,

UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET INFORMATIQUE
DÉPARTEMENT DE MATHÉMATIQUES

Mémoire de Master

Option : Contrôle et Analyse de système

Présenté par : Mlle Maroua BESSAFI
Mlle Karima BEKOUIDER

Le problème du recouvrement d'un graphe par des sommets

Président		**U.**
Examinateur	**BELGACEM. R**	**U. CHLEF**
Encadreur	**ABLAOUI .H**	**U. MOSTAGANEM**

Année universitaire : 2015-2016

Table des matières

Remerciments

Avant tout, nous devons remercier ALLAH le Tout Puissant de nous avoir donné le courage et la patience de mener à bien ce travail.

Nous tenons à remercier chaleureusement notre encadreur M.ABLAOUI, pour le sujet qu'il nous a proposé et le temps qu'il nous a consacré à la direction de ce mémoire et les précieux encouragements durant cette année.

Nos remerciements vont aussi à tous les enseignants du département mathématiques, qui ont participé à notre formation.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Maroua BESSAFI

Karima BEKOUIDER

INTRODUCTION

Le problème de recouvrement minimal d'un graphe est un problème NP, où l'objectif est d'extraire un sous ensemble de cardinalité minimal de sommets pour couvrir toutes les arêtes du graphe. Richard Karp [9] a été le premier à avoir prouvé que le problème de détermination d'une couverture minimale par des sommets est un problème NP-complet : Cela veut dire que l'on ne connait pas d'algorithmes efficaces (en un temps raisonnable) de détermination d'une couverture minimale de l'ensemble des arêtes par des sommets. D'où le recours à des méthodes approchées de résolution du problème : on peut se contenter d'une solution approximative aussi proche que possible de la solution optimale. Il arrive souvent, que l'on préfère une solution approchée à moindre frais qu'une solution optimale très couteuse. Chvatal a été le premier à avoir proposé un algorithme d'approximation basé sur une sélection aléatoire d'un sommet faisant parti de la solution MVC (CHVATAL, 1979). Plus tard, cette méthode a été modifié par Clarkson[9] en commençant par sélectionner d'abord un noeud de degré maximum pour enfin proposer sa nouvelle approche connue par la méthode MDG (le degré maximal gourmand)[2]. L'approche MIS[9] (maximum independent set) a été présentée par (Halldorsson et Radhakrishnan, 1997), et est basée sur la sélection d'un nœud avec un degré minimum. Singh et Gupta (2006) ont proposé une approche hybride qui combine l'algorithme génétique avec une heuristique.

Dans ce travail, nous présentons l'algorithme dit de liste de gauche (Avis et Imamura, 2007)[1], une méthode appelée liste droite (Delbot & Laforest, 2008)[1], qui utilise une approche similaire au premier et enfin une méthode de suppression de sous ensemble.

Notre travail se divise en qautre chapitre :

Le premier chapitre est consacré aux rappels et notations de base utilisés pour faciliter la compréhension.

Dans **le deuxième chapitre**, nous introduisons le problème de recouvrement minimal par des sommets.

Le troisième chapitre est consacré à la description de quelques méthodes de résolution, puis **au dernier chapitre** nous présentons quelques tests numériques.

Résumé

Dans ce mémoire, on s'intéresse au problème du recouvrement de l'ensemble des arêtes d'un graphe par des sommets. Le problème étant difficile à résoudre, donc on va se concentrer sur les méthodes approchées de résolutions : quatre heuristiques seront présentées plus loin. Toutes ces méthodes ont été programmées en language Matlab et une étude comparative a été faite pour pouvoir apprécier de l'efficacité de ces méthodes.

Chapitre 1

Rappels et notations

Dans cette partie, nous donnons quelques rappels sur la théorie des graphes (définitions et notations), qui sont nécessaires pour la bonne compréhension de ce qui va suivre.

1.1 Les Graphes

Définition 1.1.1 *Un graphe $G = (X, U, e)$ est défini par la donnée :*

1. D'un ensemble fini $X = \{v_1 , v_2 , \cdots , v_n\}$ dont les éléments sont appelés sommets (ou noeuds).
2. D'un ensemble fini $U = \{u_1, u_2, ..., u_m\}$ dont les éléments sont appelés arêtes.
3. D'une application e qui associe à toute arête u_k ses extrémités v_i et v_j et l'on note $u = (v_i, v_j)$ (ou $u = (v_i v_j)$).

Remarque 1.1.1 *Une arête $u_k = (v_i, v_i)$ est dite boucle si $u_k = (v_i, v_j)$ est une arête de U on dira que v_i et v_j sont adjacents, que v_i est adjacent à u_k.*

Remarque 1.1.2 *Par abus d'écriture, on notera un graphe $G = (X, U, e)$ par $G = (X, U)$ tout simplement.*

Définition 1.1.2 *L'ordre d'un graphe G est égal au cardinal de l'ensemble de ses sommets.*

Définition 1.1.3 *Le degré d'un sommet v_i de X, noté $d(v_i)$, est défini par : $d(v_i) = card\{ u_k, u_k \in U \ / \ u_k$ adjacent à $v_i\}$.*

Définition 1.1.4 *La matrice d'adjacence A_G d'un graphe G est une matrice booléenne de taille $n \times n$ définie par :* $A_G = \begin{cases} 1 \text{ si l'arête } (v_i,v_j) \in U \\ 0 \text{ sinon} \end{cases}$.

