Notions de centralité dans les graphes

QI Zhirui//ZHU Zhihao//HUANG Bozhang

September 11, 2019

Index

Contents

1 Contexte de la centralité **3**
- 1.1 Définitions de base sur les graphes [1, 2] . . . 3
- 1.2 Définitions et les caractéristiques des centralités [3] . . . 4
  - 1.2.1 Centralité de degré . . . 4
  - 1.2.2 Centralité de proximité . . . 4
  - 1.2.3 Centralité harmonique . . . 4
  - 1.2.4 Centralité d'intermédiarité . . . 4

2 Partie algorithmique **5**
- 2.1 Structure de donnée choisie pour charger le graphe en mémoire — Liste d'adjacence . . . 5
- 2.2 Calcul de distance . . . 5
- 2.3 Calcul des centralités différentes . . . 6
  - 2.3.1 Centralité de degré . . . 6
  - 2.3.2 Centralité de proximité . . . 7
  - 2.3.3 Centralité harmonique . . . 7
  - 2.3.4 Centralité d'intermédiarité (pas terminé) . . . 8

3 Partie implémentation **9**
- 3.1 Environnement . . . 9
- 3.2 Temps de calcul . . . 9
  - 3.2.1 Explication de la démarche . . . 9
  - 3.2.2 Génération de graphes aléatoires [4] . . . 10
  - 3.2.3 Vérification de la complexité théorique . . . 10

4 Partie analyse de données **11**
- 4.1 Motivation . . . 11
- 4.2 Coefficient Pearson [5] . . . 11
  - 4.2.1 Définition et Explication . . . 11
- 4.3 Coefficient Spearman [6] . . . 11
  - 4.3.1 Définition et Explication . . . 11
- 4.4 Comparaison pratique des classements sur différents jeux de données . . . 11
- 4.5 Conclusion . . . 12

5 Résumé **12**
- 5.1 Résumé et bilan . . . 12
- 5.2 Perspective . . . 12

References **13**

1 Contexte de la centralité

1.1 Définitions de base sur les graphes [1, 2]

Un graphe non-orienté: un couple $G = (V, E)$ comprenant:

- V un ensemble de nœuds.
- $E \subset \{(x, y)|x, y \in V, x \neq y\}$ un ensemble d'arêtes (appelées aussi ensemble de liens). Les arêtes sont des couples de nœuds. Autrement dit, une arête est associée à deux nœuds distincts et on note **n** le nombre de nœuds ($|V|$) et **m** celui d'arêtes ($|E|$).

Chemin : un chemin d'origine x et d'extrémité y, noté $\mu[x, y]$, est défini par une suite finie d'arêtes consécutives, reliant x à y.

Graphe connexe : un graphe non orienté $G = (V, E)$ est dit connexe si quels que soient les nœuds u et v de G, il existe un chemin reliant u et v.

Densité d'un graphe : Pour un graphe non orienté $G = (V, E)$ la densité est definie par

$$\delta = \frac{2|E|}{|V| \cdot (|V| - 1)}$$

Composante connexe d'un graphe : un sous-graphe connexe maximal de ce graphe, c'est à dire qu'il y a toujours un chemin qui relie deux nœuds dans le sous-graphe. Avec cela, on note $PGCC$, la plus grand composant connexe d'un graphe en nombre de nœuds parmi toutes ses composantes connexes.

Distance: Dans un graphe connexe ou une composante connexe d'un graphe non connexe, on appelle distance entre deux nœuds la longueur (i.e. le nombre de nœuds) minimal d'un chemin reliant ce deux nœuds. Par exemple, au-dessous, la distance entre a et h est 2 ($a - d - h$) , et celle entre k et b est 3 ($b - c - f - k$ et $b - e - f - k$).

