Projet De Théorie des Graphes

Réalisé par : Lahyani Zakaria
Année universitaire : 2015/2016

Problème Clique Maximal

Étant donné un graphe, le problème de la clique maximale consiste à trouver le plus grands nombre de sommets qui sont deux à deux adjacent (le plus grand sous graphe complet).
il existe plusieurs algorithmes pour résoudre ce problème.

Algorithme Implémenté

```
function clique(U,size)
1:    if|U|= 0then
2:        ifsize>maxthen
3:            max:=size
4:            New record; save it.
5:            found:=true
6:        end if
7:        return
8:    end if
9:    whileU≠∅do
10:       ifsize+|U|⩽maxthen
11:           return
12:       end if
13:       i:=min{j | v_j ∈ U}
14:       ifsize+c[i]⩽maxthen
15:           return
16:       end if
17:       U:=U\{v_i}
18:       clique(U∩N(v_i),size+1)
19:       iffound=truethen
20:           return
21:       end if
22:   end while
23:   return
function new
24:   max:=0
25:   fori:=ndownto1 do
26:       found:=false
27:       clique(S_i∩N(v_i),1)
28:       c[i]:=max
29:   end for
30:   return
```

Le principe consiste a chercher à chaque itération une clique plus grande d'un sommet que la clique précédente, ça peut paraître simple, mais ça permet un gain de temps énorme sur des graphes de tailles importante.

Soit C la clique dans un graphe G, La fonction c (i) renvoi la plus grande clique dans S. Il est évident que pour tout 1 □ i □ n-1, on a c (i) = c (i + 1) ou C (i) = c (i + 1) + 1.
autrement dit, on a C (i) = c (i + 1) + 1 s'il existe une clique maximale dans S qui inclut le sommet v(i).

Description du code :

le code implémenté permet de lire le fichier binaire du graphe en entrée, et faire sortir la clique maximale trouvé.

Durant la phase d u développement, je me suis rendu compte que même si l'algorithme est simple, le traitement en objet va prendre plus de temps vu que la gestion des objets et plus coûteuse, l'avantage de c++ c'est qu'il permet l'utilisation des fonctions de c, qui sont proche au système, ce qui permet un gain de temps important. Donc afin de minimiser le temps des calculs, ce qui permet d'économiser les interactions.

La Classe Matrice : permet de lire le fichier d'entrée, garder l'adjacence entre les sommets de la clique, et faire sortir la taille de la clique maximale trouvé.

elle construit la matrice d'adjacence à partir de fichier choisi. la matrice est stocker en bits

Methodes :

afficher() : affiche la matrice construite par le constructeur.

Adjacence() : renvoie tous les sommets adjacents à un sommet.

Taille(): renvoi la taille de graphe (le nombre de ses sommets).

CliqueMax : renvoie la clique maximale

maxiC : teste si la clique peut s'étendre, si oui, elle met la clique trouvé comme clique maximum

SousGraphes :est une structure de contrôle des sous graphes testés, elle contient les méthodes nécessaires pour manipuler le graphes d'entrée.

Résultat de test :

le résultat de quelques tests effectuer s'illustre dans le tableau suivant :

Graphe	Clique	Temps
hamming10-2	512	1,30s
san200-0.7-1	30	58 min 47s
MANN-a27	126	10 min 5s
brock200_1	17	1s
sanr200_0.9	32	6m33s
c-fat200-5	33	0.0015s

Source :

http://users.aalto.fi/~pat/papers/tournament.pdf

les graphes venant de site de benchmark ci-dessous :

http://iridia.ulb.ac.be/~fmascia/files/DIMACS