Définition 1.1.5 *La matrice d'icidence I_G d'un graphe G est une matrice booléenne de taille $n \times m$ définie par :* $I_G = \begin{cases} 1 \text{ si le sommet } v_i \text{ est une extrémité de l'arête } u_k \\ 0 \text{ sinon} \end{cases}$.

Définition 1.1.6 *Une chaine C reliant deux sommets v_i et v_j est une séquence d'arêtes $(u_1, u_2, ..., u_m)$ telle que : v_i est adjacent à u_1 , les arêtes u_i et u_{i+1} sont adjacentes et enfin v_j est une extémité de u_m. Une chaine est simple si elle ne passe qu'une seule fois par chacune de ses arêtes.*

Définition 1.1.7 *Soit $G = (X, U)$ un graphe et soit $A \sqsubset X$. une arête u_k est dite couverte par A si cette dernière est adjacente à un sommet de A.*

Notation : Pour tout ensemble E, on note par $|E|$ le cardinal de E.

1.1.1 Différents types de graphes

Graphe simple et multigraphe

Définition 1.1.8 *Un graphe est simple si deux sommets quelconques de X sont reliés par au plus une arête et il est dit un multi-graphe dans le cas contraire.*

Graphe complet

Définition 1.1.9 *Un graphe est complet si pour toute paire de sommets v_i et v_j de G il existe une arête u_k qui les relient.*

graphe connexe

Définition 1.1.10 *Un graphe est connexe si il existe toujours une chaine entre deux sommets quelconques.*

Graphe biparti

Définition 1.1.11 *Un graphe biparti si l'ensemble de ses sommets X peut être partitionné en deux sous-ensembles A et B de sorte que toutes les arêtes du graphe relient un sommet dans A à un sommet dans B uniquement.*

Graphe régulier

Définition 1.1.12 *Un graphe est régulier si tous les sommets ont le même degré. Si le degré commun est r, alors on dit que le graphe est r-régulier.*

1.1.2 Quelques propriétés des graphes

1. La somme des degrés des sommets d'un graphe est égal à 2 fois son nombre d'arêtes.
2. Le nombre de sommets de degré impair d'un graphe toujours est pair.
3. Le nombre d'arêtes incidentes à un sommet est le nombre d'arêtes sortant ou rentrant du sommet.
4. Deux sommets d'un graphe sont voisins ou adjacents s'ils ont au moins une arête incidente en commun.
5. Deux arêtes d'un graphe sont adjacentes si elles ont au moins un sommet en commun.
6. Un sommet est isolé si $d(x) = 0$.
7. Un sommet est pendant si $d(x) = 1$.

Chapitre 2

Problème de Recouvrement Minimal

2.1 Notre problème

Définition 2.1.1 *Soit G=(V, E) un graphe simple non orienté. Un sous-ensemble de sommets C, $C \subseteq V$, est appelé couverture minimale, si toute arête e de E possède au moins une de ses extrémités qui est dans C.*

2.1.1 Formulation mathématique du problème

Etant donné un graphe $G = (V, E)$. Le problème de recouvrement de cardinalité minimal (qu'on notera dans tout ce qui suit par MVC) peut être formulé par le programme linéaire (P) suivant :

$$(P)\begin{cases} \min \sum_{v \in V} x_v & (1) \\ x_v + x_u \geq 1; \forall\,(u,v) \in E & (2) \\ x_v \in \{0,1\}\,; \forall v \in V & (3) \end{cases}.$$

$$x_v = \begin{cases} 1, \text{ si } v \in C \\ 0, \text{ si } v \notin C \end{cases}.$$

(1) :fonction objectif qui consiste à minimiser le nombre de sommets.

(2) :Toutes les arrêtes de E sont couvertes.

(3) :Contraite d'intégrité.

2.1.2 Exemple pratique

Problème de Surveillance

Question : Où placer **optimalement** des gardiens (ou des caméras orientables) pour surveiller toutes les galeries ?

On peut représenter le musée par un graphe non orienté G = (X, E) où X est l'ensemble des croisements de galerie et une arête joint deux croisements d'une même galerie. C'est **un Problème de Recouvrement** des arêtes par les sommets : il s'agit de trouver un sous-ensemble C de sommets tel que toute arête e ait au moins une extrémité dans C (i.e : $e \cap C \neq \emptyset$). **Un Recouvrement** est un recouvrement de cardinal minimal, c'est à dire ayant un minimum de sommets.

2.1.3 La complexité du problème

Comme il a été déja signalé auparavant, le problème (MVC) est un problème NP-complet [9]. Si l'on essaie de résoudre naivement le problème en utilisant la méthode d'énumération complète il nous faut un nombre d'itérations de l'ordra de $O(2^n)$: en effet il faut d'abord tester si un seul sommet peut couvrir le graphe : cela nécessite n opérations ou C_n^1. Puis voir si deux sommets peuvent couvrir le graphe : cela nécessite C_n^2 opérations. Ensuite tester si trois sommets peuvent couvrir le graphe, il faut encore C_n^3 opérations, et de manières générale pour que k sommets couvrent le graphe il nous faut C_n^k opérations. Au total la méthode nécessite $(2^n - 1)$ opérations qui est de l'ordre $O(2^n)$.

