UNIVERSITÉ DE NAMUR

FACULTÉ D'INFORMATIQUE

Projet 2 : Analyse de données

Auteur Christian NAZILI W.
Prof Renaud LAMBIOTTE
Assistant Vsevolod SALNIKOV

Cours : Graph Mining - SDASM101

Rue Grandgagnage, 21 ● B-5000 Namur (BELGIUM

Contents

1 Introduction

Dans ce projet, il nous a été demandé d'analyser le graphe de karaté club [1] dans le cadre du cours de Graph Mining [2]. L'analyse est faite en fonction des certains concepts vu au cours et d'autre propriété que j'ai voulu voir comment le graph se comportait.

Un petit mot sur le graph Karaté Club. Il y avait essentiellement un club de karaté avec un administrateur "John A" et un instructeur "M. Salut ".

Les élèves sont les nœuds de notre graphique et les contours, ou liens, entre les nœuds sont le résultat d'interactions sociales extérieures au club entre les élèves.

Durant les lignes qui vont suivre, nous allons appliquer certaines propriétés pour voir comment le graphe se comporte. Nous allons pouvoir mesurer la distribution de degrés et de différents centralités, calculer le coefficient de clustering.

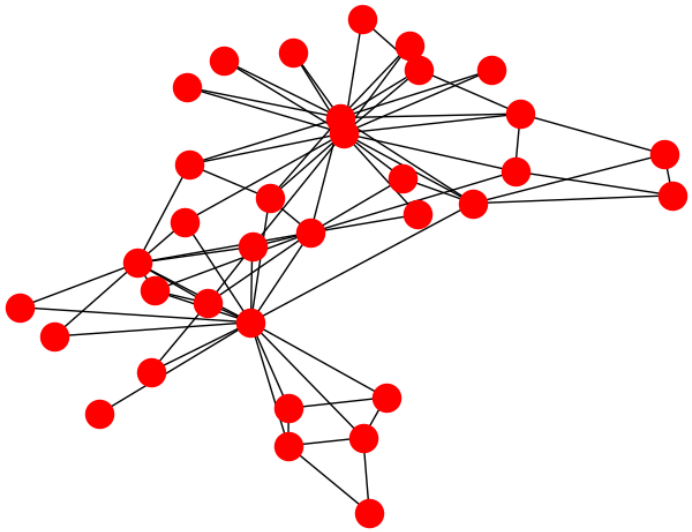
Nous allons appliquer au graphe un modèle de configuration ainsi que l'algorithme de Louvain pour calculer le nombre de communauté du graphe. Puis nous allons pouvons comparer certains résultat avec un ensemble de 100 graphes conçus aléatoirement mais qui partagent les mêmes propriétés que notre graphe de départ.

Nous allons essayés d'appliquer certaines algorithmes de clustering pour voir la différent avec le clustering de départ. Par soucis de prévoyance et pour ne pas à chaque fois perdre le travail déjà fait, j'ai préféré faire le projet avec **Anaconda**. *La vérification analytique de tout ce qui sera dit dans ce projet, peut être vérifier à partir de la simulation [3] [4].*

2 Visualisation du graphe

Le graphe a été importé directement en ligne avec le librairie networkx [5] [1] qui nous a permis de ne pas télécharger les données en local. Par conséquence pour exécuter le script, il vous faut une connexion internet.

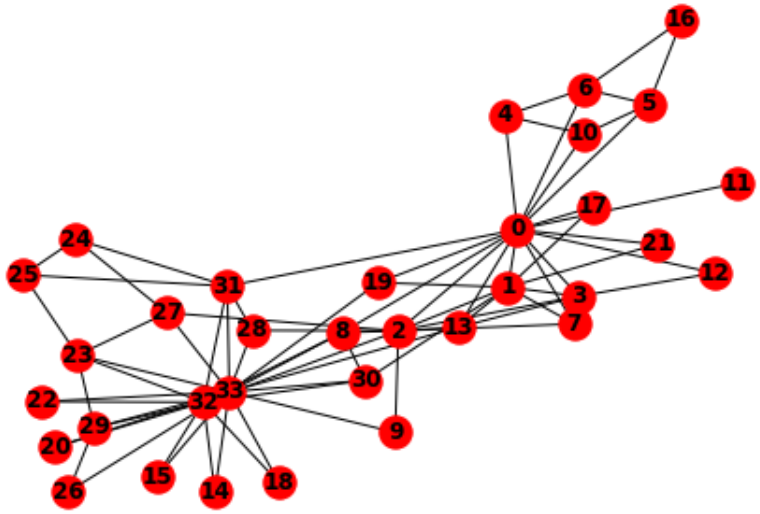
Voici le graphe du karate club :



[1] NetworkX est une bibliothèque Python pour étudier les graphes et les réseaux. NetworkX est un logiciel libre publié sous la licence BSD-new.

