommet $u \in U$ pour lequel $L(u) \setminus L(A)$ est maximum ;
     - 2.2.2 Poser $U = U \setminus \{u\}$. Si $A \cap N(u) \neq \emptyset$, alors poser $A = A \cap N(u)$
   - 2.3 Choisir un sommet $a \in A$ et poser $\sigma(a) = i$;
   - 2.4 Pour chaque voisin non numéroté v de a, ajouter i à L(v)

COMPLEXITÉ : $\mathcal{O}(n \times min(p, \bar{p})$.

contenant ni trou, ni taureau.
Nous allons indiquer le théorème qui confirme que l'ordre produit par l'al-

gorithme LexBFS* d'un graphe de la classe $\mathcal{C}$ est un ordre d'élimination P_3-simplicial .
Pour prouver ce théorème nous avons besoin du lemme suivant :

Lemme 6.1

Dans un graphe $G \in \mathcal{C}$, soit $P = a_0 - a_1 - ... - a_r$ un chemin sans corde avec $r \geq 4$. Soit u un sommet qui voit a_0 et a_r. Alors l'une des propositions suivantes est vraie :

1. u voit les sommets de P
2. r est pair, u voit $a_0, a_2, ..., a_r$ et manque $a_1, a_3, ..., a_{r-1}$
3. r=4, u voit a_2 et exactement l'un de a_1, a_3

Dans chacun des cas u voit a_2 et a_{r-2}

Voilà notre théorème principal :

Théorème 6.1

Soit G un graphe dans $\mathcal{C}$, l'algorithme LexBFS* appliqué à G produit un ordre d'élimination P_3-simplicial.

6.1.2 L'algorithme OrdreCosine*

Premierement, l'algorithme Cosine* est un cas particulier de l'algorithme Cosine, la différence est que les sommets du graphe en entrée de Cosine* sont ordonnés selon un ordre σ. Les cas d'égalité de Cosine sont départagés en utilisant cet ordre.
Alors l'algorithme OrdreCosine* se produit en combinant les idées des algorithmes OrdreCosine et OrdreContraction.
Pour prouver que l'algorithme OrdreCosine* transforme un ordre d'élimination $\bar{P}_3$-simplicial d'un graphe G de la classe $\mathcal{B}$ en un ordre de contraction amical, on doit ajouter la notion de quasi-$\mathcal{B}$ et un lemme que nous allons citer par la suite

Définition 6.2

Un graphe G est quasi-$\mathcal{B}$ si G ne contient ni trou impair, ni antitrou et G possède un sommet appelé pivot, qui est l'oreille de tous les taureaux de G

Algorithme 27 Algorithme OrdreCosine*

ENTRÉE : Un graphe G et un ordre σ sur ses sommets
SORTIE : Un ordrede contraction $\{x_1^1, ..., x_1^{k_1}, ..., x_l^1, ..., x_l^{k_l}$ et la coloration associée
CALCUL :

1. Soient c=1 et i=1 ;
2. Tant qu'il existe des sommets non coloriés :
   2.1 S'il existe un sommet non colorié qui n'a pas de voisin colorié c ;
      2.1.1 Soit A l'ensemble des sommets non coloriés qui ont un voisin colorié c ;
      2.1.2 Choisir un sommet non colorié v qui n'a pas de voisin colorié c sont départagés en prenant le sommet minimum pour σ
      2.1.3 Colorier v avec la couleur c, poser $x_c^i = v$ et i=i+1
   2.2 Sinon, poser c=c+1 et i=1

COMPLEXITÉ : $\mathcal{O}(np)$.

