N° d'ordre : 05/2010-M/MT

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté des Mathématiques

MEMOIRE
Présenté pour l'obtention du diplôme de MAGISTER
EN : MATHEMATIQUES
Spécialité : Recherche Opérationnelle (Génie Mathématiques)

Par : **HABLAL Houria**

THÈME

Sur les graphes k-Super-Triangulés

Soutenu publiquement le 20/10/2010 devant le jury composé de :

M.	M. MOULAI	Professeur à l'U.S.T.H.B	Président
M.	H. AIT HADDADENE	Professeur à l'U.S.T.H.B	Directeur de mémoire
M.	M. AIDER	Professeur à l'U.S.T.H.B	Examinatreur

Table des matières

Table des matières 2

1 Concepts fondamentaux de la théorie des graphes 8
1.1 Notions de base 9
1.1.1 Voisinage ouvert et voisinage clos 10
1.1.2 Sous-graphe et graphe partiel 10
1.1.3 Chaîne, cycle, trou et anti-trou 12
1.1.4 Connexité des graphes 13
1.2 Coloration des graphes 13
1.3 Graphes parfaits 15
1.4 Complexité algorithmique 17
1.4.1 Les problèmes P, NP, CoNP et NP-complets 18
1.5 Quelques classes de graphes 20
1.5.1 Graphes d'intervalles 20
1.5.2 Graphes de comparabilité 22
1.5.3 Graphes de permutation 23

2 Les graphes triangulés 25
2.1 Les graphes triangulés 25
2.2 Quelques caractéristiques des graphes triangulés 27
2.3 Le parcours en largeur lexicographique LexBFS 33
2.3.1 Algorithme LexBFS (parcours en largeur lexicographique) 33
2.3.2 Les propriétés du parcours LexBFS 33
2.4 La reconnaissance 35
2.5 La coloration 39

3 Les graphes faiblement triangulés 41
3.1 Les graphes faiblement triangulés 41
3.2 La relation entre les graphes triangulés et les graphes faiblement triangulés 44
3.3 La reconnaissance 47
3.4 La coloration 50

4 Les graphes k-Super-Triangulés 51
4.1 Les graphes (k, l)-chordal et k-chordal 52
4.1.1 Les graphes k-chordal 53
4.2 Les graphes k-Super-Triangulés 58
4.3 Reconnaissance des graphes STR_k 59

4.3.1 Algorithme de reconnaissance 59

5 Partie programmation **61**
5.1 Le langage de programmation Matlab 61
5.2 Algorithme de reconnaissance des graphes triangulés 62
5.3 Algorithme de reconnaissance des graphes faiblement triangulés 63
5.4 Algorithme de reconnaissance des graphes STR_k 63
5.4.1 Implémentation de l'algorithme de reconnaissance des graphes STR_k 64
5.4.2 Quelques simulations 65

Conclusion **67**

Bibliographie **67**

♣REMERCIEMENTS♣

Je remercie en premier lieu Dieu le tout puissant de m'avoir accordé la puissance et la volonté pour terminer ce travail.

Je tiens à exprimer ma reconnaissance à M. H. AIT HADDADENE , professeur à l'USTHB, pour avoir dirigé ce mémoire, pour son suivi permanent, ses lectures attentives, sa disponibilité, ses conseils judicieux et le soutien constant qu'il m'a prodigué au cours de l'élaboration de ce travail.

Je remercie vivement le Président de jury M M. MOULAI professeur à l'USTHB, pour l'honneur qu'il me fait en acceptant de présider le jury de mon mémoire.

J'adresse mes respectueux remerciements à M. M. AIDER professeur à l'USTHB, pour le temps et l'effort qu'il a bien voulu consacrer à la lecture et la correction de ce mémoire.

Mes remerciements vont aussi à mes enseignants de première année PG et aux organisateurs de notre promotion.

Enfin je n'oublie pas de remercier mes parents pour leur amour et leur soutien tout au long de mes études, tous les membres de ma famille et tous ceux qui, de près ou de loin ont apporté un plus à ce travail (en particulier Yamou et Lila).

♡DEDICACES♡

Je dédie ce travail à mes parents qui m'ont toujours entouré de leur amour et soutenu le long de ma vie.
Un grand merci à vous,
Pour vos mains qui ont tant travaillées,
Pour votre cœur qui m'a tant donné
Pour votre sourire qui m'a tant réchauffé,
Pour vos yeux qui furent parfois mouillés,
Pour vous qui m'avez tant aimé.

Je le dédie à ma famille que j'adore ; ma sœur Karima et mes frères Ali, Mouhamed, Hamid sa femme Aldja et leur adorable fils Axel, Kamel et Achour. Un grand merci pour le support moral et financier et pour leur amour.

A mes oncles Ahcene et Aissa, leurs femmes et leurs enfants.

A la mémoire de mon cousin Cherif qui nous a quitté si tôt, nous ne t'oublierons jamais.

A la mémoire de mes grands parents.

A mes tantes Nadia, Djazira et Tounsia ainsi que leurs maris et leurs enfants.

A toutes mes amies en particulier Yamina, Salima, Lila, Nacera et Lila, Fatima et Sassia.

A ma promotion.

A tous ceux qui aiment Dieu.

$\mathcal{R}$ésumé

Dans cette thèse, nous avons tracé deux objectifs :
Le premier est de réaliser une synthèse sur deux classes de graphes : les graphes triangulés et les graphes faiblement triangulés. Nous nous sommes intéressés en particulier aux algorithmes de reconnaissance et de coloration de ces deux classes. Nous avons aussi donné les caractéristiques les plus importantes de ces graphes.

Le second est d'introduire et d'étudier les graphes k-super-triangulés (qu'on a noté les graphes STR_k), k étant un nombre fixe supérieur strictement à 2. Les graphes STR_k sont les graphes n'ayant ni de cycle ni d'anti-cycle sans corde de longueur supérieure à k. Ainsi les graphes STR_3 sont les graphes n'ayant pas de cycle ni d'anti-cycle sans corde de longueur supérieure à 3, ils sont donc inclus dans la classe des graphes triangulés. Les graphes faiblement triangulés sont exactement les graphes STR_4. De plus la classe des graphes STR_5 contient strictement celle des graphes faiblement triangulés. Hayward a donné un algorithme pour la reconnaissance des graphes "k-chordal" (les graphes n'ayant pas de cycles sans corde de longueur supérieure à k). Se basant sur cet algorithme nous avons proposé un algorithme de reconnaissance des graphes k-super-triangulés. Le principe de notre algorithme est simple, on applique l'algorithme de Hayward sur le graphe G sur lequel on fait notre test, puis on l'applique sur le graphe complémentaire de G. Ainsi, on peut tester si un graphe est STR_k en temps $O(n^{k+1})$ qui est la complexité de l'algorithme de Hayward pour la reconnaissance des graphes "k-chordal". On peut utiliser aussi l'algorithme de Spinrad pour reconnaitre les graphes STR_k en $O(n^{k-2}T)$. En dernier lieu, nous avons implémenté trois algorithmes de reconnaissance l'un des graphes triangulés, le second des graphes faiblement triangulés et le dernier des graphes STR_k.

Introduction Générale

La théorie des graphes est un domaine très vaste. On l'utilise souvent comme outil de modélisation des problèmes pratiques. En effet, la résolution d'un problème modélisé par un graphe revient en général à chercher certaines caractéristique du graphe ou à le colorer ou bien encore à tester son appartenance à l'une des classes de graphes connues. Ce test qu'on effectue pour décider de l'appartenance ou non d'un graphe à une classe donné de graphe est connu sous le nom de la «reconnaissance des graphes». Il existe plusieurs classes de graphes dans la littérature. Leur importance réside dans leur utilité dans la pratiques et les diverses situations qui peuvent être modélisées par des graphes appartenant à ces classes.

Les graphes faiblement triangulés sont les graphes ne contenant pas de cycle sans corde de longueur supérieure ou égale à 4. Ces graphes trouvent plusieurs applications dans divers domaines comme la biologie, l'informatique (allocation de registres), et autres domaines, de plus les quatre problèmes d'optimisation son résolus pour cette classe. Depuis leur introduction, plusieurs études ont été effectuées sur les graphes triangulés et leurs propriétés ce qui a donné un grand nombre d'articles consacrés à leur étude. La nécessité d'établir un Survey sur ces graphes est donc évidente.

Une autre classe de graphes qu'on rencontre dans la théorie des graphes est la classe des graphes faiblement triangulés qui sont une généralisation des graphes triangulés (on autorise les cycles sans corde de longueur 4). Il est presque impossible de parler des graphes triangulés sans citer les graphes faiblement triangulés. Ceci est dû aux similitudes existantes entre les deux classes. En effet les graphes triangulés sont des graphes faiblement triangulés, de plus l'arête dans un graphe faiblement triangulé joue le rôle d'un sommet dans un graphe triangulé. Tout comme les graphes triangulés les quatre problèmes sont résolus en temps polynomial pour les graphes faiblement triangulés. Nous avons jugés donc important de consacrer une partie de cette thèse aux graphes triangulés. On y trouve plusieurs caractéristiques et algorithmes de coloration et de reconnaissance de cette classe.

Les graphes k-chordal sont les graphes n'ayant pas de cycles sans corde de longueur supérieure à k, c'est donc une classe plus vaste et plus générale que celle des graphes triangulés et faiblement triangulés. Notre étude concernera les graphes STR_k qui sont des graphes n'ayant ni de cycle ni d'anticycle de longueur supérieure à k. Nous nous intéressons en particulier au problème de reconnaissance de cette classe de graphes.

Chapitre 1 : Ce chapitre est consacré aux notions de base de la théorie des graphes. On trouve dans ce dernier la notion de graphes et les différentes façons de le représenter, la notion de voisinage les concepts de chaine, cycle, trou et antitrou, les notions de graphe partiel et sous graphe, la connexité des graphes ainsi que quelques sous classes des graphes

triangulés.

Chapitre 2 : Ce chapitre est dédié aux graphes triangulés. Dans ce chapitre, on donne les caractéristiques des graphes triangulés, leurs algorithmes de reconnaissance et de coloration. On trouve aussi le parcours en largeur lexicographique LexBFS qui est un outil de reconnaissance des graphes chordal. On a consacré une section à ce parcours où on donne l'interet et le fruit du parcours pour les différentes classes de graphes.

Chapitre 3 : Dans le troisième chapitre, on parle des graphes faiblement triangulés. Un aperçu sera donné sur les propriétés les plus importantes de cette classe de graphe comme celle de la deux-paire. On donnera aussi les différents outils et notions utilisés pour la reconnaissance et la coloration de cette classe de graphes. Pour finir on donne la fameuse relation entre les graphes triangulés et les graphes faiblement triangulés.

Chapitre 4 : Le quatrième chapitre est celui des graphes k-super-triangulés. On commence ce chapitre par la définition des graphes (k, l)-chordal et graphes k-chordal et quelques sous-classes de ces graphes. On enchaine par l'introduction des graphes k-super-triangulé qu'on notera STR_k. On s'intéresse en particulier aux algorithmes de reconnaissance des graphes STR_k.

Chapitre 5 : Le dernier chapitre de la thèse est consacré à la partie programmation. En effet, on a choisi quelques algorithmes cités dans les chapitres précédents et nous les avons programmé. On a aussi programmé l'algorithme de reconnaissance des graphes STR_k. L'implémentation des algorithmes a été faite en langage de programmation Matlab.

Chapitre 1

Concepts fondamentaux de la théorie des graphes

Sommaire

1.1 Notions de base 9
1.2 Coloration des graphes 13
1.3 Graphes parfaits 15
1.4 Complexité algorithmique 17
1.5 Quelques classes de graphes 20

Introduction

La théorie des graphes est un outil puissant de modélisation des problèmes mathématiques ou ceux d'optimisation. La modélisation est souvent la première étape de résolution des problèmes d'optimisation. Plusieurs problèmes peuvent être modélisés par des graphes, d'où l'intérêt de la théorie des graphes. Ce chapitre est dédié aux notions de base de la théorie des graphes. Le lecteur peut trouver dans ce qui suit les notions les plus utilisées dans la théorie des graphes, ou du moins celle dont on aura besoin dans la suite de la thèse.

Le chapitre commence par l'introduction de quelques définitions des concepts élémentaires de la théorie des graphes comme celles d'un graphe, matrice d'adjacence, matrice d'incidence. On trouve ensuite les notions de voisinage ouvert et voisinage clos. Ce qui est suivi par les concepts de sous-graphe, graphe partiel, chaine, cycle, trou et anti-trou qu'on trouve dans les chapitres qui suivent. On donne aussi quelques notions sur les composantes connexes, la connexité d'un graphe, ainsi que la complexité algorithmique. Pour finir par la coloration des graphes, les graphes parfaits ainsi que quelques classes de graphes qui sont des sous graphes des graphes triangulés ou faiblement triangulés qu'on abordera en détail dans les chapitres qui suivent.

1.1 Notions de base

Les notions qui vont suivre se trouvent dans la plupart des ouvrages de la théorie des graphes. On cite entre autres le livre de Didier Maquin [61] et celui de Didier Müller [64].
.Un *graphe orienté* fini G = (V, E) est défini par un ensemble V de n *sommets* ou nœuds, et un ensemble V de m *arcs* reliant ces sommets.
On notera $n = |V|$ le nombre de sommets de G et $m = |E|$ le nombre d'arcs.
.Un arc reliant le sommet x au sommet y est noté (x, y) ou $\{x, y\}$, xy désignera aussi l'arc $\{x, y\}$. On dit que l'arc xy est *incident* aux sommets x et y
.Deux sommets reliés par un arc sont dits *adjacents* ou *voisins*.
.Une *boucle* est un arc partant d'un sommet et allant vers lui-même.
.Un graphe est dit *simple*, s'il ne contient pas de boucles et s'il n'y a pas plus d'une arête reliant deux mêmes sommets.
.Si deux sommets peuvent être reliés par plusieurs arcs, on parle de *multigraphe*.
.Un graphe ou graphe *non orienté* non orient´e G = (V,E) est la donnée de deux ensembles, un ensemble fini de sommets V et un ensemble fini d'arêtes E. Une arête $e \in E$ est une paire de sommets (u, v), notée e = uv ou bien e = vu, où u et v sont les extrémités de e. On dira dans ce cas que x et y sont adjacents et que e est incidente à x et à y. .On appelle *ordre* d'un graphe le nombre de sommets (n) de ce graphe.
.La *taille* d'un graphe est le nombre de ses arcs (arêtes pour graphe non orienté) (m).
.Un graphe est *complet* si tous ses sommets sont adjacents deux à deux.
.Un graphe complet à n sommets est noté K_n.
.Le *complémentaire* d'un graphe non orienté G, noté $\bar{G}$, est le graphe dont l'ensemble des sommets est V et l'ensemble d'arêtes $\{(x, y) \in V^2 / x \neq y\} \setminus E$ (si deux sommets distincts x et y sont adjacents dans G, ils ne le seront pas dans $\bar{G}$ et inversement).
Soit A un ensemble de sommets de G. On note par $G \backslash A$ ou (G-A) le graphe obtenu en supprimant les sommets de A du graphe G ainsi que les arêtes qui lui sont incidentes.
Le *degré* d'un sommet $v \in V$, noté $d_G(v)$, est le nombre d'arêtes incidentes à v.
Un sommet de degré nul est dit *isolé*.
Un graphe n'ayant que des sommets de degré d est appelé graphe *d-régulier*.
$\triangle(G)$ et $\delta(G)$ représenteent resp. le degré maximum et minimum dans G. **Propriétés d'un graphe**

Un graphe G= (V, E) peut posséder une ou plusieurs des propriétés suivantes :
Symétrie : Un graphe est dit symétrique si et seulement si :
si $uv \in V$, alors $vu \in V$
Antisymétrie : Un graphe est dit antisymétrique si et seulement si :
si $uv \in V$, alors $vu \notin V$.
Transitivité : Un graphe est dit transitif si et seulement si :
si $uv \in V$ et $vw \in V$, alors $wu \in V$

Les représentations d'un graphe :
Un graphe peut être représenté de plusieurs manières dont la matrice d'adjacence, la matrice d'incidence, la représentation sagittale...etc.
On se contentera ici de définir la matrice d'adjacence et la matrice d'incidence qu'on utilise le plus souvent comme outil de représentation d'un graphe.

La matrice d'adjacences (Adjacency Matrix) :
La matrice d'adjacence est une matrice carrée contenant des 0 et des 1, dont les lignes et les colonnes sont classées par sommets.
- Un 1 en position (i, j) signifie qu'il y a une arête (ou arc) du sommet i au sommet j.
- Un 0 indique qu'il n'y a aucune arête ou arc.

La matrice d'incidences (Incidence Matrix) :
La matrice d'incidences est une matrice contenant des 0 et des 1 et -1 dont les lignes sont indexées par les sommets du graphe et dont les colonnes sont indexées par les arêtes.
- Un 1 à la position (i, j) de la matrice signifie que le sommet i est l'extrémité initiale de l'arête j.
- Un -1 signifie que le sommet i est l'extrémité terminale de l'arête j.
- Un 0 indique que l'arc j n'est pas incident au sommet i.

1.1.1 Voisinage ouvert et voisinage clos

Vue leur importance en particulier dans la reconnaissance des graphes triangulés, on donne les définitions des voisinages clos et ouverts.

Définition 1.1.
Le voisinage ou voisinage ouvert d'un sommet v est l'ensemble de ses sommets adjacents, il est noté N(v).

Définition 1.2.
Le voisinage clos d'un sommet v est défini par $N[v] = N(v) \cup v$. C'est donc le voisinage ouvert de v auquel on ajoute le sommet lui même.

Deux sommets v_1 et v_2 sont dits *vrais jumeaux* s'ils ont le même voisinage et sont adjacents, c'est-à-dire qu'ils ont le même voisinage clos $N[v_1] = N[v_2]$

Deux sommets v_1 et v_2 sont dits *faux jumeaux* s'ils ont le même voisinage mais ne sont pas adjacents, soit $N(v_1) = N(v_2)$.
On définit aussi le *non-voisinage* de v comme étant $N(v) = V \setminus N[v]$.

1.1.2 Sous-graphe et graphe partiel

Soit G = (V, E) un graphe.
Pour un sous-ensemble de sommets A inclus dans V, le *sous-graphe* de G *induit* par A est le graphe G' = (A, E(A)) dont l'ensemble des sommets est A et l'ensemble des arêtes E(A) est formé de toutes les arêtes de G ayant leurs extrémités dans A. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.

Le graphe G' = (V, E') est un *graphe partiel* de G, si E' est inclus dans E. Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G.
Un graphe partiel d'un sous-graphe de G est un sous-graphe partiel de G.
On appelle *clique* un sous-graphe complet de G.

On appelle *triangle* toute clique ayant trois sommets.
On appelle *stable* un sous-graphe de G sans arêtes.

1.1.3 Chaîne, cycle, trou et anti-trou

Une *chaîne* dans G, est une suite de la forme $(v_0, e_1, v_1, e_2, ..., v_{k-1}, e_k, v_k)$ ayant pour éléments alternativement des sommets (v_i) et des arêtes (e_i), commençant et se terminant par un sommet, et telle que les extrémités de e_i soient v_{i-1} et v_i, $i = 1, ..., k$.