A première vu, nous voyons que le graphe a un certain nombre de sommet et chaque sommet est relie à une arête. Pour moi c'est clairement un graphe orienté, aussi pour la première constatation c'est un graphe qui n'a ni arête multiple ni boucle, ni arête parallèle; mais cela va se vérifier avec le script.

J'ai voulu mettre un chiffre à chaque sommet pour voir si il y a combien de nœuds dans le graphe avant de pouvoir le vérifié. Voici le graphe que cela a donné:



Le graphe de Karate clud compte enrivon 33 nœuds qui sont connectés entre eux.

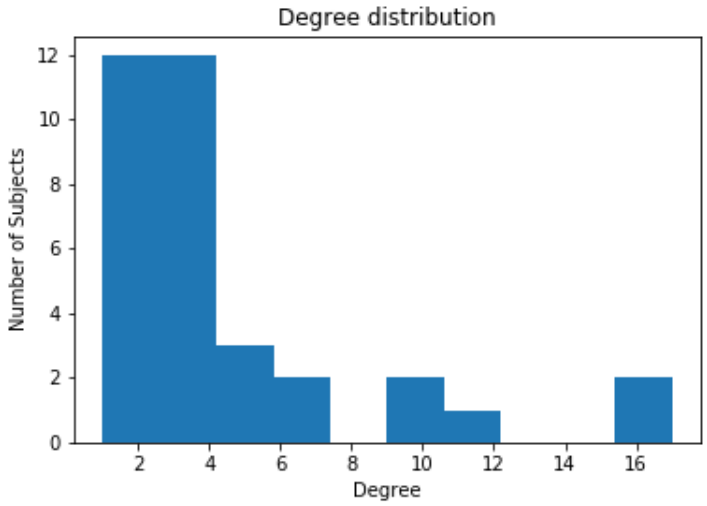
Analytiquement, j'ai trouvé après vérification que le graphe compté évidemment 34 nœuds connectés les liens, les autres. Il compte aussi un nombre 78 liens entre les nœuds, pas de boucle; car selon le script pour la vérification des boucles le résultat est une liste vide.

Vu que le graphe est un graphe connecté, alors j'ai voulu calculer le degré moyenne du graphe. Le résultat est de 2.2941176470588234. C'est-à-dire que chaque sommet a au moins deux arêtes qui partent de lui.

3 Distribution de degrée

Nous savons qu'en théorie des graphes, le degré ou plus simplement appelé aussi *valence* d'un sommet d'un graphe est le nombre de liens (arêtes ou arcs) reliant ce sommet [2].

[2] Définition du cours de théorie de graphe - SMATB254 [6]



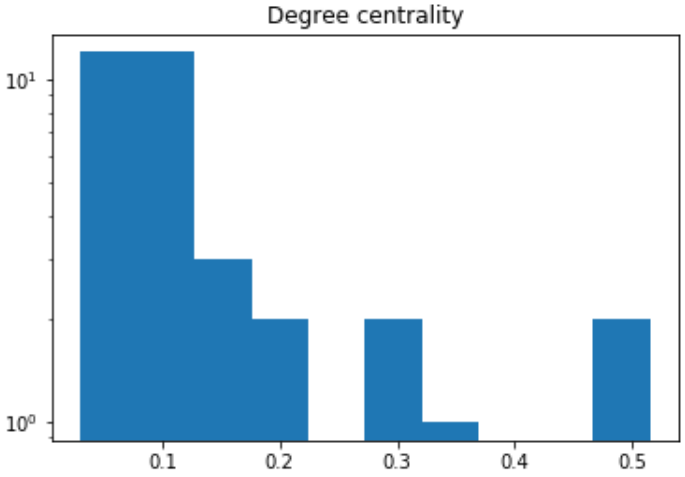
Nous voyons à travers cet histogramme que la plus tard des commets ont une variance de deux. Par contre les quatre premiers sommets ont une variance allant jusqu'à 12. Ils ont des valeurs qui sont très écartées les unes des autres. Par contre ceux qui au plus 3 et au moins 2 nœuds, ont des valeurs qui sont rapprochés. L'histogramme nous montre que nous n'avons pas de sommet qui n'ont des variance nulle. Une variance nulle prouve que toutes les valeurs sont identiques.

4 Différentes centralités

Les mesures de centralité visent à quantifier l'importance des nœuds d'un réseau. Dans les lignes qui suivent nous allons voir quelques degré de centralité sur notre graphes. Tous les centralités ont mis selon une échelle logarithmique.