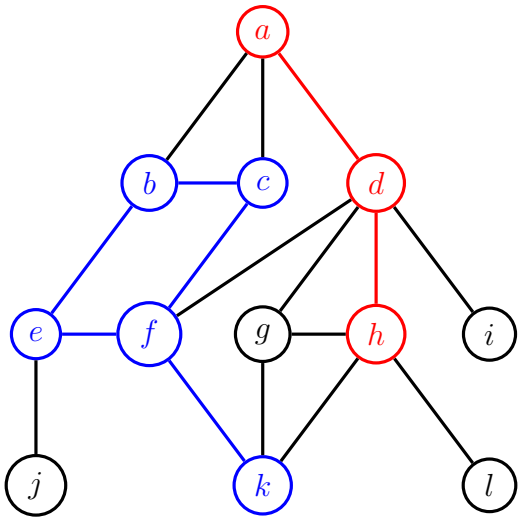


Figure 1: exemple d'un graphe connexe.

1.2 Définitions et les caractéristiques des centralités [3]

Les centralités sont des mesures censées capturer la notion d'importance dans un graphe, en identifiant les nœuds les plus significatifs. Par exemple, les applications de ces indicateurs incluent l'identification de la ou des personnes les plus influentes dans un réseau social, les nœuds clés dans une infrastructure comme internet ou un réseau urbain, donc c'est important de trouver la centralité pour décrire les structures d'un graphe.

1.2.1 Centralité de degré

Définition de la centralité de degré

Soit $u \in V$ un nœud , le degré de u est le nombre de ses voisins, soit $k(u) = |\{v|(u,v) \in E\}|$. Par cette centralité, on peut décrire le nombre de connexions d'un nœud dans le graphe. On la note C_D

1.2.2 Centralité de proximité

Définition de la centralité de proximité

Dans un graphe connexe, la centralité de proximité C_C d'un nœud x dans la composante connexe G=(V,E) est définie par la formule:

$$C_C(x) = \frac{1}{\sum_{y \in V, y \neq x} d(x,y)}$$

avec d la distance, elle traduit à quel point un nœud est proche des autres nœuds de la composante.

1.2.3 Centralité harmonique

Définition de la centralité harmonique

La centralité harmonique C_H d'un nœud est la somme des nombres inverses des distances entre lui et les autres nœuds et elle est définie pour un graphe non connexe en considérant que $d(x,y) = +\infty$ si x et y ne sont pas dans la même composante.

Formule:

$$C_H(x) = \sum_{y \neq x} \frac{1}{d(y,x)}$$

1.2.4 Centralité d'intermédiarité

Définition de la centralité d'intermédiarité

La centralité d'intermédiarité compte le nombre de fois où un nœud agit comme un point de passage le long d'un plus court chemin entre deux autres nœuds. On la note C_I

Formule:

$$C_I(x) = \sum_{\substack{s \neq v, v \neq t, s \neq t \\ s,v,t \in V}} \frac{\sigma_{st}(x)}{\sigma_{st}}$$

où σ_{st} est le nombre total de plus courts chemins du nœud s au nœud t et $\sigma_{st}(x)$ est le nombre de tels chemins qui passent par x.

2 Partie algorithmique

Dans cette partie on étudie toujours dans la *PGCC*

2.1 Structure de donnée choisie pour charger le graphe en mémoire — Liste d'adjacence

Définition de liste adjacence

La liste d'adjacence d'un graphe non orienté, est la liste des voisins de chaque nœud, on utilise un tableau de listes d'adjacence pour stocker un graphe.

Avantage du formet liste d'adjacence

On peut le comparer avec une autre forme de stockage, la matrice d'adjacence. La complexité pour dire si i est voisin de j pour la matrice d'adjacence est $\Theta(1)$ et $\mathcal{O}(k_i)$ ou $\mathcal{O}(k_j)$ (le degré de i et de j) pour la liste d'adjacence. Cepedant, le stockage en mémoire pour la matrice est $\Theta(n^2)$ et $\Theta(m+n)$ pour la liste d'adjacence. Comme on souhaite pouvoir traiter les graphes réels, qui sont normalement assez grands, on peut donc réduire l'espace de mémoire en utilisant le structure de la liste d'adjacence au lieu de la matrice d'adjacence.

