N° D'ORDRE : 23/2011-M/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET

DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE

« HOUARI BOUMEDIENE »

FACULTE DE MATHEMATIQUES

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

EN : MATHEMATIQUES

Spécialité : Recherche Opérationnelle : Génie Mathématique

Par : KESSOURI ALI

Sujet :

Sur Quelques Problèmes d'Optimisation Combinatoire d'une Classe de Graphes

Soutenu le 24/04/2011, devant le jury composé de :

Mr-	A. KHELLADI	Professeur	à l'U.S.T.H.B	Président
Mr-	H.Ait HADDADENE	Professeur	à l'U.S.T.H.B	Directeur de Thèse
Mr-	A. SEMRI	Maître de Conférences/A	à l'U.S.T.H.B	Examinateur
Mr-	M. YAGOUNI	Maître Assistant/A	à l'U.S.T.H.B	Invité

Remerciements

Je tiens à exprimer ici toute ma reconnaissance au monsieur le Professeur H. Ait Haddadène pour l'honneur qu'il m'a fait en assurant la direction et le suivi scientifique et technique du présent mémoire. Je le remercie pour sa grande contribution à l'aboutissement de ce travail, et pour sa disponibilité malgré ses nombreuses activités.

Je remercie vivement le Professeur A. Khelladi pour l'honneur qu'il me fait en acceptant de présider le jury de ce mémoire.

Mes remerciements chaleureux s'adressent également à monsieur A. Semri et à monsieur M. Yagouni pour avoir accepté d'examiner ce travail.

Sans oublier de remercier ma femme pour son soutien et ses encouragements.

Dédicaces

$\mathcal{J}$e dédie ce modeste travail à :

* $\mathcal{M}$on épouse ;

* $\mathcal{M}$on fils Mohamed ;

* $\mathcal{M}$es parents ;

* $\mathcal{M}$on frère Chaabane, sa femme et leurs fille Ahlem ;

* $\mathcal{M}$es soeurs ;

Ali Kessouri.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

Introduction

Introduction

De nombreux problèmes d'optimisation combinatoires issus du monde industriel (par exemple, la production, telecommunication, transport,...) peuvent être modélisés par des graphes. De même pour certains problèmes de confection d'horaire ou d'ordonnancement qui se modélisent par la coloration des sommets d'un graphe.

La coloration des sommets d'un graphe consiste à attribuer une couleur à chaque sommet de tel sorte que deux sommets adjacents ne reçoivent pas la même couleur. Le problème de la coloration optimale des sommets d'un graphe quelconque est un problème connu dans les problèmes d'optimisation combinatoire comme étant un problème NP-difficile. La solution optimale de celui-ci est restreinte aux classes des graphes parfaits et aux graphes de taille réduite.

Sur des problèmes de grandes tailles, l'utilisation d'une méthode exacte s'avère inutile faute de temps de calcul et d'espace mémoire, sur ceux, des heuristiques ont vu le jour, elles donnent des solutions approchées de la solution optimale.

Dans le cadre de ce mémoire, nous nous intéressons à la coloration d'une classe de graphes où chaque sommet est scindé c'est la classe des graphes localement scindés, cette classe n'appartient pas aux graphes parfaits.

Dans le premier chapitre, nous introduisons les notions de base sur la théorie des graphes en général et sur les graphes parfaits en particulier, ensuite, nous présentons quelques classes des graphes parfaits.

Dans le chapitre 2, nous nous intéressons aux problèmes de coloration et nous présentons les différentes heuristiques appliquées à la coloration des sommets d'un graphe.

La première partie du troisième chapitre sera consacré à la reconnaissance d'un graphe scindé et à la reconnaissance d'un graphe localement scindé, la deuxième partie consiste à décrire la recherche taboue adaptée à la coloration des graphes localement scindés ensuite, nous présenterons l'algorithme Dsatur et enfin nous comparerons les résultats donnés par les deux méthodes appliquées sur des instances tirées de DIMACS.

Généralités sur les Graphes

CHAPITRE 1

Généralités sur Les Graphes

Dans ce chapitre, nous allons donner quelques notions de bases sur la théorie des graphes, nous présentons les graphes parfaits et la validité de la conjecture forte des graphes parfaits et enfin, nous essayons de décrire quelques classes de ces graphes.

1.1 Notations et définitions

Un graphe $G = (X, E)$ non orienté est constitué de deux ensembles finis d'éléments non vides $X = \{x_1, ..., x_n\}$ appelé ensemble des sommets et $E = \{e_1, ..., e_m\}$ appelé ensemble d'arêtes. Deux sommets x et y sont dits adjacents s'ils sont liés par une arête.

Un graphe est dit orienté si tout couple de sommets adjacents possèdent une extrémité initiale et une extrémité terminale. On parlera alors, d'arc.

Un graphe est dit simple s'il est sans boucle et tout couple de sommets sont reliés par une et une seule arête.

Deux arêtes sont adjacentes si elles ont une extrémité commune.

Une chaîne (resp.un chemin dans le cas d'un graphe orienté) est une suite (non vide) finie d'arêtes(resp.d'arcs) adjacentes.

Un cycle(resp.un circuit) est une chaîne(resp.un chemin) dont les deux extrémités sont confondues, ils sont élémentaires s'ils n'utilisent pas plus d'une fois le même sommet.

Un voisin d'un sommet x est un sommet y dont il est adjacent (c'est à dire il existe une arête (arc) qui relie ces deux sommets.
Dans le cas des graphes orientés, x a pour successeur (resp. prédécesseur) y si l'arc $(\overrightarrow{xy})$ (resp. $(\overrightarrow{yx})$) existe.

L'ensemble de tous les voisins d'un sommet x d'un graphe G=(X,E) est appelé le voisinage de x et sera noté N(x).

Le degré d'un sommet x, noté d(x) est égale au nombre d'arêtes adjacentes à x.

Dans le cas des graphes orientés, le degré d'un sommet est égale à la somme de demi-degré extérieur et intérieur sachant que le demi-degré extérieur d'un sommet est le nombre d'arcs dont il est une extrémité initiale et le demi-degré intérieur d'un sommet est le nombre d'arcs dont il est une extrémité finale.

Un graphe est dit connexe si quelque soient les sommets x et y distincts, il existe un chaîne joignant x à y.

Une corde est une arête qui relie deux sommets non consécutifs d'une chaîne ou d'un cycle.

Un cycle élémentaire sans corde de longueur au moins quatre (04) est appelé trou son complémentaire est un anti trou.

Un graphe est complet si chaque sommet du graphe est relié directement à tous les autres sommets.

Un graphe $\grave{G} = (X, \grave{E})$ est un graphe partiel de G=(X,E), si $\grave{E}$ est inclus dans E.
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 deux extrémités dans A.
Un graphe partiel d'un sous-graphe est un sous-graphe partiel de G.

Une clique est un ensemble de sommets dont chaque sommet est adjacent à tous les autres, c'est à dire, une clique est un graphe d'ordre n, on le note K_n et il possède $\frac{n(n-1)}{2}$ arêtes. On note $\omega(G)$ la taille maximale d'une clique de G.

Un stable de cardinalité n, est un graphe dont ces sommets sont deux à deux non adjacents, on note $\alpha(G)$ la taille maximale d'un stable de G.

Matrice d'adjacence

Soit G=(X,E) un graphe simple, la matrice booléenne est une matrice A à coefficients égaux à 0 ou 1, où chaque ligne correspond à un sommet de G et où chaque colonne correspond à un sommet de G.

$$A = (a_{i,j}), (i,j) \in [1,n]^2$$

$$a_{(i,j)} = \begin{cases} 1, & \text{si le sommet i est adjacent au sommet j ;} \\ 0, & \text{sinon.} \end{cases}$$

1.1.1 Coloration des sommets d'un graphe

Définition 1.1.1. Une k-coloration des sommets d'un graphe G=(X,E) est une application qui est définit comme suit :

$$c : X \longrightarrow \{1,2,...,k\} \ \ tel \ \ que \ \ \forall x,y \in X, xy \in E \Longrightarrow c(x) \neq c(y)$$

Autrement dit, une coloration de G=(X,E) est une partition de X en stables.

le nombre chromatique de G est le plus petit nombre de couleurs nécessaire pour colorier le graphe qui est noté $\gamma(G)$.

1.1.2 Notions sur la théorie de la complexité algorithmique

La complexité algorithmique permet de déterminer mathématiquement si le problème étudié est considéré comme "facile" ou "difficile" à résoudre ainsi, la complexité cherche à classifier les problèmes en fonction de leurs difficulté.

Un algorithme est une procédure formée d'instructions élémentaires qui pour toute instance d'un problème donné transforme les entrées (données) en sorties (résultats).

Les problèmes habituellement considérés dans la théorie de la complexité sont les problèmes de décision et les problèmes d'optimisation. Les problèmes de décision sont des problèmes pour lesquels on cherche à répondre à une question par oui ou par non. Dans les problèmes d'optimisation on cherche à trouver une solution au problème considéré.

Pour un algorithme, il existe deux types de complexité : la complexité temporelle qui renseigne sur le temps de calcul nécessaire pour obtenir une solution optimale et la complexité spatiale qui renseigne sur l'espace mémoire nécessaire.

Un algorithme est dit polynomial si son temps d'exécution est borné par une fonction polynômiale de la taille de l'instance. On dit qu'il a une complexité 0(p(n)), où p est le polynôme et n est la taille de l'instance.

Les problèmes appartenant à la classe $\mathcal{P}$ sont des problèmes pouvant être résolus en temps polynomial. Les problèmes polynomiaux sont considérés comme étant des problèmes "facile". La classe $\mathcal{NP}$ contient les problèmes pour lesquels, il existe un algorithme non déterministe polynomial pour les résoudre.

La classe NP-complet est une sous-classe de NP. Un problème est NP-complet si tous les problèmes de NP se réduisent polynomialement à lui.

Remarque 1.1.1. Les problèmes du stable maximum, clique maximum, coloration minimum et la partition minimale en cliques sont des problèmes d'optimisation NP-complet.

1.2 Les graphes parfaits

Claude Berge a été le premier qui a introduit la notion des graphes parfaits au début des années 60 [5], mais la véritable origine revient aux travaux de Shannon [56] en théorie de l'information sur les codages optimaux et la capacité d'un canal de communication. Depuis, nombreux ceux qui travaillent sur ce type de graphe ainsi, de nombreuses classes sont incluses dans la classe des graphes parfaits. Elle permet de généraliser un certain nombre de résultats connus sur ces classes particulières.

Pour un graphe G=(X,E), nous désignons par :

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

Les notions $\alpha - parfait$ et $\gamma - parfait$ revient à Claude Berge[5, 4].

Définition 1.2.1. Un graphe G est dit $\gamma - parfait$ (respectivement $\alpha - parfait$) si pour tout sous graphe induit H de G, $\gamma(H) = \omega(H)$ (respectivement $\theta(H) = \alpha(H)$).

Claude Berge proposa alors deux conjectures, la conjecture faible des graphes parfaits et la conjecture forte des graphes parfaits.

Théorème 1.2.1 (Conjecture forte des graphes parfaits)**.** *Un graphe G est $\alpha-parfait$ (respectivement $\gamma - parfait$) si et seulement s'il ne contient ni trou impair, ni anti trou impair.*

Théorème 1.2.2 (Conjecture faible des graphes parfaits)**.** *Un graphe G est $\alpha - parfait$ si et seulement s'il est $\gamma - parfait$.*

La conjecture faible des graphes parfaits a été démontrée par Laslo Lovasz en 1972 [45, 46].

Théorème 1.2.3 (Théorèmes des graphes parfaits [45])**.** *Soit un graphe G=(X,E), les assertions suivantes sont équivalente :*

- G *est* $\alpha - parfait$
- G *est* $\gamma - parfait$
- *pour tout sous-graphe induit* H *et* G, $\omega(G)\alpha(G) \geq |X(H)|$

Les notions de graphe $\alpha - parfait$ et de graphe $\gamma - parfait$ sont remplacées depuis par la notion de graphe parfait.