4.1 Degré de centralité

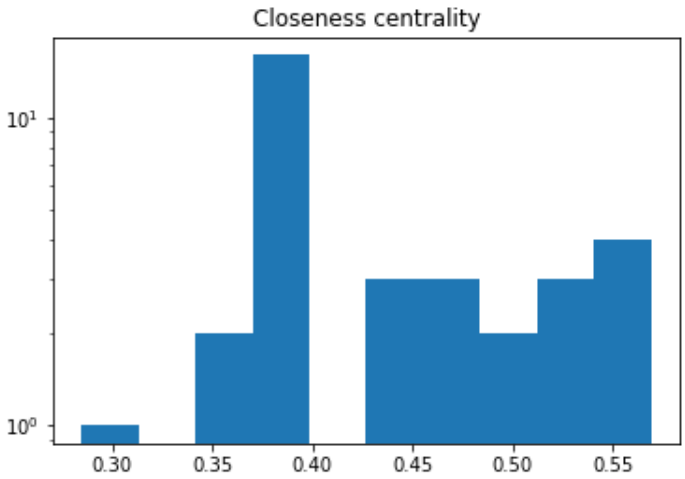
Le degré de centralité, c'est le plus simple, mais son efficace est restreint parce que dans certain cas cela ne marche pas très bien.



En voyant le graphique, nous voyons qu'il y a un sommet qu'il a plus de arête qui viennent vers lui. Selon les résultats analytique c'est le nœud 34 le plus important de tous les nœuds, c'est-à-dire qu'il est le plus significatifs. Car son degré de centralité est de 0.5151515151515151. Donc nous pouvons dire que la personne numéro 34 a plus d'influence dans le réseau. En contre partie la personne qui se trouve à la place 11 n'a pas trop d'influence, il n'est que de 0.030303030303030304. Il y a un grand nombre de nœuds qui ont un degré de centralité au environ de 0.12.

4.2 Closeness centrality

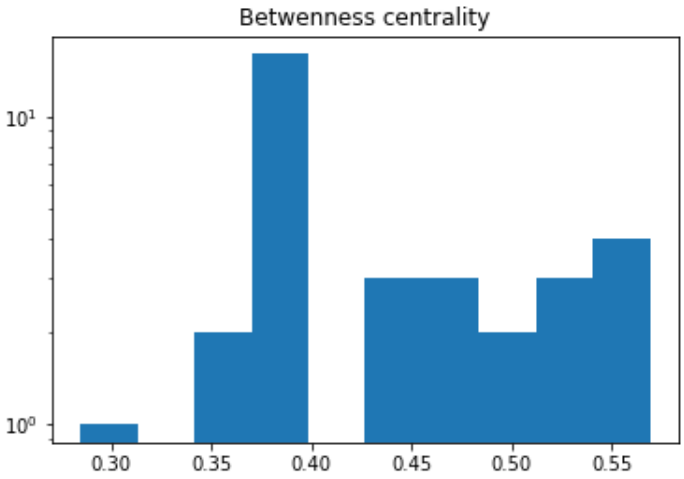
La centralité de proximité est une mesure de centralité la plus repandue qui est basée sur la distance interpoler des paires de noeuds. C'est la longueur moyenne du plus court chemin entre le nœud et tous les autres nœuds du graphe. Ainsi, plus un nœud est central, plus il est proche de tous les autres nœuds.



Dans un graphe connecté, la centralité de proximité (ou la proximité) d'un nœud est une mesure de la centralité dans un réseau, calculée comme étant l'inverse de la somme de la longueur des chemins les plus courts entre le nœud et tous les autres nœuds du graphe. Ainsi, plus un nœud est central, plus il est proche de tous les autres nœuds. Dans notre graphe le noeud le plus proche de tous les noeuds est le premier, car il a un degré de proximité le plus grand. Le 34 noeud vient en deuxièmen position. Avec notre graphique c-haut, nous pouvons dire que la plus part de noeuds se trouvent rapprocher les uns les autres.

4.3 Betwenness centrality

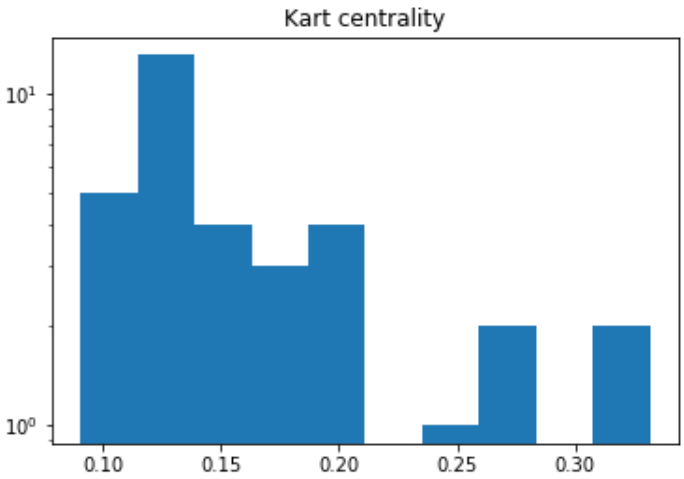
Betwenness centrality est définie comme la fraction du chemin les plus courts passant par le noeud. Cette mesure est calculé en faisant la moyenne sur toutes les paires de noeuds. La centralité de Betweenness quantifie le nombre de fois où un nœud agit comme un pont sur le chemin le plus court entre deux autres nœuds.