2.2 Calcul de distance

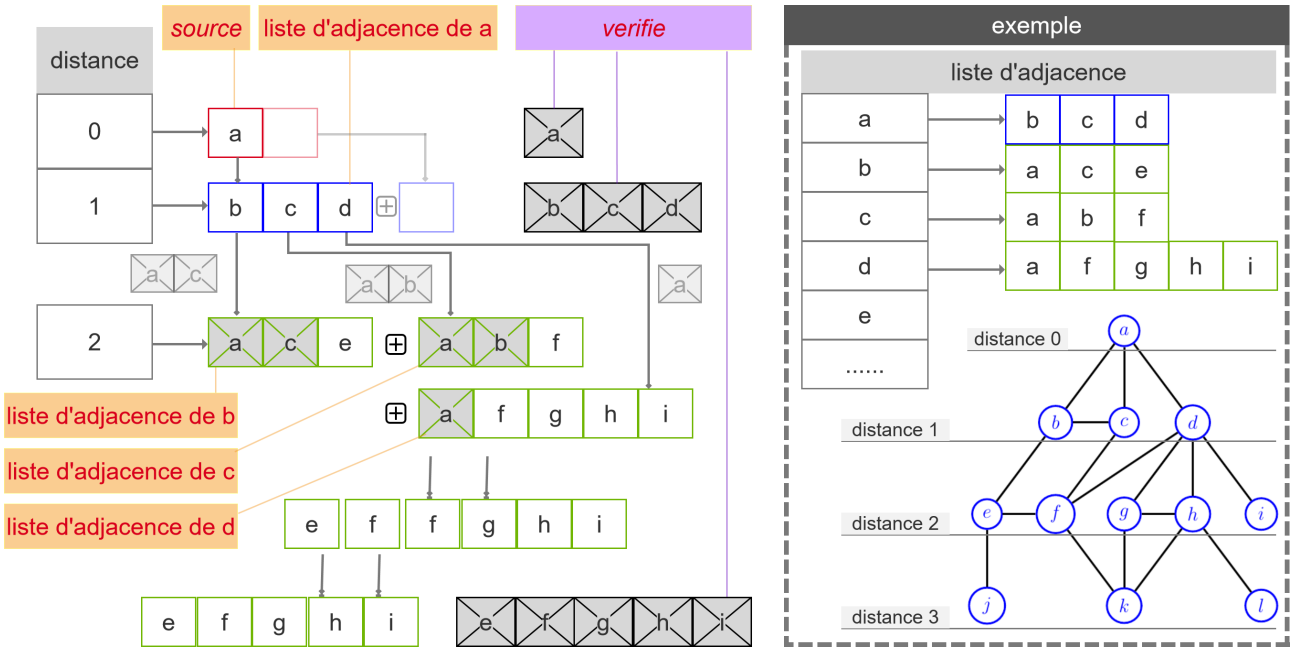


Figure 2: la principe du calcul de distance

Explication

Pour calculer la distance entre le nœud *Source* et les autres nœuds, nous prenons le nœud *Source* et le graphe g comme entrée. Nous parcourons le g et calculons les distances. Nous créons un tableau d où l'indice i représente la distance et la i-ème case contient une liste de tous les nœuds à distance i de *Source*.

Pour éviter qu'un nœud soit vu plusieurs fois, nous créons un tableau $verifie$ pour noter les nœuds, nous notons tous les nœud qui ne sont pas utilisés dans les listes d'adjacence à aide de

$verifie$. Le boucle s'arrête quand $d[i]$ est vide, cela signifie qu'il a aucun nœud à la distance i par rapport au source c. Autrement dit le nœud le plus loin de la source est à distance $i-1$. C'est un algorithme basé sur le parcours en largeur du graphe.

On obtient d : un tableau dont $d[i]$ qui contient la liste des nœuds à la distance i.

La complexité de calcul de distance : $\Theta(m)$

($m = |E|$) Car pour obtenir la distance, on parcourt toutes les nœuds et leur listes d'adjacence. Car c'est un graphe non orienté, on parcourt une arête 2 fois. Donc, finalement, on parcourt toutes les arêtes 2 fois.

