Université de Bourgogne

Projet : Dégénérescence des grands graphes réels

Licence L3 Informatique - Graphes

AUGIER Pierre
MANZANO Lilian

Introduction

Dans ce projet, notre objectif principal est de réussir à mettre sous forme de code l'algorithme de calcul de la dégénérescence de graphes.

Celui-ci nous permet de déterminer la densité de graphes ou grands graphes ainsi que le numéro de centre de chaque sommet.

Voici l'explication précise de la situation :

« La notion de dégénérescence est liée à la notion de k-centre (k-core en anglais): un k-centre d'un graphe non orienté G est un sous-graphe maximal H de G où tous les sommets sont de degré au moins k dans H. Le plus grand k pour lequel G contient un k-centre non nul est sa dégénérescence. Deux autres définitions équivalentes sont qu'un graphe a une dégénérescence de k si tout sous-graphe contient un sommet de degré inférieur ou égal à k ou bien s'il existe un ordre sur les sommets tel que tout sommet a au plus k voisins qui le précèdent dans cet ordre. »

L'algorithme le plus simple permettant de calculer la dégénérescence consiste à supprimer les sommets petit à petit en partant de ceux qui sont inférieurs ou égaux à k = 1 en continuant avec k = k + 1 une fois que tous les sommets pour k sont supprimés.

Une fois qu'il n'en reste plus, nous obtenons alors la dégénérescence de notre graphe ainsi que leur numéro de centre puisqu'il s'agit de la valeur de k au moment où le sommet est supprimé.

Principaux structures et algorithmes utilisés

Pour récupérer le graphe, Nous avons créé 2 fonctions, la première «importListeAretes» permet de récupérer dans un vector<string> les lignes comportant les arêtes à partir du fichier donné.

```
vector<string> importListeAretes(string cheminFichier){
    string ligne;
    vector<string> lignes;
    ifstream fichier(cheminFichier);
    if(!fichier.is_open()){
        cerr << "Le fichier suivant '"<<cheminFichier<<"' n'a pas pu etre ouvert"<< endl;
    }
    else{
        while(getline(fichier, ligne))
        {
            lignes.push_back(ligne);
        }

        for(int i=0;i<lignes.size();i++)
        {
            ligne=lignes[i];
            if (ligne[0]!='0' && ligne[0]!='1' && ligne[0]!='2' && ligne[0]!='3' && ligne[0]!='4' && ligne[0]!='5' && ligne[0]!='6' && ligne[0]!='7' && ligne[0]!='8' && ligne[0]!='9')
            {
                lignes.erase(lignes.begin()+i);
                i--;
            }
        }
    }
    fichier.close();
    return lignes;
}
```

Ensuite, il y a la fonction «importGraphe» qui permet elle de remplir un tableau avec uniquement les nombres représentant les sommets, elle retire donc des chaines de caractères précédemment listées, les nombres en ignorant les espaces et autre caractères, puis les convertis en entier.

```
void importGraphe(int nbAretes, int aretes[][2], vector<string> lignes){
    string ligne;
    for(int i=0;i<nbAretes;i++){
        ligne=lignes[i];

        int debutNombre1=0;
        int finNombre1=0;
        while(ligne[finNombre1+1]=='0' || ligne[finNombre1+1]=='1' || ligne[finNombre1+1]=='2' || ligne[finNombre1+1]=='3' || ligne[finNombre1+1]=='4' || ligne[finNombre1+1]=='5' || ligne[finNombre1+1]=='6' || ligne[finNombre1+1]=='7' || ligne[finNombre1+1]=='8' || ligne[finNombre1+1]=='9')
            {finNombre1++;}
        int debutNombre2=finNombre1+1;
        while(ligne[debutNombre2]!='0' && ligne[debutNombre2]!='1' && ligne[debutNombre2]!='2' && ligne[debutNombre2]!='3' && ligne[debutNombre2]!='4' && ligne[debutNombre2]!='5' && ligne[debutNombre2]!='6' && ligne[debutNombre2]!='7' && ligne[debutNombre2]!='8' && ligne[debutNombre2]!='9')
            {debutNombre2++;}
        int finNombre2=ligne.size()-1;

        int tailleNombre1=finNombre1-debutNombre1+1;
        int tailleNombre2=finNombre2-debutNombre2+1;

        string nb1;
        nb1=ligne.substr(debutNombre1,tailleNombre1);
        string nb2;
        nb2=ligne.substr(debutNombre2,tailleNombre2);

        aretes[i][0]=stoi(nb1);
        aretes[i][1]=stoi(nb2);
    }
}
```