Nous voyons tout de suite la ressemblance frapante avec le closeness centrality. Les restes de noeuds font passer leur information par beaucoup plus par le 1er noeud et aussi le 34e noeud. Nous pouvons dire que les chemins le plus court passe par soit l'un soit l'autre, voir même les deux noeuds ensembles. Dans notre un réseau du graphe, ces deux noeuds auraient demander danvatage de contrôle cause de leur degré de centralité plus grande.

4.4 Kart centrality

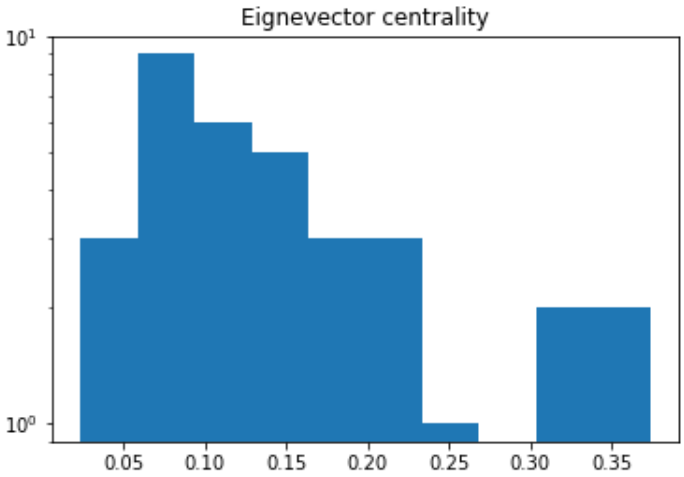
La centralité de Katz calcule l'influence relative d'un noeud au sein d'un réseau en mesurant le nombre de voisins immédiats (noeuds du premier degré) ainsi que tous les autres nœuds du réseau qui se connectent au noeud en question via ces voisins immédiats.



C'est le dernier noeuds suivi du premier noeud qui ont des degrés de centralité les plus élevés. leur influence par rapport au réseau sont plus important, cela confirme le résultat obtenu dans les sections précédentes.

4.5 Eigenevector centrality

Eigenevector centrality est une mesure qui permet de calculer aussi l'influence d'un noeud dans un réseau. Les scores sont attribués à tous les noeuds du graph. Le score le plus élevé de vecteur propre signifie qu'un noeud est connecté à de nombreux noeuds.



Analytiquement, nous voyons que cela vient encore confirmer ce que les sections précédent nous ont montré, à savoir que les noeuds 1 et 34 sont les plus importants du graphe.

5 Coefficient de clustering

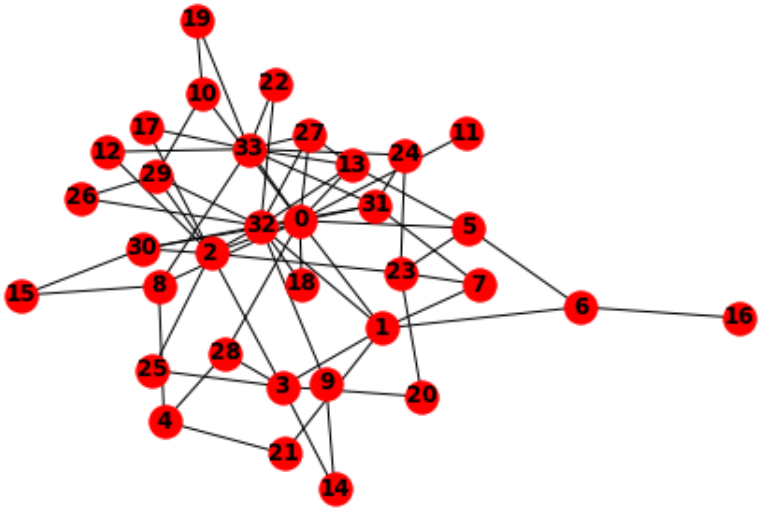
Dans la théorie des graphes, un coefficient de clustering est une mesure qui nous permet de calculer la concentration de noeuds dans un graphique. Nous pouvons calculer le coefficient de clustering de chaque noeud, mais cela n'est pas vraiment pertinent. Voilà pourquoi j'ai opté pour faire moyenne des coefficients de clustering de notre graphique, et cela revient à 0.5706384782076823. Les noeuds ont une concentration d'au moins 50 %.