Chapitre 3

Méthodes de résolution

Dans cette partie, nous présentons quelques méthodes heuristiques de résolution du problème MVC : l'algorithme LL dit "liste gauche" (ou en anglais : List-left), l'algorithme LR appelé "liste droite" (List-Right), l'algorithme de Greedy MDG (Maximum Degree Greedy) et l'algorithme SE dit "Elimination de Sous ensemble" (ou en englais : Subset Elimination).

Les heuristiques

Définition 3.0.2 *Règles empiriques simples qui ne sont pas basées sur l'analyse scientifique. Elles sont basées sur l'expérience et les résultats déjà obtenus et sur l'analogie pour optimiser les recherches. Généralement, on n'obtient pas la solution optimale mais une solution approchée.*

Algorithme de liste

Définition 3.0.3 *Un algorithme de liste parcourt les sommets du graphe un par un et dans un ordre déterminé à l'avance (une liste). Lorsque l'algorithme traite un sommet, il prend une décision irrévocable à son sujet (soit le sommet fait partie de la solution, soit il n'en fait pas partie).*

Une liste

Définition 3.0.4 *Soit $\mathcal{L}$ une liste quelconque (c'est-à-dire une permutation quelconque des sommets du graphe).*

3.1 Algorithme (LL) : Liste Gauche

C'est une méthode gloutonne : le principe de l'algorithme est, étant donné une liste $\mathcal{L}$, de parcourir cette dernière de la gauche vers la droite et de choisir si un sommet peut faire parti de la solution ou pas. Il a un rapport d'approximation $r = (\frac{\sqrt{\triangle}}{2}) + \frac{3}{2}$, où $\triangle$ est le degré maximum de graphe.

Soit de manière formelle l'algorithme (LL) :

Soit $G = (V, E)$ un graphe simple non-orienté et $\mathcal{L}$ une liste associée à G.
Algorithme LL
Debut
Donner n (le nombre des sommets)
Donner m (le nombre d'arêtes)
Donner A (la matrice d'ajacence associé à ce graphe)
Prendre une liste $\mathcal{L}$ (cette liste est prise aléatoirement de 1 à n)
Poser : $C \leftarrow \varnothing$; ($C \sqsubseteq V$, l'ensemble constituant la couverture)
pour i variant de 2 à n faire (parcourir la liste $\mathcal{L}$ de la gauche vers la droit)
  $k \leftarrow 1$;
  tant que($k <= i - 1$) (tester les sommets qui sont à gauche de le ième sommet)
    si (($A(\mathcal{L}(i), \mathcal{L}(k)) = 1$)) (si le *ième* sommet a un voisin à gauche)
      si ($\mathcal{L}(k)$ n'est pas dans C)
        $C \leftarrow C \cup \{\mathcal{L}(i)\}$ (alors le sommet de la position i entre à C)
      fin si
    fin si
    $k \leftarrow k + 1$;
  fin tant que
fin pour
fin algorithme

Exemple 3.1.1 *On prend l'exemple de 7 sommets avec la matrice d'ajacence suivante :*

$$\text{D}= \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

n=7 (le nombre des sommets), m= 12 (le nombre d'arêtes).

$\mathcal{L} = \{1, 2, 3, 4, 5, 6, 7\}$.

$U = \{(1,2), (1,3), (1,4), (2,4), (3,4), (3,5), (3,6), (4,5), (4,7), (5,6), (5,7), (6,7)\}$ l'ensemble des arêtes.

$C = \varnothing$ la solution courante.

On parcourt $\mathcal{L}$ de la gauche vers la droite :

1• le sommet 2 admet un sommet adjacent à gauche n'est pas dans C (le sommet 1), alors le sommet 2 entre à C et $C = \{2\}$.

2• le sommet 3 admet un sommet adjacent droit n'est pas dans C (le sommet 1), alors le sommet 3 entre à C et $C = \{2, 3\}$.

3• le sommet 4 admet un sommet adjacent droit n'est pas dans C (le sommet 1), alors le sommet 4 entre à C et $C = \{2, 3, 4\}$.

4•le sommet 5 n'admet pas un sommet adjacent droit n'est pas dans C, alors le sommet 3 n'entre pas à C et $C = \{2, 3, 4\}$.

5•le sommet 6 admet un sommet adjacent droit n'est pas dans C (le sommet 5), alors le sommet 6 entre à C et $C = \{2, 3, 4, 6\}$.

6•le sommet 7 admet un sommet adjacent droit n'est pas dans C (le sommet 5), alors le sommet 7 entre à C et $C = \{2, 3, 4, 6, 7\}$.

Alors la couverture finale est $C = \{2, 3, 4, 6, 7\}$.

3.2 Algorithme (LR) : Liste Droite

Contrairement à l'algorithme (LL), cet algorithme, qui est basé sur le même principe, parcourt la liste de droite vers la gauche et il a pour rapport d'approximation $r = \triangle$.