Algorithm 1: Calculer distance

Entrée:
Graphe : g ;
Source : c ;

Sortie :
Tableau de liste de Distances : d

```
1  ∀n ∈V, verifie[n] ← 0 ;
2  verifie[source] ← 1;
3  d[0] ← c;
4  d[1] ← liste d'adjacence de c ; (i.e. les voisins qui sont à la distance 1 de c)
5  i ← 1;
6  while d[i] n'est pas null do
7  |  i ← i + 1 ;
8  |  Initialiser d[i] à [ ];
9  |  for x ∈ d[i − 1] do
10 |  |  for y ∈ Liste adjacent de x do
11 |  |  |  if verifie[y] est 0 then
12 |  |  |  |  d.append(y) ;
13 |  |  |  |  verifie[y] ← 1;
14 |  |  |  end
15 |  |  end
16 |  end
17 end
```

2.3 Calcul des centralités différentes

2.3.1 Centralité de degré

Explication

Pour calculer la centralité de degré, on doit parcourir le tableau de liste d'adjacence et le degré de chaque nœud est la longeur de la liste de ses voisins.

La complexité de calcul de centralité de degré : $\Theta(m)$

($m = |E|$) Car finalement on parcours tous les arêtes 2 fois (c'est à dire $2 * m$), la complexité : $\Theta(m)$.

2.3.2 Centralité de proximité

Explication

Pour calculer la centralité de proximité, il faut d'abord parcourir le graphe pour obtenir la somme des distances entre le nœud *Source* et les autres nœuds.

On obtient res: une liste dont $res[i]$ représente la valeur de la centralité de proximité de noeud i.

La complexité de calcul de centralité de proximité : $\Theta(n * (max(n, m))$

($n = |V|$ et $m = |E|$. Généralement, $max(n, m) = m$) Car la boucle **for** en ligne 1 fait n fois ; la boucle **for** en ligne 4 coût $\Theta(n)$ parce que la somme des longeurs des listes dans d est le nombre de nœuds ($= |V| = n$) ; le calcul de distance en ligne 3 coût $\Theta(m)$. Donc, finalement, dans une boucle n fois, on fait le calcul coût $\Theta(max(n, m))$. La complexité : $\Theta(n * (max(n, m)))$.

Algorithm 2: Calculer centralité de proximité

Entrée: Graphe : g ;
Sortie : Tableau : res ;

```
1 for nœud ∈ g do
2 |   somme ← 0.0;
3 |   d ← tableau de la liste de la distance pour noeud;
4 |   for i ∈ [1 , max distance de d ] do
5 |   |   somme ← somme + i * len(d[i]);
6 |   end
7 |   res[noeud] ← 1/somme ;
8 end
```

2.3.3 Centralité harmonique

Explication

Pour calculer la centralité harmonique d'un nœud, on doit parcourir le graphe et calculer l'inverse des distances entres lui et les autres nœuds et à la fin faire une somme de tous ces inverses.

La complexité de calcul de centralité harmonique : $\Theta(n * (max(n, m))$

Car les algorithmes de centralité harmonique et centralité de proximité sont similaires.

Algorithm 3: Calculer centralité harmonique

Entrée: Graphe : g ;
Sortie : Tableau : res ;

```
1 for nœud ∈ g do
2 |   somme ← 0.0;
3 |   d ← tableau de la liste des distances pour noeud;
4 |   for n ∈ [1 , max distance de d ] do
5 |   |   somme ← somme + len(d[n])/n;
6 |   end
7 |   res[noeud] ← somme ;
8 end
```

2.3.4 Centralité d'intermédiarité (pas terminé)

Explication Pour calculer la centralité d'intermédiarité :

Tout d'abord, on définit la méthode pour stocker les chemins. On pose la liste des nœuds dans un chemin entre deux nœuds comme la liste de chemin. Par exemple, dans Figure 3 à droite, entre a et k, il y a un chemin $a-b-e-f-k$. Donc, on pose la liste $[b, e, f]$ pour représenter ce chemin. Donc, d'après cette définition, on peut stocker les chemins entre deux nœuds fixés avec la liste des listes de chemin.