On peut définir aussi une chaîne comme étant une suite d'arcs tels que chaque arc est adjacent à l'arc qui le précède dans cette suite par son extrémité initiale et à l'arc qui le suit par son extrémité terminale.

Si $v_0 = a$ et $v_k = b$, on dira que la chaîne relie a et b. En plus, on dira que la chaîne a une longueur k (c'est le nombre d'arêtes de la chaîne).
Une chaîne doit comporter au moins une arête.
On appelle *distance* entre deux sommets la longueur de la plus petite chaîne les reliant.
On appelle *diamètre* d'un graphe la plus longue des distances entre deux sommets.
Une chaîne est élémentaire si chaque sommet y apparaît au plus une fois.
Une chaîne est simple si chaque arête apparaît au plus une fois.
Un *cycle* est une chaîne telle que $v_0 = v_k$.
Une *corde* est un arc reliant deux sommets non-consécutifs d'une chaine ou d'un cycle.
On note une chaine (resp cycle) sans corde de longueur k P_k (resp C_k).
Un *trou* est un cycle de longueur 5 ou plus.
Un *anti-trou* est le complémentaire d'un trou.

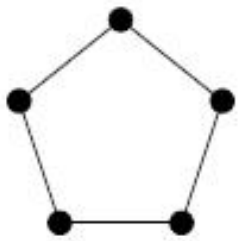

FIG. 1.1 – Un trou de longueur 5

1.1.4 Connexité des graphes

Définition 1.3. Un graphe G est *connexe* s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres sommets du graphe. Autrement dit il existe une chaîne reliant toute paire de sommets du graphe.

La connexité est une relation d'équivalence, dont les classes d'équivalences sont appelées *composantes connexes* du graphe G. Une composante connexe est donc un ensemble maximal de sommets tel qu'on peut effectuer une marche (une chaîne) d'un sommet à l'autre pour toute paire de sommets de cet ensemble, sommets qu'on dit alors connectés.

Un graphe connexe est un graphe n'ayant qu'une seule composante connexe.
Un graphe non connexe se décompose donc en composantes connexes.
Si $\overline{G}$ est connexe, on dit que G est *anti connexe*.
On appelle *composantes anti connexes* de G les complémentaires des composantes connexes de $\overline{G}$.
Il est à noter que pour tout graphe G, l'un au moins de G, $\overline{G}$ est connexe.
La *composante co-connexe* d'un graphe G est une composante connexe du graphe complémentaire de G.
On appelle *ensemble d'articulation* de G tout ensemble F de sommets tel que $G \setminus F$ n'est pas connexe.
Si pour tout ensemble F de k-1 sommets de G, le graphe $G \setminus F$ est connexe, alors on dit que G est *k-connexe*.
Un arbre est un graphe non orienté où toute paire de sommets est connecté par exactement une chaine.

1.2 Coloration des graphes

On trouve dans la pratique divers problèmes qui peuvent être modélisées comme problèmes de coloration de graphes comme l'ordonnancement de tâches, l'allocation de fréquences, l'allocations de registres, de répartition de services dans les réseaux. La coloration des graphes est un moyen de résolution de problèmes.

Définition 1.4. La *coloration* d'un graphe consiste à attribuer des couleurs aux sommets de telle sorte que deux sommets adjacents quelconques ne reçoivent pas la même couleur.

Définition 1.5. Une *k-coloration* (k-coloring) d'un graphe G est une application c : $V(G) \to C$, où C est un ensemble de k couleurs (généralement, $C = 1, 2, \ldots, k$), telle que pour toute arête uv de E(G), $c(u) \neq c(v)$.

Un graphe qui admet une k-coloration est dit *k-coloriable* (k-colorable).

Définition 1.6. Si C est une k-coloration de G et i ($1 \leq i \leq k$) une couleur, alors $c^{-1}(i)$ est une classe de couleur (color-class).

Définition 1.7. Le *nombre chromatique* (chromatic number) $\chi(G)$ d'un graphe G est le plus petit entier k tel que G admet une k-coloration. Si $\chi(G) = k$, le graphe G est dit *k-chromatique* (k-chromatic).

Il est évident que toute coloration d'un graphe contenant une clique de taille k nécessite au moins k couleurs. Plus précisément :

Propriété 1.1. Pour tout graphe G, $\chi(G) \geq \omega(G)$.

où $\omega(G)$ désigne la taille maximale d'une clique dans G (clique number). De la même façon, chaque classe de coloration étant un stable, nous avons :

Propriété 1.2. Pour tout graphe G à n sommets, $\chi(G) \geq n/\alpha(G)$.

où $\alpha(G)$ désigne le nombre de stabilité de G (independence or stability number) (c'est-à-dire la taille maximale d'un ensemble stable).
Considérons un graphe G à n sommets et ordonnons ses sommets de façon quelconque : $V(G) = v_1, v_2, \ldots, v_n$. Prenons l'ensemble C = 1, 2, ... comme ensemble de couleurs. L'un des algorithmes de coloration les plus simple est l'algorithme de glouton (qu'on appelle aussi algorithme First-Fit) qui est le suivant :

- colorions les sommets dans l'ordre $v_1, v_2, \ldots, v_n$,
- pour colorier le sommet v_i, utilisons la plus petite couleur possible (en d'autres termes, affectons à v_i la plus petite couleur non encore utilisée par les voisins de v_i déjà coloriés).

Clairement, cet algorithme produit une coloration de G utilisant au maximum $\Delta(G) + 1$ couleurs ($\Delta(G)$ étant le degré maximum des sommets de G). Ainsi :

Propriété 1.3. Pour tout graphe G de degré maximum $\Delta(G)$, $\chi(G) \leq \Delta(G) + 1$.

Finalement, pour tout graphe G à n sommets, nous avons :
$\max \{\omega(G), n/\alpha(G)\} \leq \chi(G) \leq \Delta(G) + 1$

1.3 Graphes parfaits

Au début des années 1960, Claude Berge a défini la classe des graphes parfaits qui est apparue ensuite comme une classe assez générale pour laquelle le problème de coloration pouvait être résolu efficacement.

En même temps que Claude Berge définissait les graphes parfaits, il proposa la conjecture forte des graphes parfaits. Ce n'est qu'en 2002 qu'un groupe de chercheurs (Chudnovsky, Robertson, Seymour et Thomas) a démontré cette conjecture.

Définition 1.8. (Graphes parfaits)
La classe des graphes parfaits (perfect graphs) est déninie comme étant la classe des graphes G tels que pour tout sous-graphe H de G, $\omega(H) = \chi(H)$.

Les graphes parfaits, outre leur utilité indéniable dans le monde de l'algorithmique, sont connus pour les 3 théorèmes suivants, qui furent pendant longtemps des conjectures.

Théorème 1.1. *(La Conjecture faible des graphes parfaits)[60, 59].*
Soit G un graphe parfait, alors $\overline{G}$ le complémentaire de G est également parfait.

On trouve aussi ce théorème sous une autre formulation.

Théorème 1.2. *[60, 59].*
Un graphe G est parfait, si et seulement si $\overline{G}$ est parfait.

Le théorème qui suit a été énoncé par Berge et résolu 12 ans plus tard en 1972 par Laslo Lovasz. C'est la célèbre conjecture des graphes parfaits.

Théorème 1.3. *(La Conjecture forte des graphes parfaits) [14].*
Un graphe G est parfait si et seulement si ni lui ni son complémentaire ne contiennent de cycle impair induit de longueur au moins cinq (trou de longueur impair).

Un graphe n'ayant ni de cycle ni d'anti-cycle impair induit de longueur impair est appelé graphe de Berge. Donc un graphe G est parfait si et seulement s'il est de Berge.

Théorème 1.4. *(La reconnaissance en temps polynomial des graphes parfaits).*
Il existe un algorithme terminant en temps polynomial déterminant si un graphe simple G est parfait ou non.

Il est à noter que la conjecture forte des graphes parfaits et le problème de l'existence d'un algorithme de reconnaissance ont tous deux été résolus en 2002 par Seymour et Al. Leur démonstration fait appel à la notion de graphes minimalement imparfait.
Un graphe minimalement imparfait est un graphe non parfait dont tous les sous-graphes induits sont parfaits.

Définition 1.9. Une paire d'amis est une paire de sommets telle que tous les chemins les reliant soient de longueur paire.

D'après le théorème suivant, les paires d'amis sont un bon outil pour prouver la perfection d'une classe de graphes.

Théorème 1.5. *(Meyniel [63])*
Un graphe minimalement imparfait ne possède pas de paire d'amis.

Néanmoins, il existe d'autres caractérisations des graphes minimalement imparfaits comme :

Théorème 1.6. *(Chvàtal [15]*
Un graphe minimalement imparfait n'a pas d'étoile d'articulation.

Théorème 1.7. *(Tucker [79]*
Un graphe minimalement imparfait ne contient pas de stable d'articulation.

Théorème 1.8. *(Lovasz [60]*
Un graphe minimalement imparfait n'a pas de jumeaux.

Théorème 1.9. *(Olariu [67]*
Un graphe minimalement imparfait ne contient pas d'anti-jumeaux.

1.4 Complexité algorithmique

Cette section est dédiée à la complexité algorithme qui représente un outil puissant pour la comparaison entre les différents algorithmes.
La notion d'algorithme n'est pas facile à définir, mais on peut le définir comme suit :

Un algorithme est un objet mathématique formé d'instructions qui sont souvent écrites sous un langage qu'on appelle langage de programmation. Ces instructions commence souvent par un begin ou début et se termine par end ou fin.

Avant de définir la complexité d'un algorithme, on doit définir c'est quoi un problème pour lequel est conçu l'algorithme pour mieux comprendre la notion de complexité.

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

Un problème de décision est un problème dont la réponse ne peut être que "oui" ou "non".
Dans un langage de programmation, un algorithme est une suite finie d'instructions ou d'opérations élémentaires.
La taille d'une instance est le nombre de symboles nécessaires à sa représentation.

Ce qui diffère d'un algorithme à un autre est le nombre d'opérations élémentaires nécessaires à l'exécution de l'algorithme. Ce qu'on appelle la complexité de l'algorithme. Utilisant la notation de Landau, on dira par exemple qu'un algorithme est de complexité $0(n^5)$ s'il existe une constante c telle que le nombre d'opérations pour exécuter l'algorithme sur une instance de taille n est inférieur à $c \times n^5$.

On dit qu'un algorithme est efficace s'il est en temps polynomial, c'est-à-dire si sa complexité est majorée par un polynôme dans la taille de l'instance. Il semble être que c'est Cobham [17] qui est le premier à avoir remarqué que beaucoup d'algorithmes classiques s'exécutent en temps polynomial, indépendamment de tout choix raisonnable de représentation des données. Mais c'est Jack Edmonds [24] qui l'a utilisé comme critère de classification.

Les problèmes indécidables sont les problèmes pour lesquels il n'existe pas d'algorithme. Remarquons que si l'entrée d'un algorithme est un graphe alors la taille de l'instance est égale à la somme du nombre de sommets du graphe et du nombre d'arêtes : n+m. Il est donc impossible d'espérer une complexité meilleure que O(n+m) pour un problème non-trivial. Un algorithme ayant une complexité O(n+m) est dit linéaire ou parfois optimal.

Sauf mention contraire, un graphe G = (V, E) en entrée d'un algorithme est représenté à l'aide de listes d'adjacence, soit pour chaque sommet v ∈ V la liste des voisins de v ; une telle représentation requiert un espace de stockage de taille O(n + m). Nous dirons qu'un problème de décision ou d'optimisation prenant un graphe en entrée peut être résolu en temps et espace linéaire s'il existe un algorithme permettant de résoudre ce problème en temps et espace O(n +m) sur le modèle RAM ; un algorithme en temps et espace linéaire est généralement le mieux que l'on puisse espérer pour traiter un tel problème.

1.4.1 Les problèmes P, NP, CoNP et NP-complets

Les algorithmes peuvent être classés dans l'une des classes (P, NP, CoNP et NP-complets) qui sont définies comme suit :

La classe P :
Un problème appartient à la classe P s'il existe un algorithme en temps polynomial pour le résoudre.

Les classes NP et coNP :
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 la réponse au problème est "oui" (resp. "non"). Un bon certificat est un certificat dont la taille est bornée par un polynôme en taille de l'instance, et qui répond "oui" (resp. "non") si et seulement l'entrée est un certificat du oui (resp. du non) pour cette instance.
On dit qu'un problème appartient à la classe (resp. coNP) s'il existe un bon certificat du oui (resp du non). Les notions de certificats ont été introduites par Jack Edmonds [44]. Il est clair qu'un problème P est toujours NP et coNP. Une conjecture célèbre affirme que la réciproque est également vraie :

Conjecture 1.1. *(P=NP ∩ coNP)*
Si un problème admet un bon certificat du oui et un bon certificat du non, alors il existe un algorithme en temps polynomial pour le résoudre.

Rappelons qu'une fonction booléenne à n variables est une fonction f de $\{0,1\}^n$ satisfait f si $f(\xi) = 1$.
Pour toute variable booléenne x sur $\{0,1\}$, on écrit $\bar{x} := 1 - x$, et on appelle x et $\bar{x}$ des littéraux. Une instance de 3-SAT est une fonction booléenne f donnée comme un produit booléen de clauses, chaque clause étant la somme booléenne de trois littéraux. La question de 3-SAT est de décider si on peut satisfaire f. Steve Cook a démontré un extraordinaire théorème :

Théorème 1.10. *(Cook, [32])*
Si l'on dispose d'un algorithme en temps polynomial pour résoudre 3-SAT, alors pour n'importe quel problème NP on peut donner un algorithme en temps polynomial.

La classe NP-complet :
On dit que le problème 3-SAT est NP-complet. Plus généralement, tout problème II de la classe NP est dit NP-complet si pour chaque problème II' de NP, il existe une réduction de Turing au problème II. C'est-à-dire que pour chaque problème II' de NP, on doit pouvoir fournir un algorithme en temps polynomial invoquant une sous-routine de résolution de II. Si un problème n'appartenant pas à la classe NP peut se réduire de la sorte, on dit qu'il est NP-difficile.

En pratique, pour montrer qu'un problème est NP-complet ou NP-difficile, il suffit de supposer qu'on dispose d'un algorithme en temps polynomial pour le résoudre, puis de

montrer qu'on peut utiliser cet algorithme pour résoudre en temps polynomial un problème qu'on sait déjà être NP-complet, comme SAT-3 par exemple.

Plusieurs problèmes de décision sont NP-complets, et le premier exemple en théorie des graphes est le problème d'existence d'une clique de taille k donné par Cook.
L'existence d'un algorithme en temps polynomial pour résoudre 3-SAT est une question fondamentale qui reste ouverte à ce jour.
L'opinion majoritaire est qu'un tel algorithme n'existe sans doute pas :

Conjecture 1.2. *(P≠NP)*
Il n'existe pas d'algorithme en temps polynomial pour résoudre 3-SAT.

1.5 Quelques classes de graphes

Pour ne pas trop charger le chapitre qui suit, on a préféré consaccrer cette section à quelques classes de graphes qu'on a jugé utile de citer vue leur relation directe avec les graphes triangulés. On done donc dans cette section un aperçu sur ces classes tout en citant parfois quelques résultats sur la coloration et la reconnaissances de ces dernières. Le lecteur trouvera dans cette section les graphes d'intervalles, les graphes d'intersection et les graphes de comparabilité.

1.5.1 Graphes d'intervalles

Définition 1.10. (Graphe d'intervalle)
Soit G = (V,E) un graphe simple. On dit que G est un graphe d'intervalle, si à chaque sommet $v \in V$ on peut associer I_v un intervalle de $\mathbb{R}$ tel que :
$\forall u; v \in V; uv \in E$ ssi $I_v \cap I_u \neq \emptyset$;

Remarque 1.1.
-Dans la représentation graphique précédente, on voit que la taille de la clique maximale est simplement le nombre d'intervalles maximum s'intersectant en un point donné.
-La classe des graphes d'intervalle vérifie la condition d'hérédité, ce qui simplifie la preuve de perfection de cette classe.
-On remarque qu'on peut choisir que tous les intervalles démarrent à une date différente.

On peut aussi définir les graphes d'intervalles comme suit :

Définition 1.11. Un graphe G= (V,E) est un graphe d'intervalles s'il est graphe d'intersection d'un ensemble de réels, c'est à dire qu'il existe un ensemble d'intervalles $I_i \subset \mathbb{R}$, $1 \leq i \leq n$,appelé réalisateur (ou réalisation) de G tel que :
$\forall i \neq j \in [1, n]$, $(v_i, v_j) \in E \Leftrightarrow I_i \cap I_j \neq \emptyset$

Théorème 1.11. *(Caractérisation des graphes d'intervalles [32]).*
Un graphe est un graphe d'intervalles si et seulement s'il admet un arbre des cliques qui est une chaîne.

Intérêt des graphes d'intervalle

Les graphes d'intervalles représentent un outil très intéressant qu'on utilise souvent pour représenter les problèmes d'ordonnancement. En effet, un problème d'ordonnancement se représente souvent comme un certain nombre de tâches, avec un temps de début, de fin, une durée et des chevauchements temporels de tâches. C'est exactement ce qu'est un graphe d'intervalles : un graphe associé à un ensemble d'intervalles qui peuvent s'intersecter. De plus, ces graphes ont l'avantage de posséder un algorithme de reconnaissance en temps linéaire.

Il existe plusieurs algorithmes de reconnaissance des graphes d'intervalles. Il est sans doute difficile de présenter touts les algorithmes de reconnaissance des graphes d'intervalles. On s'est contenté donc de donner un résumé des différents algorithmes tout en

donnant les idées principales utilisées pour les établir.

- 1962 Lekkerkerker et Boland [58] on donné un algorithme de complexité $O(n^4)$ utilisants les sommets simpliciaux et triplets astéroïdes.
- 1969 Fulkerson et Gross [27] on donné un algorithme de complexité $O(n^4)$ en utilisant les 1-consécutifs pour matrice des cliques max
- 1975 Booth, Lueker [8, 9] on établi un algorithme de complexité $O(n + m)$ avec utilisation des PQ-trees
- 1986 Korte et Mohring [54, 55] $O(n + m)$ utilisation de MPQ-trees et LexBFS.
- 1990 Ramalingam, Pandu Rangan [68] on établi un algorithme séquentiel et parallèle de complexité $O(n^2)$.
- 1991 Hsu, Ma [50, 49] on donné un algorithme de complexité $O(n + m)$ utilisation de décomposition modulaire et LexBFS
- 1992 Simon [72] on donné un algorithme de complexité $O(n + m)$ en utilisant 4 LexBFS successifs.
- 1993 Hsu [48] on donné un algorithme de complexité $O(n + m)$ qui est une variante de [50] pour la décomposition modulaire.
- 1989 Corneil, Olariu, Stewart [19] on donné un algorithme de complexité $O(n + m)$ 4 LexBFS successifs
- 2000 Habib, Paul, McConnell et Viennot [35] on donné un algorithme de complexité $O(n + m)$ utilisant LexBFS et affinage de partition sur les cliques max.

On présente entre outre l'algorithme de reconnaissance de Michel Habib, Christophe Paul et Ross M. McConnell qui se base sur l'approche par la chaîne des cliques maximales.

Algorithme (Reconnaissance des graphes d'intervalles [35]).
. Calcul à l'aide d'un parcours LexBFS d'un arbre des cliques maximales (il y en a O(m+ n)), aux arcs orientés depuis les dernières vers les premières cliques trouvées.

- Affinage de partition en utilisant les séparateurs maximaux et en commençant par la dernière clique trouvée dans l'arbre appelée C_1 : la chaîne de cliques recherchée est de la forme (C_1 (les cliques qui contiennent les éléments du séparateur maximal entre C_1 et sa clique "mère" dans l'arbre des cliques) (les autres cliques)).
- Vérification finale : chaque sommet doit se trouver dans des cliques consécutives au sein de la chaîne de cliques. Sinon, le graphe n'est pas un graphe d'intervalles.

De nombreuses classes de graphes admettent des caractérisations par un ordre sur les sommets du graphe. L'algorithme de reconnaissance est alors de la forme :

- Calculer un ordre,
- Vérifier l'ordre trouvé.

Il existe plusieurs caractéristiques des graphes d'intervalles. L'une d'elles est la suivante :

Propriété 1.4. (Caractérisation des graphes d'intervalles [66]).
G est un graphe d'intervalles ssi il existe un ordre σ sur ses sommets tel que $\forall x \preceq_\sigma y \preceq_\sigma z$, si (x, z) $\in$ E alors (x, y) $\in$ E

Théorème 1.12. *(Perfection).*
Tout graphe d'intervalle est parfait.

Théorème 1.13. *(Théorème fort des graphes parfaits - graphes d'intervalle).*
Si G = (V ;E) est un graphe d'intervalles associé à I, alors ni lui ni $\overline{G}$ ne contient de trou impair.

Calcul du nombre chromatique

Il est facile de trouver un algorithme polynomial pour le calcul du nombre chromatique d'un graphe d'intervalle.

En effet, on dispose d'un algorithme linéaire de reconnaissance d'un tel graphe, qui calcule un ensemble d'intervalles associés au graphe de départ. Il suffit ensuite de trier ces intervalles par extrémité gauche croissante. On tient ensuite un compteur qui augmente de 1 dès qu'on arrive sur un nouvel intervalle, et diminue de 1 dès qu'on sort d'un autre. Il suffit ainsi de parcourir l'ensemble des intervalles, il y en a n.
Le coût est donc O (n.log(n)) du fait du tri.

1.5.2 Graphes de comparabilité

Nous allons à présent donner la classe des graphes de comparabilité.

Définition 1.12. Un graphe G est un graphe de comparabilité s'il est possible d'orienter ses arêtes de façon que le graphe orienté (antisymétrique) ainsi obtenu soit transitif.

La définition suivante est équivalente à la précédente.

Définition 1.13. (Graphe de comparabilité)
Soit un graphe G = (V, E). On dit que G est un graphe de comparabilité s'il existe une orientation de ses arêtes telle que la relation résultante sur son ensemble des sommets est un ordre partiel P.

Autrement dit, un graphe de comparabilité est le graphe d'un ordre partiel.

Une orientation transitive des arêtes est une orietation qui satisfait la condition suivante :
S'il existe un arc allant du sommet x vers le sommet y et un arc allant du sommet y vers le sommet z, alors il existe aussi un arc allant de x vers z. Un graphe est dit de comparabilité si ses arêtes peuvent être transitivement orientées.

Déterminer une orientation des arêtes de façon à ce que celle-ci soit transitive si et seulement si le graphe est de comparabilité peut être fait en temps et espace linéaire [62]. D'un autre côté, tester si une orientation est transitive ou non se réduit au problème de la multiplication de matrices, pour lequel le meilleur algorithme connu à ce jour prend un temps $O(n^{2,376})$ [18].

Les problèmes de la clique maximum et de la coloration minimum peuvent être résolus en temps et espace linéaire, étant donnée en entrée une orientation transitive du graphe de comparabilité [33] et [62]). Les problèmes du stable maximum et de la partition minimum en cliques se réduisent au couplage maximum dans un graphe biparti [46, 1, 25]. Pour plus de détails sur ces graphes et leurs propriétés, nous renvoyons le lecteur à [33].
Le théorème qui suit montre la perfection des graphes de comparabilité.

Théorème 1.14. *([Admis] Perfection des graphes de comparabilité).*
Si G est un graphe de comparabilité, alors G est parfait.

Mentionnons enfin la classe des graphes des ordres partiels séries-parallèles. Communément appelés cographes, ceux-ci correspondent exactement aux graphes sans P_4 [71]. Le complément d'un cographe étant aussi un cographe, ceux-ci forment une sous-classe des graphes de permutation. Ils peuvent être reconnus et coloriés en temps et espace linéaire [21, 71].

1.5.3 Graphes de permutation

Une sous-classe des graphes de comparabilité connue pour ses nombreuses applications est la classe des graphes de permutation. Soit $V = 1, 2, \ldots, n$ et $\pi = [\pi[1]; \pi[2], \ldots, \pi[n]]$ une permutation de V . Le graphe $G(\pi)$ = (V,E) est défini tel que $ij \in E$ si et seulement si $(i - j)(\pi^{-1}[i] - \pi^{-1}[j]) < 0$ où $\pi - 1[i]$ dénote la position du nombre i dans π.

Définition 1.14. Un graphe G est dit de permutation s'il existe une permutation π tel que G soit isomorphe à $G(\pi)$.

Le graphe de la figure suivante est un graphe de permutation dont la permutation est $\sigma \equiv \{4, 3, 5, 1, 2\}$.
On peut aussi définir un graphe de permutation de la manière suivante.

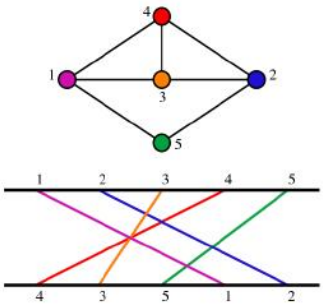

FIG. 1.2 – Un graphe de permutation

Définition 1.15. (Graphe de permutation)
On dit que G=(V ;E) est un graphe de permutation s'il existe σ une permutation de $\{1, , \mid V \mid\}$ telle que : $\forall u, v \in V; uv \in E$ ssi $(u < v)$ et $\sigma(u) > \sigma(v)$

Il existe une caractérisation très intéressante de la classe des graphes de permutation :

Lemme 1.1. *Si on dispose d'un graphe de permutation $G = (\pi, V, E)$, alors (π^{rev}, V, E) est un graphe de permutation également, et de plus $(\pi^{rev}, V, E) = \overline{G}$.*

Pour obtenir donc le complément $\overline{G}(\pi)$ d'un graphe de permutation, il suffit de renverser la séquence π. Ainsi, le complément $\overline{G}(\pi)$ est aussi un graphe de permutation.
Une autre caractérisation intéressante des graphes de permutation est :

Théorème 1.15. *(Caractérisation).*
Si G est un graphe de permutation, alors G et $\overline{G}$ sont des graphes de comparabilité. (La condition est même suffisante).

On en déduit donc immédiatement que la classe des graphes de permutation est une sous-classe des graphes parfaits. On en déduit également une preuve de la conjecture faible des graphes parfaits pour cette classe particulière. On trouve aussi une caractérisation similaire à celle des graphes d'intervalles, qu'on lance ici :

Propriété 1.5. (Caractérisation des graphes de permutation).
G est un graphe de permutation si et seulement s' il existe un ordre σ sur ses sommets tel que
$\forall x \preceq_\sigma y \preceq_\sigma z$:
- si $(x, z) \in E$ alors $(x, y) \in E$ ou $(y, z) \in E$,
- si $(x, z) \in E$ alors $(x, y) \in E$ ou $(y, z) \in E$.

La reconnaissance et la coloration

Les graphes de permutation sont reconnaissables en temps $O(n^2)$ [74] ; lorsque le test est positif, l'algorithme de reconnaissance fournit en plus la permutation π en sortie.
Les problèmes de la clique maximum et de la coloration minimum (dont une application est le tri de permutations à l'aide du minimum de files) sont solubles en temps $O(nlogn)$, étant donnée en entrée une permutation π représentant le graphe. Pour plus de détail, consulter cette [33].

Chapitre 2

Les graphes triangulés

Sommaire

2.1 Les graphes triangulés 25
2.2 Quelques caractéristiques des graphes triangulés 27
2.3 Le parcours en largeur lexicographique LexBFS 33
2.4 La reconnaissance 35
2.5 La coloration 39

Introduction

Dans ce chapitre, on va présenter la classe des graphes triangulés. Depuis son introduction, cette classe a motivé de nombreuses recherches pendant des années. C'est une classe de graphes qui a trouvé une place importante dans la pratique en particulier dans les problèmes d'allocation de registres. Dans ce chapitre on trouve la définition des graphes triangulés, quelques propriétés ainsi qu'un Survey sur les algorithmes de reconnaissance et de coloration concernant cette classe. Étant donné qu'il n'est pas possible de donner tous les algorithme de reconnaissance et de coloration des graphes triangulés, on s'est contenté de donner quelques uns et citer seulement les autres. Le lecteur pourra toutefois trouver dans ce chapitre les algorithmes de reconnaissance et de coloration les plus utilisés pour les graphes triangulés.

2.1 Les graphes triangulés

La notion de graphe triangulé a été introduite par A. Hajnal et J. Surányi en 1958 [36]. Les graphes triangulés sont aussi connus sous le nom de "chordal graphs" dans la littérature anglophone. On trouve aussi ces graphes sous le nom de graphes à élimination parfaite. Les graphes triangulés correspondent exactement aux graphes d'intersection des sous-arbres dans un arbre, ce qui leur confère certaines applications dans le domaine de la classification.

Définition 2.1. Un graphe G est dit triangulé s'il ne contient pas de cycle sans corde de longueur supérieure ou égale à 4.

Remarque 2.1. Le fait d'être triangulé est une propriété héréditaire, en effet tous les sous-graphes induits d'un graphe triangulé sont des graphes triangulés.

Avant même que C. Berge n'eut défini les graphes parfaits, A. Hajnal et J. Surányi [36] ont montré que le nombre de stabilité d'un graphe triangulé est égal à la taille minimale d'une partition en cliques. Il a donc démontré la perfection des graphes triangulée avant l'introduction de la notion de graphe parfait. D'où le corollaire suivant :

Corollaire 2.1. *Tout graphe triangulé est parfait.*

On a vu dans le chapitre précédent la classe des graphes d'intervalles. Nous rappelons la définition d'un graphe d'intervalle et donnons ensuite un théorème relatif à cette classe de graphe.

Définition 2.2. Un graphe d'intervalles est le graphe d'intersection d'intervalles sur une droite.

Remarque 2.2. Les graphes d'intervalles sont triangulés.

Cette remarque est aussi considérée comme un théorème. Pour la preuver, il suffit de démontrer qu'un graphe d'intervalle ne peut pas contenir un cycle sans corde de longueur supérieure à 3, et donc triangulé. La classe des graphes d'intervalles est donc l'une sous-classes de celle des graphes triangulés.

Une autre sous-classe intéressante des graphes triangulés est celle des graphes scindé introduits en 1977 par S. földer et P. L. Hammer. Ils sont définis comme suit :

Définition 2.3. Un graphe scindé est un graphe dont les sommets admettent une partition en deux sous-ensembles S et C, où S est un stable et C est une clique.

Un graphe scindé admettant une partition en un stable S et une clique C est noté G = (S, C, E) où E est l'ensemble des arêtes du graphe G.

Remarque 2.3. Le complément d'un graphe scindé est aussi un graphe scindé car le complémentaire d'un stable est une clique et le complémentaire d'une clique est un stable.

La relation entre les graphes scindés et les graphes triangulés peut être résumée par le théorème suivant :

Théorème 2.1. *(S. földer et P. L. Hammer)*
Un graphe G est scindé si et seulement si G et $\overline{G}$ sont triangulés [33].

Les graphes scindés sont caractérisés par le théorème suivant :

Théorème 2.2. *Soit G=(V, E) un graphe, les propriétés suivantes sont équivalentes :*

- *G est un graphe scindé.*
- *G et $\bar{G}$ sont des graphes triangulés.*
- *G ne contient pas de sous-graphe isomorphe à un K_2, C_4 ou C_5.*

Il est à noter que les graphes scindés peuvent être reconnus en temps et espace linéaire [33]. Pour plus de détails sur les graphes scindés et leurs propriétés, le lecteur peut consulter la référence précédente [33].

2.2 Quelques caractéristiques des graphes triangulés

Nous avons donné quelques caractéristiques des graphes triangulés dans la section précédente. Dans ce qui suit, on donne d'autres caractéristiques plus ou moins intéressantes des graphes triangulés.

Le séparateur minimal

L'une des caractéristiques les plus importantes des graphes triangulés est basée sur les notions de séparateur et séparateur minimal qui forment un outil puissant et essentiel pour la reconnaissance des graphes triangulés.

Définition 2.4. Un sous-ensemble W de sommets dans un graphe connexe G=(V, E) est un séparateur si G (V-W) n'est pas connexe.

W est dit ab-séparateur si a et b sont dans deux composantes connexes différentes de G (V-W).

Définition 2.5. Si un séparateur S est minimal pour l'inclusion parmi tous les séparateurs de G (aucun sous-ensemble de S n'est séparateur de G), alors on dit que S est un *séparateur minimal* de G.

Propriété 2.1. [23].
Pour tout graphe triangulé G, tout séparateur minimal de G est une clique.

Théorème 2.3. *Si G est un graphe triangulé, alors chaque ensemble d'articulation minimal est une clique.*

Triangulation d'un graphe

Définition 2.6. Triangulation
Un graphe triangulé H = (V, E U F) est dit une triangulation de G = (V, E).

Une triangulation est minimale si et seulement si pour toute arête e de F,
H'= $(V, (EUF) \setminus \{e\})$ n'est pas triangulé. Dans ce cas, F est appelé un fill-in minimal.

Propriété 2.2. La propriété de corde unique
Une triangulation H de G est minimale si et seulement si pour tout $e \in F$, e est l'unique corde d'un certain 4-cycle de H.

On entend par un k-cycle un cycle de longueur k.
La figure suivante montre deux graphes ; le premier est un fill-in minimal, le second ne l'est pas car pour les deux arêtes de F il existe deux cordes de l'unique 4-cycle de H.
Dans ce qui suit CC(S) dénotera l'ensemble des composantes connexes du graphe $G(V \setminus S)$.

Lemme 2.1. *Aucune arête d'un fill-in minimal de G ne peut joindre 2 composantes connexes dans CC(S), où S est un séparateur clique de G (un séparateur qui est en même temps une clique).*

Propriété 2.3. Propriété du séparateur minimal
Soit H une triangulation minimale de G. Alors tout ab-séparateur minimal de H est aussi un ab-séparateur minimal de G.

FIG. 2.1 – Exemples de fill-in

La notion de simplicialité

L'une des notions qu'on trouve particulièrement pour la classe des graphes triangulés est la notion de simplicialité. On peut même dire que cette dernière est propre aux graphes triangulés.

Définition 2.7. Un sommet v est simplicial si $N(v) = K_n$.

Autrement dit, un sommet simplicial de G est un sommet dont le voisinage est une clique. n ici un entier naturel et n'est pas forcement le nombre de sommets du graphe G.

Remarque 2.4. Tout sommet de degré 1 est trivialement simplicial.

Lemme 2.2. *[23].*
Tout graphe triangulé possède au moins un sommet simplicial.

Preuve 1. *Soit un graphe triangulé G = (V,E), on procède par induction sur $n = |V|$ pour prouver la propriété suivante qui est plus forte que le lemme :*
Tout graphe triangulé possède au moins un sommet simplicial, et si ce n'est pas une clique, alors il possède au moins deux sommets simpliciaux non adjacents.
L'initialisation étant triviale pour $n = 1$, supposons que tout graphe triangulé à moins de n sommets en possède un sommet simplicial, et démontrons que G ayant n sommets en a un aussi. Si G est une clique, alors tout sommet est simplicial. Si ce n'est pas le cas, alors il possède deux sommets non adjacents a et b. Soit S un $\{a, b\}$-séparateur minimal de G. $G[V \setminus S]$ a un certain nombre de composantes connexes, appelons A (respectivement B) l'ensemble des sommets de celle qui contient a (respectivement b). Montrons que A contient un sommet simplicial (on procède de même pour B).
-Soit $G[A \cup S]$ est une clique. Alors a est simplicial dans $G[A \cup S]$. Or tous les voisins de a appartiennent à $G[A \cup S]$, donc a est simplicial dans G. Soit $G[A \cup S]$ n'est pas une clique. Comme b n'appartient pas à $A \cup S$, $G[A \cup S]$ a moins de n sommets, donc on applique l'hypothèse de récurrence : $G[A \cup S]$ a deux sommets simpliciaux non adjacents qu'on appelle x et y. Trois cas se présentent alors :
-soit $x \in A$, alors tous ses voisins sont dans $A \cup S$, donc x est aussi simplicial dans G.
-soit $y \in A$, alors de même y est simplicial dans G.
-soit $x \in S$ et $y \in S$, x et y ne sont pas adjacents, mais S est un séparateur minimal d'un graphe triangulé donc ce devrait être une clique d'après la propriété précédente : contradiction ! Finalement, dans tous les cas, on a montré que A et B contiennent chacun au moins un sommet simplicial pour G, ce qui conclut la démonstration.

Lemme 2.3. *[23].*
Tout graphe triangulé non complet possède au moins deux sommets simpliciaux non adjacents.

Théorème 2.4. *Tout graphe triangulé G qui n'est pas une clique contient au moins deux sommets simpliciaux et non adjacents.*

Preuve 2. *Si G ne contient que deux sommets, alors G est constitué de deux sommets isolés qui sont simpliciaux et non adjacents. Supposons donc que le théorème vrai pour tout graphe ayant moins de n sommets et soit $|V = n|$. Soit W un séparateur minimal et $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux composantes connexes de $G[V - W]$. On a vu que W est une clique.*
- Si $G[V_1 \cup W]$ est une clique alors choisissons x dans V_1 : x est simplicial dans $G[V_1 \cup W]$.
- Sinon, par hypothèse d'induction, il existe deux sommets simpliciaux non adjacents dans $G[V_1 \cup W]$, et comme W est une clique, l'un de ces sommets qu'on appellera x est dans V_1. Dans chacun des deux cas on a déterminé un sommet x simplicial dans $G[V_1 \cup W]$. De même, on peut déterminer un sommet y simplicial dans $G[V_2 \cup W]$. Ces deux sommets x et y sont simpliciaux dans G et non-adjacents.

Ayant défini les sommets simpliciaux, on peut à présent donner la définition d'un ordre d'élimination simpliciale.

Définition 2.8. Un ordre d'élimination simpliciale de G est un ordre $[v_1, v_2, ..., v_n]$ de ses sommets tel que tout v_i, $1 \leq i \leq n-1$ est un sommet simplicial dans le sous-graphe $G[v_i, ..., v_n]$.

L'ordre d'élimination simpliciale est appelé aussi ordre d'élimination parfaite ou simplement ordre parfait.

Théorème 2.5. *Un graphe est triangulé si et seulement s'il possède un schéma d'élimination parfait.*

Le théorème précédent est l'un des outils de reconnaissance des graphes triangulés.
Étant donné qu'un graphe triangulé qui n'est pas une clique possède deux sommets simpliciaux non adjacents, et que tout sous graphe induit d'un graphe triangulé est triangulé, il est possible de choisir n'importe quel sommet de G et de décider de le mettre comme le dernier sommet de l'ordre d'élimination parfait, choisir ensuite l'un de ses voisins et le poser dans la position n-1, et ainsi de suite en choisissant les sommets dans l'ordre inverse de l'ordre d'élimination parfait.
Plusieurs algorithmes de reconnaissance des graphes triangulés sont basés sur cette idée. Le lecteur peut consulter l'article de D. J. Rose et R. E. Tarjan [70].
On dit qu'un graphe est parfaitement ordonnable s'il admet un ordre parfait.

Théorème 2.6. *Les graphes triangulés sont parfaitement ordonnables.*

Preuve 3. *Montrons que l'ordre inverse d'un schéma d'élimination parfait est un ordre parfait. Orientons donc x y si et seulement $y < x$ dans le schéma d'élimination parfait. Considérons une chaîne sans corde [a, b, c, d]. Si a b alors b c car sinon b précéderait a et c dans le schéma d'élimination parfait et on devrait donc avoir une corde reliant a à c. Par un même raisonnement, l'orientation b c implique c d. On ne peut donc pas avoir simultanément a b et d c.*

Lemme 2.4. *[27].*
Un graphe est triangulé si et seulement s'il possède un ordre d'élimination simpliciale.

Preuve 4.
⇒ *Cette implication découle de l'application récursive du lemme précédent : dans un graphe triangulé, on trouve un sommet simplicial, on le supprime, le graphe obtenu reste un graphe triangulé*
⇐ *Si un graphe n'est pas triangulé, alors il possède un cycle de plus de 3 sommets sans corde, et aucun sommet de ce cycle ne peut être éliminé "simplicialement".*

Définition 2.9. Un ensemble d'articulation relatif (à u et v) est un sous-ensemble de sommets dont la suppression déconnecte deux sommets (u eu v) fixés du graphe.

Les trois théorèmes qui suivent donnent diverses caractérisations des graphes triangulés. Vu les différences existantes entre les trois théorèmes, nous avons préféré les lancer tous malgré les répétitions.

Théorème 2.7. *Soit G= (V, E) un graphe. Chacune des assertions suivantes est une condition nécessaire et suffisante pour que le graphe G soit triangulé :*
(i) Tout ensemble d'articulation relatif minimal de G est une clique.
(ii) Tout sous-graphe induit de G contient un sommet simplicial.
(iii) Tout sous-graphe induit de G est soit une clique, soit contient deux sommets simpliciaux non adjacents.
(iv) G admet un ordre d'élimination simplicial de ses sommets.
(v) Tout sous-graphe induit connexe à k sommets contient au plus k-1 cliques maximales.

Théorème 2.8. *[23, 27, 31].*
Soit G= (V, E) un graphe. Les propriétés suivantes sont équivalentes :
(vi) G est triangulé.
(vii) Tout sous-graphe induit de G contient un sommet simplicial.
(viii) Tout sous-graphe induit de G est soit une clique, soit contient deux sommets simpliciaux non adjacents.
(ix) G admet un ordre d'élimination simplicial de ses sommets.

Théorème 2.9. *(Caractérisation des graphes triangulés)[23, 27, 31].*
Soit G un graphe. Les quatre propositions suivantes sont équivalentes :
(i) G triangulé.
(ii) G admet un schéma d'élimination simpliciale.
(iii) G admet un arbre de cliques maximales.
(iv) G est le graphe d'intersection d'une famille de sous-arbres d'un arbre.

En plus de son rôle dans la reconnaissance des graphes triangulés, le schéma d'élimination simpliciale peut servir à la recherche d'une clique max de ces derniers.

Propriété 2.4. On peut trouver une clique max d'un graphe triangulé en temps linéaire comme suit :
Parmi tous les sommets dans le schéma d'élimination simpliciale, on choisit le sommet v qui maximise $1 + |N(v)|$. Ce sommet et son voisinage constituent une clique max.

La notion d'LB-simplicialité

Lekkerkerker et J.C. Boland ont utilisé une autre notion qui est la notion de sommet LB-simplicial pour caractériser les graphes triangulés.

Définition 2.10. On dit qu'un sommet est LB-simplicial si tous les séparateurs minimaux inclus dans son voisinage sont des cliques.

Caractérisation 2.1. [58] Un graphe est triangulé si et seulement si pour tout sommet x, tous les séparateurs minimaux inclus dans N(x) sont des cliques.

Théorème 2.10. *[58] Un graphe G = (V,E) est triangulé si et seulement si tout sommet de V est LB-simplicial.*

Autres caractérisations

Il existe d'autres caractérisations des graphes triangulés.

Définition 2.11. Soit G un graphe, on définit star(G) comme étant le graphe obtenu par l'addition d'un seul sommet v à G, avec des arcs de v vers tous les sommets de G.
On note star(n) le star sur un graphe trivial de n sommets (sans arcs). Donc star(n)=$K_{\{}1, n\}$ est d'ordre n+1 avec un sommet de degré n et n sommets de degré 1. La roue est définie comme étant star(C_n).

Définition 2.12. Une arête uv est une arête splitting si $N[u] \subset N[v]$ ou $N[v] \subset N[u]$. Si uv est une arête splitting, on doit supposer que les sommets sont ordonnés de telle sorte que $N[u] \subset N[v]$

Lemme 2.5. *Un graphe G a la propriété que tous les sous graphes induits (non triviaux) H contiennent une arête splitting si et seulement si G est triangulé.*
Non trivial veut dire que H contient au mois un arc.

Lemme 2.6. *Pour tout graphe triangulé G il existe un arc splitting e tel que G\{e} est triangulé.*

Définition 2.13. [3]
Un graphe G est décomposable si ou bien

1. G est complet, ou
2. On peut exprimer V par $V = A \cup B \cup C$ où
   (a) A, B et C sont disjoints,
   (b) A et C sont non-vide,
   (c) B est complet,
   (d) B sépare A et C dans G, et
   (e) A $\cup B$ et $B \cup C$ sont décomposables.

Une clique maximale d'un graphe G est une clique de G maximale pour l'inclusion. Un arbre de cliques d'un graphe G = (V,E) est un arbre $T = (V_T, E_T)$ où V_T est un ensemble des cliques de G qui contient toute les cliques maximales de G.

Définition 2.14. Un arbre de jonction d'un graphe G est un arbre de cliques de G qui satisfait la condition suivante :
Pour toutes cliques C_1 et C_2 dans l'arbre, toute clique du chemin reliant C_1 et C_2 contient $C1 \cap C2$.

Se basant sur cette définition, on trouve la caractérisation suivante des graphes triangulés :

Théorème 2.11. *Les propriétés qui suivent sont équivalentes.*
1. G est triangulé.
2. G est décomposable.
3. G est récursivement simplicial.
4. G a un arbre de jonction.

2.3 Le parcours en largeur lexicographique LexBFS

Le parcours en largeur lexicographique LexBFS a été introduit par Rose, Tarjan et Lueker pour reconnaitre les graphes triangulés. Cet algorithme est considéré comme l'un des algorithmes les plus populaires vu son utilisation dans divers problèmes comme la reconnaissance de quelques familles restreintes de graphes et la détermination d'une paire dominante dans un graphe AT-free, et l'approximation du diamètre d'une famille restreinte de graphes.

Vu son importance pour la reconnaissance des graphes triangulés et autres classes de graphes, nous avons consacré cette section aux parcours en largeur lexicographique qu'on note généralement par LexBFS. Un rapport de synthèse sur les applications de l'algorithme LexBFS a été présenté par Derek Corneil à la conférence WG 2004 [Cor04].

Il est à noter qu'il existe deux types de parcours qu'on utilise souvent dans le domaine de la théorie des graphes, le premier étant le parcours en largeur d'abord BFS (Breadth First Search) le second est le parcours en profondeur DFS (Depth First Search). Pour notre part, nous allons nous intéresser au parcours LexBFS pour lequel on donne les caractéristiques les plus intéressantes et qui font de lui un outil important de l'algorithmique de la théorie des graphes.

.

2.3.1 Algorithme LexBFS (parcours en largeur lexicographique)

L'algorithme suivant décrit le parcours en largeur lexicographique.
Données : un graphe G = (V,E) et un sommet source s
Résultat : un ordre total σ de V.
1 : Affecter l'étiquette $\emptyset$; à chaque sommet.
2 : label(s) :={n}
3 : Pour i de n à 1 faire :
4 : Choisir un sommet v d'étiquette lexicographique maximale.
5 : $\sigma(i) := v$
6 : Pour chaque sommet non numéroté ω de $N(v)$ faire :
7 : label(ω) := label(ω) $\cup\, i$ { Concaténation de i à la fin de l'étiquette de ω }

2.3.2 Les propriétés du parcours LexBFS

Le parcours LexBFS possède diverses propriétés dont :

Propriété 2.5. Pour un parcours LexBFS, on a :
-si G est triangulé, le dernier sommet visité est simplicial,
-si G est un cographe, le dernier sommet visité est un jumeau,
-si G est un graphe d'intervalles, la dernière clique maximale visitée peut être choisie comme clique extrême dans une chaîne de cliques maximales,
-si G est un graphe de comparabilité, le dernier sommet visité par un LexBFS sur G peut être choisi comme source d'une orientation transitive de G.

Théorème 2.12. *(Caractérisation LexBFS) [69]*
Un graphe G est triangulé si et seulement si tout ordre LexBFS de G est simplicial.

Les propriétés du parcours LexBFS sont particulièrement intéressantes sur les classes de graphes héréditaires (quand on enlève un sommet au graphe, le graphe obtenu reste dans la même classe).

2.4 La reconnaissance

D. J. Rose, R. E. Tarjan et GS. Lucker (1976) [70] ont présenté un algorithme polynomial de reconnaissance des graphes triangulés en $O(n+m)$ en utilisant l'ordre d'élimination simpliciale. Ils montrent qu'un ordre d'élimination parfait d'un graphe triangulé peut être trouvé de manière efficace en utilisant l'algorithme de recherche lexicographique en largeur d'abord. De même F. Gavril [28] a présenté des algorithmes polynomiaux pour résoudre les quatres problèmes d'optimisation (coloration minimum, clique maximum, coverture minimum par des cliques, stable maximum) pour cette classe de graphes.
Une implémentation naïve de l'idée précédente (ordre d'élimination parfait) est la suivante :
.Chercher un sommet simplicial et l'éliminer du graphe.
.Répéter cette procédure jusqu'à ce qu'il ne reste aucun sommet (dans le cas où le graphe initial est triangulé).
Si à une certaine étape, on ne trouve plus de sommet simplicial, alors le graphe initial n'est pas triangulé.
Cette procédure est en $O(n^4)$. En effet tester si un sommet est simplicial prend un temps $O(n^2)$, et comme la recherche d'un sommet simplicial peut requière le test de tout les sommets du graphe G, ceci prendra $O(n^3)$. Etant donné que cette procédure doit être répétée n fois si le graphe est triangulé, la complexité de l'algorithme précédent est donc $O(n^4)$.
Recherche de cardinalité maximum
L'algorithme connu sous le nom de recherche de cardinalité maximum "Maximum Cardinality Search" (MCS) [77] fait la reconnaissance des graphes triangulés en déterminant un ordre d'élimination simplicial σ en temps $O(n+m)$. Son principe st le suivant :
-MCS associe à tout sommet v un poids $\lambda(v)$, qui est initialement égal à 0.
-A chaque étape MCS ajoute à σ le sommet v non encore visité et qui a le plus grand poids.
-MCS incrémente séquentiellement par 1 le poids des voisins de v, et commence une autre phase.

Théorème 2.13. *(Tarjan and Yannakakis, 1984) [77]*
Un graphe non orienté G est triangulé si et seulement si la procédure de recherche de cardinalité maximum du graphe G produit un ordre d'élimination parfait σ.

Le principe de l'algorithme est le suivant :

- Numéroter les sommets de n à 1 dans un ordre décroissant.
- Coisir pour prochain sommet à numéroter le sommet adjacent au plus grand nombre de sommets ayant été numérotés auparavant et répéter cette phase .

L'algorithme réalise la recherche de cardinalité maximum sur un graphe non orienté et lorsque le graphe G est triangulé il donne un ordre d'élimination parfait s.
Procédure de recherche de cardinalité maximum G, σ.
1 : pour tout sommet v de G faire
2 : poser label[v] à zéro.
3 : fin pour

4 : pour tout i de n à 1 faire
5 : choisir un sommet non numéroté v ayant le plus grand label
6 : poser $\sigma(i)$ à i (numéro du sommet v)
7 : pour tout sommet w non numéroté et adjacent à v faire
8 : incrémenter label[w] par 1
9 : fin pour
10 : fin pour
11 : fin procédure

Théorème 2.14. *La procédure précédente peut être implémentée pour effectuer la recherche de cardinalité maximum en temps $O(n+m)$.*

Comme nous l'avons vu, un graphe est triangulé si l'ordre obtenu par la procédure MCS est un ordre parfait. Pour reconnaitre donc un graphe triangulé, il suffit de cérifier si cet ordre est parfait.
Une procédure naïve pour tester si σ est un ordre d'élimination parfait et que G est triangulé consiste à simuler le processus d'élimination des sommets et pour tout sommet à éliminer tester si ses voisins qui restent forment une clique. Cette procédure se fait en un temps $O(nm)$.
Soit σ la permutation des sommets de G produite par la procédure de recherche de cardinalité maximum(G,σ).
L'algorithme suivant teste si σ est un ordre d'élimination parfait. Il est dû aux mêmes auteurs.
Fonction ordre d'élimination parfait(G,σ)
1 : pout tout i de 1 à n - 1 faire
2 : mettre v à $\sigma^{-1}(i)$
3 : mettre m[v] to $\sigma^{-1}(min\{\sigma(w) \mid w \in adj(v), \sigma(w) > \sigma(v)\})$
4 : pout tout les sommets w adjacents au sommet v faire
5 : si $\sigma(m(v)) > \sigma(w)$ et $w \notin adj(m(v))$ alors
6 : retourner faux
7 : fin si
8 : fin pour
9 : fin pour
10 : retourner vraie
11 : fin fonction

Théorème 2.15. *L'algorithme précédent peut être implémenté pour tester un ordre d'élimination parfait en temps $O(n+m)$.*

Soit $X(v) = \{w \in adj(v) \mid \sigma(v) < \sigma(w)\}$.

Lemme 2.7. *Toute clique maximale d'un graphe triangulé G= (V,E) est de la forme $\{v\} \cup X(v)$, pour un certain sommet $v \in V$.*

Lemme 2.8. *Un graphe triangulé possède au plus n cliques maximales.*

Soit σ l'ordre d'élimination produit par la procédure de recherche de cardinalité maximum(G,σ).
L'algorithme suivant énumére toutes les cliques maximales de G.

Procédure toutes les cliques maximales(G,σ)
1 : pour tout sommet v de G faire
2 : poser size[v] à zéro
3 : fin pour
4 : pour i allant de 1 à n faire
5 : poser v à $\sigma^{-1}(i)$
6 : si deg(v)=0 alors
7 : retourner maximal clique $\{v\}$
8 : fin si
9 : mettre X à $\{w \in adj(v) \mid \sigma(v) < \sigma(w)$
10 : si $X \neq \emptyset$ alors
11 : si size[v]$<\mid X \mid$ alors
12 : retourner maximal clique $\{V\} \cup X$
13 : fin si
14 : mettre m[v] à $\sigma^{-1} min(\{\sigma(w) \mid w \in X\})$
15 : mettre size[m[v]] à max{size[m[v]],$\mid X \mid -1$}
16 : fin si
17 : fin pour
18 : fin procédure

Lemme 2.9. *L'algorithme précédent peut être implémenté pour énumérer toutes les cliques maximales en un temps O(n+ m).*

Il existe d'autre algorithmes pour la reconnaissance des graphes triangulés.
On cite entre autres l'algorithme de Fulkerson et Gross [27], celui de Tarjan, et celui de Lekkerkerker and Boland.

Algorithme de Fulkerson et Gross (1965) [27]

1. Poser G'=G ;
2. Si G' est vide alors G est triangulé : STOP
3. Si G' ne contient pas de sommet simplicial alors G n'est pas triangulé.
4. Ôter un sommet simplicial de G' et retourner à (2).

Algorithme de Tarjan (1976)

1. Poser $N := \emptyset$ et $W := V$;
2. Si $W = \emptyset$ alors STOP : G est triangulé
3. Choisir le sommet x de W ayant le plus de voisins dans N.
4. Si les voisins de x dans N ne forment pas une clique alors STOP : G n'est pas triangulé
5. Rajouter x dans N, le retirer de W, et retourner à (2).

Algorithme de Lekkerkerker and Boland :
Entrée : un graphe connexe G = (V,E)
Sortie : une réponse à la question : "G est-il triangulé ?"

1. G' = G
2. tant que ∃ un sommet simplicial dans G' faire
3. trouver un sommet simplicial u ∈ G' ;
4. enlever u et toutes les arêtes uv, v ∈ G', de G' ;
5. fin tant que
6. si G =∅, alors
7. retourner (G est triangulé) ;
8. sinon
9. return (G n'est pas triangulé) ;
10. fin si

Pour finir, on donne l'algorithme basé sur la caractérisation de Lekkerkerker et Boland [58] (La notion de sommet LB-simplicial)

Algorithme Reconnaissance des graphes triangulés :
Entrée : un graphe connexe G = (V,E)
Sortie : une réponse à la question : "G est-il triangulé ?"

1. pour tout v ∈ G faire
2. si v n'est pas un sommet LB-simplicial alors
3. retourner (G n'est pas triangulé) ;
4. fin si
5. fin pour
6. retourner (G est triangulé) ;

2.5 La coloration

D. J. Rose, R. E. Tarjan et GS. Lucker (1976) ont présenté un algorithme polynomial de coloration des graphes triangulés, il est de complexité $O(n + m)$, son principe est le suivant :

Algorithme de coloration d'un graphe triangulé G= (V, E).

1. Déterminer un schéma d'élimination parfait $v_1 < \cdots < v_n$.
2. Colorer G séquentiellement selon l'ordre inverse $v_n < \cdots < v_1$.

Les problèmes de la clique maximum, de la coloration minimum, du stable maximum et de la partition minimum en cliques peuvent être résolus en temps et espace linéaire pour la classe des graphes triangulés [28] et [33].

Au cours de l'exécution d'un programme, le processeur effectue un grand nombre d'accès à la mémoire afin de lire et écrire les valeurs des variables du programme. Ces accès sont naturellement gourmands en temps. Pour accélérer l'exécution des programmes, le processeur est muni d'un petit nombre de zones de stockage a accès beaucoup plus rapide, les registres. En général, le nombre de registres d'un processeur est nettement inferieur au nombre de variables utilisées dans un programme.
Le but de l'allocation de registres est de déterminer ou sont stockées les variables d'un programme a tout moment de son exécution : soit en registres si ces derniers sont disponibles, soit en mémoire le cas échéant. La difficulté est de proposer une affectation optimale des registres.
Dans le cas d'une allocation de registres, le graphe à colorier est un graphe d'interférences, dont les sommets représentent les variables du programme à compiler. Les arêtes sont de deux types. Les arêtes d'interférence relient tous les sommets qui représentent des variables qui ne doivent pas occuper les mêmes registres à un instant donné de l'exécution du programme. Les arêtes de préférence relient tous les sommets qui représentent des variables telles qu'il existe une instruction d'affectation entre celles-ci (et il n'existe pas d'arête d'interférence entre ces sommets). Un poids est associé à chaque arête afin de tenir compte de la fréquence d'exécution des instructions, ainsi que de la fréquence d'utilisation des variables.
Dans ce qui suit, nous définissons un graphe G comme étant un triplet (S, I, P), où G est le graphe dont l'ensemble des sommets est S, l'ensemble des arêtes d'interférence est I et l'ensemble des arêtes de préférences est P. De plus, les graphes formés par S et I d'une part et S et P d'autre part sont respectivement appelés interf-graphe et pref-graphe de G.

L'un des algorithmes utilisés pour la coloration des graphes triangulés est l'algorithme de coloration gourmande qu'on présente ici.
Cet algorithme est utilisé pour le test de k-colorabilité. C'est un algorithme qui fournit une coloration optimale si l'ordre dans lequel sont coloriés les sommets est l'ordre inverse d'un ordre d'élimination simplicial [28]. Ces résultats ont été prouvés en Coq. Il en découle que les graphes triangulés peuvent être colorés de façon optimale grâce à cet algorithme puisque comme on l'a vu, la recherche d'ordre d'élimination simplicial est un problème pour lequel il existe divers algorithmes polynomiaux.
Dans leur article, Sandrine Blazy, Benoît Robillard et Éric Soutif [7] ont écrit en Coq un algorithme de recherche d'un ordre d'élimination simplicial relativement naïf mais dont

la correction est plus simple à montrer que pour les algorithmes les plus efficaces et dont la complexité (en temps) est comparable.
L'algorithme consiste informellement à chercher un sommet simplicial s, le retirer du graphe et itérer ce procédé.
Cette technique fonctionne car tout graphe induit d'un graphe triangulé est lui même triangulé et possède donc un ordre d'élimination simplicial.
L'appel à la fonction is_{sv}(s, S - T) teste si s est un sommet simplicial dans le graphe induit par S -T. Plus précisément, il est testé si s et ses voisins de S -T forment une clique. De plus, un graphe est triangulé si et seulement s'il admet un ordre d'élimination simplicial [26]. Aussi, l'ordre renvoyé par l'algorithme 1 est un ordre d'élimination simplicial si et seulement si le graphe G est triangulé.
L'algorithme [7] est le suivant :
Algorithme peo search (G)
Entrée : Un graphe G=(S, I, P) dont l'interf-graphe est triangulé.
Sortie : Un ordre $x = x_1, x_2, ...x_n(G)$ d'élimination simplicial de l'interf-graphe.
1 : i := 0,U :=$\emptyset$;
2 : tant que $i < n(G)$ faire
3 : trouvé := faux, T := U
4 : tant que trouvé = faux faire
5 : choisir s dans S - T
6 : si is-sv(s, S - T) alors
7 : x_{i+1} := s
8 : trouvé := vrai
9 : sinon
10 : T := T $\cup$ s
11 : fin si
12 : fin tant que
13 : i := i + 1 ; U := U $\cup$s
14 : fin tant que

Chapitre 3

Les graphes faiblement triangulés

Sommaire

3.1 Les graphes faiblement triangulés 41
3.2 La relation entre les graphes triangulés et les graphes faiblement triangulés 44
3.3 La reconnaissance 47
3.4 La coloration 50

Introduction

Dans ce chapitre nous allons présenter la classe des graphes faiblement triangulés, quelques propriétés ainsi que les algorithmes de reconnaissance et de coloration les plus rencontrés dans la littérature pour cette classe de graphes.

3.1 Les graphes faiblement triangulés

Les graphes faiblement triangulés ont été introduits par R. Hayward en 1985 [41]. Cette classe de graphes contient celle des graphes triangulés.

Définition 3.1. Les graphes faiblement triangulés sont les graphes n'ayant aucun cycle ou anti-cycle induit de longueur supérieure ou égale à 5.

Dans l'article où il introduit la classe des graphes faiblement triangulés, R. Hayward [41] montra que ces graphes sont des graphes parfaits.
Dans le même article, il montra aussi que les graphes triangulés sont des graphes faiblement triangulés. La classe des graphes triangulés est donc une sous classe de celle des graphes faiblement triangulés.

Théorème 3.1. *[41] Un graphe G=(V,E) est triangulé ⇒ G est faiblement triangulé.*

L'une des caractérisations des graphes faiblement triangulés est la suivante :

Théorème 3.2. *G est faiblement triangulé si et seulement si son complémentaire est faiblement triangulé.*

Mais la caractérisation la plus répondue des graphes faiblement triangulés est celle se basant sur la notion de deux-paire. Avant de lancer cette caractérisation, on introduit quelques notions nécessaires pour pouvoir la définir.
On dit que W est un ab-séparateur minimal si W est un ab-séparateur et que tout sous-ensemble strict de W n'est pas un ab-séparateur.

Définition 3.2. Une deux-paire est un couple de sommets $\{x, y\}$ tels que toute chaine sans corde reliant x et y est de longueur deux.

Définition 3.3. Une co-paire est une deux-paire du graphe $\overline{G}$ (le complémentaire de G).

Définition 3.4. On dit que $\{x, y\}$ est une deux-paire si et seulement si $S_{xy} = N(x) \cap N(y)$ est un xy-séparateur. Dans ce cas, S_{xy} est l'unique séparateur minimal.

Théorème 3.3. *(Hayward [41])*
Soit G un graphe faiblement triangulé. Alors on a l'une des assertions suivantes :
. G est une clique ;
. G est le complémentaire d'un couplage parfait ;
. G possède une étoile d'articulation [1]*.*

Ryan Hayward, Chinh T. Hofing et Frédéric Maffray [43] on établit le théorème qui suit :

Théorème 3.4. *(Hayward, Hoàng et Maffray [43])*
Soit G un graphe faiblement triangulé quelconque. Alors G est ou bien une clique ou contient une deux-paire.
De plus si C est un séparateur minimal alors ou bien C est une clique ou il contient une deux-paire.

Le théorème précédent est équivalent au théorème suivant dont la démonstration est donnée ci-après.

Théorème 3.5. *Tout graphe faiblement triangulé qui n'est pas une clique possède une deux-paire.*

Pour la preuve du théorème, on donne l'assertion forte que tout graphe faiblement triangulé qui n'est pas une clique vérifie l'une des deux propriétés suivantes :
(1) Si G n'a pas de séparateur qui est une clique, alors tout séparateur de G contient une deux-paire.
(2) G contient une deux-paire.

[1] Un ensemble d'articulation sous forme d'une étoile

Preuve 5.
La preuve est établie par induction (récurrence) sur le nombre de sommets du graphe.
Le premier cas est vérifié facilement en considérant que les plus petits graphes qui sont faiblement triangulés et ayant des séparateurs minimaux qui ne sont pas des cliques sont P_4 la chaine de longueur 4 et C_4 le cycle sans corde de longueur 4. La base du second cas est vérifiée en considérant le graphe trivial constitué de deux sommets.
Par l'hypothèse de récurrence, on suppose que tout graphe faiblement triangulé avec moins de sommets que G satisfait les propriétés (1) et (2).
Soit C un séparateur minimal de G et soit H_C le sous-graphe induit par C.
Nous avons deux cas différents :
Cas 1 : *H_C est une clique. On doit démontrer que G contient une deux-paire. Soient $B_1, B_2, \cdots, B_t$ les ensembles des sommets des composantes connexes de $G - C$. Si l'un des $G - B_j$ n'est pas une clique, alors par l'hypothèse d'induction $G - B_j$ contient une deux-paire. Celle-ci est une deux-paire de G car toute chaine sans corde ayant les deux extrémités terminales dans $G - B_j$ est entièrement contenue dans $G - B_j$. Donc, $t = 2$, et on x,y est une deux-paire quand $x \in B_1$ et $y \in B_2$.*
Cas 2 : *H_C n'est pas une clique. On doit montrer que C contient une deux-paire de G. Ici aussi, on distingue entre deux cas. Soit H_C^c le complément de H_C.*
Cas 2a : *H_C^c n'est pas connexe. Soit D l'ensemble des sommets d'une certaine composante connexe de H_C^c ayant au moins deux sommets (comme C n'est pas une clique, il doit y avoir un tel ensemble D). Comme D est composante de H_C^c, tout sommet de C-D est adjacent à tout sommet de D. Pour la même raison, il est clair que D n'est pas une clique. On suppose donc que D est un séparateur minimal de G-(C-D). Sinon, $D' \subset D$ est un séparateur de G-(C-D). Et on a donc $(C-D) \cup D'$ est un séparateur de G, contradiction avec la minimalité de C. Par l'hypothèse de récurrence, D contient une deux-paire, {x, y} de G-(C-D). Comme tout sommet de D est adjacent à tous les sommets de C-D {x, y} est une deux-paire de G.*
Cas 2b : *Supposons que H_C^c est connexe. Soient $B_1, B_2, \cdots, B_t$ les ensembles des sommets des composantes connexes de G-C. D'après le théorème 1 de [9], il s'en suit que dans chaque $B_1, B_2, \cdots, B_t$, il existe un certain sommet qui est adjacent à tous les sommets de C dans G. Notons $z(B_j)$ un tel sommet pour tout j.*
Cas 2bI : *Si $|B_j| = 1$ pour tout j. Donc tout B_j est constitué d'un seul sommet qu'on a noté $z(B_j)$. Comme H_C est un graphe faiblement triangulé qui n'est pas une clique, par l'hypothèse de récurrence H_C contient une certaine deux-paire {x, y} , qui est clairement une deux-paire de G, puisque x et y sont tout les deux adjacents à $z(B_j)$ pour tout j.*
Cas 2bII :*Supposons maintenant que $|B_p| \geq 2$ pour un certain $p \in 1, \cdots, t$. Soit z un sommet quelconque de B_j qui soit adjacent à tous les sommets de C; soit D l'ensemble des sommets de C qui sont adjacents à un certain sommet de $B_j - z$. A présent D est un séparateur minimal de $G - z$. Notons que D n'est pas vide, et n'est pas une clique (Autrement $D \cup \{x, y\}$ est un séparateur qui est une clique de G, contradiction). D'où par l'hypothèse de récurrence D contient une deux-paire de $G - z$ qui est aussi une deux-paire de G.*
Pour prouver (2), on doit supposer que G a un séparateur qui est une clique. Soient $B_1, B_2, \cdots, B_t$ les ensembles des sommets des composantes connexes de $G - C$. Si l'un $G - B_j$ n'est pas une clique alors par l'hypothèse d'induction, $G - B_j$ contient une deux-

paire ; comme toutes chaine sans corde de G ayant les deux extrémités dans $G - B_j$ est totalement contenu dans $G - B_j$, cette deux-paire est une deux-paire aussi de G. Donc on doit supposer que tous $G - B_j$ sont des cliques. Ceci implique que t=2 et que $\{x, y\}$ est une deux-paire quand $x \in B_1$ et $y \in\in B_2$.

Le théorème qu'on donne juste après est utilisé pour la construction d'un algorithme de reconnaissance des graphes faiblement triangulés.

Théorème 3.6. *Soit G un graphe et $\{x, y\}$ une deux-paire de G. Notons $G' = G \cup \{x, y\}$ le graphe obtenu à partir de G en rajoutant l'arête xy. Alors G est faiblement triangulé si et seulement si G' est faiblement triangulé.*

C. Hoàng,et F. Maffray [45] ont montré que les graphes faiblement triangulés sont des graphes de quasi-parité stricte.

3.2 La relation entre les graphes triangulés et les graphes faiblement triangulés

Hayward [37] a établi une relation intéressante entre les graphes triangulés et les graphes faiblement triangulés en définissant un schéma de décomposition des graphes faiblement triangulés similaire à celui des graphes triangulés.

Il remarqua que la classe des graphes triangulé peut être générée en ajoutant répétitivement un sommet qui n'est pas le milieu d'un P_3 (le sommet ajouté est simplicial). Il montra aussi que les graphes faiblement triangulés peuvent être générés en commençant par un ensemble de sommets sans aucune arête, et en ajoutant répétitivement une arête qui n'est pas le milieu d'un P_4.

Ce résultat a donné l'idée qu'une arête dans un graphe faiblement triangulé joue le même rôle que joue un sommet dans un graphe triangulé.

Cette idée a été utilisée plus tard par Berry, Bordat et Heggernes [6] pour donner une extension du théorème de Lekkerkerker et Boland [58] : "Un graphe est triangulé si et seulement si tous les séparateurs minimaux inclus dans le voisinage du sommet sont des cliques".

Définition 3.5. On dit qu'une arête e=ab de E est LB-simpliciale si, pour tout séparateur minimal S inclus dans le voisinage de e, e est S-saturante.

Caractérisation 3.1. (Berry, Bordat and Heggernes [5])
Un graphe G est faiblement triangulé si et seulement si toute arête est LB-simpliciale.

On observe deux observations importantes :

Observation 3.1. *Un cycle sans corde de longueur 5 est isomorphique à son complémentaire.*

Observation 3.2. *Le complémentaire de tout cycle sans corde de longueur ≥ 6 contient un cycle sans corde de longueur 4.*

On ajoute à ces observations deux autres observations. La première est donnée par Hayward [37], la seconde par Kratsch [56] :

Observation 3.3. *[37]*
Les graphes triangulés peuvent être générés en ajoutant répétitivement un sommet qui n'est pas le sommet du milieu d'un P_3 (une chaine sans corde de longueur 3). Les graphes faiblement triangulés le peuvent en ajoutant répétitivement un arc qui n'est pas l'arc du milieu d'un P_4.

Observation 3.4. *[56]*
Dans un graphe triangulé G = (V;E), pour tout séparateur minimal s, toute composante C de G (V - s) contient un point de confluence, pour un graphe faiblement triangulé G, pour tout séparateur minimal s, toute composante complète [2] *C contient ou bien un point de confluence* [3] *ou une arête de confluence* [4]*.*

Ces deux observations montrent qu'une arête dans un graphe joue un rôle similaire à celui d'un sommet dans un graphe triangulé.
On peut à présent donner le théorème de Ryan B. Hayward [39] :

Théorème 3.7.
Tout graphe sans C_k pour $k \geq 5$ et sans $\bar{C}_k$ a un certain sommet qui n'est pas le sommet du milieu d'un P_5. De plus, si le graphe n'est pas une clique alors il a deux tels non-adjacents sommets.

On donne maintenant la définition d'une arête s-saturante, qui est une version plus forte d'une arête de confluence.

Définition 3.6. (Hayward [38]) Par rapport à un ensemble de sommets S, une arête de $G \backslash S$ est dite S-saturatante si, pour toute composante S_j de $\bar{G}(S)$, au moins un sommet de e voie tous les sommets de S_j.

Comme Kratsch , Hayward [38] montra que dans toute composante fortement connexe d'un séparateur minimal dans un graphe faiblement triangulé, il y a ou bien un point de confluence ou une arête s-saturante. Plus tard, Berry, Bordat, et Heggernes [6] ont donné une nouvelle caractérisation des graphes faiblement triangulés, ils ont défini la notion d'arête LB-simpliciale basée sur le rôle que joue une telle arête dans un graphe faiblement triangulé.

Définition 3.7. [6]
Une arête e est LB-simpliciale si pour tout séparateur minimal s inclus dans le voisinage de e, e est s-saturante.

[2] Une composante C de $\mathcal{C}(S)$ est complète si N(C)=S, où $\mathcal{C}(S)$ est l'ensemble des composantes conexes de G(V-S)
[3] x est un point de confluence de $X \subseteq V$ si touts les ommets de N(X) voient x
[4] Une arête de confluence de C est une arête e tel que $N(C) \subseteq N(e)$.

Le théorème qui permet d'avoir la nouvelle caractérisation des graphes faiblement triangulés est donc :

Théorème 3.8. *[6]*
Un graphe G=(V, E) est faiblement triangulé si et seulement toute arête de E est LB-simpliciale.

Pour prouver ce théorème, les auteurs ont introduit deux lemmes en plus pour pouvoir appliquer le théorème de l'article de Hayward où il a introduit les graphes faiblement triangulés.

Lemme 3.1. *[6]*
Dans un graphe donné, une arête appartenant à un trou ne peut pas être LB-simpliciale.

Lemme 3.2. *[6]*
Dans un graphe donné, tout anti-trou contient une arête qui n'est pas LB-simpliciale.

Théorème 3.9. *[41]*
Soit G un graphe faiblement triangulé, et soit s un séparateur minimal de G tel que $\bar{G}(s)$ est connexe. Alors dans toute composante complète C de C(s), il existe un sommet qui voie (est adjacent a) tous les sommets de s.

En se basant sur ces résultats, on peut donner leur preuve du théorème précédent.