Soit $G = (V, E)$ un graphe simple non-orienté , $\mathcal{L}$ une liste associée à G.

Algorithme LR
Debut
Donner n (le nombre des sommets)
Donner m (le nombre d'arêtes)
Donner A (la matrice d'ajacence associée à ce graphe)
Prendre une liste $\mathcal{L}$ ($\mathcal{L}$ est prise aléatoirement de 1 à n)
Poser $C \leftarrow \varnothing$; ($C \sqsubseteq V$, une couverture minimale du sommets)
pour i variant de $n-1$ à 1 par pas égal à (-1) faire (parcourir la liste $\mathcal{L}$ de la droite vers la gauche)
  $k \leftarrow n-1$;
  tant que $(k >= i+1)$ (tester les sommets qui sont à droite de le ième sommet)
    si $((A(\mathcal{L}(i), \mathcal{L}(k)) = 1))$ (si le *ième* sommet a un voisin à droite)
      si ($\mathcal{L}(k)$ n'est pas dans C)
        $C \leftarrow C \cup \{\mathcal{L}(i)\}$; (alors le sommet de la position i entre à C)
      fin si
    fin si
    $k \leftarrow k-1$;
  fin tant que
fin pour
fin algorithme

Exemple 3.2.1 *On prend l'exemple de 7 sommets avec la matrice d'ajacence suivante :*

$$\text{D}=\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

n=7 (le nombre des sommets), m= 12 (le nombre d'arêtes).

$\mathcal{L}$={1,2,3,4,5,6,7} l'ensemble des sommets.

U={(1,2),(1,3),(1,4),(2,4),(3,4),(3,5),(3,6),(4,5),(4,7),(5,6),(5,7),(6,7)} l'ensemble des arêtes.

C=$\varnothing$ la solution courante.

On parcourt $\mathcal{L}$ de la droite vers la gauche ($7 \rightarrow 1$) :

1• le sommet 6 admet un sommet adjacent droit n'est pas dans C (le sommet 7), alors le sommet 6 entre à C et $C = \{6\}$.

2• le sommet 5 admet un sommet adjacent droit n'est pas dans C (le sommet 7), alors le sommet 5 entre à C et $C = \{6, 5\}$.

3• le sommet 4 admet un sommet adjacent droit n'est pas dans C (le sommet 7), alors le sommet 4 entre à C et $C = \{6, 5, 4\}$.

4•le sommet 3 n'admet pas un sommet adjacent droit n'est pas dans C, alors le sommet 3 n'entre pas à C et $C = \{6, 5, 4\}$.

5•le sommet 2 n'admet pas un sommet adjacent droit n'est pas dans C , alors le sommet 2 n'entre pas à C et $C = \{6, 5, 4\}$.
6•le sommet 1 admet un sommet adjacent droit n'est pas dans C (le sommet 2 et 3), alors le sommet 1 entre à C et $C = \{6, 5, 4, 1\}$.
Alors la couverture finale est $C = \{6, 5, 4, 1\}$.

Le théorème suivant montre que l'algorithme de List-Right renvoie une couverture des sommets minimal pour l'inclusion :

Théorème 3.2.1 *L'algorithme de Liste Droite (LR) donne une couverture des sommets de G qui est minimale (au sens de l'inclusion).*

Preuve. Soient $e = uv$ une arête de G et C la solution courante. Supposons, sans perte de généralité que u se trouve à gauche de v dans $\mathcal{L} = ...u...v....$ Lorsque l'algorithme de LR considère le sommet u : si v est déjà dans C alors l'arête e est couverte (au moins par v) ; sinon, u possède un voisin droit "v" qui n'est pas dans C donc u est ajouté dans C ; dans ce cas, e est couvert par u. C est donc une couverture des sommets de G. De plus, comme C ne peut contenir un sommet u et tout son voisinage (un sommet u ne peut être ajouté à la solution uniquement s'il possède un voisin droit qui n'est pas dans C), C est minimal pour l'inclusion. □
Il y'a aussi un autre théorème qui fait la comparaison entre les deux algorithmes (LL et LR), mème il montre que la couverture donnée par l'algorithme de LR est de cardinalité minimale que celle donnée par l'algorithme LL.

Théorème 3.2.2 *Pour tout liste, l'algorithme de Liste Droite donne une couverture de sommet dont la taille est inférieure ou égale à celle donnée par l'algorithme de Liste Gauche* (*ie :* $\forall \mathcal{L}, |C_R(\mathcal{L})| \leq |C_L(\mathcal{L})|$).

Preuve. Un sommet est dit monotone gauche s'il possède au moins un voisin gauche et aucun voisin droit. Soit $\mathcal{L}$ une liste quelconque. On note $LM(\mathcal{L})$ l'ensemble des sommet monotones gauche de $\mathcal{L}$ et $ISOL(\mathcal{L})$ l'ensemble des sommets isolés (c'est-à-dire l'ensemble des sommets de degré 0). Soit $C_R(\mathcal{L})$ (resp $C_L(\mathcal{L})$) l'ensemble des sommets donnés par l'algorithme de Liste Droite (resp : Liste Gauche) lorsqu'il s'éxécute sur la liste $\mathcal{L}$. Il est facile de

voir que $C_R(\mathcal{L})$ ne contient aucun sommet de $ISOL(\mathcal{L})$, ni aucun sommet de $LM(\mathcal{L})$ car une condition nécessaire pour qu'un sommet soit ajouté dans $C_R(\mathcal{L})$ est qu'il possède au moins un voisin droit, ce qui n'est ni le cas des sommets de $ISOL(\mathcal{L})$ ni le cas des sommets de $LM(\mathcal{L})$.