Pour simplifier le problème, on le décompose par 3 étapes:

1. On crée une fonction $fixeUnNoeud(g, source)$ qui fixe un nœud $source$ et calcule tous les plus courts chemins dont la source est $source$. La sortie de cette fonction est un tableau de listes de listes dont l'élément (liste de listes) $res[i]$ contient tous les chemins entre $source$ et le nœud i. Le chemin décrit comme une liste des nœuds. Le principe de ce algorithme : (semblable à celui de distance)

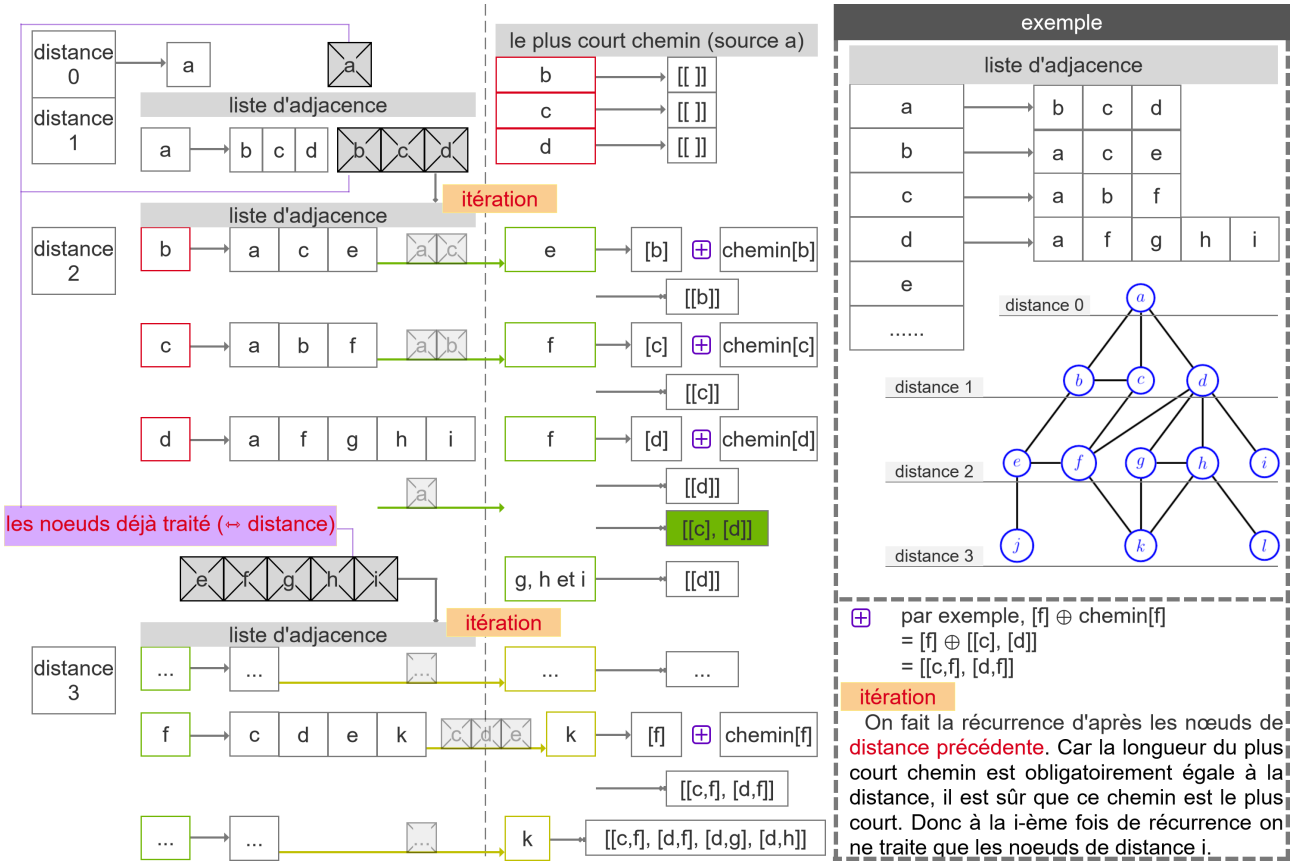


Figure 3: la principe de fonction : $fixeUnNoeud$

On cherche les plus courts chemins avec une source fixée. Ici, par exemple, on fixe le nœud a comme $source$. On initialise les chemins entre a et les nœuds dans la liste d'adjacence de a sont la liste de liste vide ([[ ]]). Alors, on stocke les chemins entre a et b comme $res[b] = [[\,]]$, pareil pour c et pour d. (Algorithme : si $x \in voisins(a)$, $res[x] \leftarrow [[\,]]$)