Preuve 6. $\Rightarrow$ *Soit G un graphe dans lequel toute arête est LB-simpliciale. Alors d'après le premier lemme G ne peut pas contenir un trou, et par le second G ne peut pas contenir un anti-trou. Donc G doit être faiblement triangulé.*
$\Leftarrow$ *Soit G un graphe faiblement triangulé, et supposant une certaine arête ab paraît être LB-simpliciale. Soit s = N(C) un séparateur minimal contenu dans le voisinage de ab pour lequel ab parait être s-saturante, soit s1 une composante connexe de G(s) telle que ni a ni b voie tous les sommets de s1, et considérant le sous-graphe G0 induit par C [s1[ab]]. Comme tout sous-graphe d'un graphe faiblement triangulé est lui même faiblement triangulé, G0 doit être faiblement triangulé. s1 est un séparateur minimal de G0, avec deux cliques, et G0 (s1) est connexe. Ni a ni b voie touts les sommets de s1, contradiction.*

3.3 La reconnaissance

Hayward a proposé un algorithme en $O(n^5)$ pour la reconnaissance de cette classe de graphes, il s'est basé sur l'existence d'un trou dans le graphe. Ce qui a été amélioré ensuite par la procédure de Spinard pour la recherche d'un trou. Plus tard, Hayward, Hoàng et Maffray [42] ont caractérisé les graphes faiblement triangulés par le fait d'avoir une deux paire ; plusieurs résultats ont été obtenus en se basant sur cette propriété. Arikati et Rangan ont donné un algorithme efficace pour trouver une deux-paire. Spinrad et Sritharan (1995) [75] ont utilisé ce dernier pour améliorer la reconnaissance des graphes faiblement triangulés en temps $O(n^2m)$. Ils ont remarqué que si un graphe G possède une deux-paire alors le graphe obtenu en reliant les deux sommets de la deux-paire est faiblement traingulé si et seulement si le graphe G est faiblement triangulé. Le principe de leur algorithme consiste donc à chercher une deux-paire $\{a, b\}$ et ajouter l'arc ab jusqu'à l'obtention d'une clique. Ils ont montré aussi que l'ajout d'un arc entre deux sommets d'une deux-paire ne change pas la propriété pour le graphe d'être faiblement triangulé car deux sommets formant une deux-paire ne peuvent pas appartenir à un trou ou un anti-trou. Hayward, Spinrad et Sritharan [76] ont amélioré cet algorithme en $O(m^2)$. Il cherche une co-paire et l'élimine jusqu'à ce qu'il ne reste plus d'arête dans le graphe. Il utilise pour cela un schéma d'élimination des arêtes.
Il existe d'autres algorithmes comme celui de Hayward, Spinard et Sritharan (1985) [44] qui est de complexité $O(m^2)$ qui utilise une structure appelée "handle", et celui de Berry, Bordat et Heggernes. On cite aussi un algorithme de Samuel Hym, Todinca et Thuillier (2001) [51] qui est une amélioration de l'algorithme de Spinard et Sritharan et qui est de complexité O(mn).
On donne ici un résumé sur les algorithmes de reconnaissance des graphes faiblement triangulés tout en donnant pour chaque algorithme sa complexité.
Hayward [40], $O(n^5)$
Spinrad [73], $O(n^{4.376})$
Spinrad er Sritharan [75], $O(n^2.m)$
Hayward et al [44], $O(m^2)$
Berry et al [5], $O(m^2)$
Nikolopoulos et Palios [65], $O(m^2)$
Chandrasekhran et al [11], $O(logn)$, $O(n^5)$
Chong, Nikolopoulos et Palios [12], $O(log^n)$, $O(m^2/logn)$
Vu le grand nombre d'algorithmes de reconnaissance des graphes faiblement triangulés, on va pas tous les donner mais on se contentera de citer deux algorithmes seulement.
Le premier algorithme qu'on donnera a été proposé par J.Spinard et R.Sritharan [75]. Il repose sur la détection des deux paires.

Algorithme 1 G est-il faiblement triangulé
Calcul de $\bar{G}$, complémentaire de G
Contient-une-deux-paire ← vrai
Tant que Contient une deux-paire **faire**
Si il existe une deux-paire {x,y} dans $\bar{G}$ **alors**
Ajouter l'arête xy dans $\bar{G}$
Sinon

Contient-une-deux-paire ← faux
Finsi
Fin tant que
Si $\bar{G}$ est un graphe complet **alors**
Retourner "G est faiblement triangulé"
Sinon
Retourner "G n'est pas faiblement triangulé"
Fin si L'algorithme que nous donnons ci-dessous est basé sur la notion d'arête LB-simpliciale.

Algorithme de reconnaissance des graphes faiblement triangulés
Entrée : un graphe connexe G = (V,E)
Sortie : une réponse à la question : "G est-il faiblement triangulé ?"

1. Pour tout e ∈ E faire
2. si e n'est pas une arête LB-simpliciale alors
3. retourner (G n'est pas faiblement triangulé) ;
4. fin si
5. fin pour
6. retourner (G est faiblement triangulé) ;

L'algorithme suivant est dû à J. Spinrad et R. Sritharan [75] et il est de complexité $O(m.n^2)$.

L'algorithme de J. Spinrad et R. Sritharan [75]
1 : Calcul de $\bar{G}$, complémentaire de G
2 : Contient-une-deux-paire ← vrai
3 : **Tant que** Contient une deux-paire **faire**
4 : **Si** il existe une deux-paire {x,y} dans $\bar{G}$ **alors**
5 : Ajouter l'arête xy dans $\bar{G}$
6 : **Sinon**
7 : Contient-une-deux-paire ← faux
8 : **Finsi**
9 : **Fin tant que**
10 : **Si** $\bar{G}$ est un graphe complet **alors**
11 : Retourner "G est faiblement triangulé"
12 : **Sinon**
13 : Retourner "G n'est pas faiblement triangulé"
14 : **Fin si**
Le dernier algorithme qu'on donne est celui de Samuel Hym [51]. Cet algorithme permet de reconnaître les graphes faiblement triangulés en temps $O(m^2)$.

Algorithme 2 G est-il faiblement triangulé
1 : Calcul des co-paires de G
2 : **Tant** que G contient une co-paire {x,y} **faire**
3 : Suppression des co-paires de la forme {a,b} avec a dans la composantes co-connexe de

x dans $G \setminus (N_{\bar{G}}(x) \cap N_{\bar{G}}(y))$ et b dans celle de y (en utilisant l'algorithme 3)
4 : Suppression de l'arête xy du graphe
5 : Calcul des co-paires d'extrémité x
6 : Calcul des co-paires d'extrémité y
7 : **Fin tant que**
8 : **Si** le graphe obtenu est stable **alors**
9 : Retourner "G est faiblement triangulé"
10 : **Sinon**
11 : Retourner "G n'est pas faiblement triangulé"
12 : **Fin si**

3.4 La coloration

Hayward, Hoàng et Maffray [42] ont obtenu un algorithme en $O(m.n^3)$ pour trouver une clique de taille maximum et une coloration minimale d'un graphe faiblement triangulé. Raghunathan, Arvind ont amélioré cet algorithme en 1989 en proposant un algorithme en $O(m.n^2)$ pour trouver une clique de taille maximum et une coloration minimale d'un graphe faiblement triangulé. Ils donnèrent aussi un algorithme en $O(n^5)$ pour la version pondérée du même problème. Le meilleur algorithme à ce jour pour la coloration des graphes faiblement triangulés est celui de R. Hayward, J. Spinrad et R. Sritharan (2000), il a pour complexité $O(n.m)$.
L'algorithme qu'on donne ici est celui de Hayward, Hoàng et Maffray [42] dont la comlpexité est $O(m.n^3)$.
Soit $G(x, y \rightarrow z)$

Algorithme OPT(G)
Entrée : un graphe faiblement triangulé G.
Sortie : une clique de taille maximum et une coloration minimale f_G.
Etape 1 : chercher une deux-paire {x, y} de G.
si G ne contient pas de deux-paire, alors
(a) $K_G \leftarrow V(G)$,
(b) pour i=1 à n faire $f_G(v_i) \leftarrow i$, et
(b) Stop.
Etape 2 : $J \leftarrow G(x, y \rightarrow z)$.
Etape 3 : K_J, $f_J \leftarrow OPT(J)$.
Etape 4a : si $z \notin K_G$, alors $K_G \leftarrow K_J$, sinon $z \in K_J$ et $\cdots$
si x voie tous les sommets de $K_G - \{z\}$ alors $K_G \leftarrow K_J - \{z\} + \{x\}$,
sinon $K_G \leftarrow K_J - \{z\} + \{y\}$
Etape 4b : $f_G(x) \leftarrow f_G(x) \leftarrow f_J(z)$;
pout tout $v_i \in J - \{x, y\}$ faire
$f_G(v_i) \leftarrow f_J(v_i)$.
Fin.

Chapitre 4

Les graphes k-Super-Triangulés

Sommaire

4.1 Les graphes (k, l)-chordal et k-chordal 52
4.2 Les graphes k-Super-Triangulés 58
4.3 Reconnaissance des graphes STR_k 59

Introduction

On a vu dans les chapitres précédents les graphes triangulés et faiblement triangulés. Ces deux classes peuvent être définis comme des graphes «(k, l)-chordal». Cette classe de graphes est général, il suffit de varier les paramètres k et l pour tomber sur un nombre considérable de graphes comme les graphes triangulés, HHD-free ainsi que d'autres types de graphes. Lorsque l est égal à 1, les graphes (k-l)-chordal sont appelés «k-chordal». Pour notre part on introduira les graphes k-Super-Triangulés qui sont en relation directe avec les graphes «k-chordal» et pour lesquels on donnera des algorithmes de reconnaissance.

4.1 Les graphes (k, l)-chordal et k-chordal

Avant de parler de la classe des graphes k-Super-Triangulés, on donne un rappel sur les graphes (k, l)-chordal et graphes k-chordal ainsi que leurs propriétés et plus particulièrement leurs algorithmes de reconnaissance.
Les graphes (k, l)-chordal est une classe de graphe qui contient un grand nombre de graphes comme les graphes triangulés et faiblement triangulés.

Définition 4.1. (Auiello, D'Arti, Moscarini [2])
Un graphe est (k, l)-chordal si tout cycle de G de longueur au moins k contient au moins l cordes.

L'intérêt des graphes (k, l)-chordal réside dans le fait qu'il existe plusieurs classes de graphes qui peuvent être considérés comme des graphes (k, l)-chordal avec des valeurs fixes de k et l, ou peuvent être définis à l'aide de cette classe de graphes. On va donner dans ce qui suit quelques sous-classes des graphes (k, l)-chordal.
La première classe qu'on cite est celle des graphes triangulés. En effet les graphes triangulés sont des graphes (4,1)-chordal.
Une autre sous-classe intéressante des graphes (k, l)-chordal est la classe des graphes HHD-free. Les graphes HHD-free sont les graphes sans maison ni trou ni domino. La proposition de F. F. Dragan montre la relation entre les graphes (k, l)-chordal et les graphes HHD-free.
La proposition suivante est une caractérisation des graphes (5,2)-chordal.

Propriété 4.1. Un graphe est (5,2)-chordal si et seulement s'il est HHD-free.

Les graphes sans corde peuvent être définis par les graphes (k, l)-chordal. Ils sont les graphes (5, 1)-chordal.

Théorème 4.1. *[16] Un graphe G ne contient pas de trou si et seulement si chaque graphe induit H de G ayant au moins une arête contient une arête qui n'est pas l'arête du milieu d'un P_4 dans H.*

Un graphe G est à distance héréditaire si G est connexe et toute chaine induite dans G est isométrique. Tout trou C_k, $k \geq 5$ n'est pas à distance héréditaire.

Théorème 4.2. *[47]*
Un graphe a au moins deux "crossing" cordes dans tout cycle de longueur au moins 3 si et seulement si il est à distance héréditaire.

4.1.1 Les graphes k-chordal

Comme nous l'avons vu dans la section précédente les graphes (k, l)-chordal sont les graphes dont tout cycle de longueur au moins k contient au moins l cordes. On a aussi vu qu'on variant les deux paramètres k et l on tombe sur diverses classes de graphes. Pour la valeur 1 de l les graphes (k, 1)-chordal sont nommés k-chordal et dans cette section on va jeter un peu de lumière sur cette classe de graphes pour laquelle on ne trouve pas beaucoup de résultat.

Définition 4.2. Un graphe est k-chordal si et seulement s'il ne contient aucun cycle sans corde de longueur supérieure à k.

Les graphes k-chordal sont connus aussi sous le nom de "k-bounded hole family" [30] ou bien la famille à trous k-bornés.

Problème de reconnaissance

Le problème de déterminer si un graphe contient un cycle sans corde de longueur supérieure ou égale à k, pour une certaine valeur fixe de $k \geq 4$, est résolu en temps $O(n^k)$ par Hayward [40]. Spinrad [73] améliora la comlexité de l'algorithme de Hayward en proposant un autre algorithme de complexité $O(n^{k-3}M)$, où $M \simeq n^{2.376}$ est le temps nécessaire pour la multiplication de deux matrices carrées $n \times n$. Notons que le problème de détermination si un graphe contient un cycle de longueur supérieure ou égale à 5 est résolu en un temps O(n+m). En effet c'est le problème de reconnaissance des graphes triangulés qui est un cas particulier de reconnaissance des graphes k-chordal.
Etant donné que les graphes faiblement triangulés peuvent être définis en fonction des graphes 5-chordal, les algorithmes de Hayward [40] et de Spinrad [73] peuvent être utilisés pour reconnaitre les graphes faiblement triangulés en temps $O(n^5)$ et $O(n^{4.376})$ respectivement. Il suffit de les appliquer sur le graphe G puis sur son complémentaire $\overline{G}$.

Gavril [30] a démontré que la taille de la plus longue chaine induite d'un graphe n'ayant pas de cycles induits de taille plus grande que k sommets peut être calculés en temps $O(n^k m)$ où n est le nombre de sommets et m le nombre d'arêtes. Tetsuya Ishizeki, Yota Otachi et Koichi Yamazaki [52] ont présenté un algorithme en temps $O(n^k)$ pour le même problème.

Le nombre d'arêtes d'un cycle induit de taille maximum d'un graphe G est appelé la cordalité de G. La cordalité d'un graphe G est définie aussi comme étant le plus petit entier k tel que G est k-chordal. Les arbres sont par convention, de cordalité 2. Les graphes k-chordal ont été étudié dans la littérature dans [4, 5, 14, 13, 20, 22, 41, 40, 53].
Bodlaender and Thilikos [4]ont prouvé qu'un graphe k-chordal avec dégénérescence d contient un séparateur de taille $O((dn)^{\frac{k-3}{k-2}})$, peut être calculé en temps linéaire.

Il existe plusieurs classes de graphes qui sont connues avoir une cordalité bornée. Par exemple les graphes triangulés (et donc les graphes d'intervalles) sont 3-chordal, les graphes faiblement triangulés, les graphes à distance-héréditaires et les graphes de tolérance sont 4-chordal, les graphes "AT-free" et les graphes de co-comparabilité sont 5-chordal. Il est à noter qu'il y a divers autres graphes de cordalité bornée.
Chandran et Ram ont montré que le nombre de séparateurs minimum dans les graphes k-chordal est au plus $\frac{(k+1)n}{2} - k$.

Algorithme de Hayward

Hayward [40] donna un algorithme de complexité $O(n^k)$ pour décider si un graphe contient un trou de longueur au moins k ($k \geq 4$). Son algorithme peut être utilisé pour reconnaitre les graphes "k-chordal" en $O(n^{k+1})$. Le principe de l'algorithme de Hayward est le suivant :
Soit n le nombre de sommets du graphe G et m est le nombre des arêtes.
Soit P une chaine de longueur k-2. Il est possible de décider si la chaine P peut être complétée pour donner un trou de longueur supérieure ou égale à k en $O(m)$.
Soit x et y les extrémités de P. Créer un graphe G' par élimination de tous les sommets internes de P et leur voisins sauf x et y. Il est facile de voir qu'il existe un trou de longueur k qui inclut P si et seulement s'il existe une chaine de x à y dans G'. Comme il y a $O(n^{k-2})$ chaines possibles dont chacune peut être retrouvée en temps $O(m)$, on peut donc déterminer si un graphe possède un trou de longueur $\geq$ k en temps $O(n^{k-2}m)$ ou bien en $O(n^k)$.
On peut utiliser l'algorithme de Hayward pour reconnaitre les graphes "k-chordal" comme suit :

Algorithme k-chordal
1 : Déterminer toutes les chaines possibles de longueur k-1 qu'on peut avoir des sommets du graphe G.
2 : Vérifier si les chaines déterminées dans 1 existent dans le graphe G.
3 : Choisir une chaine P d'extrémités x et y non encore testé (parmi celles qui existent dans le graphe G).
4 : Éliminer les sommets de la chaine P sauf x et y.
5 : Éliminer les voisins des sommets de la chaine P sauf x et y.
6 : Chercher dans le graphe obtenu une chaine entre x et y.
7 : S'il existe une chaine entre x et y dans G' alors il existe un trou de longueur supérieure à k passant par la chaine P.
8 : Sinon : il n'existe pas de trou de longueur supérieure à k passant par P, revenir à 3.

Algorithme de Spinrad

Spinrad [73] donna un autre algorithme de reconnaissance des graphes (k - 1)-chordal en temps $O(n^{k-3}T)$, ce dernier peut être modifié pour déterminer si un graphe non pondéré a une chaine induite de longueur au moins l en un temps $O(n^{l-3}T)$, où T est temps nécessaire pour vérifier si un graphe à O(n) sommets est transitif.
Le principe de l'algorithme de Spinrad est le même que celui de Hayward. Il consiste à teter si une chaine de longueur k-3 peut être complétée pour donner un trou de longueur au moins k. Sa procédure holetest peut déterminer si une chaine P de longueur k-3 fait partie d'un trou de longueur au moins k.

Procédure holetest(P) Soit x et y les extrémités d' une chaine P de longueur k-3.
V1 :=V-P-les voisins des sommets de P ;

G_1 :=le sous graphe de G induit par V1 ;
C :=l'ensemble des composantes connexes de G1 ;
créer un digraphe G2 avec un sommet pour tout élément de C, N(x) et N(y)
V_{2a} :=N(x)-N(y) ;
V_{2b} :=1
V_{2c} :=N(x)-N(y)-P ;
pour tout v2a ∈ V2a v2c ∈ V2C faire
pour tout v2a ∈ V2a, v2c ∈ V2c faire :
créer une arête (v2a, v2c) si et seulement si (v2a, v2c) ∈ G ;
pour tout v2a ∈ V2a, v2b ∈ V2b faire :
créer une arête (v2a, v2b) si et seulement si v2a est relié à un sommet de la composante de v2b dans G1 ;
pour tout v2b ∈ V2b, v2c ∈ V2c faire :
créer une arête (v2b, v2c) si et seulement si v2c est relié à un sommet de la composante de v2b dans G1 ;
P peut être complété pour être un trou de longueur ≥ k si et seulement si G2 n'est pas transitif.

Lemme 4.1. *Soit P une chaine de longueur k-3. La procédure holetest(P) détermine correctement si P peut être une partie d'un trou de longueur ≥ k.*

Preuve 7. *Supposons que P peut être complété pour être un trou H de longueur ≥ k. Soit u un sommet de H qui apparait immédiatement avant P et soit v le sommet qui apparait immédiatement après P dans H. Tous les éléments de H-P-u-v seront tous dans une seule composante connexe c. Dans G2 il y aura des arêtes de u à c et de c à v mais pas d'arêtes de u vers v. D'où G2 n'est pas transitif et holetest va donner une réponse à la question P peut-elle être complété pour donner un trou.*
Supposons que holetest peut répondre si P peut-être complétée. Il devra y avoir une paire d'arêtes (u, c), (c, v) dans G2 tels que (u, v) n'est pas une arête. Il existe une chaine de u à v qui utilise seulement les sommets de c comme sommets intermédiaires. La plus courte chaine P2 de u vers v qui vérifie cette condition doit comporter au mois deux arêtes, comme u et v ne sont pas adjacents alors elle doit être sans corde. Il y a un trou P, u, P2, u dans G qui est de longueur au moins k.

Théorème 4.3. *On peut déterminer si G a un trou de longueur ≥ pour un k fixe en temps* $0(n^{k-3}T)$*, où T est le temps nécessaire pour vérifier qu'un graphe ayant O(n) sommets est transitif .*

Algorithme de Tetsuya Ishizeki

Le problème de la plus longue chaine induite pondérée (W)LIP ((weighted) longest induced path problem) est de trouver la plus longue chaine induite dans un graphe dont les sommets sont pondérés. Ce problème est NP difficile pour le cas général (graphes) [29]. Gavril [30] a démontré un algorithme polynomial pour le problème WLIP pour les graphes k-chordal. Sa complexité est en temps $O(|V|^k|E|)$ pour les graphes k-chordal à sommets

pondérés. Kratsch et al. [57] ont donné un autre algorithme de complexité $O(|V||E|^2)$ toujours pour le problème WLIP pour les graphes AT-free à sommets pondérés. Il est connu que les graphes AT-free sont des graphes qui sont 5-chordal, donc l'algorithme de Gavril marche aussi pour le problème WLIP dans le cas des graphes AT-free, mais sa complexité est $O(|V|^5|E|)$.

L'algorithme que nous allons donner ci-après est celui de Tetsuya Ishizeki et all [52]. Cet algorithme est conçu pour le problème WLIP pour les graphes k-chordal. Sa complexité est $O(|V|^k)$.
Pour un graphe à sommets pondérés, w_G ou bien w représente la fonction poids de G. Pour un sous ensemble S de V, $N(S)$ représente le voisinage de l'ensemble S.
Soit G=(V, E) un graphe et soit $P = (v, v_1, v_2, \cdots, v_p)$ une chaine induite de G. Pour un sommet v tel que $v \in N(v_1)$ et v n'appartenant pas à $N(\{v_2, \cdots, v_p\})$, on note la chaine induite $(v, v_1, v_2, \cdots, v_p)$ par vP. De la même façon on note aussi Pu. On désignera par len(P) la longueur de P, qui représente le poids total des sommets de V(P) (resp. des arêtes dans E(P)) si G est un sommet (resp. arête) pondérée, maxlen(Q) : $Q = (u_1, u_2, ..., u_q)$ est une chaine induite telle que $v_i = u_i \ \forall 1 \leq i \leq p, p \leq q\}$ by lip(P), et la longueur de la plus longue chaine induite par lip(G). On note aussi l'ensemble $\{P = (v_1, v_2, ..., v_l)$: P est une chaine induite dans G $\}$ par $P_l(G)$ (noter qu'on distingue entre $(v_1, v_2, ..., v_l)$ et $(v_l, v_{l-1}, ..., v_1)$).
Gavril [30] montra le lemme suivant.

Lemme 4.2. *Soit G=(V, E) un graphe k-chordal, $P = (v_1, v_2, ..., v_p)$ une chaine induite de G ayant au moins k - 1 sommets. Alors $(v, v_1, v_2, ..., v_{k-1})$ est une chaine induite de G si et seulement si vP est une chaine induite dans G.*

Le corollaire suivant découle du lemme précédent.

Corollaire 4.1. *Soit vQ une chaine induite dans $P_{k-1}(G)$, et notons S(vQ) l'ensemble $\{u : vQu \in P_k(G)\}$. Alors*

$$lip(vQ) = \begin{cases} len(vQ), & si\ S(vQ) = \emptyset\ ; \\ w(v) + max_{u \in S(vQ)} lip(Qu), & sinon. \end{cases}$$

Pour l'algorithme Tetsuya Ishizeki et all [52], le problème WLIP pour les graphes k-chordal est réduit au problème de la plus longue chaine (pas nécessairement induite) pour les graphes orientés acycliques dont les arcs sont pondérés.
Considérons un graphe orienté $\acute{D}_G$ obtenu à partir d'un graphe k-chordal G tel que $V(\acute{D}_G) = \{P : P \in P_{k-1}(G)\}$ et $E(\acute{D}_G) = \{(P_1, P_2) : \exists Q \in P_{k-2}(G)$ tel que $(P_1 = vQ) \wedge (P_2 = Qu)$ pour certains v et u et $vQu \in P_k(G)\}$. Considérons alors un arc pondéré d'un graphe orienté D_G obtenu à partir de $\acute{D}_G$ tel que $V(D_G) = V(\acute{D}_G) \cup \{t\}$ où $t / \in V(\acute{D}_G)$ et $E(D_G) = E(\acute{D}_G) \cup (P, t) : P \in T$ où $T = \{P \in V(\acute{D}_G)$: le degré extérieur de P est 0 dans $\acute{D}_G\}$. L'arc $(P, t) \in E(D_G)$ est pondéré par len(P) et l'arc $(vQ, Qu) \in E(D_G)$ par w(v). Par la manière de construction de D_G et le corollaire précédent, on peut dire que lip(G) est égal à la longueur de la plus longue chaine (pas nécessairement induite) de D_G.

Par conséquent, on a le théorème suivant.

Théorème 4.4. *Pour tout graphe G k-chordal, lip(G) peut être calculé en temps $O(n^{\frac{k-1}{2}}m)$ si k est un nombre impair et en temps $O(n^{\frac{k}{2}})$ si k est un nombre pair.*

Comme corollaire du théorème précédent, on a le résultat suivant.

Corollaire 4.2.
Pour tout graphe k-chordal G, lip(G) peut être calculé en temps $O(m^k)$.

Du théorème précédent, WLIP peut être calculé en temps $O(n^2m)$ pour les graphes 5-chordal, ce qui donne la même complexité pour les graphes AT-free dans [57]. Comme les graphes 5-chordal contiennent complètement les graphes AT-free, ce résultat fait étendre le résultat de [57] à une classe plus large sans diminuer la complexité en temps.

4.2 Les graphes k-Super-Triangulés

Dans cette section, on introduira les graphes k-Super-Triangulés. On donnera ensuite les classes de graphes contenues dans cette classe et les différentes inclusions qui existent entre les graphes k-Super-Triangulés. On termine par donner deux algorithmes de reconnaissance des graphes k–Super-Triangulés k étant un nombre fixe.

Remarque 4.1. Dans tout ce qui suit, un graphe k-Super-Triangulé sera noté STR_k.

La notion de graphe k-Super-Triangulés est une généralisation de celle d'un graphe faiblement triangulé.

Définition 4.3. STR_k **:** Un graphe est STR_k si et seulement s'il ne contient aucun cycle ni anti-cycle sans corde de longueur supérieure à k.

Les définitions suivantes se déroulent.

Définition 4.4. STR_3 **:** Un graphe est STR_3 si et seulement s'il ne contient aucun cycle ni anti-cycle sans corde de longueur supérieure à 3.

Les graphes STR_3 sont les graphes faiblement triangulés qui ne contienent ni triangle ni deux arcs (complémentaire d'un carré) comme sous-graphe induit.

Remarque 4.2.

1. La classe des graphes STR_3 est une sous classe des graphes triangulés.
2. Un graphe G est STR_3 si et seulement si G et $\overline{G}$ sont triangulés.

Comme nous l'avons vu au chapitre 2, un graphe G est scindé si et seulement si G et son complémentaire sont triangulés. Les graphes STR_3 sont donc exactement les graphes scindés et la reconnaissance des graphes STR_3 peut être faite en utilisant l'un des algorithmes de reconnaissance des graphes triangulés. Pour cela il suffit d'appliquer cet algorithme sur le graphe G, l'appliquer ensuite sur le graphe $\overline{G}$.

Remarque 4.3. Les graphes STR_3 sont des graphes parfaits.

Définition 4.5. STR_4 **:** Un graphe est STR_4 si et seulement s'il ne contient aucun trou ni anti-trou de longueur supérieure à 4.

Remarque 4.4. La classe des graphes STR_4 est exactement celle des graphes faiblement triangulés.

Définition 4.6. STR_5 **:** Un graphe est STR_5 si et seulement s'il ne contient aucun trou ni anti-trou de longueur supérieure à 5.

Remarque 4.5. La classe des graphes STR_5 est une généralisation des graphes faiblement triangulés. En effet, on autorise les cycles sans corde de longueur 5.

Définition 4.7. STR_6 **:** Un graphe est STR_6 si et seulement s'il ne contient aucun cycle sans corde de longueur supérieure à 6.

Il est facile de voir que la suite des inclusions suivante est vraie :
$STR_2 \subset STR_3 \subset STR_4 \subset STR_5 \subset \cdots \subset STR_k \cdots STR_{(k+1)} \cdots$

4.3 Reconnaissance des graphes STR_k

Dans cette section, nous allons donner deux algorithmes de reconnaissance des graphes STR_k. Leur principe consiste à utiliser des algorithmes qui existent pour le graphe G puis son complémentaire.

4.3.1 Algorithme de reconnaissance

On donne ici un algorithme général pour la reconnaissance des graphes STR_k k étant une valeur fixe.

Spinrad (1991) [73] donna un algorithme de reconnaissance des graphes «k-chordal» en temps $O(n^{k-2}T)$, ce dernier peut être modifié pour déterminer si un graphe non pondéré a une chaine induite de longueur au moins l en un temps $O(n^{l-3}T)$, où T est le temps nécessaire pour vérifier si un graphe à $O(n)$ sommets est transitif.

Pour reconnaitre les graphes STR_k, il suffit de remarquer ce qui suit :

Théorème 4.5. *Un graphe est STR_k si et seulement si G et son complémentaire $\overline{G}$ sont k-chordaux.*

Preuve 8. *Un graphe est STR_k s'il ne contient pas de cycles ni d'anti-cycle sans corde de longueur supérieure à k. On peut le définir donc de la manière suivante :*
Un graphe G est STR_k si si et seulement si G et $\overline{G}$ ne contiennent pas de cycle sans corde de longueur supérieure à k. D'où et d'après la définition des graphes k-chordaux, on a
Un graphe G est STR_k si si et seulement si G et $\overline{G}$ sont tous les deux k-chordal.

Pour reconnaitre qu'un graphe est STR_k il suffit donc d'appliquer l'algorithme de Spinrad ou celui de Hayward sur le graphe G pour reconnaitre qu'il ne contient pas de cycle sans corde de longueur supérieure à k, l'appliquer ensuite sur le graphe $\overline{G}$ pour vérifier que G ne contient pas d'anti-cycle de longueur supérieure à k.
On présente ici le principe de l'algorithme de reconnaissance des graphes STR_k.

Théorème 4.6. *Un graphe G est STR_k si et seulement si G et $\overline{G}$ sont k-chordal.*

L'algorithme 1
1 : utiliser l'algorithme "k-chordal" pour tester la k-cordalité du graphe G.
2 : utiliser l'algorithme "k-chordal" pour tester la k-cordalité du graphe $\overline{G}$.
3 : si G et $\overline{G}$ sont tous les deux k-chordaux ; écrire "G est STR_k".
4 : sinon écrire "G n'est pas STR_k".
L'algorithme 2
1 : utiliser l'algorithme de Spinrad pour tester la k-chordalité du graphe G.
2 : utiliser l'algorithme de Spinrad pour tester la k-chordalité du graphe $\overline{G}$.
3 : si G et $\overline{G}$ sont tous les deux k-chordaux ; écrire "G est STR_k".
4 : sinon écrire "G n'est pas STR_k".
Complexité
La complexité du premier algorithme est celle de l'algorithme de Hayward pour la reconnaissance des graphes k-chordaux, il est donc de comlpexité $O(n^{k+1})$. Pour l'algorithme 2, il est de complexité $O(n^{k-2}T)$. Pour les graphes 5-chordaux, la complexité de l'algorithme de reconnaissance des graphes STR_5 est $O(n^3T)$.

Remarque 4.6. Pour les graphes STR_3, on utilise les algorithmes de reconnaissance des graphes triangulés. On applique l'un de ces algorithmes sur le graphe G puis sur le graphe $\overline{G}$. On peut donc reconnaitre les graphes STR_3 en temps $O(n+m)$

Chapitre 5

Partie programmation

Sommaire

5.1 Le langage de programmation Matlab 61
5.2 Algorithme de reconnaissance des graphes triangulés 62
5.3 Algorithme de reconnaissance des graphes faiblement triangulés 63
5.4 Algorithme de reconnaissance des graphes STR_k 63

Introduction

Nous avons jugé utile de programmer quelques algorithmes parmi ceux cités dans les chapitres précédents. Pour plus de clarté, nous avons préféré mettre la partie programmation à part et de lui consacrer ce chapitre. On a choisi quelques algorithmes vus précédemment et nous les avons implémentés en utilisant le langage de programmation Matlab. Le choix du langage est justifié dans la section qui suit.
Les classes de graphes pour lesquels nous avons programmé des algorithmes sont les graphes triangulés, faiblement triangulés et les graphes STR_k. Nous avons programmé l'un des algorithmes de reconnaissance connus dans la littérature pour les deux premières classes ainsi que notre algorithme de reconnaissance des graphes STR_k.

5.1 Le langage de programmation Matlab

Il est indispensable de donner quelques caractéristiques du langage de programmation utilisé et qui est ici le langage Matlab qu'on trouve souvent lorsque on tente de programmer des algorithmes concernant la théorie des graphes.
Matlab pour " MATtrix LABoratory ", est un logiciel qui a été conçu pour fournir un environnement de calcul numérique de haut niveau. Il est particulièrement performant pour le calcul matriciel car sa structure de données interne est basée sur les matrices. Il dispose également de grandes capacités graphiques pour, par exemple, la visualisation d'objets mathématiques complexes. Son fonctionnement repose sur un langage de programmation interprété qui permet un développement très rapide.

Matlab est un logiciel de calcul numérique produit par MathWorks. Il est disponible sur plusieurs plateformes. Matlab est un langage simple et très efficace, optimisé pour le traitement des matrices, d'où son nom. Pour le calcul numérique, Matlab est beaucoup plus concis que les "vieux" langages (C, Pascal, Fortran, Basic). Un exemple : plus besoin de programmer des boucles modifier pour un à un les éléments d'une matrice. On peut traiter la matrice comme une simple variable. Matlab contient également une interface graphique puissante, ainsi qu'une grande variété d'algorithmes scientifiques.
On peut enrichir Matlab en ajoutant des "boîtes à outils" (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières (traitement de signaux, analyses statistiques, optimisation, etc.).
L'Université de Genève dispose d'une centaine de licences Matlab qui sont à la disposition de la communauté universitaire. Il suffit d'être connecté au réseau de l'université pour pouvoir l'utiliser. Plusieurs toolbox sont aussi disponibles, leur nombre dépend de l'environnement, il est parfois plus restreint.

Matlab est l'outil de référence pour la simulation numérique, notamment en ce qui concerne l'Automatique. Il offre des possibilités avancées que ce soit en matière d'identification ou de commande. Il permet, de manière plus générale, de résoudre une grande diversité de problèmes de simulation, dans des domaines aussi variés que le traitement du signal, les statistiques ou la vision, pour ne citer que quelques exemples. L'apprentissage de Matlab se fera en s'appuyant sur l'étude d'un moteur à courant continu.
Avec Matlab les calculs sont numériques (une variable doit avoir une valeur) et basés sur la manipulation de scalaires, de vecteurs et de matrices.

5.2 Algorithme de reconnaissance des graphes triangulés

L'algorithme que nous avons choisi pour cette classe est l'algorithme de D.R. Fulkerson and O.A Gross [27] dont le principe consiste à tester l'existence d'un ordre d'élimination simplicial.

Algorithme de Fulkerson et Gross (1965) [27]

1. Poser G'=G ;
2. Si G' est vide alors G est triangulé : STOP
3. Si G' ne contient pas de sommet simplicial alors G n'est pas triangulé.
4. Ôter un sommet simplicial de G' et retourner à (2).

L'algorithme consiste à tester l'existence d'un sommet simplicial v dans le graphe G considéré. Si un tel sommet existe, on l'élimine, puis on cherche un sommet simplicial dans le graphe G' obtenu après élimination du sommet v. On refait cette étape jusqu'à ne plus trouver de sommet simplicial ce qui signifie que le graphe de départ n'est pas triangulé. Dans le cas d'épuisement des sommets du graphe, le graphe initial est triangulé et l'ordre des sommets obtenu est un ordre d'élimination simplicial.
Le graphe de départ est représenté par une matrice carré qui est la matrice d'adjacence du graphe G. On calcule pour chaque sommet v du graphe son voisinage qu'on a noté par V qui est ici un tableau de taille n. On teste pour chaque sommet si le voisinage V

est une clique. Si V est une clique, le sommet v dont le voisinage est V est simplicial. La boucle est itérée jusqu'à rencontrer un sommet simplicial, dans ce cas, on sort de la boucle pour supprimer le sommet v du graphe. Si un tel sommet n'existe pas on sort de la boucle complètement et on affiche un message "Le graphe G n'est pas triangulé". Si le sommet simplicial existe, sa suppression se fait par un décalage des lignes et colonnes de la matrice A qui représente la matrice d'adjacence du graphe à chaque itération. On décrémente l'indice k d'une unité a chaque fois qu'on supprime un sommet pour gagner du temps et ne pas parcourir les éléments nuls de la matrice A. Si à la fin de la boucle l'indice est nul (la matrice est vide et donc le graphe est vide) on affiche le message suivant "Le graphe G est triangulé".

5.3 Algorithme de reconnaissance des graphes faiblement triangulés

Comme nous l'avons vu, il existe plusieurs algorithmes de reconnaissance des graphes faiblement triangulés. L'algorithme que nous avons choisi [51] se base sur la recherche des deux-paires ou plus exactement des copaires dans un graphe faiblement triangulés et la suppression de ces dernières. Cette suppression n'a aucun effet sur l'appartenance du graphe obtenu à la classe des graphes faiblement triangulés. On refait cette procédure jusqu'à épuisement du graphe ce qui signifie l'obtention d'un stable (dans ce cas le graphe initial est faiblement triangulé) ou non existence d'une copaire à supprimer (ce qui signifie que le graphe initial n'est pas faiblement triangulé).
On explique dans ce qui suit comment calculer la complexité de l'algorithme qu'on a programmé.
Calcul des co-paires de G {O(mn)}
Tant que G contient une co-paire {x,y} **faire** {Boucle itérée m fois}
Suppression des co-paires de la forme {a,b} avec a dans la composantes co-connexe de x dans G ($N_{\bar{G}}(x) \cap N_{\bar{G}}(y)$) et b dans celle de y (algorithme 3) {O(m)}
Suppression de l'arête xy du graphe {O(1)}
Calcul des co-paires d'extrémité x {O(m)}
Calcul des co-paires d'extrémité y {O(m)}
Fin tant que
Si le graphe obtenu est un stable **alors** Retourner "G est faiblement triangulé"
Sinon Retourner "G n'est pas faiblement triangulé" **Fin si**
Cet algorithme permet donc de reconnaître les graphes faiblement triangulés en temps $O(m^2)$

5.4 Algorithme de reconnaissance des graphes STR_k

Cette section est consacrée à la programmation de l'algorithme de reconnaissance des graphes k-super-triangulés, k étant un entier fixe.
On a donné au chapitre 4 l'algorithme de reconnaissance des graphes STRk. Cet algorithme se base sur l'algorithme de Hayward [40] ou celui de Spinrad [73]. Nous avons choisi de programmer celui de Hayward vu qu'il est plus facile à implémenter.

Le principe de l'algorithme consiste à chercher toutes les chaines de longueur k-2, pour toute chaine P, éliminer tous les sommets de la chaine et leurs voisins sauf les extrémités x et y de la chaine. Tester ensuite s'il existe dans le graphe obtenu une chaine entre x et y. S'il existe une chaine entre x et y, alors la chaine P peut être complétée pour donner un trou de longueur $\geq$ k et le graphe G n'est pas k-chordal et donc n'est pas STRk. Sinon tester une autre chaine P'. Après avoir testé toutes les chaines du graphe, refaire cette procédure sur le graphe $\overline{G}$.
La complexité de notre algorithme est $O(n^{k+1})$.
Nous rappelons l'algorithme de reconnaissance des graphes "k-chordal".
Algorithme k-chordal : Reconnaissance des graphes "k-chordal" :
1 : Déterminer toutes les chaines possibles de longueur k-2 qu'on peut avoir des sommets du graphe G.
2 : Vérifier si les chaines déterminées dans 1 existent dans le graphe G.
3 : Choisir une chaine P d'extrémités x et y non encore testé (parmi celles qui existent dans le graphe G).
4 : Éliminer les sommets de la chaine P sauf x et y.
5 : Éliminer les voisins des sommets de la chaine P sauf x et y.
6 : Chercher dans le graphe obtenu une chaine entre x et y.
7 : S'il existe une chaine entre x et y dans G' alors il existe un trou de longueur $\geq$ k passant par la chaine P.
8 : Sinon il n'existe pas de trou de longueur $\geq$ k passant par P, revenir à 3.

5.4.1 Implémentation de l'algorithme de reconnaissance des graphes STR_k

Comme nous l'avons vu, le principe de l'algorithme que nous avons implémenté est le suivant :
1 : Mettre G'=G.
2 : Générer toutes les chaines possibles de longueur k-1 ; 3 : Vérifier pour toute chaines obtenues si elles existent dans le graphe G'.
4 : Pour une chaine qui existe dans G' :
5 : Elimine les sommets de la chaine ainsi que leurs voisins sauf les deux extrémités de la chaine x et y.
6 : S'il existe une chaine entre les deux sommets x et y (les deux sommets sont dans la même composante connexe) , alors il existe un trou de longueur supérieure à k passant par cette chaine. On arrête, le graphe G' n'est pas k-chordal. G n'est pas donc STR_k.
7 : Sinon, il n'existe pas de trou de longueur supérieure ou égale à k passant par cette chaine, on passe à une autre chaine.
8 : Poser G'=$\overline{G}$ et aller à 3. 9 : Si à la fin de la procédure on ne trouve pas de trou de longueur supérieure à k (ni dans G ni dans $\overline{G}$), on dit que le graphe G ne contient pas de trous ni d'anti-trou de longueur supérieure à k, et donc il est STR_k.

La recherche des chaines de longueur k-1 :
Pour la recherche des chaines de longueur k-1, on a utilisé deux procédures ; l'une pour

la recherche des combinaisons possibles de k-1 parmi n, n étant le nombre de sommet du graphe G, l'autre pour la recherche de toutes les permutations possibles de k-1 éléments. pour la deuxième procédure, on utilise les combinaisons générées par la première procédure. Pour toute combinaison obtenue par la première procédure, on génère toutes les permutations possibles.

Vérification de l'existence des chaines générées : Pour toute chaine $P = v_1, \ldots, v_{k-1}$ de longueur k-1 générée par la procédure précédente, on teste l'existence des arêtes (v_i, v_{i+1}) i=1,k-2. L'existence de ces arêtes veut dire que la chaine P existe dans le graphe G.

Elimination des sommets intérieurs de la chaine et leurs voisins :
Pour l'élimination des sommets intérieurs de la chaine P et leurs voisins nous avons créé une liste de sommets. Cette liste contient tous les sommets du graphe G sauf les sommets intérieurs de la chaine P. Elle sera utilisée pour vérifier l'existence d'une chaine entre les extrémités de P dans le nouveau graphe.

L'existence d'une chaine entre les extrémités de la chaine :
Pour l'existence d'une chaine entre les sommets x et y dans le graphe obtenu après élimination des sommets intérieurs de la chaine et leur voisins, on utilise la procédure suivante : On cherche la composante connexe de x, puis on teste si y appartient à cette composante. Si y est dans la même composante connexe que x, alors x et y sont reliés par une chaine. Sinon x et y ne sont pas reliés par une chaine.

5.4.2 Quelques simulations

Nous avons réalisé quelques simulation sur notre algorithme de reconnaissance des graphes STR_k. Pour les simulations, nous avons généré des matrices binaires carrées symétriques et dont la diagonale est nulle. Ces matrices sont donc des matrices d'adjacence d'un graphe. Nous avons ensuite appliqué notre algorithme sur ces matrices pour différentes valeurs de k.
Les résultats obtenus sont représentés par le tableau suivant :

Le graphe	Nombre de sommets	k	Temps d'exécution	k-cordalité du graphe
G1	7	6	0.0936	oui
G2	7	5	0.0780	non
G3	8	5	0.0624	non
G4	10	7	1.9656	oui
G5	10	6	0.2028	non
G6	11	5	0.1092	non
G7	11	6	0.2964	non
G8	11	7	7.7688	oui
G9	11	7	6.7392	non
G10	13	5	0.0936	non

D'après les simulations on remarque que le temps d'exécution est petit, il augmente lorsqu'on augmente la taille du graphe ou le paramètre k. On remarque aussi que le temps de

reconnaitre qu'un graphe est STR_k est plus grand que celui de décider qu'il ne l'est pas car ceci nécessite la vérification de toutes les chaines du graphes, alors que pour décider qu'un graphe n'est pas STR_k peut demander la vérification d'une seule chaine car il suffit de trouver un trou de longueur supérieure à k pour dire que G n'est pas STR_k.

Conclusion

Dans cette thèse, nous avons vu les classes de graphes triangulés et faiblement triangulés. Nous avons donné les différentes caractéristiques de ces deux classes ainsi que leurs algorithmes de reconnaissance et de coloration les plus connus, nous avons aussi présentés leurs sous-classes et leurs algorithmes de reconnaissance.
Nous avons ensuite introduit une nouvelle classe de graphes qu'on a nommé les graphes STR_k ou bien les graphes k-super-triangulés.
Les graphes STR_k sont les graphes n'ayant ni de trou ni d'anti-trou sans corde de longueur supérieure à k.
Le dernier chapitre est consacré aux graphes k-Super-Triangulés. On y trouve deux algorithmes de reconnaissance dont les complexités en temps sont (n^{k+1}) et $O(n^{k-3}T)$. Ces complexités sont respectivement celle celle des algorithmes de Hayward (1987) et de Spinrad (1991) pour la recherche d'un trou de longueur supérieure à k.
Pour finir, nous avons aussi ajouté un dernire chapitre qu'on a dédié à la programmation de quelques algorithmes de reconnaissance des graphes triangulés, faiblement triangulés et STR_k.
Il sera intéressant d'étudier les autres problèmes sur la classe des graphes STR_k ; la coloration, stable max et couvrement par un nombre min de cliques.
Il sera aussi intéressant d'améliorer la complexité de notre algorithme ou de trouver un autre algorithme de complexité plus petite que celui que nous avons proposé dans ce travail.

Bibliographie

[1] H. Alt, N. Blum, K. Mehlhorn et M. Paul (1991). Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/logn})$. Information Processing Letters 37, pp. 237-240.

[2] G. Ausiello, A. D'Atri, M. Mosscarini, Chordality properties on graphs and minimal conceptual connexions in semantic data models, J. Comput. Syst Sci. 33, 179-202, 1986.

[3] Peter. Bartlett. Undirected Graphical Models : Chordal Graphs, Decomposable Graphs, Junction Trees, and Factorizations. October 2003.

[4] C. Berge, Färbung von Graphen deren sämtliche bzderen ungerade Kreise starr sind, Wiss. Zeitschrift, Mathematisch-Naturwissenschaftliche Reihe, Martin- Luther-Univ. Halle-Wittenberg, 114-115, 1961.

[5] A. Berry, J. P. Bordat, and P. Heggernes, Recognizing weakly triangulated graphs by edge separability, Nordic J. Computing 7, 164-177, 2000.

[6] Anne Berry, Jean Paul Bordat, et Pinar Heggernes. Recognizing weakly triangulated graphs by edge separability. In LNCS 1851, Proceedings of Seventh Scandinavian Workshop on Algorithm Theory, pages 139–149, 2000.

[7] Sandrine Blazy, Benoît Robillard et Éric Soutif. Vérification formelle d'un algorithme d'allocation de registres par coloration de graphe inria-00202713, version 1 - 7 Jan 2008

[8] Kellogg S. Booth and George S. Lueker. Linear algorithms to recognize interval graphs and test for the consecutive ones property. In Proceedings of the seventh Annual ACM Symposium on Theory of Computing (STOC'75), pages 255-265, 1975.

[9] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Science, 13 :335-379, 1976.

[10] A. Brandstädt, V.B. Le et J.P. Spinrad, Graph Classes : A Survey, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[11] N. Chandrasekharar, V.S. Lakshmanan, et M. Medidi, Efficiant parallel algorithms for finding chordless cycles in graphs, Parallel Process. Letters 3, 165-170,1993.

[12] K.W. Chong, S.D. Nikolopoulos, and L. Palios, An optimal parallel coconnectivity algorithm, Theory of Computing Systems (to appear) ; Technical Report 27- 02, Dept of Computer Science, Univ. of Ioannina, 2002.

[13] M. Chudnovsky and P.D. Seymour, Recognition algorithm for Berge graphs, preprint, 2003.

[14] M. Chudnovsky, N. Robertson, P. Seymour et R. Thomas, The strong perfect graph theorem. Manuscript 2002

[15] V. Chvàtal. Star-cutsets and perfect graphs. Journal of Combinatorial Theory, Series B, 39 :189–199, 1985.

[16] V. Chvàtal, I. Rusu, A note on graphs with no large holes, presented at DIMACS Workshop on Perfect Graphs (1993).

[17] A. Cobhan. The intrinsic computational difficulty of function. In Proc. 1964 International Congress for Logic Methodology and Philosophy Science, volume 10, pages 24-30. North Holland, 1964.

[18] T. Cormen, C. Leiserson et R. Rivest (1994). Introduction à l'Algorithmique. Dunod, Paris, France. (édition française)

[19] Derek G. Corneil, Stephan Olariu, and Lorna K. Stewart. The ultimate interval graph recognition algorithm ? In Proceedings of the ninth annual ACM-SIAM Symposium on Discrete algorithms (SODA'98), pages 175-180, 1998.

[20] G. Cornuéjols, X. Liu, and K. Vušković, Perfect graph recognition, Proc. 44th IEEE Symp. on Foundations of Computer Science (FOCS 2003), 2003.

[21] D.G. Corneil, Y. Perl et L. Stewart (1985). A linear recognition algorithm for cographs. SIAM Journal on Computing 4, pp. 926-934.

[22] E. Dahlhaus, J. Gustedt, and R.M. McConnell, Efficient and practical algorithms for sequential modular decomposition, J. Algorithms 41, 360-387, 2001.

[23] Gabriel A. Dirac. On rigid circuit graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 25 :71–76, 1961.

[24] J. Edmond. Paths, trees and flowers. Canad. J. Math., 17 : 449-467, 1964.

[25] T. Feder et R. Motwani (1991). Clique partitions, graph compression and speeding up algorithms. In Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, pp. 123-133. ACM Press, New York, NY.

[26] D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. Pac. J. Math., 15 :835-855, 1965.

[27] Delbert R. Fulkerson and O. A. Gross. Incidence matrices, interval graphs, and seriation in archaeology. Pacific Journal of Mathematics, 28 :565-570, 1969.

[28] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM Journal on Computing 1(2), pp. 180-187, 1972.

[29] M.R. Garey, D.S. Johnson, Computers and Intractability : A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.

[30] F. Gavril, Algorithms for maximum weight induced paths, Information Processing Letters 81 : 203–208, 2002.

[31] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. Journal of Combinatorial Theory Series B, 16 :47-56, 1974.

[32] Paul C. Gilmore and Alan J. Hoffman. A characterization of comparability graphs and of interval graphs. Canadian Journal of Mathematics, 16 :539-548, 1964.

[33] M.C. Golumbic. Algorithmic Graph Theory and Perfect Graphs. Computer Science and Applied Mathematics Series, Academic Press, New York, NY, 1980.

[34] Michel Habib and Marie-Catherine Maurer. On the X-Join decomposition for undirected graphs. Discrete Applied Mathematics, 3 :201-207, 1979.

[35] Michel Habib, Christophe Paul, Ross M. McConnell, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. Theoretical Computer Science, 234 :59-84, 2000.

[36] A. Hajnal, J. Surányi - Uber die Auflösung von Graphen in vollständige Teilgraphen, Ann. Univ. Sci. Budapest Eötvös, Sect. Math. 1 (1958), 113–121.

[37] R. Hayward. Generating weakly triangulated graphs. J. Graph Theory, 21 :67–70, 1996.

[38] R. Hayward. Meyniel weakly triangulated graphs - 1 : co-perfect orderability. Discrete Applied Mathematics, 73 :199–210, 1997.

[39] R. Hayward. Meyniel weakly triangulated graphs II : A Theorem of Dirac. 1997.

[40] R.B. Hayward. Two classes of perfect graphs, PhD Thesis, School of Computer Science, McGill Univ., 1987.

[41] R. Hayward. Weakly triangulated graphs. J. Comb. Theory, 39 :200–208, 1985.

[42] R.B.Hayward, C. Hoàng,et F. Maffray, Optimizing weakly triangulated graphs, Graphs, J Combin. Theory B, 5 :339-349, 1989.

[43] R. Hayward, C.T. Hoàng, and F. Maffray. Optimizing weakly triangulated graphs. Graphs and Combinatorics, 5 :339–349, 1989. See also Erratum in Volume 6 (1990), 33–35.

[44] R.B. Hayward, J. Spinrad, and R. Sritharan, Weakly chordal graph algorithms via handles, Proc. 11th ACM-SIAM Symp. on Discrete Algorithms (SODA 2000), 42-49, 2000.

[45] C. Hoàng, et F. Maffray. Weakly triangulated graphs are quasi-parity. Technical Report Rutcor Research Report 6-86, Rutgers University, 1986.

[46] J.E. Hopcroft et R.M. Karp (1973). An $O(n^{5/2})$ algorithm for maximum matching in bipartite graphs. SIAM Journal on Computing 2, pp. 225-231.

[47] E. Howorka, A characterisation of distance-hereditary graphs, Quart. J. Math. Oxford, Ser. 2, 28 (1977), 417-420.

[48] Wen-Lian Hsu. A simple test for interval graphs. In Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'92), volume 657 of Lecture Notes in Computer Science, pages 11-16. Springer-Verlag, 1993.

[49] Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. SIAM Journal on Computing, 28(3) :1004-1020, 1999.

[50] Wen-Lian Hsu and Tze-Heng Ma. Substitution decomposition on chordal graphs and applications. In Proceedings of the second International Symposium on Algorithms (ISA'91), volume 557 of Lecture Notes in Computer Science, pages 52-60. Springer-Verlag, 1991.

[51] Samuel HYM, Ioan. TODINCA et Henri.THUILLIER. Reconnaissance des graphes faiblement triangulés. Rapport de stage MIMI École Normale Supérieure de Lyon. Juillet 2001.

[52] Tetsuya Ishizeki, Yota Otachi, Koichi Yamazaki. An improved algorithm for the longest induced path problem on k-chordal graphs. Discrete Applied Mathematics 156 : 3057–3059, 2008.

[53] H. Ito and M. Yokoyama, Linear time algorithms for graph search and connectivity determination on complement graphs, Inform. Process. Letters 66, 209-213, 1998.

[54] Norbert Korte and Rolf H. Möhring. A simple linear-time algorithm to recognize interval graphs. In Proceedings of the twelfth International Workshop on Graph-Theoretic Concepts in Computer Science (WG'86), volume 246 of Lecture Notes in Computer Science, pages 1-16, 1986.

[55] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. SIAM Journal on Computing, 18(1) :68-81, 1989.

[56] D. Kratsch. The structure of graphs and the design of efficient algorithms. Habilitation thesis, Friedrich-Schiller Universität, Jena, Germany, 1995.

[57] D. Kratsch, H. Müller, I. Todinca, Feedback vertex set and longest induced path on AT-free graphs, in : 29th Workshop on Graph Theoretic Concepts in Computer Science, WG 2003, in : Lecture Notes in Computer Science, vol. 2880, 2003, pp. 309–321.

[58] C. Lekkerkerker and J.C. Boland. Representation of a finite graph by a set of intervals on the real line. Fund. Math., 51 :45–64, 1962.

[59] L. Lovásk. A characterisation of perfect graphs.J. Combin. Theory. Ser B, 13 : 95-98, 1972.

[60] L. Lovásk. Normal hypergraphs and the perfect graphs conjecture .Discrete Math., 2 : 235-267, 1972.

[61] Didier Maquin. Éléments de Théorie des Graphes INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE Ecole Nationale Supérieure d'Electricité et de Mécanique. Version provisoire du 3 mai 2003.

[62] R.M. McConnel et J. Spinrad (1997). Linear-time transitive orientation. In Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 19-25. ACM Press, New York, NY.

[63] H. Meyniel. A new property of critical imperfect graphs and some consequences. European Journal of Combinatorics, 8 :313–316, 1987.

[64] Didier Müller. Introduction à la théorie des graphes. www.apprendre-en-ligne.net/graphes. 2008.

[65] Stavros D. Nikolopoulos Leonidas Palios. Hole and Antihole Detection in Graphs, Proc. 15th ACM-SIAM Symp. on Discrete algorithms (SODA'04).

[66] Stephan Olariu. An optimal greedy heuristic to color interval graphs. Information Processing Letters, 37(1) :21-25, 1991.

[67] Stephan Olariu. No antitwins in minimal imperfect graphs. J. Comb. Theory,B45, 255-257, 1988.

[68] Ganesan Ramalingam et Chandrasekaran Pandu Rangan. New sequential and parallel algorithms for interval graph recognition. Information Processing Letters, 34(4) :215-219, 1990.

[69] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. SIAM Journal on Computing, 5(2) :266-283, 1976.

[70] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput., 5(2) :266–283, 1976.

[71] D. Seinsche. On a property of the class of n-colorable graphs. Journal of Combinatorial Theory Series B 16, pp. 191-193, 1974.

[72] Klaus Simon. A new simple linear algorithm to recognize interval graphs. In Proceedings of the International Workshop on Computational Geometry (CG'91), volume 553 of Lecture Notes in Computer Science, pages 289-308. Springer-Verlag, 1992.

[73] Jeremy P. Spinrad. Finding large holes. Information Processing Letters, 39(4) : 227-229. 1991.

[74] J.P. Spinrad. On comparability and permutation graphs. SIAM Journal on Computing 14, pp. 658-670. 1987.

[75] J. Spinrad et R. Sritharan. Algorithms for weakly triangulated graphs. Discrete Applied Mathematics, 59 :181–191, 1995.

[76] R. Hayward, J. Spinrad, et R. Sritharan. Weakly chordal graph algorithms via handles. In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2000).

[77] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput., 13(3) :566–579, 1984.

[78] Nicolas Trotignon. Graphes parfaits structure et algorithmes. Thèse de Doctorat. Université Grenoble I, Joseph Fourier, 2004.

[79] A. Tucker. Coloring graphs with Stable Cutsets, J. Comb. Theory,B34, 258-267.