L'algorithme principal de notre programme est celui du calcul de la dégénérescence, il consiste dans un premier temps à copier notre graphe dans un tableau afin de pouvoir marquer les arêtes supprimées ultérieurement. Il nous faut aussi un tableau avec les sommets, le degré de nos sommets ainsi que le marquage de leur suppression éventuelle. Ensuite tant qu'il y a des sommets, on calcule leur degré et on regarde s'il est inférieur ou égal à k, si oui il est marqué comme supprimé et on modifie notre tableau en conséquence.

```c
int calculDege(int nbAretes, int aretes[][3], int nbSommets, int centres[][2]){

    int copieAretes[nbAretes][3];
    copierGraphe(nbAretes, aretes, copieAretes);

    int sommets[nbSommets][3];
    remplirTabSommets(nbAretes, aretes, nbSommets, sommets);

    int k=0;
    int nbNonMarques=nbSommets;
    int nbSommetSup=0;

    while(nbNonMarques>0)
    {
        k++;
        do
        {
            nbSommetSup=0;
            calculDegre(nbAretes, copieAretes, nbSommets, sommets);

            for(int i=0;i<nbSommets;i++)
            {
                if (sommets[i][2]==0 && sommets[i][1]<=k)
                {
                    printf("sommet supprime: %d avec k=%d\n",sommets[i][0],k);
                    centres[i][0]=sommets[i][0];
                    centres[i][1]=k;
                    int sommetSup=sommets[i][0];
                    sommets[i][2]=1;
                    marquerSommet(nbAretes, copieAretes, sommetSup);
                    nbSommetSup++;
                }
            }
            nbNonMarques=nbNonMarques-nbSommetSup;
        }while (nbSommetSup>0);
    }
    return k;
}
```

Pour une meilleure lisibilité et un code propre, la fonction de calcul de la dégénérescence utilise elle-même plusieurs autres fonctions, tel que par exemple des fonctions permettant de copier des tableaux, calculer les degrés de chaque sommets, ou encore marquer toutes les lignes du tableaux contenant un sommet donné.

Et pour terminer, voici comment nous dessinons le graphe sous forme de cercles avec les plus gros centres à l'intérieur et les plus petits à l'extérieur. Pour réaliser ceci, nous avons décidé d'écrire dans un fichier postscript (.ps) que nous créons s'il n'existe pas, le dessin se faisant à l'aide d'indication plutôt simple dans ce type de fichier. Puis seulement une fois le fichier postscript crée et rempli par les bonnes indications, nous le convertissons en PDF si le système le permet à l'aide de la commande système «ps2pdf».

Tout d'abord nous trions les centres à l'aide de la fonction «trierCentres», ceci facilitera le dessin du graphe, nous pourrons ainsi procéder cercle par cercle.

```c
void trierCentres(int nbSommets, int centres[][2], int degenerescence){
    for(int i=0; i<nbSommets; i++)
    {
        int minimum=degenerescence;
        int indice=-1;
        for (int j=i; j<nbSommets; j++)
        {

            if(centres[j][1]==minimum && centres[j][0]<centres[i][0])
            {
                indice=j;
            }
            if(centres[j][1]==minimum && indice== -1)
            {
                indice=i;
            }
            if(centres[j][1]<minimum)
            {
                minimum=centres[j][1];
                indice=j;
            }
        }
    int tempSommet=centres[i][0];
    int tempValeur=centres[i][1];
    centres[i][0]=centres[indice][0];
    centres[i][1]=centres[indice][1];
    centres[indice][0]=tempSommet;
    centres[indice][1]=tempValeur;
    }
}
```

Ensuite, on calcul les coordonnées de chaque sommet à l'aide de formule trigonométrique simple.

```
int nbSommetsParCentres[degenerescence];
for(int i=0;i<degenerescence;i++)
{
    int nb=0;
    for(int j=0;j<nbSommets;j++)
    {
        if(centres[j][1]==i+1)
        {
            nb++;
        }
    }
    nbSommetsParCentres[i]=nb;
}

double coordSommet[nbSommets][4];
int rayonCercle=295/degenerescence;
int indice=0;
double pi = acos(-1);
for(int i=0;i<degenerescence;i++)
{
    int angle=360/nbSommetsParCentres[i];
    for(int j=0;j<nbSommetsParCentres[i];j++)
    {
        coordSommet[indice][0]=centres[indice][0];
        coordSommet[indice][1]=centres[indice][1];
        coordSommet[indice][2]=306+((degenerescence-coordSommet[indice][1]+1)*rayonCercle)*cos(((double)j*(double)angle)/180*pi);
        coordSommet[indice][3]=391+((degenerescence-coordSommet[indice][1]+1)*rayonCercle)*sin(((double)j*(double)angle)/180*pi);
        indice++;
    }
}
```