Après, pour obtenir le plus court chemin, on fait l'itération d'après la liste des nœuds la distance précédente. C'est expliqué dans la figure 3 en bas à droite. (Algorithme : si $x \in$ distance précédente, si $y \in voisins(x)$, $res[y] \leftarrow res[x] \oplus [x]$)

Comme chaque itération on retire les nœuds déjà traités avant, on évite de stocker le chemin qui n'est pas le plus court.

Par exemple, la liste de distance 1 est b, c, d. On cherche les listes d'adjacence de b, c, d et retire les nœuds déjà traités avant (c'est-à-dire les nœuds avec plus courts distances). S'il y a plusieurs chemins (par exemple de a à f), on ne réunit que les chemins dans même niveau d'itération.

2. On change la source qu'on fixe dans la première étape. Donc, on peut parcourir tous les paires de nœuds dans ce graphe.

3. On utilise tous les chemins qu'on trouve et calcule la centralité d'intermédiarité.

Algorithm 4: Etape 2 et 3

Entrée:
Graphe : g ;
Sortie :
Tableau : res ;

```
1  for x ∈ g do
2      somme ← 0.0 ;
3      for y ∈ g do
4          chemin ← fixeUnNud(g , y) ;
5          for z ∈ g do
6              if x , y et z sont différents deux à deux then
7                  Calculer la longueur de chemin[z] noté n ;
8                  Calculer le nombre de chemin qui passe de x noté nx;
9                  somme ← somme + nx/n ;
10             end
11         end
12     end
13     res[x] ← somme ;
14 end
```

3 Partie implémentation

On va vérifier expérimentalement les complexités qui étaient calculées théoriquement.

3.1 Environnement

Utilisation de dictionnaire

Comme nous avons utilisé python, un tableau de listes d'adjacence sera représenté par un dictionnaire de listes. Il facilite beaucoup le stockage et la récupération des données.

3.2 Temps de calcul

3.2.1 Explication de la démarche

On a généré des graphes aléatoires simples puis on sélectionne leur composantes connexes pour tester la complexité des algorithmes.

3.2.2 Génération de graphes aléatoires [4]

Graphe d'Erdos-Rényi

Un graphe d'Erdos-Rényi est un graphe dans lequel on fixe **n** et **m**, puis on tire aléatoirement les paires de nœuds connectés.

Conditions pour créer le graphe d'Erdos-Rényi

1. On fixe le nombre de nœuds **n** et le nombre d'arêtes **m**.
2. On tire aléatoirement deux nœuds distincts.
3. On teste si l'arête tirée existe déjà pour éviter le doublement.

3.2.3 Vérification de la complexité théorique

Explication

À la base des temps utilisés par des fonctions, on fait un modèle de régression, celui-ci est réalisée avec le module python [Numpy et Scipy], qui effectue une régression selon la méthode moindres carrés. Dans notre expérience, on fixe la densité (D) et le nombre de noeud et on calcule le temps utilisé. On obtient un point. En auguementant le nombre de noeuds, on obtient une série de points. On trace le courbe avec ces points en utilisant la méthode moindres carrés. Finalement, on change la densité pour obtenir des courbes différentes.

Exemple et graphe

On a étudié la complexité theorique de calcul de temps pour la centralité proximité et la centralité de degré puis on donne leurs graphes.

On suppose que la complexité de la centralité de proximité est $\Theta(n * (max(n, m))$. Si on fixe la densité D, alors d'après $D = \frac{2m}{n \cdot (n-1)}$ on a $\Theta(m)$ équivalent à $\Theta(n^2)$ et la complexité est de $\Theta(n^3)$. Par ce graphe, on vérifie qu'un modèle en polynome de degré 3 correspond bien aux graphes d'Erdos-Rényi. On fait également pour la centralité de degré et la complexité est de $\Theta(n^2)$. D'ailleurs, plus la densité est grande, plus le temps de calcul est long.