6 Modèle de Configuration & Fraction à éliminer & Densité

6.1 Modèle de configuration

Le modèle de configuration est une méthode permettant de générer des graphes aléatoires à partir d'une séquence de degrés donnée.

J'ai appliqué le modèle de configuration sur les degrés du graphe. Le graphique produit est presque semblable à celui de départ. A la seule différent que lorsqu'on exécute le code une nouvelle fois, on obtient juste un sommet qui se détache de tous les autres noeuds.



6.2 Fraction à éliminer

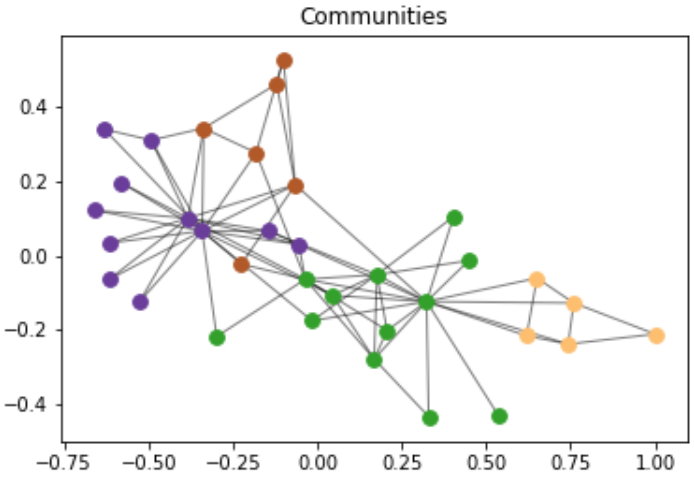
J'ai remarqué dans la simulation que le fraction à éliminer ne change rien sur notre graphique. Il garde les mêmes propriétés que le graphique de départ [3].

6.3 Densité

La densité de notre graphique est de seulement 0.13903743315508021. Notre graphe n'est pas assez dense, c'est-à-dire qu'il y a un certain nombre de noeud qui n'ont pas beaucoup de connexion avec d'autre, au min une seule connextion.

7 Nombre de communauté

Dans cette section, nous allons voir si les noeuds faisant partis d'un même réseau peuvent être facilement regroupés en ensembles de noeuds, de manière à ce que chaque ensemble de noeuds soit connecté de manière assez dense en interne.



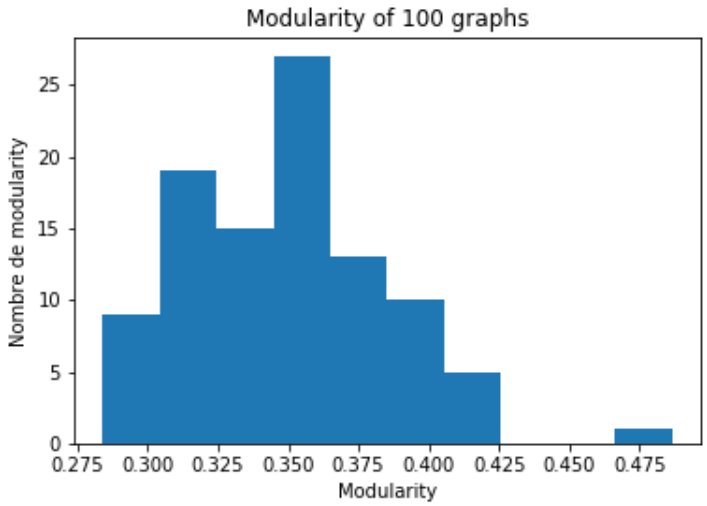
[3] Confère simulation

Pour notre graphique tous les noeuds peuvent être regroupés en 4 communauté avec une modularité de 0.4188034188034188.

8 100 graphes aléatoires

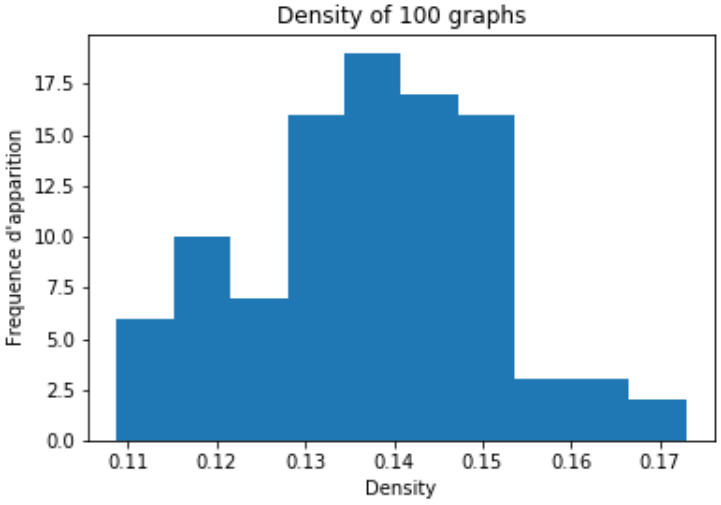
Dans cette section, je vais générer 100 graphes aléatoirement qui partagent certaines propriétés avec notre graphe de Karate club. Pour cela je vais utiliser un Erdos-Renyi avec la même densité d'arêtes que nous avons obtenus.

8.1 Modularités



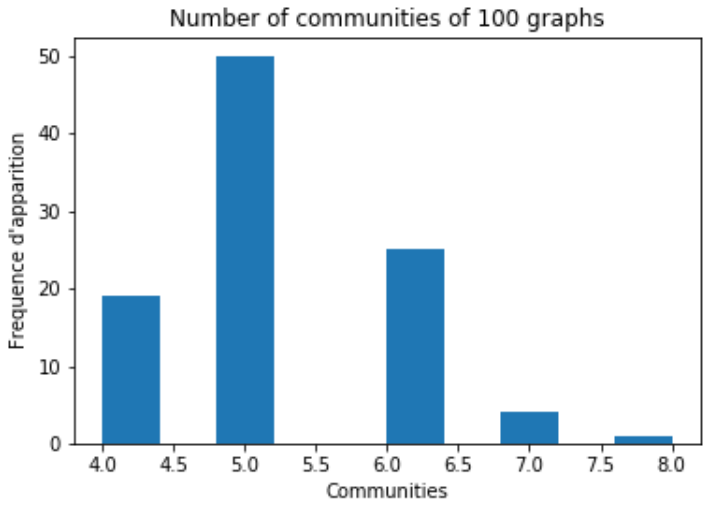
Il y a un grand nombre de graphique générée aléatoirement que ont une modularité égal à peut près 0.350. Nous voyons aussi qu'il y a au environ de 5 graphes qui ont la même modularité que le graphe de départ. Et remarquons aussi à l'extrême qu'il y a certains graphes générer qui ont une modularité égal à 0.475, c'est plus que le graphe de départ. De l'autre extrême certains graphes aussi ont une modularité beaucoup plus petit (au environ de 0.279).

8.2 Densités



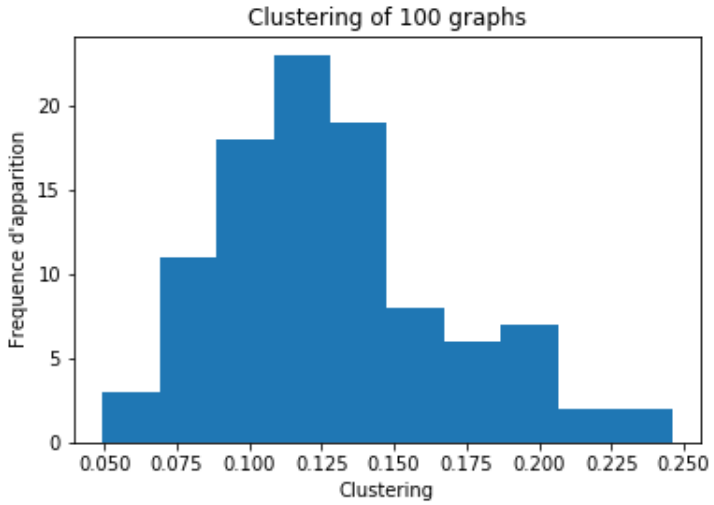
Nous voyons clairement dans l'histogramme ci-haut qu'il y a quand un certain nombre de graphiques qui partagent la même densité que le graphe de départ. Voir même il y a d'autres qui ont une densité plus grande aussi à attendre 0.14. Et aussi nous remarquons que d'autres graphes ont une densité plus petit, et un certains nombre de graphes (nombre pas assez grand) qui ont une densité allant jusqu'à 0.17.

8.3 Communautés



On remarque un nombre très important graphes ont des communautés de 5. Ensuite , vient un certain nombre de graphes qui ont une communauté de 6, en troisième position ce sont de graphes qui partagent le même nombre de communauté que le graphe de départ. On regarde aussi que la plus grande communauté est de 8. C'est le 42ieme graphe qui a cette communauté. *A chaque exécution de code, le nombre de communauté peut changer.*

8.4 Clustering



Aucun de graphes générés ne partagent le même coefficient de clustering que notre graphe de départ. Ils ont tous un coefficient de clustering inférence. Et le coefficient de clustering du graphe ayant la plus grande communauté est de 0.13726387991093875.

9 Quelques algorithmes de Clustering

Dans cette section, nous allons étudier quelques algorithmes dans Python afin de comparer si nous aurons le même nombre de cluster que ceux optenus.