Soit $W = \mathcal{L} - C_R(\mathcal{L}) - ISOL(\mathcal{L}) - LM(\mathcal{L})$. Par définition, W ne contient aucun sommet de $C_R(\mathcal{L})$, de $ISOL(\mathcal{L})$, ou de $LM(\mathcal{L})$. Nous avons vu que $C_R(\mathcal{L})$ ne contient aucun sommet de $ISOL(\mathcal{L})$, ni aucun sommet de $LM(\mathcal{L})$. Et par définition, $LM(\mathcal{L})$ ne contient aucun sommet de $ISOL(\mathcal{L})$. Les ensembles $W, C_R(\mathcal{L}), ISOL(\mathcal{L})$ et $LM(\mathcal{L})$ forment donc une partition de $\mathcal{L}$.

Pour montrer le théorème, nous allons montrer que $C_L(\mathcal{L}) = C_R(\mathcal{L}) \cup W$. Clairement, $C_L(\mathcal{L})$ ne contient aucun sommet de $ISOL(\mathcal{L})$ car une condition nécessaire pour qu'un sommet soit ajouté dans $C_L(\mathcal{L})$ est qu'il possède au moins un voisin droit. Soit u un sommet de $LM(\mathcal{L})$. Comme l'algorithme de Liste Gauche parcourir la liste de la gauche vers la droite, lorsque u est considéré, tous ses voisins (qui sont tous des voisins gauches) ont déjà été parcourus et ajoutés dans C la solution courante (car lorsqu'ils ont été considérés, u été un voisin droit qui n'était pas encore dans C). Ainsi, u ne possède pas de voisin qui ne soit pas déjà dans C et donc u n'est pas ajouté dans C (et il ne sera pas ajouté plus tard car toute décision est irrévocable). Les arguments précédents montrent que $C_L(\mathcal{L})$ ne contient aucun sommet de $ISOL(\mathcal{L}) \cup LM(\mathcal{L})$. Considérons maintenant les autres sommets de la partition de $\mathcal{L}$: l'ensemble $C_R(\mathcal{L}) \cup W$. Tout sommet u dans cet ensemble possède au mois un voisin droit w. Lorsque u est considéré, w n'est pas encore dans C puis qu'il n'a pas encore été parcouru ; ainsi l'algorithme de Liste Gauche va ajouter u dans C.

Tous les arguments précédents montrent que $C_L(\mathcal{L}) = C_R(\mathcal{L}) \cup W$. Comme $C_R(\mathcal{L}) \subseteq C_R(\mathcal{L}) \cup W$. On a $C_R(\mathcal{L}) \subseteq C_L(\mathcal{L})$ et donc $|C_R(\mathcal{L})| \leq |C_L(\mathcal{L})|$, ce qui prouve le théorème. □

3.3 Algorithme (MDG) : méthode gloutone

L'heuristique est basée sur la considération suivante :

Soit G un graphe possédant un sommet de degré supérieur à k, avec k la taille d'une couverture minimum de G. Supposons qu'un algorithme choisisse de ne pas sélectionner le sommet de degré maximum. Il va donc être obligé de sélectionner tout son voisinage, sinon certaines

arêtes ne seraient pas couvertes. Or, la taille du voisinage étant supérieure à k, l'algorithme est sûr de ne pas retourner une solution optimale. Outre le fait que Maximum Degree Greedy maximise le nombre d'arêtes couvertes à chaque étape, sélectionner tous les sommets dont le degré est supérieur à la taille de la couverture minimum semble être une bonne idée [2].
Le but de cet algorithme est de trouver C, un sous ensemble de V, qui soit une couverture minimale du graphe G.
Le principe de cet algorithme est de selectionner, à chaque étape, un sommet de la liste $\mathcal{L}$ qui soit de degré maximum. Le rapport d'approximation de MDG est $r = \log(\triangle)$.

Soit $G = (V, E)$ un graphe simple non-orienté, $\mathcal{L}$ une liste associée à G.
Algorithme MDG
Début
Donner n (le nombre des sommets)
Donner m (le nombre d'arêtes)
Donner D (la matrice d'ajacence associé à ce graphe)
Prendre une liste $\mathcal{L}$ (cette liste contient des sommets de V, est prise aléatoirement de 1 à n)
$C \leftarrow \varnothing$; ($C \sqsubseteq V$, une couverture minimale du sommets)
$E \leftarrow \varnothing$; (l'ensemble des arêtes)
$k \leftarrow 1$;
tant que $k <= n$
  pour $i = 1$ à n
    si $(D(\mathcal{L}(i), \mathcal{L}(k)) = 1)$ (si le *ième* sommet a un voisin à gauche)
      $E \leftarrow E \cup \{(\mathcal{L}(i), \mathcal{L}(k))\}$ (ajouter l'arête $(\mathcal{L}(i), \mathcal{L}(k))$ à l'ensemble E)
    fin si
  fin pour
  $k \leftarrow k + 1$
fin tant que
calculer le degré du chaque sommet de $\mathcal{L}$
tant que $E \neq \varnothing$ (tant qu'il existe des arêtes non couvertes)
  pour $i = 1$ à n
    sélectionner le sommet $\mathcal{L}$(i) de degré maximum
      $C \leftarrow C \cup \{\mathcal{L}(i)\}$ (le sommet de la position i entre à C)
      $\mathcal{L} \leftarrow \mathcal{L} - \{\mathcal{L}(i)\}$ (le sommet de la position i sorte de $\mathcal{L}$)
    rendre la ligne et la colonne corréspondant à $\mathcal{L}(i)$ nuls
  fin pour
fin tant que
fin algorithme