Lemme 6.2

Dans un graphe G quasi-$\mathcal{B}$, soit $P = a_0 - a_1 - ... - a_r$ un chemin sans corde avec $r \geq 5$, où a_0 est un pivot de G et u un sommet qui voit les deux extrémités a_0, a_r de P, alors u voit a_2

Voilà notre théorème principal :

Théorème 6.2

Soit G un graphe dans $\mathcal{B}$ et σ un ordre d'élimination $\bar{P}_3$-simplicial de G. Alors l'algorithme OrdreCosine* appliqué sur cette entrée produit un ordre de contraction amical de G

Résumé

On peut donc colorier optimalement un graphe G sans taureau, en appliquant l'algorithme LexBFS* sur $\bar{G}$, pour obtenir un ordre $\bar{P}_3$-simplicial, suivi de l'algorithme OrdreCosine* sur G pour obtenir un ordre de contraction amical. Pour obtenir un algorithme de coloration robuste et plus rapide, il suffit d'ajouter à ces deux algorithmes l'algorithme Clique ci-dessous et de vérifier que l'ensemble trouvé est bien une clique.

Algorithme 28 Algorithme Clique

ENTRÉE : Un graphe G et une coloration de ses sommets utilisant l couleurs
SORTIE : Un ensemble Q composé de l sommets de G
CALCUL :
— Soit $Q = \emptyset$
— Pour tout sommet x, soit q(x)=0 ;
— Pour c=l,...,1 :
  — Choisir un sommet x de couleur c qui maximise q(x)
  — Poser $Q = Q \cup \{x\}$;
  — Pour tout voisin y de x, poser q(y)=q(y)+1 ;
COMPLEXITÉ : $\mathcal{O}(n + p)$.

6.2 Graphes de Meyniel

Définition 6.3

On rappelle qu'un graphe est de Meyniel, si tout cycle impair de longueur supérieure ou égale à cinq contient au moins deux cordes.

Remarque :
Les graphes de Meyniel sont des graphes parfaits.

6.2.1 Algorithme de coloration

L'algorithme LexColor 6 est l'algorithme le plus rapide qui peut donner une coloration optimale pour un graphe de Meyniel, de plus si on veut une coloraion et une clique de même taille on peut utiliser l'algorithme Clique

6.2.2 Obstruction

Définition 6.4

Une maison est un cycle impair de longueur au moins cinq avec une seule corde qui est courte. Dans une maison, le sommet qui forme un triangle avec l'unique corde est appelé le toit de la maison

Définition 6.5

Une obstruction de Meyniel est un trou impair ou une maison

Remarque :
Il est facile de voir qu'un graphe est de Meyniel si et seulement s'il ne contient pas une obstruction de Meyniel.

Théorème 6.3

Pour tout graphe de G, soit G contient une clique et une coloration de même taille soit G contient une obstruction de Meyniel

Pour trouver une obstruction de Meyniel dans un graphe quelconque, on va appliquer le théorème précédent.

Méthode de recherche

Soit l le nombre totale de couleurs utilisées par LexColor pour colorier un graphe quelconque.Ensuite, en appliquant l'algorithme Clique sur G, à chaque étape on vérifit que le sommet x de couleur c qui est séléctionné satisfait q(x)=l-c. Si celà est vérifié au cours de toute l'éxécution de l'algorithme, alors l'ensemble Q final est une clique de taille l et la coloration est optimale, Sinon d'après le théorème précédent, l'existence d'une obstruction de Meyniel est garantie. Pour plus de détails voir [2] page :77. On peut par la suite conclure que le graphe n'est pas de Meyniel.

6.3 Graphes d'Artémis

Définition 6.6

Un graphe d'Artémis est un graphe qui ne contient ni trou impair, ni antitrou, ni prisme

Remarque :
Les graphes d'Artémis sont des graphes parfaits car ils sont parfaitement contractibles

Définition 6.7

Une paire d'amis est dite *spéciale* si le graphe G/ab ne contient pas de prisme

Lemme 6.3

Si G est un graphe d'Artémis et a,b est une paire d'amis spéciale, alors G/ab est un graphe d'Artémis

Définition 6.8

- Un sommet de $S \setminus X$ est dit X-complet s'il voit tous les sommets de X. C(X) l'ensemble des sommets X-complets de $S \setminus X$
- Un ensemble non vide $T \subseteq S$ est dit *intéressant* si G[T] est co-connexe et G[C(T)] n'est pas une clique ($|C(T)| \geq 2$)
- Un ensemble intéressant est maximal s'il n'est pas strictement inclus dans un autre ensemble intéressant
- Un chemin T-sortant est un chemin sans corde dont les extrémités sont dans C(T) et dont les sommets intérieurs sont tous dans $S \setminus (T \cup C(T))$
- Un chemin T-sortant P est minimal s'il n'y a pas de chemin T-sortant dont les sommets intérieurs sont strictements contenus dans les sommets intérieurs de P