Nous allons utiliser scikit-leanr, les pandas comme librairies. Afin de colorer les noeuds d'élèves en fonction de leur appartenance à un club, nous utilisons la classe Normalize de matplotlib pour adapter le nombre de clubs à l'intervalle (0, 1). Nous le faisons parce que plus tard, nous allons voir que certains algorithmes de clustering ne fonctionne pas très bien. Car nous voulons voir si nous pouvons aller plus loin que 4 communautés.

Pour comparer les performances de notre algorithme, nous voulons les vraies étiquettes, c'est-à-dire où chaque élève se retrouve après la fission du club. Malheureusement, les véritables étiquettes ne sont pas fournies dans le jeu de données networkx, mais nous pouvons les récupérer à partir de l'étude elle-même. Avant cela, nous devons pré-traiter les données en transformant le graphique en matrice car il s'agit du format requis pour les clusterers.

9.1 Algorithmes

Pour cela nous allons utiliser l'apprentissage non supervisé de machine learning. Nous allons passer en revue quelques algorithmes connus pour leur très bon fonctionnement et voir comment ils se comportent sur notre graphique. Nous ne savons pas du tout quel algorithme utiliser

9.1.1 K-means

K-means définit des groupes sphériques séparables de manière à ce que la valeur moyenne converge vers le centre du groupe. De ce fait, K-Means peut parfois sous-performer.

Pour simplement construire et former un modèle K-means, nous pouvons utiliser le package de sklearn. Avant cela, nous allons définir le nombre de grappes que nous savons vraies (deux), une liste contenant les résultats (étiquettes) et un dictionnaire contenant chaque algorithme que nous finissons par essayer.

9.1.2 Agglomerative Clustering

Le principe de Agglomerative Clustering est que chque noeud commence d'abord dans sa propre grappe, puis des paires de grappes se fusionnenet de manière récursive pour augmenter le distance minimal de liaison. L'avantage est qu'il n'est pas nécessaire de spécifier le nombre de clusters, cela va jouer sur la performance.

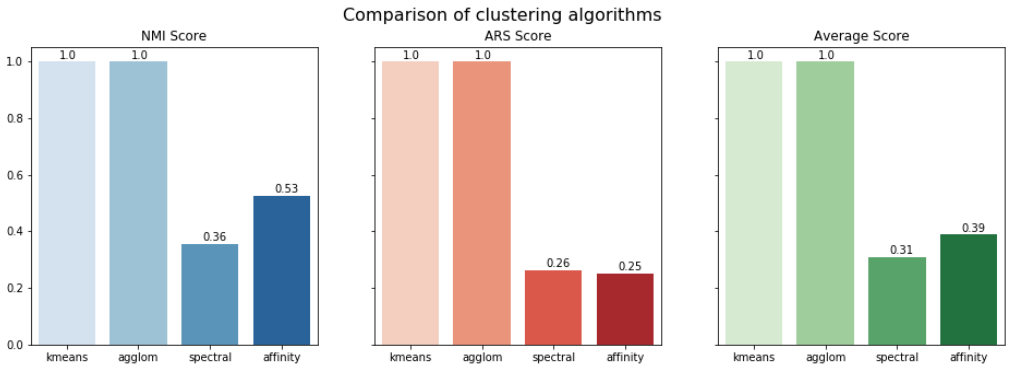
9.1.3 Spectral Clustering

Cette technique utilise la classification à une projection du laplacien normalisé. Laplacien est utilisé pour trouver une représentation matricielle d'un graphe. Ça fonctionne bien pour le regroupement d'image.

9.1.4 Affinity propagation

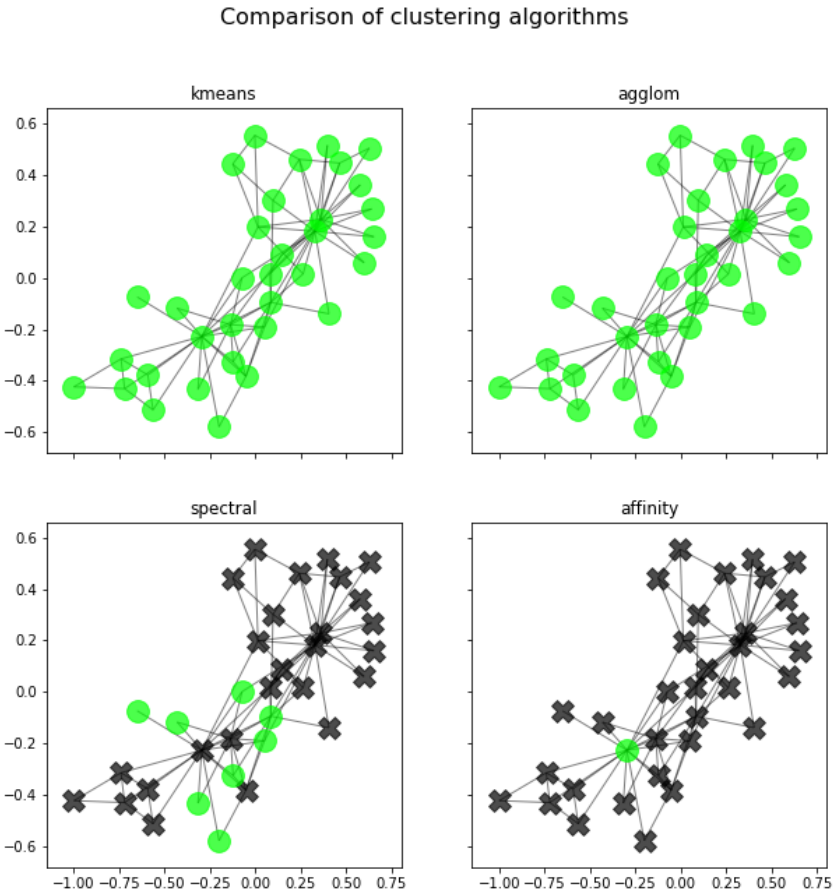
Elle est un peu différente, car elle ne nécessite pas de déterminer le nombre de grappes avant d'exécuter l'algorithme.

Voyons maintenant qu'il algorithme est le mieux adapté à notre graphe, en calculant les mesures du score de rand ajusté (ARS), calcule une mesure de similarité entre deux clusters, et de l'information normalisée mutuelle (NMI), est une mesure de la dépendance mutuelle entre deux variables allant 0 (aucune information mutuelle) et 1 (coorélation parfaite), pour une interprétation plus facile.



Comme nous pouvons remarquer dans le graphique obtenu, la classification par K-moyennes et par agglomération produit le meilleur résultat possible pour notre graphique. Bien entendu, cela ne signifie pas que Spectral et Agglomerative sont des algorithmes peu performants, mais simplement qu'ils nont moins performant pour notre graphique. Peut être que dans un autre ensemble de donner ils vont bien fonctionner.

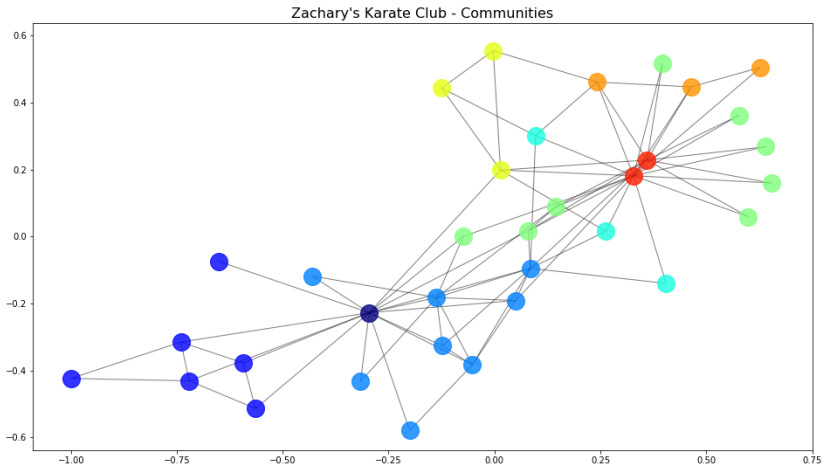
Voyons maintenant où chaque algorithme a mal fonctionné en comparant les grappes Par curiosité, créons une nouvelle fonction pour tracer où chaque algorithme a mal tourné en comparant les clusters d'élèves prédites aux vraies clusters d'élèves.



Ce le graphique ci-haut, décrit les affectations de cluster correcters (cercle vert) et incorrectes (X noir) de chaque algorithme.

De plus ce graphique nous montre que l'algorithme d'affinité a un score moyen plus élevé. Voyons maintenant comment nous pouvons examiner les informations de cluster produit par cet algorithme.

L'algorithme d'affinité nous donne un nombre de 8 Clusters, c'est deux plus que celle que nous avons obtenu juste pour le graphe de départ sans utilisation de l'algorithme.



References

[1] karate Club, "Zachary's karate club". `https://en.wikipedia.org/wiki/Zachary%27s_karate_club`, dernière consultation le 28 Octobre 2018.

[2] R. Lambiotte, *Graph Mining - SDASM101.* World Scientific, 2016.

[3] Python, "python learn". `https://pythonprogramminglanguage.com/getting-started/`, dernière consultation le 27 Octobre 2018.

[4] Python, "python documentation". `https://docs.python.org/3/`, dernière consultation le 30 Novembre 2018.

[5] Networkx, "Networkx". `https://networkx.github.io/`, dernière consultation le 01 Janvier 2019.

[6] R. Lambiotte, *SMATB254 Théorie des graphes.* 2016.