Exemple 3.3.1 *On prend l'exemple de 7 sommets avec la matrice d'ajacence suivante :*

$$\text{D}=\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

n=7 (le nombre des sommets), m= 7 (le nombre d'arêtes)

$\mathcal{L} = \{1, 2, 3, 4, 5, 6, 7\}$.

$E = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ (l'ensemble des arêtes)

$C = \varnothing$

Itération 1 :

d=[3,2,2,2,1,1,1]

on séléctionne de sommet de degré maximum (n=1)

$C = [1]$

$\mathcal{L} = \{2, 3, 4, 5, 6, 7\}$.

Itération 2 :

d=[0,1,1,1,1,1,1]

on séléctionne de sommet de degré maximum (n=2)

$C = [1, 2]$

$\mathcal{L} = \{3, 4, 5, 6, 7\}$.

Itération 3 :

d=[0,0,1,1,1,1,0]

on séléctionne de sommet de degré maximum (n=3)

$C = [1, 2, 3]$

$\mathcal{L} = \{4, 5, 6, 7\}$.

Itération 4 :

d=[0,0,0,1,0,1,0]

on séléctionne de sommet de degré maximum (n=4)

$C = [1, 2, 3, 4]$

$\mathcal{L} = \{5, 6, 7\}$.

Itération 5 :

d=[0,0,0,0,0,0,0]

$C = [1, 2, 3, 4]$

$\mathcal{L} = \{5, 6, 7\}$.
$MVC = \{1, 2, 3, 4\}$
Alors la couverture finale est MVC=$\{1, 2, 3, 4\}$.

Remarque 3.3.1 *S'il existe plusieurs sommets qui sont de degré maximum, on en choisira un quelconque (en général, pour des raisons de programmation on choisit le sommet de plus bas indice).*

3.4 Algorithme (SE) : Elimination de Sous-ensemble

Dans ce qui suit nous présentons l'algorithme SE (Subset Elimination) qui est une amélioration de l'algorithme MDG. Cet algorithme est basé sur la suppression de sous ensemble de sommets de la solution courante qui couvrent des arêtes déja couvertes par d'autres sommets. Soit de manière formelle l'algorithme SE :

Algorithme SE
Début
Donner n (le nombre des sommets)
Donner m (le nombre d'arêtes)
Donner D (la matrice d'incidence associé à ce graphe)
numéroter les arêtes dans l'odre décroissant $(u_1, u_2, \dots u_m)$
$C \leftarrow \varnothing$ (ensemble des arêtes du graphe qui sont couvertes)
$MVC \leftarrow \varnothing$ (la couverture courante)
calculer le degré de chaque sommet
tant que (il reste un arête dans le graphe original)
  poser : $MVC - noeud \leftarrow$sommet de degré maximum
  faire : $d(MVC - noeud) \leftarrow 0$
  éléminer toues les arêtes couvertes par ce sommet et ajouter toutes ses arêtes à l'ensemble C
  $MVC \leftarrow MVC \cup (MVC - noeud)$
  $compteur \leftarrow compteur + 1$
  si $|MVC| \succ 1$
    pour tout sommet $x \in MVC$
      prendre un sommet x
      soit $S_x = \{$arêtes couvertes par le sommet $x\}$
      $S_{MVC-\{x\}} = \{$arêtes couvertes par les sommets de MVC différentes à les arêtes de S_x $\}$
      si $S_x \subset S_{MVC-\{x\}}$
        $MVC = MVC - \{x\}$
        $compteur \leftarrow compteur - 1$
      fin si
    fin pour
  fin si
fin tant que
fin algorithme

Exemple 3.4.1 *On prend l'exemple de 7 sommets avec la matrice d'incidence suivante :*

$$\text{D=}\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$n = 7$ (le nombre des sommets) , $m = 13$ (le nombre d'arêtes).

$\mathcal{L} = \{1, 2, 3, 4, 5, 6, 7\}$.

$C \leftarrow \varnothing$ (ensemble des arêtes du graphe qui sont couvertes)

$MVC \leftarrow \varnothing$ (la couverture courante)

Itération1 :

d=[3,2,2,2,1,1,1] (degré de chaque sommet)

$MVC - noeud = 1$

$d(MVC - noeud) \leftarrow 0$

$C = \{u_2, u_3, u_5\}$

$MVC = \{2\}$, $|MVC| \prec 1$.