Lemme 6.4

Pour tout graphe G, les conditions suivantes sot équivalentes :
- G n'a pas d'ensemble intéressant
- Tout sommet de G est simplicial
- G est une union disjointe de cliques

De plus, si G n'est pas une union disjointe de cliques, alors tout sommet non-simplicial forme un ensemble intéressant

Lemme 6.5

Soit G un graphe d'Artémis qui contient un ensemble intéressant et soit T un ensemble intéressant maximal de G. Si T n'a pas de chemin T-sortant, alors toute paire d'amis spéciale du sous-graphe G[C(T)] est une paire d'amis spéciale de G

Lemme 6.6

Soient T un ensemble intéressant maximal dans le graphe G et a,b deux sommets son-adjacents de C(T). Soit C'(T) l'ensemble des sommets T-complets dans G/ab.Si C'(T) n'est pas une clique, alors T est un ensemble intéressant maximal de G/ab

6.3.1 Algorithme d'ensemble intéressant maximal

Algorithme 29 Algorithme Intéressant

ENTRÉE : Un graphe G
SORTIE : Un ensemble intéressant maximal T de G ou la réponse "G est une union disjointe de cliques"
CALCUL :

Etape 1 : Chercher un sommet non-simplicial t
- Calculer les composantes connexesde G ;
- Si tout sommet est de degré égal à la taille de sa composante moins 1, retourner "G est une union disjointe de cliques".
- Sinon, soit u un sommet dont le degré est strictement plus petit que la taille de sa composante moins 1.Exécutons un parcours en largeur à partir de u, soit v un sommet à distance 2 de u, et soit t le père de v dans le parcours.

Etape 2 : Construction de T à partir de t
- Soient T=t, C=N(t), $U = S(G) \setminus (T \cup C)$, $Z = \emptyset$;
- Tant qu'il existe $u \in U$:
  - Si $N(u) \cap C$ est une clique, déplacer u de U vers Z
  - Si $N(u) \cap C$ n'est pas une clique, déplacer u de U vers T et déplacer tout sommet de $C \setminus N(u)$ de C vers U
- Retourner l'ensemble T

COMPLEXITÉ : $\mathcal{O}(max(n+p, p(n-k)))$où k est le nombre de sommets de C(T) de l'ensemble T retourné (si aucun ensemble T n'est retourné, nous considérons k=n et donc la compléxité est $\mathcal{O}(max(n+p))$

6.3.2 Algorithme du Chemin sortant

Après avoir un ensemble intéressant maximal, cherchons maintenant un chemin sortant minimal.

Algorithme 30 Algorithme Chemin Sortant

ENTRÉE : Un graphe G et ensemble intéressant maximal T de G
SORTIE : Un chemin T-sortant minimal ou la réponse "G n'a pas de chemin T-sortant"

CALCUL :
- Tout les sommets de $S(G) \setminus (T \cup C(T))$ sont non-marqués ;
- Tant qu'il existe un sommet non-marqué r dans $S(G) \setminus (T \cup C(T))$:
  - Effectuer un parcours en largeur de r vers C(T) dans G, en notant V les sommets rencontres, M l'ensemble $V \cap C(T)$ et en arrêtant le parcours dès que M n'est plus une clique ; soit F les sommets qui sont encore dans la file lorsque le parcours est arrêté
  - Si M n'est pas une clique :
    - Soit x le dernier sommet ajouté à M et $M_x = M \cap N(x)$
    - Effectuer un parcours en largeur à partir de x dans le sous-graphe $G[V \setminus (F \cup M_x)]$
    - Soit y le premier sommet de $M \setminus M_x$ qui est atteint par ce parcours
    - Retourner le chemin de ce parcours qui va de x à y
  - Sinon, marquer les sommets de S ;
- Retourner "G n'a pas de chemin T-sortant"

COMPLEXITÉ : $\mathcal{O}(lp)$ où l est le nombre de composante de $G \setminus (T \cup C(T))$

6.3.3 Algorithme de la paire d'amis spéciale

Finalement, voici l'algorithme qui nous aide à chercher une paire d'amis spéciale

Algorithme 31 Algorithme Paire d'amis spéciale

ENTRÉE : Un graphe G, un ensemble intéressant maximal T et un chemin T-sortant minimal$x - z_1 - ... - z_p - y$
SORTIE : Une paire d'amis spéciale de G
CALCUL :

1. Soient $A = (N(z_1) \cap C(T)) \setminus N(y)$ et $B = (N(z_p) \cap C(T)) \setminus N(x)$
2. Effectuer un parcours en largeur de B vers N(A) dans $G \setminus (T \cup A)$ et soit K les sommets de N(A) qui sont atteints par ce parcours
3. Effectuer un parcours en largeur de A vers des sommets de N(B) qui sont atteints par ce parcours
4. Soit a un sommet de A qui voit tout K
5. Soit b un sommet de B qui voit tout L ;
6. Retourner la paire a,b

COMPLEXITÉ : $\mathcal{O}(n + p)$

Méthode de coloration

La méthode fonctionne comme suit :
Premièrement, un algorithme trouve un ensemble intéressant maximal T dans G . Puis, un deuxième algorithme trouve une paire d'amis spéciale dans C(T), en se basant sur le lemme 6.5 et la contracte. Ce deuxième algorithme est itéré tant que l'ensemble C(T) n'est pas une clique, ce qui est impossible, d'après les lemmes 6.3 et 6.6. Lorsque l'ensemble C(T) devient une clique, le premier algorithme est appelé à nouveau pour trouver un nouvel ensemble intéressant maximal. On répète tout ça jusqu'à trouver une réunion disjointe de cliques, la contraction de cette réunion va nous donner une seule clique et donc la coloration est évidente.

Conclusion

Après toutes les recherches et les études qui sont faites pour la classe des graphes parfaits dés 1960, et vu qu'elle est une classe de graphes importantes, en particulier parce qu'elle contient de nombreuses autre classes usuelles, il reste toujours des lacunes et des questions à poser à propos de cette classe.

La reconnaissance des graphes parfaits comme on a vu se fait juste dans les mesure de vérification du théorème fort des graphes parfait mais le problème reste ouvert quand on parle de la reconnaissance des graphes sans trou impair.

Une deuxième question concerne l'existence d'un algorithme en temps polynomial qui décide si un graphe possède une partition antisymétrique paire ? Un autre axe, qui semble a priori plus intéressant, serait d'obtenir un algorithme de construction explicite des graphes de Berge.
Dans le même axe, on sait qu'il existe un algorithme de coloriage polynômial, en utilisant la méthode des ellipsoïdes : mais existe-t-il un algorithme combinatoire, plus proche de la structure des graphes parfaits ? Donc, il faut bien étudier les propriétés structurelles de ces sous-classes et modifier les algorithmes de coloration associés afin d'obtenir de nouveaux algorithmes qui vont nous aider à colorier les autres sous-classes des graphes parfaits

Bibliographie

[1] Nicolas Trotignon,"Graphes parfaits :Structures et algorithmes", Thèse. 2004.

[2] Benjamin Lévêque. Coloration de graphes : structures et algorithmes. Mathématiques [math]. Université Joseph-Fourier - Grenoble I, 2007.

[3] Gérard Cornuéjols, Le théorème fort des graphes parfaits, 2006.

[4] V. Chvátal. Star-cutsets and perfect graphs. J. Combin. Ser. B,39 :189–199, 1985.

[5] Claude Berge, "Graphes et hypergraphes"

[6] Olivier Fouquet,"Théorie des graphes : une brève introduction",Université Paris XI,2013

[7] M. Chudnovsky and P. Seymour. Recognizing Berge graphs. Manuscript, 2002.

[8] G. Cornuéjols, X. Liu, and K. Vušković. A polynomial algorithm for recognizing perfect graphs. Manuscript, 2002.