Puis il suffit de créer si nécessaire et ouvrir le fichier .ps en écriture, et enfin y insérer le code postscript. Nous procédons par étape, d'abord dessiner les bases (point central, et cercles rouges), ensuite on met les sommets, puis les arêtes, et enfin les numéros de sommet :

```
    // Point central et cercles rouges
    fprintf(outfile,"%%!\n306 391 2.5 0 360 arc\nfill\n255 0 0 setrgbcolor\n");
    for(int i=1;i<=degenerescence;i++){
        int rayon=rayonCercle*i;
        fprintf(outfile,"%d 391 moveto\n306 391 %d 0 360 arc\n",306+rayon,rayon);
    }
    fprintf(outfile,"stroke\n0 0 0 setrgbcolor\n");

    // Sommets
    for(int i=0;i<nbSommets;i++){
        int x=coordSommet[i][2];
        int y=coordSommet[i][3];
        fprintf(outfile,"%d %d moveto\n%d %d 10 0 360 arc\nfill\n",x,y,x,y);
    }
    fprintf(outfile,"fill\nstroke\n");

// Arêtes
for(int i=0; i<nbAretes;i++){
    int ind1=0;int ind2=0;
    while(aretes[i][0]!=coordSommet[ind1][0]){
        ind1++;
    }
    while(aretes[i][1]!=coordSommet[ind2][0]){
        ind2++;
    }
    int x1=coordSommet[ind1][2];
    int y1=coordSommet[ind1][3];
    int x2=coordSommet[ind2][2];
    int y2=coordSommet[ind2][3];
    fprintf(outfile,"%d %d moveto\n%d %d lineto\n",x1,y1,x2,y2);
}
fprintf(outfile,"stroke\n");

// Numéros de sommets
fprintf(outfile,"255 255 255 setrgbcolor\n/Helvetica-Bold findfont\n10 scalefont\nsetfont\n");
for(int i=0;i<nbSommets;i++){
    int x=coordSommet[i][2]-5;
    int y=coordSommet[i][3]-3;
    int s=coordSommet[i][0];
    fprintf(outfile,"%d %d moveto\n(%d) show\n",x,y,s);
}
fprintf(outfile,"stroke");
```

Exemples d'utilisation et résultats

En premier exemple, essayons avec un graphe simple (le graphe d'exemple du sujet de projet) avec seulement 1O sommets et 17 arrêtes.

Voici le fichier que nous utiliserons avec notre algorithme :

*grapheExemple - Bloc-notes

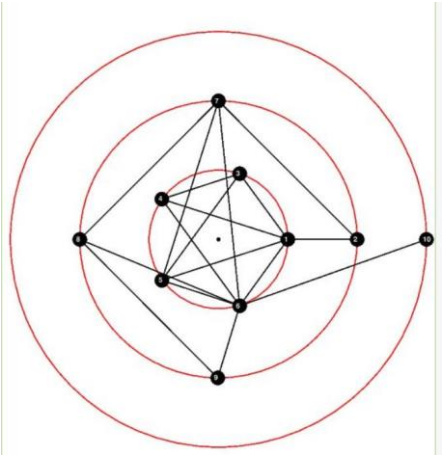
Fichier Edition Format Affichage Aide

```
1 2
1 3
1 4
1 5
1 6
2 7
3 4
3 5
4 6
5 6
5 7
6 7
6 8
6 9
6 10
7 8
8 9
```

Voici le résultat lors de l'exécution de notre programme :