Corollaire 1.2.4. *Un graphe est parfait si et seulement si son complémentaire est parfait.*

Définition 1.2.2 (Trou)**.** Un trou est un cycle sans corde de longueur au moins quatre, et l'anti-trou est son complémentaire.

(Une corde est une arête qui relie deux sommets non consécutif de ce cycle).

Définition 1.2.3. On appelle graphe de Berge un graphe n'ayant ni trou impair, ni anti-trou impair.

La conjecture fortes des graphes parfaits se résume en "Tout graphe de Berge est parfait."

1.2.1 Validité de la conjecture forte des graphes parfaits

La conjecture forte des graphes parfaits a fait l'objet de nombreuses recherches et travaux en théorie des graphes, sa validité a été finalement prouvée en 2002 par les quatres chercheurs M.CHUDNOVSKY, N.ROBERTSON, P.D.SEYMOUR et R.THOMAS [14]. Afin de résoudre cette conjecture, les travaux ont pris deux directions.

La première direction consiste à démontrer la conjecture des graphes parfaits pour des classes particulières des graphes, il existe deux classes, la classe excluant implicitement les trous impairs et les anti-trous impairs et la classe les excluant explicitement.

La seconde direction consiste à étudier la structure des graphes imparfaits critiques (imparfaits minimaux).

Définition 1.2.4. Un graphe G est imparfait critique si seulement si G n'est pas parfait et tous sous graphe propre de G induit un graphe parfait.

La conjecture de Berge est équivalente à
"Tout graphe imparfait critique est soit un trou impair, soit un anti-trou impair".

Pendant 42 ans, la conjecture forte des graphes parfaits a suscité un très grand intérêt, jusqu'à sa résolution en Mai 2002 par Maria Chudnovsky, Neil Robertson, Paul Seymour et Robin Thomas. Nous allons présenté les idées principales de la démonstration de ce théorème.

Définition 1.2.5. Un graphe G a un 2-joint si ses sommets peuvent se partitionner en deux ensembles X_1 et X_2, chacun de cardinalité au moins trois, contenant des sous-ensembles non vides disjoins $A_1, B_1 \subseteq X_1$ et $A_2, B_2 \subseteq X_2$, tels que tous les sommets de A_1 sont adjacents à tous les sommets de A_2, tous les sommets de B_1 sont adjacents à tous les sommets de B_2 et ces adjacences sont les seules entre X_1 et X_2. Les 2-joint ont été introduit en 1985 par Cornuéjols et Cunningham [11].

Théorème 1.2.5 ([11])**.** *Si un imparfait critique contient un 2-join, alors c'est un trou impair.*

Définition 1.2.6. Une anti-chaîne est un sous graphe dont le complémentaire est une chaîne, sa longueur c'est le nombre d'arêtes constituant le complémentaire de la chaîne ;

Définition 1.2.7. $\grave{X} \subset X$ est connexe si $G \backslash \grave{X}$ est connexe ou si $\grave{X}$ est vide.

Définition 1.2.8. $\grave{X} \subset X$ est anti-connexe si $\overline{G} \backslash \grave{X}$ est connexe.

Définition 1.2.9. Une partition oblique dans G est une partition A,B de X tel que A n'est pas connexe et B n'est pas anti-connexe.

Définition 1.2.10. Soient A,B deux sous-ensembles disjoints de X(G).
On dit que la paire (A,B) est équilibrée, s'il n'existe pas de chaîne impaire entre des sommets non adjacents dans B avec intérieur de la chaîne dans A, et il n'existe pas d'anti-chaîne impaire entre des sommets adjacents dans A avec intérieur de la chaîne dans B.

Définition 1.2.11. Un graphe G est dit de base si G ou $\overline{G}$ est biparti ou graphe adjoint d'un biparti ou graphe scindé double.

Définition 1.2.12. Si $A, B \subset X$ sont disjoints.

- La paire (A,B) est complète si tout sommet dans A est adjacent à tout sommet dans B ;
- A est anti-complet à B s'il n'y a pas d'arête entre A et B ;

Définition 1.2.13. Un M-joint dans G=(X,E) est une partition de X en six sous-ensembles non vides {A,B,C,D,E,F} tel que :

- Tout sommet dans A, a un voisin dans B et aucun voisin dans D et vice versa ;
- (D,A), (A,E), (E,B), (B,C) sont anti-complets ;
- (C,A), (A,F), (F,B), (B,D) sont complètes.

Théorème 1.2.6 ([10])**.** *Aucun imparfait critique ne contient un M-joint.*

Définition 1.2.14. Soient m, $n \geq 2$ des entiers, et soient $\{a_1, ..., a_m\}$, $\{b_1, ..., b_m\}$, $\{c_1, ..., c_m\}$ et $\{d_1, ..., d_m\}$ des ensembles disjoints.

On définit un graphe G, tel que l'ensemble de ses sommets, est l'union des sous-ensembles précédents, et les arêtes définies comme suit :

- a_i est adjacent à b_i pour $1 \leq i \leq m$ et c_j est non adjacent à d_j pour $1 \leq j \leq n$
- Il n'existe pas d'arêtes $\{a_i, b_i\}$, $\{a_l, b_l\}$ pour $1 \leq i \leq< l \leq m$ et tout quatre arêtes entre $\{c_j, d_j\}$ et $\{c_h, d_h\}$ pour $1 \leq j < h \leq n$.

Il existe exactement deux arêtes entre $\{a_i, b_i\}$ et $\{c_j, d_j\}$ pour $1 \leq i \leq m$ et $1 \leq j \leq n$ et ces deux arêtes sont disjointes.
Alors, ce graphe est dit bicographe

1.2.2 Quelques classes des graphes parfaits

Les Graphes Bipartis

Les graphes bipartis constituent l'une des classes les plus simples de graphes.

Définition 1.2.15. Un graphe G=(X,E) est dit graphe biparti si on peut partitioner X en X_1 et X_2 de telle sorte qu'une arête ne peut relier qu'un sommet de X_1 à un sommet de X_2 c'est à dire en deux stables.

Un graphe biparti $G = (X_1, X_2, E)$(car on le note comme ça) est dit complet, si tout sommet de X_1 est adjacent à tout les sommets de X_2. Généralement, il est notée $K_{p,q}$ tel que $p = |X_1|$ $q = |X_2|$

Théorème 1.2.7 (Perfection)**.** *Tout graphe biparti est parfait*

La perfection des graphes bipartis est évidente car pour tout sous-graphe induit H de G, $\omega(G) = \gamma(G) = 2$ si H possède au moins une arête et $\omega(G) = \gamma(G) = 1$ sinon.

Les line graphes des graphes bipartis

D.König[43] a montré que les line graphes des graphes bipartis sont parfaits. Le line graphe $\grave{G}$ d'un graphe G est définit comme suit :

- à chaque arête de G correspond un sommet de $\grave{G}$
- deux sommets de $\grave{G}$ sont reliés par une arête si les deux arêtes correspondantes de G sont adjacentes.

Les Graphes de comparabilité

Un graphe est de comparabilité si on peut orienter ses arêtes de façon transitive.

Théorème 1.2.8. *Les graphes de comparabilité sont parfaits [*6 *].*

Les Graphes d'intervalles

On dit que G=(X,E) est un graphe d'intervalles, si à chaque sommet $x \in X$ on peut associer I_x un intervalle de $\mathbb{R}$ tel que :

$$\forall x_1, x_2 \in X; \quad x_1x_2 \in E \quad ssi \quad I_{x_1} \cap I_{x_2} \neq \emptyset$$

Théorème 1.2.9 (Perfection)**.** *Tout graphe d'intervalles est parfait*

La perfection de cette classe se montre en prouvant que tout graphe d'intervalles est triangulé, c'est ce qu'a utilisé Hajös [27]

Les graphes triangulés

Définition 1.2.16. Un graphe est triangulé si tout cycle, de longueur supérieur ou égal à quatre (04), a au moins une corde.

Cette classe de graphe a été introduite par Hadjnal et Suranyi en 1958 [26] avant même l'apparition des graphes parfaits par Claude Berge, les deux chercheurs ont prouvé que le nombre de stabilité d'un graphe triangulé est égal à la taille minimale d'une partition en cliques d'ou la perfection de cette classe.

Les graphes faiblement triangulés

Définition 1.2.17. Un graphe G est dit faiblement triangulé s'il ne contient ni trou ni anti-trou de longueur supérieur ou égal à cinq.

En 1985, R.B. Hayward [32] a montré la perfection de cette classe.

Les graphes fortements parfaits

Définition 1.2.18. Un graphe est dit fortement parfait si et seulement si tout sous graphe induit H de G contient un ensemble stable qui intersecte toutes les cliques maximales de H.

C. Berge et P. Duchet ont montré que tous graphe fortement parfait est parfait [7].

Les graphes préparfaits

Définition 1.2.19. Un graphe est dit préparfait si tout sous-graphe induit possède une paire de sommets dont l'un prédomine l'autre (un sommet x prédomine un autre sommet y si dans G, toute clique maximum contenant y contient aussi x).

En 1993, P.L. Hammer et F. Maffray [28] ont prouvé la perfection de cette classe.

Les graphes de parité

Définition 1.2.20. Un graphe G est de parité si et seulement si tout cycle impair, de longueur supérieur ou égal à cinq, possède au moins deux cordes croisées.

En 1970, H. Sachs [54] a montré que tout graphe de parité est parfait.

Les graphes de Meyniel

Définition 1.2.21. Un graphe G est dit de Meyniel si tout cycle de longueur supérieur ou égal à cinq possède deux cordes.

Il existe un algorithme polynomial de reconnaissance de cette classe élaboré par M. Burlet et J. Fonlupt [9] et la technique de l'échange bichromatique conduit à sa perfection.

Remarque 1.2.10. Cette technique de coloration sera détaillée dans le troisième chapitre.

Les graphes sans P_4

En 1974, D. Seinche [55] a montré la perfection de cette classe.

Les graphes de Berge sans griffe

Définition 1.2.22. Un graphe sans griffe est un graphe qui n'admet pas de $K_{1,3}$ comme sous-graphe induit.

En 1976, K.R. Parthasarathy et G. Ravindra [51] ont montré que tout graphe de Berge sans griffe est parfait.

Les graphes de Berge faiblement sans diamant

Définition 1.2.23. Un sommet x d'un graphe G est dit sans diamant si G[N(x)] est sans diamant et $d_G(x) \leq 2\omega(G) - 1$.
Un graphe G est dit faiblement sans diamant si tout sous-graphe H de G contient un sommet faiblement sans diamant.

H.Ait Haddadène et S.Gravier [1] [2] ont montré la perfection des graphes de Berge faiblement sans diamant.

Les graphes de Berge sans K_4

En 1977, A.C. Tucker [58] [59] a prouvé que tout graphe de Berge sans K_4 est parfait.

Remarque 1.2.11. Les graphes de Berge sans K_4 sont aussi appelé les graphes de Tucker.

Les graphes de Berge sans diamant (sans $K_4 - e$)

Définition 1.2.24. Un graphe sans diamant est un graphe ne contenant pas de sous-graphe isomorphe à $K_4 - e$.

En 1979, K.R. Parthasarathy et G. Ravindra [**?**] ont donné une démonstration sur la validité de cette classe mais elle s'est avéré inexacte, A. Tucker proposa alors une autre démonstration basée sur les résultats suivant :

Théorème 1.2.12 (A.Tucker [60])**.** *Soit G=(V,E) un graphe sans trou impair et sans $K_4 - e$, avec $|V| = n$ et $\omega(G) = \omega$.*
Si toute clique maximale de G est de taille supérieure ou égale à trois (03), alors G contient un sommet appartenant à au plus deux cliques maximales. de plus, $|E| < 2\omega$.

Théorème 1.2.13 (A.Tucker [60])**.** *Tous graphe sans trou impair et sans $K_4 - e$ est parfait.*

Un algorithme polynomial pour la reconnaissance de cette classe a été élaboré par J. Fonlupt et A. Zemirline [21].

Les graphes de Berge sans patte

La classe des graphes sans patte a été introduite par S. Olariu [49].

Définition 1.2.25. Un graphe est dit multiparti si l'ensemble des ces sommets admet une partition en au moins trois stables.

Théorème 1.2.14 ([49])**.** *Un graphe est sans patte si et seulement si toute composante connexe de G est soit sans triangle, soit multiparti complet.*

La perfection de ce type de graphe est une conséquence du ce théorème.

Les graphes de Berge à voisinage scindé

Définition 1.2.26. Un graphe G est dit scindé si l'ensemble des sommets de G admet une partition en deux ensembles S et K où S est un stable de G et K est une clique de G.

Définition 1.2.27. Un sommet x est dit scindé si l'ensemble de ses voisins dans G induit un graphe scindé.

Définition 1.2.28. Un graphe est dit à voisinage scindé si pour tout sous-graphe induit H de G contient un sommet x scindé.

En 1995, F. Maffray et M. Preissmann [47] ont montré que tout graphe de Berge à voisinage scindé est parfait.

Les graphes de Berge dégénérés

Définition 1.2.29. Un graphe G est dit dégénéré si tout sous-graphe induit H de G contient un sommet de degré au plus $\omega(H) + 1$.

La validité de la conjecture des graphes parfaits pour les graphes dégénérés a été prouvée par H. *Aït* Haddadène et F. Maffray et cela en utilisant la technique de l'échange trichromatrique [3].

Problèmes de Coloration

CHAPITRE 2

Problèmes de Coloration

Dans ce chapitre, nous présentons le problème de coloration des graphes ensuite, nous décrivons quelques heuristiques de coloration étant donné que ce problème est NP-difficile.

2.1 Terminologies

Le problème de coloration revient à Francis Guthrie en 1852, il s'agit de répondre à cette question "est-il possible de colorier toute carte géographique avec au plus 4 couleurs de sorte deux régions qui ont une frontière en commun aient des couleurs différentes". Ce problème s'appelle le problème de quatre couleurs. Ce problème est resté plus d'un siècle sans résolution jusqu'au 1976, Appel et Haken ont réussi à le résoudre à l'aide de l'ordinateur.
Depuis, de nombreux problèmes d'optimisation combinatoire se modélisent par la coloration de graphes, ce problème étant NP-complet, ce qui a motivé l'élaboration de nombreuses heuristiques et a suscité beaucoup d'études jusqu'à nos jours.

Définition 2.1.1. Soit G=(X,E) un graphe simple, où X, E représentent respectivement les sommets et les arêtes. Pour un entier k, on définit une k-coloration de G comme une fonction c : $V \rightarrow \{1, ..., k\}$, c(x) représente la couleur du sommet x, généralement les couleurs sont représentées par des nombres entiers. Si deux sommets adjacents x et y ont

la même couleur on dit qu'ils sont en conflit et que l'arête qui relie x à y est une arête conflictuelle. Une k-coloration sans conflit est dite légale ou propre. Le problème de coloration des sommets d'un graphe consiste à déterminer le plus petit entier k (déterminer le nombre chromatique).

La coloration de G revient à trouver une partition des sommets en k stables. C'est à dire, chaque deux sommets adjacents doivent avoir deux couleurs différentes.

Le nombre chromatique $\gamma(G)$ de G est le nombre minimum de couleurs qu'on peut affecter aux sommets de G.
Pour tout graphe G : $\gamma(G) \geq \omega(G)$, $\omega(G)$ étant la taille maximum de cliques de G et

$$\gamma(G) \leq \Delta(G)$$

$\Delta(G)$ étant le degré maximum de G sauf si G est un graphe complet ou un cycle de longueur impair.

Remarque 2.1.1. Dans la littérature, il existe d'autres colorations telle que la coloration des arêtes, la coloration de faces,... etc mais on peut toujours se ramener à la coloration des sommets.

2.2 Techniques de coloration des sommets d'un graphe

2.2.1 La coloration par contraction

Définition 2.2.1. Pour un graphe G et deux sommets non adjacents x et y, nous notons par G-x-y le graphe en supprimant les sommets x et y en rajoutant un nouveau sommet xy adjacent à tous les sommets de G-x-y qui étaient adjacents à au moins un des deux sommets x et y dans G.
Nous dirons que G-x-y est obtenu par contraction de la paire {x,y}.

Soient x et y une paire de sommets non adjacents dans un graphe G, la coloration de G revient simplement à la coloration de G-x-y car on peut affecter à x et à y la couleur du sommet contracté xy.

Définition 2.2.2. Un graphe G est contractile s'il existe une séquence $G_0, G_1, ..., G_k$ de graphes tels que :
$G_0 := G$, G_K est une clique et pour $i \leq k-1$, G_{i+1} est obtenu à partir de G_i par contraction d'une paire de sommets (x,y) de G_i.

Les graphes de Meyniel et les graphes faiblement triangulés sont contractiles.

Des heuristiques de coloration basés sur la méthode de contraction a été utilisées par Dutton et Brigham [19], ils choisissent à chaque étape deux sommets non adjacents tel que le nombre de voisins communs est le plus large possible dans G. Ainsi que par Hertz [34, 35], il fixe un sommet x et tant que x n'a pas de voisin on choisit un sommet non adjacent y de x, qui a le maximum de voisins avec x.
Tucker [61] a utilisé la méthode de contraction pour la coloration des graphes parfaits sans K_4.

2.2.2 Technique de l'échange bichromatique

Pour un graphe G et un sommet v de G, s'il existe une k-coloration de G-v tel que : soit une couleur α manque dans le voisinage $N_G(v)$ de v, soit il existe une paire (α, β) de couleurs telle que tous les sommets de couleur β dans $N_G(v)$ appartiennent à des composantes connexes du sous graphe de G engendrés par les sommets coloriés α, β , ne contenant aucun des sommets de $N_G(v)$ coloriés α (en particulier un sommet colorié β dans $N_G(v)$ n'est pas adjacent à un sommet de $N_G(v)$ colorié α), alors G est k-colorable.

Pour obtenir cette k-coloration, il suffit lorsque toutes les couleurs sont présentes dans $N_G(v)$, d'échanger, sur les composantes connexes, la couleur β avec la couleur α cette opération s'effectue en temps polynomial et s'appelle l'échange bichromatique, elle a été introduite par Meyniel.

.

2.2.3 Technique de l'échange trichromatique

Soit G un graphe de Berge et un sommet x de G, supposons qu'on a une ω-coloration de G-x où ω-coloration sont présentes dans $N_G(x)$. On considère la propriété suivante : il existe trois couleurs i, j et k tel que le sous-graphe induit par $S_i \cup S_j \cup S_k \cup x$ (où $S_i, S_j et S_k$ sont des stables de couleurs i, j et k respectivement) est sans K_4. Si cette propriété est vérifiée, alors on peut obtenir une 3-coloration avec les couleurs i, j et k du sous-graphe $S_i \cup S_j \cup S_k \cup x$ en appliquant l'algorithme de Tucker [57].

Cette méthode a été introduite par H. Ait Haddadène et F. Maffray [3]

2.3 Les heuristiques appliquées à la coloration

De nombreux problèmes réels ont été modélisés par le problème de coloration de graphe. Ce problème étant NP-complet [23, 42], les méthodes exactes s'avèrent non appropriées pour les graphes de grandes tailles bien qu'elles permettent de résoudre certaines classes de graphes, mais cependant, l'énumération de toutes les solutions permet d'avoir la solution optimale ce qui est impossible faute de temps de calcul pour la plupart des problèmes. Dans ces conditions, il vaut mieux adopter des approches heuristiques bien qu'elles ne garantissent pas la solution optimale.

IL existe différentes heuristiques selon les problèmes à traiter, on distingue trois grands types [44], bien que certaines heuristiques sont une combinaison de plusieurs types d'où l'impossibilité de les classées.

- **Les méthodes constructives :** elle construit pas à pas une seule solution à partir d'une solution initiale qui est au départ vide.
- **Recherches locales :** On part d'une solution initiale et on tente de l'améliorer par des transformations successives. Elle s'arête lorsqu'on ne peut plus améliorer la solution courante.
- **Les heuristiques évolutives :** Elle fait intervenir plusieurs solutions ou un ensemble de solutions qui s'opèrent et s'adaptent individuellement.

Dans la suite de ce chapitre, nous allons décrire pour chacune de ces trois catégories les heuristiques les plus connues et utilisées dans la littératures pour résoudre notre problème qui est la coloration des sommets d'un graphe.

2.3.1 Les heuristiques constructives

Les heuristiques constructives construisent la solution finale à partir d'une solution qui est au départ vide en effectuant à chaque étape des choix qui permettent d'aboutir à une solution de bonne qualité c'est à dire que le nombre chromatique soit le plus petit possible.

L'algorithme glouton

L'algorithme glouton est l'algorithme constructif le plus utilisé pour résoudre les problèmes de coloration des sommets d'un graphe, il affecte étape par étape, en suivant un certain ordre, à un sommet la plus petite couleur possible de manière à éviter les conflits.

Afin d'avoir un algorithme glouton de bonne qualité, l'ordre dans lequel les sommets sont parcourus doit être choisit judicieusement, ce dernier peut être statique c'est à dire l'ordre est connu à l'avance, ou dynamique dans ce cas là, le choix du prochain sommet dépend de l'affectation précédente.

Nous présentons ici l'algorithme glouton le plus simple :

`Étape 1` : Classer les sommets dans l'ordre décroissant de leur degré ;

`Étape 2` : En parcourant la liste dans l'ordre, affecter une couleur non encore utilisée au premier sommet non coloré, et affecter cette même couleur à chaque sommet non encore coloré, et non adjacent à un sommet de cette couleur ;

`Étape 3` : S'il reste des sommets non encore colorés dans le graphe, revenir à l'étape 2 ;

2.3.2 Recherches locales

Les méthodes de la recherche locales résolvent le problème de coloration de manière itérative. Elles font évoluer la solution courante en la remplaçant par une autre solution issue de son voisinage. Le voisinage d'une solution s noté N(s) est obtenu en effectuant des modifications sur s c'est ce qu'on appelle le mouvement. *À* partir d'une solution initiale obtenue généralement en appliquant une méthode constructive tel que la méthode gloutonne, une méthode de recherche locale génère des solutions $s_1, s_2, ...$ dans S (S : espace de solution) tel que $s_{i+1} \in N(s_i)$ [40, 50, 31]. Les heuristiques de recherche locale les plus connues sont La descente, le recuit simulé et la recherche tabou [24, 25].

Le recuit simulé

La méthode de recuit simulé s'inspire de recuit physique utilisé en métallurgie pour améliorer la qualité d'un solide. En partant d'une température élevée dont la matière est devenue liquide, la phase de refroidissement conduit la matière à retrouver sa forme solide par diminution progressive de la température. Chaque température T est maintenue jusqu'à ce que la matière trouve son équilibre thermodynamique.

En optimisation combinatoire, le recuit simulé consiste à accepter d'aller vers une solution voisine $\grave{s}$ moins bonne que la solution courante s avec une certaine probabilité $p(s, \grave{s}, T)$, cette probabilité dépend de l'écart entre s et $\grave{s}$. Le paramètre T décroît d'une itération à l'autre, si $f(\grave{s}) > f(s)$ alors on pose $\grave{s} = s$ avec une probabilité :

$$p(s, \grave{s}, T) = \exp \frac{f(s) - f(\grave{s})}{T}$$

`Les variantes de recuit simulé :` Johnson et al. [41] ont proposés trois variantes de recuit simulé qu'on va présenter.

• **Fonction de pénalité :** Une solution s est une partition $\{C_1, ..., C_k\}$ (avec $k \leq |V|$) de l'ensemble V. Soit s la solution courante, la solution voisine $\grave{s}$ de s est obtenue comme suit : on choisit aléatoirement $i \in \{1, ..., k+1\}$ et un sommet v, $v \in C_j$ où $j \in \{1, ..., k\}$, on déplace v de C_j vers C_i, si i=k+1 alors on crée une autre classe C_k mais le plus intéressant

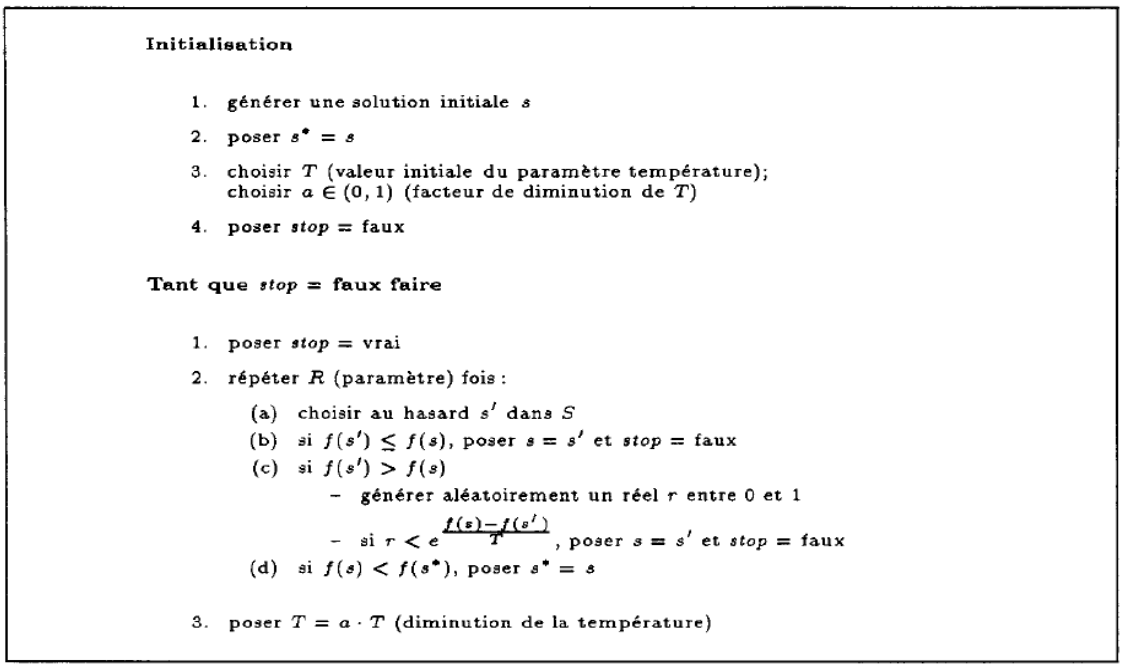

FIG. 2.1 – L'algorithme du recuit simulé

.

c'est d'avoir $i < j$ afin de minimiser le nombre de classes (nombre de couleurs).
La fonction objectif f à minimiser d'une solution $s = \{C_1, ..., C_k\}$ est définit comme suit :

$$f(s) = -\sum_{i=1}^{k} |C_i|^2 + 2\sum_{i=1}^{k} |C_i|.|E(C_i)|$$

avec $E(C_i)$: ensemble des arêtes de E qui ont leurs extrémité dans C_i.

• **Les chaînes de Kempe :** Une solution s est une partition $\{C_1, ..., C_k\}$ de l'ensemble V tel que $i \in \{1, ..., k\}$, C_i est un stable (pas de conflit). La fonction objectif est définit comme suit :

$$f(s) = -\sum_{i=1}^{k} |C_i|^2$$

Une chaîne de kempe relative à deux ensemble V_1 et V_2 est une composante connexe dans le sous ensemble de G induit par $V_1 \cup V_2$.
Soit $X\Delta Y = (X - Y)\Delta(Y - X)$ la différence symétrique entre l'ensemble X et Y, si H est une chaîne de Kempe relative à deux ensembles stables V_1 et V_2 alors $V_1\Delta H$ et $V_2\Delta H$ restent deux ensembles stables $V_1 \cup V_2$.

La génération d'une solution voisine se fait : on choisit aléatoirement C_i de s, un sommet v de C_i et une autre classe C_j, ensuite on détermine la chaîne de Kempe H relative aux classes C_i et C_j qui contient v, la solution voisine est obtenue en remplaçant C_i et C_j de s par respectivement $C_i \Delta H$ et $C_j \Delta H$, si $C_i \cup C_j$, on refuse la solution voisine $\grave{s}$.

• **Le nombre de couleurs est fixé au départ :** Une solution s est une k-coloration, la fonction objectif minimise le nombre de conflits :

$$f(s) = \sum_{i=1}^{k} |E(C_i)|$$

Si f(s)=0 alors s est une k-coloration admissible.

Au départ, k est fixé, si on peut trouver une solution s dont f(s)=0 alors, on essaye avec k-1 et ainsi de suite.
Deux solutions $s = \{C_1, ..., C_k\}$ et $\grave{s} = \{\grave{C}_1, ..., \grave{C}_k\}$ sont voisines s'il existe C_i contenant le sommet v et C_j telle que $\grave{C}_i = C_i - \{v\}$, $\grave{C}_j = C_j \cup \{v\}$ et $\grave{C}_l = C_l$ pour tout $l \in \{1, ..., k\} - \{i, j\}$. Autrement dit, on déplace un sommet v de C_i qui a plusieurs voisins dans C_i vers C_j comportant peu de sommet adjacent à v.

La recherche Tabou

La recherche taboue revient à Fred Glover en 1989 [24, 25], c'est une méthode basée sur l'exploration de voisinage.

$\grave{A}$ partir d'une solution initiale s_{ini} qui devient la solution courante s, le meilleur voisin v_{best} devient la nouvelle solution courante s, ce processus recommence jusqu'à ce qu'une condition d'arrêt soit satisfaite. Pour ce qui concerne la construction des voisins, plusieurs méthodes sont envisageables.

Ce processus peut conduire à un cyclage pour empêcher ce type de problème d'apparaître, on crie une liste de taille k et on mémorise les k dernières solutions déjà visitées dans cette liste et on interdit de retenir toute solution qui en fait déjà partie. Cette liste est appelé la **liste tabou**.

La mémorisation des solutions s'avère trop coûteuse en temps de calcul et espace mémoire d'où l'intérêt de mémoriser des caractéristiques relatives aux solutions par exemple mémoriser un attribut de cette solution.

La génération de la solution initiale se fait généralement en utilisant des heuristiques cela dépend des problèmes à traiter, dans le cas de problème de coloration des sommets d'un graphe, la méthode gloutonne donne de bonnes solutions initiales mais cela n'empêche pas l'utilisation d'autres méthodes.

La taille de la liste tabou est fixée à l'avance, dans certains cas, il se peut que certaines solutions non déjà explorées soit tabou. Pour palier ce problème, un mécanisme appelé l'aspiration est utilisé, il consiste à enlever le statut tabou d'une solution à une étape donnée de la recherche si cette solution est strictement améliorante.

Il existe d'autres techniques qui permettent d'améliorer la performance de la recherche tabou : l'intensification et la diversification. L'intensification permet de se focalise sur certaines solutions de très bonne qualité rencontrées au cours de la recherche. Par contre la diversification permet de rediriger la recherche vers des zones inexplorées dans l'espace des solutions.

2.3.3 Les heuristiques évolutives

Les algorithmes évolutifs sont des heuristiques inspirées de l'évolution biologique, elles se distinguent des autres méthodes du faite qu'elles génèrent un ensemble de solutions (individus) formant une population.

D'une génération à une autre, la population évolue et crie une autre population de meilleure qualité. Cette évolution se fait par des opérateurs d'évolution (selection, croisement et mutation) appliquer sur les individus formant la population. De nombreuses méthodes basées sur l'évolution sont apparues, on distingues les algorithmes génétiques [16] et les colonies de fourmis [18] bien qu'il existe d'autres méthodes, on se contentera de ces deux heuristiques que nous allons présenter dans la suite de ce chapitre.

Algorithme 1 Principe général d'une recherche Tabou

Pré-conditions: Qual(x) : Qualité (à maximiser) de la solution x

```
    /* Initialisation */
 1  Génération de la solution initiale s_init
 2  s = s_best = s_init ;
 3  Tℓ = ∅
    /* Boucle principale */
 4  Tant que Condition d'arrêt non satisfaite faire
 5      v_best = ∅
        /* Recherche du meilleur voisin non-tabou */
 6      Construire l'ensemble des voisins V(s) de la solution s
 7      Pour chaque v ∈ V(s) faire
 8          Si v ∉ Tℓ et Qual(v) > Qual(v_best) alors
 9              v_best = v
10          fin Si
11      fin Pour
12      s = v_best
13      Tℓ = Tℓ ∪ v_best /* Mise à jour de la liste tabou */
14      Si Qual(s_best) < Qual(s) alors
15          s_best = s /* Mise à jour de la meilleure solution trouvée */
16      fin Si
17  fin Tant que
18  retourner s_best
```

FIG. 2.2 – L'algorithme général de la recherche tabou

.

Les algorithmes génétiques

Les algorithmes génétiques font partis de la famille évolutifs, elles ont été introduits par Holland [39] en 1975, ils s'inspirent de l'évolution naturelle.

Les algorithmes génétiques se composent principalement de trois éléments, la population qui est formée d'individus (solutions) sous un codage donné on l'appelle un chromosome, la fonction d'évaluation (fitness) relatif au problème étudié, elle permet d'évaluer une solution (chromosome ou individu) et la comparer aux autres. Enfin, les opérateurs d'évolution de la population qui permettent de produire d'autres individus (enfants) à partir d'individus déjà existant (parents). On distingue trois opérateurs d'évolution :

- `La selection :` elle permet de choisir les individus qui sont apte à se reproduire par des croisements ou par un autre opérateur génétique, cela veut dire que la recherche se fait autour des meilleures solutions. L'opérateur de selection utilise la valeur d'adaptation pour sélectionner les individus.
  Dans la littérature, il existe plusieurs méthodes de selection que nous allons présenter.

  - **Selection par rang :** on attribue un numéro à chaque individu. Dans le problème de minimisation, on ordonne les individus selon l'ordre décroissant, le plus mauvais individu reçoit le numéro 1 ainsi de suite. Ensuite, on prélève une nouvelle population à partir de cet ensemble d'individus ordonnés en utilisant cette probabilité :
    $$Prob_i(parent_i) = \frac{rang(parent_i)}{\sum_{j \in population} rang(parent_j)}$$
  - **Selection par la roulette :** On attribue à chaque individu i une probabilité de selection $Prob_i$, cette probabilité est proportionnelle à sa valeur F_i de la fonction objectif.
    $$Prob_i = \frac{F_i}{\sum_{j \in population} F_j}$$
    Cella se fait dans le cas d'un problème d'optimisation de maximisation. Pour les problèmes de minimisation $Prob_j = \frac{1 - Prob_i}{N-1}$

- **Selection aléatoire :** chaque individu a la même probabilité uniforme $\frac{1}{N}$ (N : la taille de la population) d'être sélectionner.
- **selection par tournoi :** Une sous-population $M \leq N$ est prise aléatoirement, le meilleur individu du sous-ensemble est sélectionner.

– `Le croisement` : permet de produire deux individus enfants à partir de deux individus parents. Il existe plusieurs opérateurs de croisement. Les plus connues sont l'opérateur de croisement à un point et à deux points sur deux chromosomes. L'opérateur de croisement à un point consiste à diviser chacun des deux parents en deux parties à la même position, l'enfant 1 est composé par exemple de la première partie du parent1 et de la première partie du parent2 et l'enfant2 est composé de la deuxième partie du parent1 et de la deuxième partie du parent2.

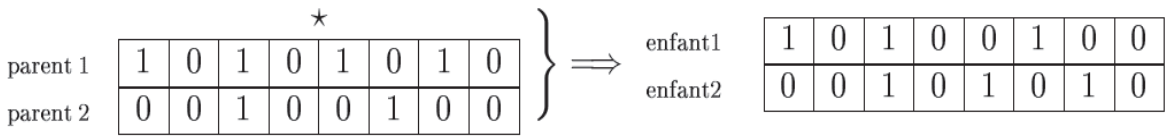

FIG. 2.3 – L'opérateur de croisement à un point

.

– `La mutation` : cette opération permet d'avoir un autre individu à partir d'un individu déjà existant en appliquant un changement sur son code (par exemple, dans un codage binaire en changeant un bit de 1 à 0 ou de 0 à 1 d'un individu devient un autre)

L'algorithme génétique en général s'opère comme suit :
A la première itération (génération), on génère une population P_0 et on applique à cette population les opérateurs d'évolution afin d'avoir une autre population différente P_1. Ce processus se répète jusqu'à ce que le test d'arrêt soit satisfait.

Remarque 2.3.1. Le test d'arrêt peut être le nombre de générations ou lorsque la population cesse d'évoluer.

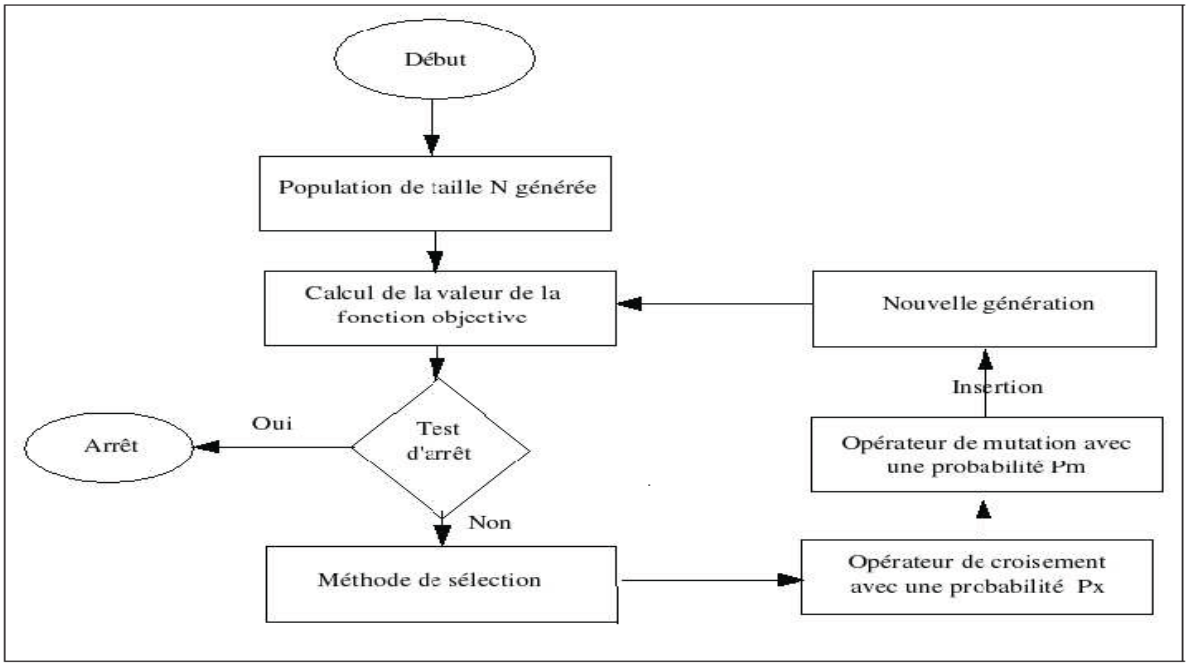

FIG. 2.4 – L'organigramme de l'algorithme génétique

.

Dans la première partie de cette section, on a présenté en général l'algorithme génétique. Dans la suite, on décrit les différentes adaptations des algorithmes génétiques aux problèmes de coloration des sommets d'un graphe. Chaque adaptation se caractérise par la manière de coder les solutions et les deux opérateurs d'évolution (croisement et mutation). Hertz a proposé cinq adaptations que nous allons essayer de présenter.

L'algorithme de Davis : Davis utilise l'espace des solutions admissibles partielles (sans conflits) et une fonction objectif qui maximise le nombre de sommets colorés. L'ordre des sommets est utilisé pour codage des solutions et pour construire une solution en utilisant l'algorithme glouton, celui-ci affecte au sommet v la plus petite couleur i tel que $i \in \{1, ..., k\}$ de façon à ne pas générer des conflits, en cas de conflit le sommet v reste incoloré. Si à une étape tous les sommets sont colorés alors il répète le processus avec k-1.

Pour l'opérateur de croisement, Davis procède comme suit :
Soient deux chromosomes parents O_1 et O_2 (le chromosome est l'ordre des sommets d'un graphe). En appliquant le croisement de deux chromosomes enfants sont crées O_3 et O_4 en générant un vecteur binaire $\{0, 1\}$ de taille $|V|$.
Le chromosome O_3 est composé des sommets de O_1 aux positions où b contient la valeur 1 et en plaçant les autres sommets de O_1 selon l'ordre d'adaptation dans O_2.

Le chromosome O_4 est composé des sommets de O_2 aux positions où b contient la valeur 0 et en plaçant les autres sommets de O_2 selon l'ordre d'adaptation dans O_1.
L'opérateur de mutation consiste à interchanger deux sommets en deux positions différentes dans un chromosome.

Le premier algorithme de Fleurent et Ferland [20] : Ces deux chercheurs utilisent le même codage que Davis mais dans l'espace de k-coloration. L'algorithme glouton affecte à chaque sommet la plus petite couleur i, $i \in \{1, ..., k\}$ qui ne génère pas de conflit, si c'est le cas alors, le sommet v recevra la couleur qui génère moins de conflit. La fonction objectif est le nombre de conflits donc une solution dont le nombre de conflits est nul est une solution admissible du graphe. Pour l'opérateur de mutation, il est identique à celui de Davis.

Le second algorithme de Fleurent et Ferland [20] : Dans ce deuxième algorithme, Fleurent et Ferland proposent une solution $s(s(1), ..., s(|V|))$ où s(v) correspond à la couleur du sommet v, k est fixé et ils minimisent le nombre de conflits. Pour le croisement, ils utilisent le croisement à un point mais ensuivant certaines règles que voici :
La génération d'un enfant O_3 à partir de deux chromosomes $O1$ et O_2 se fait de la manière suivante : pour un sommet v de G, notons $n_i(v)$ le nombre de voisins de v ayant la couleur i dans P_1 ou / et dans P_2.
Si v n'est en conflit que dans l'un des deux parents, on affecte à v la couleur qu'il a dans l'autre parent, si v n'est en conflit ni dans P_1 ni dans P_2, on choisit $O_3(v)$ aléatoirement parmi $P_1(v)$ et $P_2(v)$. Finalement, si v est en conflit dans P_1 et dans P_2, on attribue à v la couleur i qui minimise $n_i(v)$.

L'algorithme de Costa et al. : Costa et al. [12] proposent un algorithme qui s'apuit sur le nombre de conflit, ils définissent la fonction objectif comme suit :

$$f(s) = \sum_{[x,y] \in E/s(x)=s(y)} w_{[x,y]}$$

avec $w_{[x,y]}$ poids de l'arête [x,y] du graphe G c'est à dire la valeur de la solution est la somme des poids des arêtes reliant deux sommets de même couleur.

Le poids des arêtes sont fixé au départ à 1 et à chaque génération, le poids $w_{[x,y]}$ d'une arête [x,y] est augmenté de Δ (Δ est fixé) si et seulement si la population contient au moins une solution contenant un conflit.

Costa et al. définissent quelques fonctions que voici :
δ : la distance entre un sommet v et une arête [x,y] est le plus petit nombre d'arêtes sur une chaîne reliant v à une extrémité de [x,y].
$d[v, s, \delta]$: le nombre d'arêtes en conflit dans s à distance δ de v.
Soient α_1, α_2, α_3 à distance 0, 1 et 2 ($\alpha_1 > \alpha_2 > \alpha_3$)
$p(v, s)$ est la mesure de proximité, elle est définit comme suit :

$$p(v, s) = \sum_{\delta=1}^{2} \alpha_\delta d[v, s, \delta]$$

Cette fonction est utilisé pour générer un chromosome enfant O_3 à partir de deux chromosomes parents. Costa et al. ordonne les sommets de G. La couleur de chaque sommet v est déterminé comme suit :

$$Si \quad p(v, O_1) < p(v, O_2) \quad alors \quad O_3(v) = O_1(v)$$

$$Si \quad p(v, O_1) > p(v, O_2) \quad alors \quad O_3(v) = O_2(v)$$

Si $p(v, O_1) = p(v, O_2)$ alors le sommet v prend l'une des deux couleur $O_3(v) = O_1(v)$ ou $O_3(v) = O_2(v)$.

L'algorithme de Galinier et Hao : Galinier et Hao [22] travaillent dans l'espace des k-colorations et minimisent le nombree de conflits.

Pour l'opérateur de croisement, l'idée est :

Soient deux parents $P_1 = \{C_1^1, ..., C_k^1\}$ et

$P_2 = \{C_1^2, ..., C_k^2\}$ avec C_i^l est la classe des sommets des parents l colorés avec la couleur i, $i \in \{1, ..., k\}$ et l=1,2.

On retire tous les sommets en conflit. Chaque classe C_r de la solution enfant devrait avoir beaucoup de sommets en commun avec l'une des classes C_i^1 ou C_j^2 des parents.

Pour construire la solution enfant e, ils génèrent une k-coloration partielle de tel sorte que la moitié des classes de e soient des sous-classes de P_1 et l'autre moitié est constitué des sous-classes de P_2 c'est à dire l'enfant e est construit classe par classe, chaque classe devrait permettre la coloration d'un maximum de sommets.

l'opérateur de croisement construit une solution enfant $\{C_1, ..., C_k\}$ en appliquant l'algorithme suivant :

Entrée : deux k-coloration parents (partielles) $P_1 = \{C_1^1, ..., C_k^1\}$ et $P_1 = \{C_1^2, ..., C_k^2\}$

Poser i=1 ;

Tant que i ≤ k **faire**

1. **Si** i est pair, poser a = 1 ; sinon poser a = 2
2. choisir la classe C_i^a de plus grande cardinalité parmi celles de $P_1 = \{C_1^a, ..., C_k^a\}$ (choix aléatoire si plusieurs possibilité)
3. Poser $C_i = C_i^a$
4. Retirer les sommets de C_i des classes de P_1 et P_2
5. Poser i = i + 1

Donner aléatoirement une couleur entre 1 et k aux sommets de $V-\{C_1, ..., C_k\}$

Sortie : k-coloration enfant e = $\{C_1, ..., C_k\}$

FIG. 2.5 – Procédure de croisement de deux solution P_1 et P_2

.

La première classe C_1 de la solution enfant est toujours choisit au hasard dans le premier parent P_1.

Les colonies de fourmis

L'algorithme de fourmis est inspiré du comportement réel des fourmis dans la nature, chaque individu a pour rôle la recherche d'une source de nourriture. Dés qu'une fourmi trouve une source de nourriture, elle dépose sur son chemin une substance chimique appelé phéromone qui va guider les autres fourmis vers cette source.

En observant les fourmis, on constate qu'elles empruntent le plus court chemin vers la source de nourriture et dés qu'elles rencontrent un obstacle, elles le détournent.

C'est en 1997, que les premiers algorithmes de fourmis ont été appliqués aux problèmes de coloration des sommets d'un graphe.

Dans la littérature, il existe trois approches de colonies de fourmis pour le problème de coloration des sommets d'un graphe, dans la première approche, chaque fourmi est un algorithme constructive qui laisse une trace sur chaque paires de sommets non adjacents pour indiquer si ces sommets ont reçu la même couleur, la deuxième approche, les fourmis se promènent sur le graphe et tentent collectivement de modifier la couleur des sommets qu'elles visitent, cela afin de diminuer le nombre d'arêtes conflictuelle dans une k-coloration, la dernière approche, les fourmis sont des algorithmes de recherche local qui laisse des traces sur l'exploration qu'elles ont faite dans l'espace de recherche.

Chaque fourmi est un algorithme constructif : Cette approche a été proposé par Costa et Hertz [13], ils se basent sur les algorithmes Dsatur et RLF.

Soit une coloration partielle légale (sans conflit) d'un graphe G et soit U l'ensemble des sommets non encore colorés. Le degré de saturation d'un sommet v non coloré est le nombre de couleurs différentes des sommets adjacent au sommet v.
L'algorithme Dsatur color le sommet de U ayant le plus grand degré de saturation, en cas d'ex-aequo l'algorithme colore celui de degré maximal.

L'algorithme RLF construit les classes l'une après l'autre, pour chaque classe le premier sommet à en faire partie est choisit au hasard parmi ceux ayant un nombre maximal de sommets adjacents non colorés. Afin de completer chaque classe, on procède comme suit : Soit $W \subseteq U$ ensemble des sommets non colorés qui ne peuvent plus faire partie de la classe en construction (car ils sont adjacents à au moins un sommet de cette classe). Le prochain sommet introduit dans la classe appartenant à U-W ayant un nombre maximum de sommet adjacent dans W. Lorsque U=W, on passe à la classe suivante.

Dans les algorithmes de fourmis basés sur les algorithmes Dsatur et RLF, les fourmis laissent des traces qui influencent le choix du prochain sommet à colorer mais pas la couleur à attribuer à ce sommet.

Les fourmis se promènent sur le graphe : Dans cette approche, on place les fourmis sur le graphe à colorer et on les laisse explorer le graphe de sommet en sommet ainsi, chaque fourmi fournit une k-coloration qui est pas forcément sans conflits. A la fin,on aura un ensemble de solution S comportant toutes les k-colorations [38]

Chaque fourmi est une procédure de recherche locale : Les fourmis tentent de résoudre le problème de coloration des sommets d'un graphe avec k fixé. En se déplaçant d'un sommet à l'autre et en tentant de changer la couleur du sommet, chaque fourmi agit comme une procédure de recherche locale et les traces laissée lors de chaque tentative de diminuer le nombre d'arêtes conflictuelles influencent les recherches futures [38].

Coloration de la Classe des Graphes Localement Scindés

CHAPITRE 3

Coloration de la Classe des Graphes Localement Scindés

Dans ce chapitre, nous consacrons la première partie à la reconnaissance des graphes scindés et localement scindés, ensuite nous décrivons la recherche taboue adaptée à la coloration des graphes localement scindés et enfin, nous présentons l'algorithme Dsatur appliqué aux graphes quelconques.

3.1 Algorithme de reconnaissance

3.1.1 Définition

Définition 3.1.1. Reconnaître une classe de graphes, c'est donner un algorithme permettant de décider si un graphe appartient ou non à la classe.

3.1.2 Graphe scindé

La notion des graphes scindé a été introduite par S. *Földes* et P.L. Hammer [29].

Définition 3.1.2. Un graphe G= (V, E) est scindé s'il existe une partition de l'ensemble des sommets V en un stable S et une clique K.

Théorème 3.1.1 ([29]). *Soit G=(V,E) un graphe, les propriétés suivantes sont équivalentes :*

- *G est un graphe scindé ;*
- *G et $\overline{G}$ sont des graphes triangulés ;*
- *G ne contient pas de sous graphe isomorphe à un $2K_2$, C_4 ou C_5*

3.1.3 Algorithme de reconnaissance des graphes scindés

Afin d'élaborer un algorithme de reconnaissance des graphes scindés, nous appliquons le théorème P.L.Hammer et B.Simeone [30].

Théorème 3.1.2 (Thérème P.L.Hammer et B.Simeone). *Soit G= (V, E) un graphe avec une séquence de degré $d_1 \geq d_2 \geq \cdots \geq d_{n-1} \geq d_n$ où d_i est le degré du sommet v_i.*
Posons $P = \max\{i/d_i \geq i-1\}$. Alors les propriétés suivantes sont équivalentes :

- *(i) G est un graphe scindé*
- *(ii) $\sum_{i=1}^{P} d_i = P(P-1) + \sum_{i=P+1}^{n} d_i$*

De plus si le graphe est scindé alors $\{v_{i,\ldots,v_P}\}$ et $\{v_{P+1}, \ldots, v_n\}$ forment une partition scindé, $\omega(G) = \gamma(G) = P$ et $\alpha(G) = \theta(G) = n - \min\{P, d_P\}$.

Cet algorithme nous donne aussi :

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

Algorithme de reconnaissance d'un graphe scindé

1. `Donnée` : Un graphe G =(V, E) à n sommets.
2. `Sortie` : Le graphe est scindé ou pas

Complexité : O(nlogn).

Exemple d'un graphe scindé

Pour tous graphe, le programme de reconnaissance des graphes scindé permet de reconnaître les graphes scindés. Nous présentons ici un graphe qui est scindé.

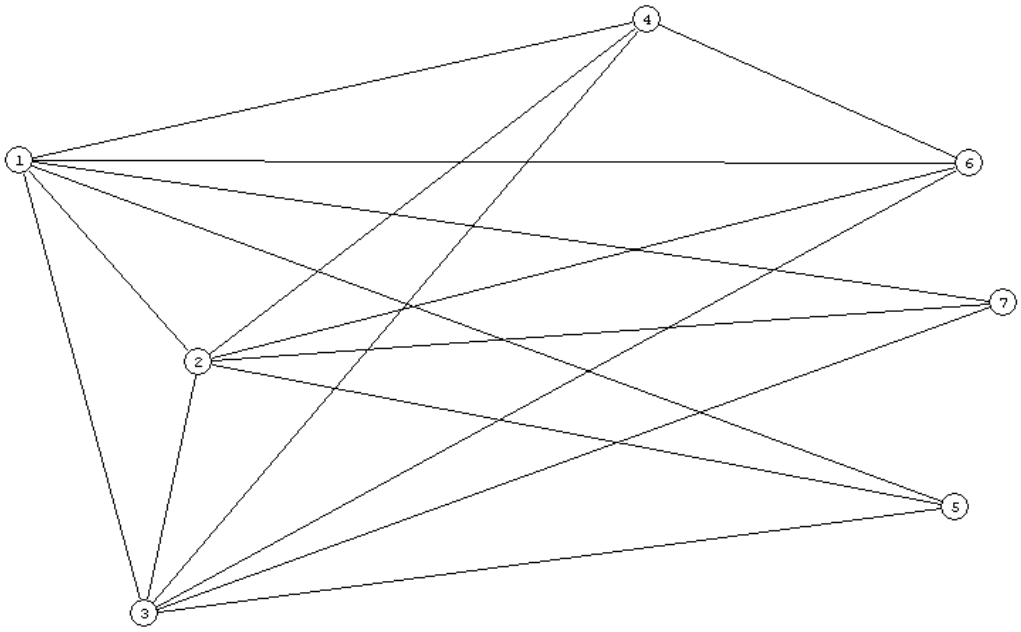

FIG. 3.1 – Exemple d'un graphe scindé

.

3.1.4 Graphes localement scindés

Définition 3.1.3. Un graphe G= (V, E) est localement scindé si tous ces sommets sont scindés.

Définition 3.1.4. Un sommet scindé est un sommet dont le voisinage peut être partitioner en une clique K et un stable S.

3.1.5 Algorithme de reconnaissance des graphes localement scindés

Pour l'algorithme de reconnaissance d'un graphe localement scindé, nous appliquons le théorème de P.L.Hammer et B.Simeone [30] sur chaque sommet de ce graphe. C'est-à-dire

nous avons appliqué ce théorème sur le voisinage $N(v_i)$ de chaque sommet v_i du graphe G = (V, E).

1. `Donnée` : Un graphe G =(V, E) à n sommets.
2. `Sortie` :
   - Le graphe est Localement Scindé ou pas
   - $\omega(G)$ sachant que $\omega(G) = \max_{\overline{1,n}} |G_i(N(v_i))|$

Complexité : $O(n^2 logn)$.

Exemple d'un graphe localement scindé

Ce graphe présenté ici est un graphe localement scindé mais qui n'est pas scindé et $\omega(G)$=5.

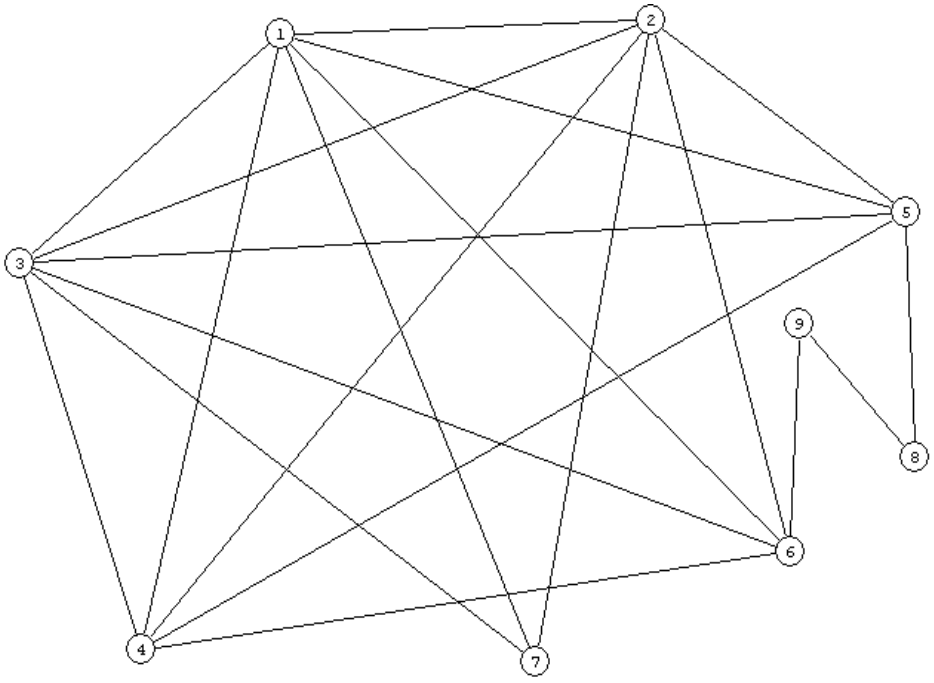

FIG. 3.2 – Exemple d'un graphe localement scindé

.

3.1.6 Expérimentations numériques

Dans ce tableau, la première, la deuxième et la troisième colonnes représentent successivement les instances (exemple de graphe), le nombre de sommets et le nombre d'arêtes, la quatrième et la cinquième colonnes désignent la reconnaissance des graphes localement scindés et la taille maximum de clique.

L'algorithme de reconnaissance des graphes localement scindés est très intéressant car il nous permet de savoir si oui ou non ce graphe appartient à cette classe, nous avons appliqué ce programme sur plusieurs instances. Les résultats sont illustrés dans le tableau ci-dessous, notons bien que l'algorithme donnera aussi la taille de clique maximum dans le cas des graphes localement scindés.

graphe	N^{br} de sommets	N^{br} d'arêtes	Reconnaissance de GLSc	$\omega(G)$
myciel	5	5	oui	2
myciel3	11	20	oui	2
myciel4	23	71	oui	2
myciel5	47	236	oui	2
myciel6	95	755	oui	2
myciel7	191	2360	oui	2
glscnsc	9	20	oui	5
$2-Insertion-5$	597	3936	oui	2
$3-Insertions-5.col$	1406	9695	oui	2
$1-Insertions-4.col$	67	232	oui	2
$1-Insertions-5.col$	202	1227	oui	2
$1-Insertions-6.col$	607	6337	oui	2
$2-Insertions-3.col$	37	72	oui	2
$2-Insertions-4.col$	149	541	oui	2
$3-Insertions-3.col$	56	110	oui	2
$4-Insertions-3.col$	79	156	oui	2
miles250	128	774	non	-
miles500	128	2340	non	-
miles750	128	4226	non	-
miles1000	128	6432	non	-
miles1500	128	10396	non	-
school1	385	19095	non	-
games120	120	1276	non	-
$le450-5a$	450	5714	non	-
$le450-5c$	450	9803	non	-
$le450-5d$	450	9757	non	-
$le450-15a$	450	8168	non	-
$le450-15b$	450	8169	non	-
$le450-15c$	450	16680	non	-

TAB. 3.1 – Tableau de reconnaissance des graphes localement scindés

3.2 Approche tabou pour la coloration d'un graphe localement scindé

Les algorithmes de la recherche tabou ont montré leurs efficacité pour résoudre des problèmes combinatoires et en particulier les problèmes de coloration des sommets d'un graphe [17, 36].

Le problème que nous allons traiter est le problème de coloration d'un graphe localement scindé. L'algorithme de reconnaissance de cette classe nous permet d'avoir la taille de la clique maximum, nous savons bien que $\gamma(G) \geq \omega(G)$, alors on a bien une borne inférieure pour pouvoir colorier ce graphe.

La solution initiale

Résoudre le problème de coloration c'est de trouver un entier k (k étant le nombre de couleurs) c'est à dire, k-coloration et l'affectation de ces couleurs aux sommets.

Étant donné que nous avons la borne inférieure, la méthode tabou commence par affecter les $\omega(G)$=k couleurs aux sommets du graphe arbitrairement, cette affectation peut être admissible c'est à dire sans conflits ou avec des conflits.

Le voisinage

L'affectation de k couleurs au graphe est une partition des sommets de ce graphe en sous-ensemble $V_1, V_2, ..., V_k$ tel que si cette affectation est une coloration alors les sous-ensemble sont des stables, sinon, il existe des sommets adjacents appartenant au même ensemble alors il sont en conflit.
Le voisinage est constitué de l'ensemble des partitions possible des sommets en sous-ensemble pouvant être obtenu en changeant la couleur i d'un sommet v qui est en conflit avec au moins un autre sommet en une autre couleur j.

La liste tabou

Lorsqu'un sommet x est déplacé de l'ensemble V_i vers l'ensemble V_j, ce mouvement sera interdit pendant un nombre fixé d'itérations, qui correspond à la taille de la liste tabou et le mouvement (x,i) sera placé dans cette liste.

La liste tabou évite de retourner vers des solutions déjà explorées, mais cela empêche parfois d'explorer des solutions non encore visitées et qui peut être peuvent conduire à une meilleure solution. Afin de remédier à ce problème, on utilisé la fonction d'aspiration FIFO (First In First Out), divers fonctions d'aspiration sont décrites par A. Hertz et D. de Werra [37].

Le critère d'arrêt

À la première itération, la méthode prend $k = \omega(G)$ (borne inférieure) et elle affecte arbitrairement les couleurs aux sommets, le résultat est bien souvent avec des conflits, alors avec l'exploration du voisinage des sommets qui sont en conflits ; la méthode cherche à éliminer tous ces conflits.

Si le nombre de conflits est égal à zero, alors la méthode s'arrête et cette affectation est une coloration, si par contre la méthode n'arrive pas à éliminer ces conflits (le nombre de conflits est différent de zéro) et le nombre d'itérations fixé au départ est dépassé alors on incrémente k (k :=k+1) et le processus recommence.

3.2.1 Algorithme tabou

Cette algorithme est adapté à la coloration des graphes localement scindés, dont les principales étapes sont décrites.

1. – Fixer le nombre d'itérations maximum (nbritmax)
   – Fixer la taille de la liste tabou (TLT)
2. Appliquer l'algorithme de reconnaissance des graphes localement scindés ($\omega(G)$).
   K :=W(G)

3. Affecter K couleurs arbitrairement aux N sommets, itération :=0 ;
4. Si cette affectation est une coloration (pas de conflits) alors aller à l'étape (9), sinon (ie l'affectation contient des conflits, c-à-d des arêtes dont les extrémités ont la même couleur) aller à l'étape (5)
5. Affecter toutes les couleurs possibles pour chaque sommet x (x a au moins un conflit), on prend l'affectation qui donne le minimum de conflits.
   Soit x ce sommet, (x,i) entre dans la liste tabou sachant que i est l'ancienne couleur de x, la recherche tabou interdit au sommet x de reprendre la couleur i pendant TLT itérations, aller à l'étape (6)
6. Si le nombre de conflits est égale à 0 (pas de conflits) alors aller à l'étape (9)
   Sinon, aller à l'étape (7).
7. itération :=itération + 1 ;
   si itération > nbritmax alors aller à l'étape (8), sinon aller à l'étape (5)
8. K :=K+1, aller à l'étape (3)
9. Le nombre de couleurs optimale est K

la coloration recherchée est la coloration sans conflits.

3.2.2 Expérimentations numériques

Il existe en général deux types d'instances pour tester les méthodes de coloration à savoir les graphes générés aléatoirement et les graphes tirés de DIMACS disponibles sur le site internet http ://mat.gsia.cmu.edu/COLOR/instances.html. Ces instances représentent des problèmes réels.

La méthode tabou adaptée au problème de coloration des graphes localement scindés a été implémentée en Delphi 7 sur un PC Pentium (4), 1.60 GHZ avec 1 Go de RAM.

Dans ce tableau, la première, la deuxième et la troisième colonnes représentent successivement les instances (exemple de graphe), le nombre de sommets et le nombre d'arêtes, la quatrième et la cinquième colonnes désignent la reconnaissance des graphes localement

scindés et la taille maximum de clique, la sixième colonne correspond au nombre de couleurs donné par la méthode et la dernière colonne est réservé pour le temps de calcul de chaque instances (en seconde).

D'après les résultats présentés dans le tableau (Tab 3.2), on constate que les treize premiers graphes sont des graphes localement scindés et la méthode donne la coloration de ces graphes. Le reste des graphes ne sont pas des graphes localement scindés d'où l'impossibilité d'appliquer la recherche tabou adaptée à cette classe.

graphe	n	m	RGLS	$\omega(G)$	N^{br} couleurs	Temps (en seconde)
myciel	5	5	oui	2	3	01
myciel3	11	20	oui	2	4	01
myciel4	23	71	oui	2	5	01
myciel5	47	236	oui	2	6	01
myciel6	95	755	oui	2	7	02
myciel7	191	2360	oui	2	8	03
glscnsc	9	20	oui	5	5	01
$3-Insertions-3.col$	56	110	oui	2	4	01
$2-Insertions-3.col$	37	72	oui	2	4	01
$1-Insertions-4.col$	67	232	oui	2	5	01
$1-Insertions-5.col$	202	1227	oui	2	6	02
$2-Insertions-4.col$	149	541	oui	2	5	01
$4-Insertions-3.col$	79	156	oui	2	4	01
jean	80	508	non	-	-	-
miles250	128	774	non	-	-	-
miles500	128	2340	non	-	-	-
miles750	128	4226	non	-	-	-
miles1000	128	6432	non	-	-	-
miles1500	128	10396	non	-	-	-
school1	385	19095	non	-	-	-
games120	120	1276	non	-	-	-
fpsol2.i.1.col	496	11654	non	-	-	-
mulsol.i.1	197	3925	non	-	-	-
$le450-5a$	450	5714	non	-	-	-
$le450-5c$	450	9803	non	-	-	-
$le450-5d$	450	9757	non	-	-	-
$le450-15a$	450	8168	non	-	-	-
$le450-15b$	450	8169	non	-	-	-
$le450-15c$	450	16680	non	-	-	-

TAB. 3.2 – Résultats de la recherche tabou adaptée à la coloration des graphes localement scindés

3.3 L'heuristique DSATUR

Les méthodes constructives de coloration des sommets d'un graphe parcourent séquentiellement les sommets, en attribuant à chaque sommets la plus petite couleur possible, la qualité de la solution ainsi obtenue dépend fortement de l'ordre dans lequel les sommets sont considérés, un ordre des sommets peut être statique ou dynamique. L'ordre statique si on peut le déterminer à l'avance par contre l'ordre est dynamique si le choix du prochain sommet à colorer dépend des affectations précédentes.

La méthode DSATUR est une méthode dynamique, elle a été introduite par Brélaz [8]. Il définit le degré de saturation d'un sommet v comme étant le nombre de couleurs différentes déjà affectées aux voisins de v.
À la première itération, la couleur 1 est attribuée au sommet de degré maximum, pour les sommets non encore colorier le degré de saturation est utilisé. Nous détaillons les principales étapes de cette heuristique dans ce qui suit :

3.3.1 Principe de l'algorithme

1. Ordonner les sommets par ordre décroissant des leurs degré
2. Colorier un sommet de degré maximum avec la couleur 1
3. Choisir un sommet de DSAT maximum (en cas d'ex-aequo en prend celui de degré maximal) et le colorier avec la plus petite couleur possible
4. Si tous les sommets sont coloriés alors stop, sinon aller à l'étape (3)

DSAT(v)= le nombre de couleurs différentes utilisées dans le premier voisinage de v.

3.3.2 Expérimentations numériques

Dans ce tableau, la première, la deuxième et la troisième colonnes représentent successivement les instances (exemple de graphe), le nombre de sommets et le nombre d'arêtes, la quatrième colonne correspond au nombre de couleurs donné par la méthode et la dernière colonne désigne le temps de calcul de chaque instances (en seconde).

La méthode DSATUR est implémentée pour la coloration des graphes quelconques, le temps d'execution de tous les testes sont inférieur à (01) seconde. En général, la méthode donne des résultats satisfaisants.

3.4 Comparaison entre la Recherche Tabou Adaptée à la coloration des graphes localement scindés et l'heuristique Dsatur :

Après avoir programmer la recherche tabou adaptée à la coloration des graphes localement scindés et l'heuristique Dsatur, il nous reste plus qu'à faire des tests et à évaluer la qualité de nos méthodes. Pour cela, nous avons utilisé des instances dont les graphes s'avèrent des graphes localement scindés.

D'après les résultats expérimentaux présentés dans le tableau ci-dessous, on constate que les deux méthodes donnent le même nombre de couleurs mais en ce qui concerne le temps d'exécution, on remarque sur certaines instances que l'heuristique Dsatur s'avère plus rapide que la recherche tabou.

graphe	n	m	N^{br} de couleurs	Temps de calcul (en seconde)
myciel	5	5	3	01
myciel3	11	20	4	01
myciel4	23	71	5	01
myciel5	47	236	6	01
myciel6	95	755	7	01
myciel7	191	2360	8	01
glscnsc	9	20	5	01
$queen5-5$	25	320	5	01
$queen7-7$	49	952	11	01
$queen8-8$	64	1456	11	01
$queen8-12$	96	2736	14	01
$queen9-9$	81	2112	13	01
huck	74	602	11	01
jean	80	508	10	01
miles250	128	774	8	01
miles500	128	2340	20	01
miles750	128	4226	31	01
miles1000	128	6432	42	01
miles1500	128	10396	73	01
school1	385	19095	15	01
games120	120	1276	9	01
fpsol2.i.1.col	496	11654	38	01
mulsol.i.1	197	3925	49	01
$le450-5a$	450	5714	10	01
$le450-5c$	450	9803	12	01
$le450-5d$	450	9757	12	01
$le450-15a$	450	8168	16	01
$le450-15b$	450	8169	16	01
$le450-15c$	450	16680	21	01

TAB. 3.3 – Résultats de la méthode DSATUR appliquée à la coloration des sommets d'un graphe

Graphe	Nombre de sommets	Nombre d'arrêtes	Recherche Tabou Adaptée			DSATUR	
			Reconnaissance de graphe localement scindé	Nombre de couleurs	Temps de calcul (en secondes)	Nombre de couleurs	Temps de calcul (en secondes)
myciel	5	5	Oui	3	01	3	01
myciel3	11	20	Oui	4	01	4	01
myciel4	23	71	Oui	5	01	5	01
myciel5	47	236	Oui	6	01	6	01
myciel6	95	755	Oui	7	02	7	01
myciel7	191	2360	Oui	8	03	8	01
GLSCNSC	9	20	Oui	5	01	5	01
3-Insertions_3.col	56	110	oui	4	01	4	01
2-Insertions_3.col	37	72	oui	4	01	4	01
1-Insertions_4.col	67	232	oui	5	01	5	01
1-Insertions_5.col	202	1227	oui	6	02	6	01
2-Insertions_4.col	149	541	oui	5	01	5	01
4-Insertions_3.col	79	156	oui	4	01	4	01

FIG. 3.3 – Tableau de comparaison

.

Conclusion

Conclusion

Dans ce mémoire, nous avons étudié la classe des graphes scindés et la classe des graphes localement scindés.

Le travail effectué sur ces deux classes est partagé en deux parties :

Dans la première partie, nous avons implémenter un programme de reconnaissance de graphe scindé en utilisant le théorème P.L.Hammer et B.Simeone. Ce programme nous permet de savoir si "oui" ou "non" le graphe est scindé, comme il permet de donner la taille maximale d'un stable de G ($\alpha(G)$), le nombre chromatique de G ($\gamma(G)$), la taille maximale d'une clique de G ($\omega(G)$) et la taille minimale d'une partition de V en cliques ($\theta(G)$). La généralisation de ce théorème, nous a permis d'élaborer un algorithme de reconnaissance des graphes localement scindés, ce dernier, nous donne aussi la taille de la clique maximum de cette classe.

Dans la deuxième partie, nous avons abordé le problème de la coloration des sommets d'un graphe localement scindé, nous avons proposé la recherche "tabou" adaptée à la coloration de cette classe.

Afin de tester l'efficacité de cette approche, nous avons implémenté l'heuristique Dsatur pour pouvoir comparer les résultats des deux approches.

Notre travail à porter sur la résolution du problème de coloration des sommets d'un graphe localement scindé en utilisant la recherche taboue adaptée à cette classe, mais cependant, la solution donnée par cette heuristique reste une solution approximative.

La recherche de la solution optimale pour le problème de coloration des sommets d'un graphe localement scindé est un problème ouvert. Alors, notre perspective est d'essayer de trouver une solution exacte en un temps polynomial.

Bibliographie

BIBLIOGRAPHIE

[1] H. *Aït* Haddadène, S. Gravier , *On weakly diamond-free Berge graph*, Discrete mathematics, 159, 237-240, 1996.

[2] H. *Aït* Haddadène, *Sur quelques structures de graphes parfaits*, Thèse de doctorat, ES-Sciences, USTHB, 2000.

[3] H. *Aït* Haddadène and F. Maffray, *Coloring perfect degenerate graphs*, Discretes Mathematics, 163, 237-240, 1997.

[4] C. Berge, *Färbung von graphen*, deven sämtlichebz dzw .deren ungerade kreise starr sind, Wiss.Z.. Martin-Luther University, Halle-Wittenberg, 1961.

[5] C. Berge, *Graphes et Hypergraphes*, Dunod, 1973.

[6] C. Berge, *Les problèmes de coloration en théorie des graphes*, Publications de l'institut de statistiques, Université Paris 9, 123-160,1960 .

[7] C. Berge and P. Duchet, *Strongly perfect graph*, in Berge and Chvatal, 1984.

[8] D.Brélaz, *New methods to color the vertics of a graph*, Communication of the ACM, 22, 251-256, 1979.

[9] M. Burlet and J. Fonlupt, *Polynomial algorithm to recognize a Meyniel graph*, in Berge and Chvatal, 1984.

[10] V. Chvatal and N. Sbihi, *Bull-free Berge graphs are perfect*, Graphs and Combinatorics 3, 127-139, 1987.

[11] G. Cornuéjols and W.H. Cunningham, *Composition for perfect graphs* , Discrete Mathematic 55, 245-254, 1985.

[12] D. Costa, A. Hertz and O. Dubuis , *Embedding of a sequential algorithm within an evolutionary algorithm for coloring problems in graphs*, Journal of heuristics 1, 105-128, 1995.

[13] D. Costa and A. Hetz, *Ants can colour graphs*, Journal of operationel research society, vol. 48, 295-305, 1997.

[14] M. Chudnovsky, N. Robertson, P.D. Seymour and R. Thomas, *Progress On Perfect Graph*, Manuscript, November 15,2002, Revised December 18, 2002.

[15] M.Chudnovsky, N. Robertson, P.D. Seymour and R. Thomas, *The strong perfect graph theorem*, Manuscript, 2002.

[16] L. Davis, *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, 1991.

[17] I. Devarenne, H. Mabed and A. Caminada, *Self-adaptive Neighborhood Exploration Parameters in Local Search*, 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics, Universityof Málaga, Spain, 2006.

[18] M. Dorigo and L.M. Gambardella, *Ant colony system : optimisation by a coColony of Cooperating Agents*, IEEE Transactions on Systems, Man and CyberneticsPart B, 26(1), 1-13, 1996.

[19] R.D. Dutton and R.C. Brigham, *A new graph coloring algorithm*, Computer Journal 24, 85-86, 1981.

[20] C. Fleurent and J.A. Ferland, *Genetic and hybrid algorithms for graph coloring*, Annals of Operationals Research 63, 437-461, 1996.

[21] J. Fonlupt and A. Zemirline, *Polynomial recognition algorithm of perfect* $K_4 - e$*-free graphs*, Institut IMAG Rapport Technique, RT-16, 1987.

[22] P. Galinier and J.K. Hao, *hybrid evolutionary algorithms for graph coloring*, Journal of combinatorialn optimization, 379-397, 1999.

[23] M. Garey and D.S. Johnson, *Computer and Intractability*, Freeman, san Francisco, 1979.

[24] F. Glover, *Future paths for integer programming and links to artificial intelligence*, Computers and Operations Research, 13(5), 533-549, 1986.

[25] F. Glover and Laguna, *Tabu search*, Kluwer Academic Publishers, 1997.

[26] A. Hajnal, J. Suranyi, *Uber die auflosung von graphen in vollstandige Teilgraphen*, Annals of University of Sciences Budapest, *Eötvös* sect.Math. 113-121 ,1958.

[27] *G.Hajös*, *Uber eine Art Von Graphen*, Int.Math.Nachr.11, Problem 65 , 1957.

[28] P.L. Hammer and F. Maffray , *Perfect graph*, Combinatorica, 199-208, 1993.

[29] *Földes* et P.L. Hammer, *Split graph*, Proc. 8^{th} South Eastern Conference on Combinatorics Graph Theory, 311-315, 1977.

[30] P.L. Hammer and B. Simeone, *The Splittance of a graph*, Combinatorial 1, 375-384, 1981.

[31] J.K. HAO, P. Galinier and M. Habib, *Méta-heuristiques pour l'optimisation combinatoire et l'affectation sous contraintes*, Revue d'Intelligence Artificielle, 1999.

[32] B. Hayward, *weakly triangulated*, J. Comb. Theory , B39, 200–2O8, 1985.

[33] A. Hertz, *Application des métaheuristiques à la coloration des sommets d'un graphe*, 2002.

[34] A. Hertz, *A fast algorithm for coloring Meyniel graphs*, J. Comb. Theory,B50, 231-240, 1990.

[35] A. Hertz and Cosine, *A new graph coloring algorithm*, Operation Research Letters 10, 411-415, 1991.

[36] A. Hertz and D. de Werra , *Using tabu search technique for graph coloring*, Computing 39, 345-351, 1987.

[37] A. Hertz and D. de Werra, *The tabu search metaheuristic : how we used it*, Annals of Mathematics and Artificial Intelligence 1, 111-121, 1990.

[38] A. Hetz and N. Zufferey, *A new ant colony algorithm for the graph coloring problem*, Proceedings of the Workshop on Nature Inspired Cooperative Strategies forOptimization, NICSO 2006, June 29 – 30, Granada, Spain, pp. 51–60, 2006.

[39] J.H. Holland, *Adaptation in natural and artificial systems*, The university of Michigan, 1975.

[40] D.S. Johnson, C.H. Papadimitriou and M. Yannakakis, *How easy is local search ?*, Journal of Computer and systems sciences 37(1), 79-100, 1988.

[41] D.S. Johnson, C.R. Aragon, L.A. Mcgeoch et C. Schevon, *Optimisation annealing : an experimental Evaluation*, Part II, Graph Coloring and Number Partioning, operational Research 39, p 378-406, 1991.

[42] R.M. Karp, *Reducibility Among Combinatorial Problems*, Complexity of computer computations, Plenum Press, New York, 85-103, 1972.

[43] *D.König*, *Theory der endlichen und unendlichen graphen*, Chelsea, New York, 1950.

[44] P. Lacomme, C. Prins and M. Sevaux, *Algorithmes de graphes*, EYROLLES, 2007.

[45] L. Lovasz, *Normal Hypergraphs and the Perfect Graph Conjecture*, Discrect Math.2, 253–267, 1972.

[46] L. Lovasz, *A Characterisation of Perfect Graph*, J.Comb, Theory, B3, ,95-98, 1972.

[47] F. Maffray and M. Preissmann, *Split-neighbourhood graph and the strong perfect graph conjecture*, Journal of Combinatorial Theory, Serie B, 294-308, 1995.

[48] H. Meyniel, *On the perfect graph conjecture*, Discrete Mathematics 16, 339-342, 1987.

[49] S. Olariu,*Paw-free graphs*, Information Processing Letters, 28, 53-54, 1988.

[50] C.H. Papadimitriou and K. Steiglitz, *Combinatorial optimisation*, Algorithm and Complexity, Prentice Hall, New Jersey, 1982.

[51] K.R. Parthasarathy and G. Ravindra, *The strong perfect graph conjecture is true for $K_{1,3}$-free graphs*, J.Comb.Theory, B21, 212-223, 1976.

[52] K.R. Parthasarathy and G. Ravindra, *The validity of the strong perfect graph conjecture for $(K_4 - e)$-free graphs*, J. Comb. Theory, B26, 98–100, 1979.

[53] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour and K. *Vuškovič*, *Recognizing Berge graphs*, Combinatorica 25, 143-186, 2005.

[54] H. Sachs, *On the Berge conjecture concerning perfects graphs*, Combinatorial Structures and their Applications , Gordon and Breach, New York, 377-384, 1970.

[55] D. Seinche, *On a proprerty of class of n-colorable graphs*, Journal of Combinatorial Theory, B16, 191–193, 1974.

[56] C. Shannon, *The zero error capacity of a noisy channel*, IRE Trans. information theory, IT-2, 8-9, 1956.

[57] A.C. Tucker and D. Wilson, *An $O(n^2)$ algorithm for coloring perfect planar graphs*, Journal of algorithm 5, 60–68, 1984.

[58] A.C. Tucker, *Uniquely colorable perfects graphs*, Discretes Mathematics 44, 1983.

[59] A.C. Tucker, *The validity of the perfect graph conjecture for K_4-free graphs*, Topics on perfects graphs, Annals of Discretes Mathematics 21, 148-157 1984.

[60] A.C. Tucker, *Coloring perfect $K_4 - e$-free graphs*, J. Comb. Theory, B42, 313-318, 1984.

[61] A.C. Tucker, *A reduction procedure for coloring perfect K_4-e-free graphs*, Journal of Combinatorial theory, B43, 151-172, 1987.