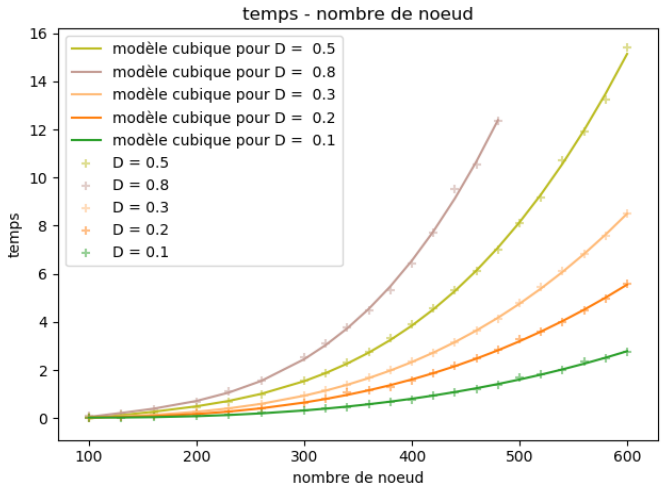


Figure 4: Centralité de Proximité

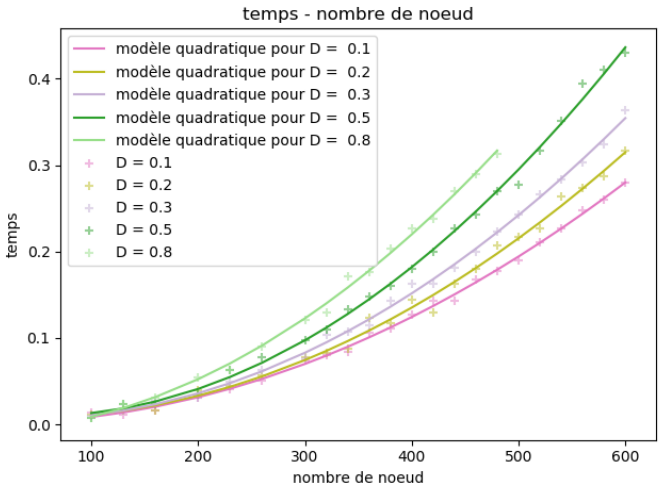


Figure 5: Centralité de Degré

4 Partie analyse de données

4.1 Motivation

Le but de cette partie est de trouver quelles centralités sont corrélées.

4.2 Coefficient Pearson [5]

4.2.1 Définition et Explication

Définition

Le coefficient de Bravais-Pearson, dit coefficient de corrélation linéaire, a pour expression :

$$r_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

ici $Cov(X,Y)$ désigne la covariance des variables X et Y, σ_X et σ_Y leurs écarts types.

Explication

Il a pour but de détecter la présence ou non d'une relation linéaire entre deux variables quantitatives continues et est compris entre -1 et 1. Le sens de la relation est indiqué par le signe de r alors que l'intensité de la relation (capacité à prédire les valeurs d'une variable par rapport à l'autre) est donnée par la valeur absolue de r. On interprète sa valeur de la façon suivante :

- Si $-1 \leqslant r < 0$: relation linéaire négative entre x et y
- Si $r = 0$: absence de relation linéaire entre x et y
- Si $1 \geqslant r > 0$: relation linéaire positive entre x et y

4.3 Coefficient Spearman [6]

4.3.1 Définition et Explication

Définition Le coefficient de Spearman est Pearson calculé sur le rang.

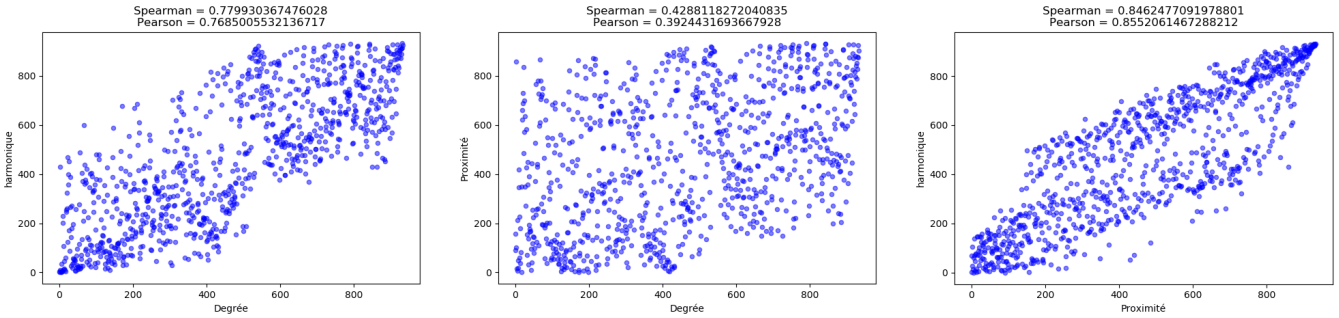
De façon similaire au coefficient de Pearson, le coefficient de Spearman aura une valeur positive lorsque la tendance est croissante et négative lorsqu'elle est décroissante.

Lorsque la tendance n'est pas monotone, il aura une valeur proche de 0.

4.4 Comparaison pratique des classements sur différents jeux de données

On a calculé et comparé les coefficients de Pearson et Spearman entre C_H et C_D, entre C_H et C_C, et entre C_D et C_c. Encore une fois,on a utilisé le graphe des rues de chicago ($chicago - net.txt$) avec 584 nœuds et 638 arêtes comme l'exemple.

Par le Figure 6(c), on constate que le nuage de points est plus proche d'une droite et les coefficients de Pearson et Spearman sont plus proches de 1, donc ça signifie que ces deux centralités sont plus corrélées. Autrement dit ces deux centralités sont plus corrélés que les autres.



(a) Degré et Harmonique (b) Degré et Proximité (c) Harmonique et Proximité

Figure 6: Comparaison pratique

4.5 Conclusion

Les centralités harmonique et de proximité sont fortement corrélées (Spearman=0.85), on pouvait s'y attendre car les deux définitions sont très proches. Les centralités de degré et harmonique sont également fortement corrélées sur ce graphe (0.78), c'est plus étonnant, surtout que les centralités de degré et de proximité sont seulement faiblement corrélées (Spearman=0.43).

5 Résumé

5.1 Résumé et bilan

En résumé, on a approfondi nos études dans le domaine de centralité. On a trouvé les algos pour les calculer et aussi essayé d'exploiter les particularités des centralité par calculer les complexités des algorithmes. on a vérifié expérimentalement la complexité temporelle de certains algorithmes. et puis employer les outils comme coefficient de Spearman et celui de Pearson afin d'examiner leur niveaux de corrélations. La procédure totale est vraiment enrichissante et elle nous a beaucoup aidé.

5.2 Perspective

La centralité est une partie très importante dans le domaine de science, on peut faciliter beaucoup d'études ou obtenir les meilleurs bénéfits par la recherche sur la centralité et c'est forcément un secteur très intéressant pour continuer le travail dessus. Nous allons le poursuivre à la suite de notre études, par exemple il nous reste encore des problèmes sur le domaine de la centralité d'intermédiarité.

References

[1] Wikipedia. Théorie des graphes — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=Th\%C3\%A9orie\%20des\%20graphes&oldid=161670233, 2019. [Online; accessed 31-August-2019].

[2] Wikipedia. Graphe connexe — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=Graphe\%20connexe&oldid=159772445, 2019. [Online; accessed 07-September-2019].

[3] Wikipedia. Centralité — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=Centralit\%C3\%A9&oldid=160813356, 2019. [Online; accessed 31-August-2019].

[4] Wikipedia. Graphe aléatoire — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=Graphe\%20al\%C3\%A9atoire&oldid=157012986, 2019. [Online; accessed 07-September-2019].

[5] SURVEY. Que signifie Coefficient de Pearson ? https://www.soft-concept.com/surveymag/definition-coefficient-de-pearson.html, 2019. [Online; accessed 07-September-2019].

[6] Wikipedia. Corrélation de Spearman — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=Corr\%C3\%A9lation\%20de\%20Spearman&oldid=160312281, 2019. [Online; accessed 07-September-2019].