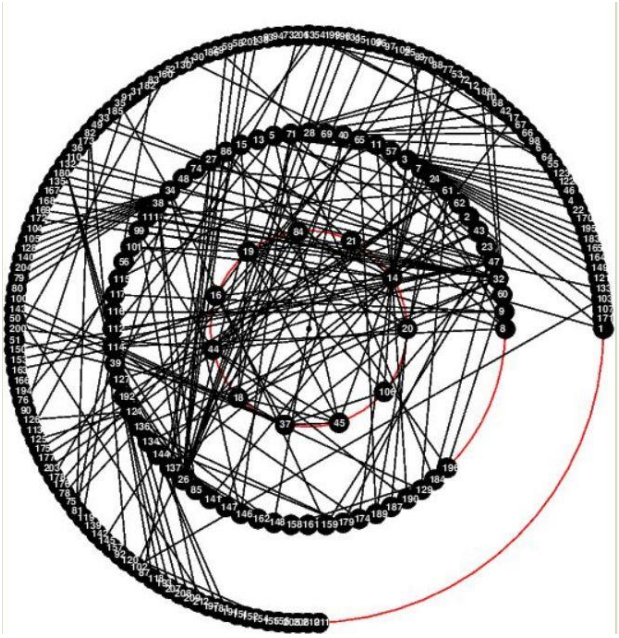
```
lm230562@aragon:/home4/lm230562/graphes/donnees$ ./projet
Entrer le chemin du fichier de donnees :
/home1/lm230562/graphes/donnees/grapheExemple.txt
sommet supprime: 10 avec k=1
sommet supprime: 2 avec k=2
sommet supprime: 9 avec k=2
sommet supprime: 8 avec k=2
sommet supprime: 7 avec k=2
sommet supprime: 3 avec k=3
sommet supprime: 4 avec k=3
sommet supprime: 5 avec k=3
sommet supprime: 6 avec k=3
sommet supprime: 1 avec k=3

La degenerescence du graphe est de 3

Tableau des centres :
Sommet: 1 2 3 4 5 6 7 8 9 10
Centre: 3 2 3 3 3 3 2 2 2 1

Le fichier .ps tente d'etre converti en .pdf si la commande 'ps2pdf' est disponible sur
votre machine
```

Et notre graphe dessiné en cercles dans le fichier créé:



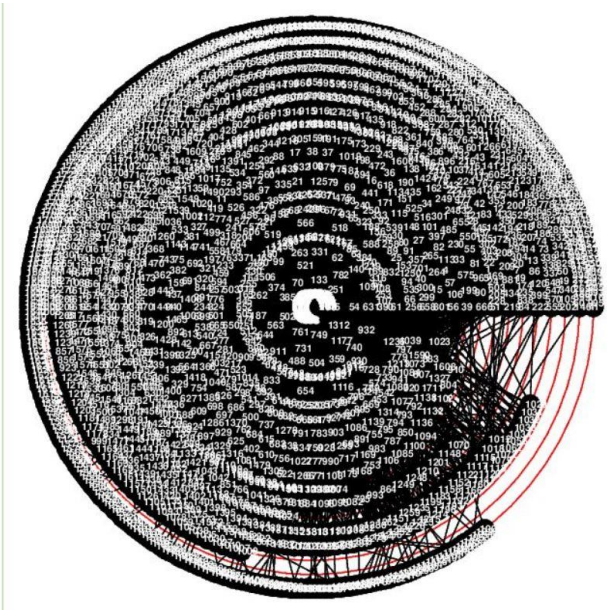
Nous pouvons donc maintenant essayer avec deux graphes plus volumineux, tout d'abord avec le fichier « out.maayan-pdzbase », qui correspond aux interactions entre des protéines (210 sommets et 244 arêtes).

Avec ce graphe nous obtenons une dégénérescence de 3, et le dessin suivant :



Pour finir, voyons encore plus grand avec le fichier « out.petster-friendships-hamster-uniq », qui correspond à des liens d'amitié entre les utilisateurs d'un site internet.

Le graphe possède 1 858 sommets et 12 534 arêtes, on obtient une dégénérescence de 20 avec le dessin suivant :



Limitations et améliorations de notre programme

Notre programme permet de gérer de grands graphes comme nous avons pu le voir dans la partie précédente, même s'il fonctionne, il pourrait certainement être davantage rapide lors de son exécution pour les graphes de grande taille.

De plus notre dessin créé peut seulement être converti en PDF si nous sommes sur un environnement Linux ou avec des librairies / API supplémentaires.

En ce qui concerne les améliorations possibles, les deux premiers points de la question 3 du sujet pourraient être implémentés pour compléter notre code.

La création d'une interface est une option pour améliorer notre projet ainsi qu'une facilité d'utilisation et d'exécution sur tous les environnements (Windows, Mac ou autres).