Itération 2 :

d=[0,1,1,1,1,1,1]

$MVC - noeud = 2$

$d(MVC - noeud) \leftarrow 0$

$C = \{u_1\}$

$MVC = \{2, 1\}$, $|MVC| \succ 1$

$S_1 = \{u_3, u_2, u_5\}$, $S_2 = \{u_1\}$

Itération 3 :

d=[0,0,1,1,1,1,0]

$MVC - noeud = 3$

$d(MVC - noeud) \leftarrow 0$

$C = \{u_6, u_5\}$

$MVC = \{2, 1, 3\}$, $|MVC| \succ 1$

Itération 4 :

d=[0,0,0,1,1,1,0]

$MVC - noeud = 4$

$d(MVC - noeud) \leftarrow 0$

$S_x = \{u_4, u_3\}$

$MVC = \{2, 1, 3, 4\}, |MVC| \succ 1$

Itération 5 :

d=[0,0,0,0,0,0,0]

$MVC = \{2, 1, 3, 4\}$, $|MVC| \succ 1$

$S_1 = \{u_3, u_2, u_5\}$, $S_{MVC-\{1\}} = \{u_1, u_2, u_4, u_6, u_3, u_5\}$.

si $S_x \subset S_{MVC-\{x\}}$

$MVC = MVC - \{x\}$

$MVC = \{2, 3, 4\}$.

Alors la couverture finale est MVC=$\{1, 2, 4, 5\}$.

Chapitre 4

Tests numériques

Nous avons programmé les différentes heuristiques citées plus haut en language MATLAB sur un pc hp : processeur intel(R) core(TM) i3-2328M, RAM : 4 Go. Les exemples ont été tirés des instances de **DIMAC the Maximum clique probleme.**

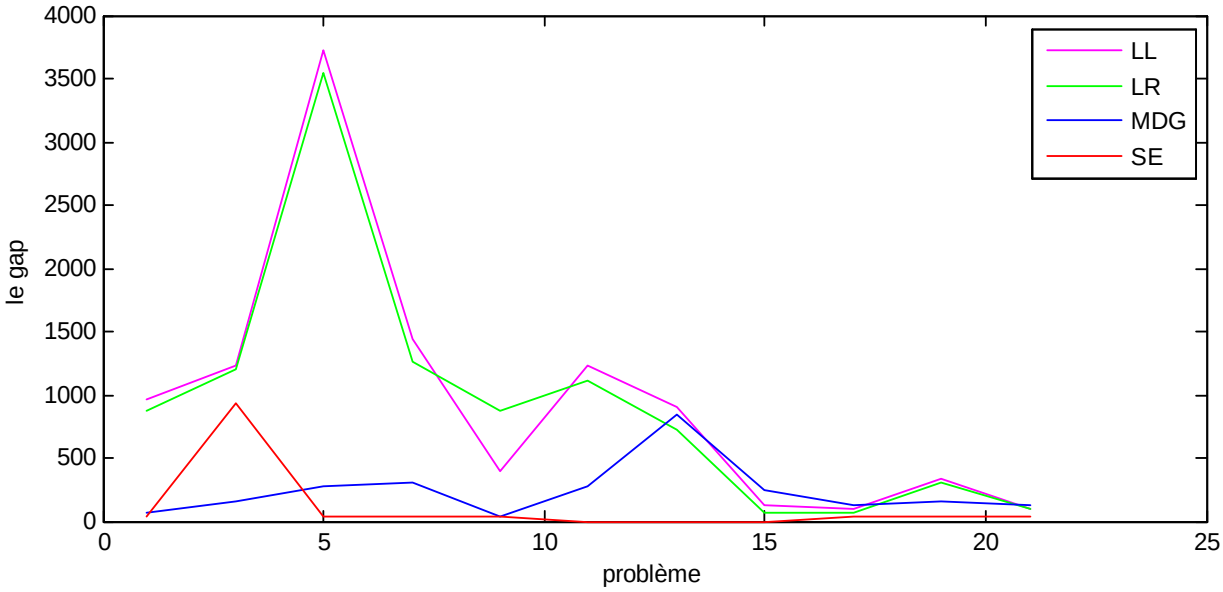
Et nous avons appliqué ces méthodes sur l'exmple choisi au chapitre 2 (Problème de Surveillance).

			LL		LR		MDG		SE	
problème	N	sol-opt	MVC	CPU	MVC	CPU	MVC	CPU	MVC	CPU
c 125	125	91	123	0.23	120	0.27	93	51.90	92	0.23
c250.g	250	206	247	0.54	246	0.77	211	285.12	237	0.87
mann_a27	378	252	376	1.62	370	1.96	261	544.06	253	3.66
c500.g	500	443	491	12.33	485	3.01	453	734.69	444	4.29
phat300_1	300	292	279	0.29	263	0.88	393	172.16	293	0.94
gen200p0.944	200	156	197	0.29	193	0.86	165	76.41	156	0.71
Hamming6_2	64	32	62	0.06	56	0.07	60	5.74	32	0.09
Hamming8_4	256	240	244	1.09	242	1.56	248	154.67	240	0.87
keller4	171	160	163	1.15	162	0.28	164	63.54	161	0.33
broc200_4	200	183	194	0.26	193	0.43	188	284.72	184	0.49
broc200_2	200	188	191	0.22	185	0.34	192	46.459	189	0.37

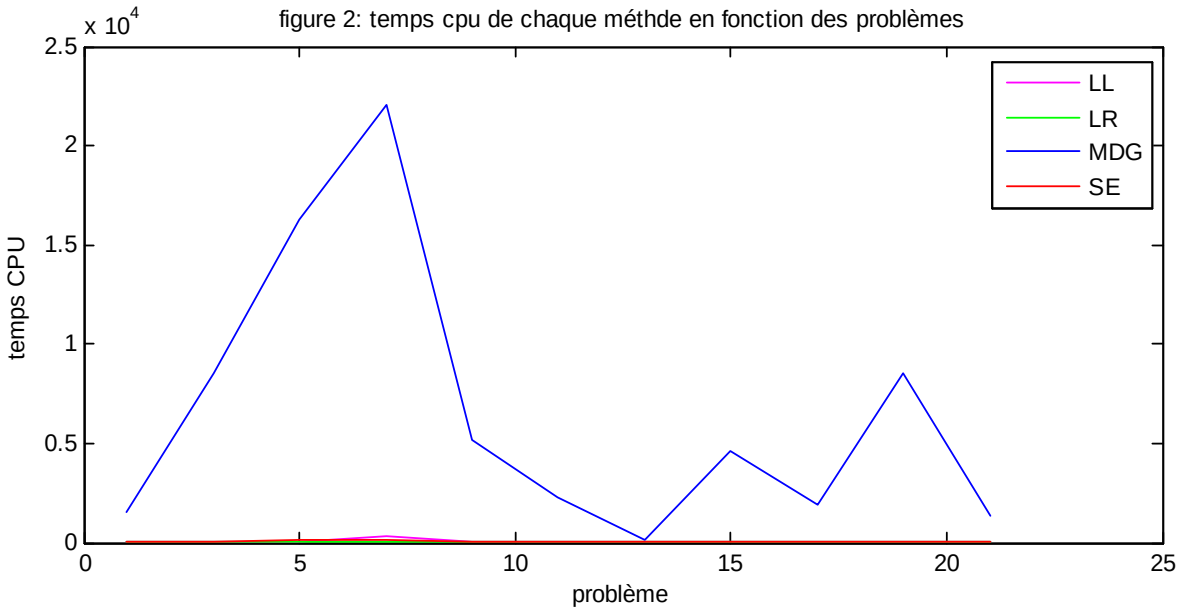
D'après les testes numériques on constate :

– Pour ce qui est de la qualité de la solution : Dans la figure suivante nous avons représenté "le gap" (différence entre la solutin obtenue et la solution optimale) en fonction des différents problèmes. Il apparait clairement la méthode (SE) donne des solutions meilleurs que les autres méthodes.

figure 1: le gap en fonction des problèmes



– Pour ce qui est du temps CPU : Nous avons représenté les temps CPU en fonction des problèmes. La méthode (LL) domine les autres méthodes mais elle nous procure des solutions de trés mauvaises qualité.Cette présentation est montré comme suit :



CONCLUSION

Le problème de recouvrement des arêtes par des sommets dans un graphe possède énormément d'applications pratiques d'où l'intérêt qu'il suscite aprés de la communauté scientifique.

Nous avons présenté quatre heuristiques de résolutions du problème pour leurs simplicités et leurs performances : la méthode (SE) semble être plus performante que les trois autres méthodes en terme de précision de la solution obtenue sur les exemples traités dans ce mémoire. Néanmoins la méthode (LL) parait plus rapide.

Mis à part qu'il faille tester ces méthodes sur des exemples de grandes taille (pour se prononcer) il parait évident que l'on doit procéder par hybridation des méthodes pour profiter des unes et des autres.

Nous pensons, à l'avenir, s'intéresser aux méthodes hybrides et pencher sur les méthodes exactes.

Bibliographie

[1] F. Delbot, C. Laforest . Information Processing Letters 107 (2008).

[2] François Delbot. Au del a de l' evaluation en pire cas : comparaison et evaluation en moyenne de processus d'optimisation pour le probl eme du vertex cover et des arbres de connexion de groupes dynamiques.. Recherche opérationnelle [cs.RO]. Université d'Evry-Val d'Essonne, 2009.

[3] D. Avis, T. Imamura. Operations Research Letters 35 (2007).

[4] Romain Campigotto. Algorithmes d'approximation a mémoire limitée pour le traitement de grands graphes : le problème du Vertex Cover. Operations Research [cs.RO]. Universit e d'Evry-Val d'Essonne, 2011..

[5] Jean-Charles Régin, Arnaud Malapert. Théorie des Graphes - 2013/2014.

[6] Préparation CAPES UCBL. Théorie des Graphes 2009/2010 .

[7] http ://gilco.inpg.fr/~rapine/Graphe/Graphe/definition.html. 30/03/2006.

[8] Imran Khan and Sangeen Khan.Experimental Comparison of Five Approximation Algorithms for Minimum Vertex Cover.2014.

[9] J.-F. Scheid.Chapitre 8 : Introduction aux méthodes heuristiques

[10] http ://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark.