République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de 8 Mai 1945-Guelma-
Faculté des Mathématique,d'Informatique et des Sciences de la matière
Département d'Informatique

Mémoire de fin d'études Master

Filiere : Informatique

Option :

Science et téchnologie de l'information et de la communication

Thème

Étude comparative des méthodes d'évaluation de la qualité des structures communautaires

Encadré par :
M^me LOUAFI WAFA

Présenté par :
M^r Talhi Fayssal

Année Universitaire 2020/2021

Remerciements

Je remercie en tout premier lieu, le bon Dieu tout puissant de nous avoir donné la volonté, la force et le courage de réaliser le présent travail.

Je tiens á remercier chaleureusement notre directeur de recherche, Madame Louafi Wafa, pour sa patience, sa disponibilité et son aide incontestable. Je lui adresse toute ma reconnaissance quant á son accompagnement régulier tout au long de l'élaboration de ce travail. J'adresse mes sincères

remerciements aux membres de jury pour leurs efforts de lecture et d'évaluation de ce mémoire.

Je remercie également tout les membres de département de l'informatique soit des enseignants qui ont veillé á nous accorder une formation de qualité, des employés de direction qui facilite tout les choses lié aux direction, des agents qui veille sur notre sécurité .

Mes sincères gratitudes vont aussi á tous ceux qui ont contribué de près ou de loin á l'accomplissement de ce travail.

Merci

Dedicaces

Je dédie ce modeste travail ;

A mes chers parents qui n'ont jamais cessé de formuler des prières á mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs, aucune dédicace, aucun mot ne pourrait exprimer á sa juste valeur la gratitude et l'amour que je leur porte sans exeption ma seul soeur khawla .

A mes uniques fréres, hazem et fateh et souleymen pour ses soutien moral, ses encouragements et ses aide précieuse.

A la mémoire de ma grand-mère maternelle, ma grand-mère paternelle, qui restera á jamais dans mon esprit et dans mon coeur.

A sara, pour son aide inestimable.

Resume

Actuellement, de nombreuses méthodes de détection communautaire dans les réseaux sociaux ont été proposées á travers plusieurs chercheurs . Il est important d'évaluer la qualité de ces structures. La capacité d'un algorithme á détecter la structure d'une communauté est généralement validée en testant l'algorithme sur des réseaux artificiels ou réels pour lesquels la division en communautés est connue. Étant donné que la disponibilité d'une structure de communauté de vérité terrain pour les grands réseaux réels est plutot difficile, Plusieurs méthodes de mesures existes, nous avons choisi deux : la modularité et l'information mutuelle normalisée (NMI) que nous allons comparer en termes de leurs précision, reproductibilité et performance.

Mots clé :
La détection de communaité, les mesures d'évaluations, la modularité, l'information mutuelle normalisée(IMN).

Abstract

Currently, many methods of community detection in social networks have been proposed through several researchers. It is important to evaluate the quality of these structures. The ability of an algorithm to detect community structure is usually validated by testing the algorithm on artificial or real networks for which the division into communities is known. Since the availability of a ground truth community structure for large real networks is difficult, several measurement methods exist, we have chosen two : modularity and normalized mutual information (NMI) which we will compare in terms of their accuracy, reproducibility and performance.

Keywords :
Community detection, Evaluation measure, Modularity, Normalized mutual information (NMI),

Table des matières

Table des figures

Introduction Generale

Les structures communautaires sont omniprésentes dans divers réseaux complexes, comme les réseaux biologiques, sociaux et technologiques. Les communautés (ou modules) sont des parties du réseau avec des connexions internes denses et des connexions externes éparses [3].

Les communautés sont étroitement liées aux unités fonctionnelles des réseaux du monde réel, comme les cycles et les voies dans les réseaux métaboliques et les complexes protéiques dans les réseaux d'interaction protéine-protéine, elles peuvent avoir des propriétés topologiques trés différentes de celles des réseaux entiers et affecter la dynamique des réseaux. Par conséquent, l'identification des communautés est importante pour comprendre les structures et les fonctions des réseaux [26]

Plusieurs méthodes ont été proposées pour détecter les structures communautaires dans les réseaux, certaines sont basées sur des mesures de similarité, d'autres font appel á la dynamique des réseaux telles que la dynamique de la marche aléatoire et la propagation des étiquettes, d'autres méthodes sont basées sur des modèles statistiques. Tandis que certaines méthodes de détection de communautés basant sur l'optimisation de fonctions de qualité. Par exemple, la célébre modularité de Newman-Girvan peut étre utilisée comme un moyen objectif d'estimer la qualité des partitions de

communautés, et implique également un type de stratégie de détection de communautés. [18] peut etre considérée comme un type de problème d'optimisation, étant donné les fonctions de qualité pour l'évaluation des structures communautaires. Par conséquent, l'optimisation des fonctions de qualité a été l'une des stratégies les plus populaires pour la détection de communautés dans des environnements complexes.
Apré la détection des communautés, la question qui se pose est comment évaluer les communautés et tester leurs qualités. Plusieurs méthodes de mesures existes dans la littérature, mais comment évaluer ses mesures lui memes, dans ce mémoire nous avons choisi les deux mesures les plus populaires qui sont modularité et l'information mutuelle normalisée (NMI) pour faire une comparaison entre eux en termes de leur précision, reproductibilité et performance.

Ce mémoire est compose de quatre chapitres, il est organisé de la manière suivante :

— Le premier chapitre : La détection de communautés

— Le deuxième chapitre : Les mesures d'évaluation

— Le Troisième chapitre : Outils et Méthodologie

— Le Quatrieme chapitre : Résultats et Interprétation

Nous terminerons notre le mémoire avec une conclusion générale et quelques perspectives.

Chapitre 1

La détection de communautés

:

1.1 Introduction

La détection de communautés consiste á regrouper un ensemble d'individus ou d'objets en fonction de différents critères ou contraintes. Le terme de détection de communautés est plutót appliqué á des individus engagés dans des rapports sociaux. D'un point de vue mathématique, nous verrons plus loin que les individus peuvent étre représentés par des sommets d'un graphe (on parle aussi des noeuds d'un graphe) et les relations qui les unissent présentées par les liaisons(arcs) du graphe [17].
Dans ce chapitre, nous allons présenter les fondements théoriques de notre travail. Nous allons présenter bravement la notion de la théorie des graphes, la détection des communautés avec leurs différentes techniques et méthodes et on termine ce chapitre par une conclusion.

1.2 Notions relatives á la théorie des graphes

La théorie des graphes est la discipline mathématique et informatique qui étudie les graphes, lesquels sont des modèles abstraits de dessins de réeaux reliant des objets. Ces modèles sont constitués par la donnée de sommets (aussi appelés noeuds ou points, en référence aux polyèdres c'est-á-dire forme géométrique á trois dimensions ayant des faces planes polygonales qui se rencontrent selon des segments de droite qu'on appelle arètes), et d'arètes (aussi appelées liens ou lignes) entre ces sommets ; ces arètes sont parfois non-symétriques (les graphes sont alors dits orientés) et sont appelées des flèches ou des arcs .

1.2.1 graphes

Un graphe est un triplet G = (V , E , S) comprenant :
V : un ensemble des sommets (aussi appeles noeuds ou points)
E : un ensemble d'aretes (aussi appeles liens ou ligne)

$$S : E \to x, y \mid (x, y) \in V2 \wedge x \neq y \tag{1.1}$$

une fonction d'incidence associant á chaque arete une paire de sommets (c'est-á-dire qu'une arete est associée á deux sommets distincts) . Pour lever toute ambiguité, ce type d'objet peut etre appelé précisément un multigraphe non orienté . V et E sont en général supposés finis, et de nombreux résultats cessent d'etre vrais (ou ont des énoncés différents) pour des graphes infinis parce que certains arguments de preuve ne se transposent pas au cas infini . De plus , V est souvent supposé non vide , mais E peut etre l'ensemble vide. L'ordre d'un graphe V est son nombre de sommets . La taille d'un graphe est E , son nombre d'aretes. Le degré ou la valence d'un sommet est le nombre d'aretes incidentes á ce sommet, oú une boucle compte double . Dans un graphe simple non orienté d'ordre n , le degré maximum d'un sommet est n-1 et la taille maximale du graphe est n (n - 1)/2.
Les aretes E d'un graphe non orienté G induisent une relation binaire symétrique sur V appelée la relation d'adjacence de G . Spécifiquement , pour chaque arete x , y , ses sommets extremes x et y sont dits adjacents l'un l'autre , ce qui est noté x - y .

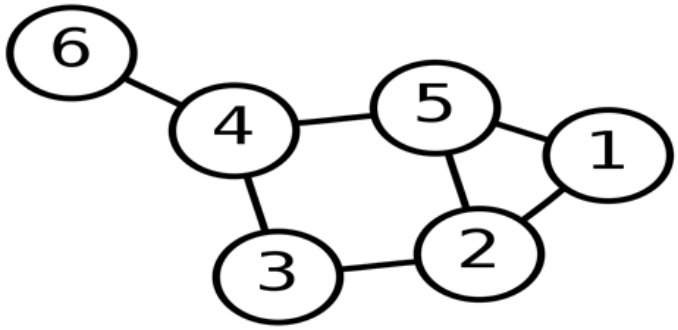

FIGURE 1.1 – Un tracé de graph

1.2.2 Topologies

Il existe trois grandes familles de graphes (strucrure,quelconques,multipolaires) et cinq catégories au total :
Structurés : il est alors possible de définir quatre identités topologiques remarquables :

- homogènes (1) : les sommets et les arètes reproduisent un schéma régulier. Le schéma le plus commun est une architecture de type matriciel aussi appelé "en filet de poisson" (mesh) ;
- hiérarchiques (2) : structure typique des graphes où les sommets s'arrangent en couches hiérarchisées et pyramidales ;
- cycliques (3) : on peut identifier des cycles dans le graphe. L'exemple le plus parlant est le graphe circulaire ;
- centralisés ou polaires (4) : c'est une architecture où tous les sommets sont rattachés á un seul sommet, le pole ;
- Quelconques (5) : aucune propriété topologique ne semble émerger ;
  Multipolaires : c'est une architecture mixte entre les graphes centralisés et décentralisés.

Les réseaux multipolaires sont très étudiés en raison de leur proximité avec de nombreux cas concrets, notamment Internet ou les réseaux de neurones [23]. Clique :

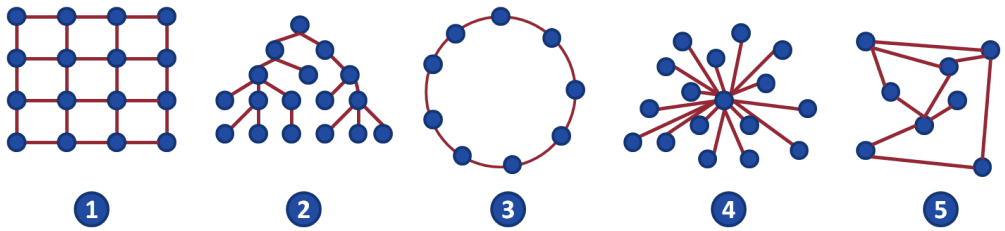

FIGURE 1.2 – Topologie typique d'un réseau multipolaire

Une clique d'un graphe est un sous-ensemble des noeuds de ce graphe dont le sous-graphe induit est complet, c'est-á-dire que deux sommets quelconques de la clique sont toujours adjacents.
Une clique est appelée maximale si elle possède le plus grand nombre de noeuds dans un graphe donné.
avant d'aborder les différentes techniques de la détection de communauté , nous expliquerons quelques terme de que nous résumerons comme suit

1.3 Problème de détections des communautés

La détection de communauté est un domaine de recherche actif depuis ces vingt dernières années, de très nombreuses approches ont été mises en oeuvre pour la détection de structures communautaires, Certaines méthodes considèrent le graphe dans son ensemble et effectuent une coupe pour trouver des communautés alors que d'autres privilègieront une approche nodale (c'est-ta-dire, un partitionnement fondée sur les propriétés de noeuds voisins).

1.3.1 Qu'est-ce qu'une communauté ?

Une communauté se définit par rapport á un graphe courant comme un groupe de noeuds qui sont fortement reliés entre eux et faiblement reliés au reste du réseau. Il peut s'agir par exemple des individus qui échangent beaucoup entre eux et peu avec les autres. Il est particulièrement intéressant d'identifier ces groupes afin de faire émerger la structure sous-jacente du graphe. On peut ainsi le diviser en groupes naturels d'individus (sans chevauchement, i.e. un noeud appartient á un unique groupe) qui peuvent etre de toute taille. Cette identification va prendre en compte uniquement la structure du graphe, ainsi que les éventuels poids d'arètes qui apportent de l'information supplémentaire, par exemple le niveau d'activité d'une relation.

1.3.2 Structure communautaire

Pour une meilleure compréhension des Réseaux Sociaux, les chercheurs ont essayé de trouver plus de caractéristiques structurelles de ces réseaux. [2] [21] ont montré que les réseaux réels ne sont pas des graphes aléatoires car ils présentent de grandes hétérogénéités, révélant un haut niveau d'ordre et d'organisation. En outre, la répartition des arètes n'est pas seulement globale, mais aussi localement non homogène, avec des concentrations élevées dans des groupes spéciaux de sommets et de faibles concentrations entre ces groupes (voir figure 1.3). Cette caractéristique des réseaux réels s'appelle la structure communautaire [9]. En effet, les graphes représentant les Réseaux Sociaux du monde réel présentent une structure modulaire avec des noeuds formant des groupes et éventuellement des groupes au sein de groupes [13] .
distinguent quatre types de communautés en ligne : les commu-

nautés de transaction, les communautés de relations, les communautés de fantaisie et les communautés d'intéret.

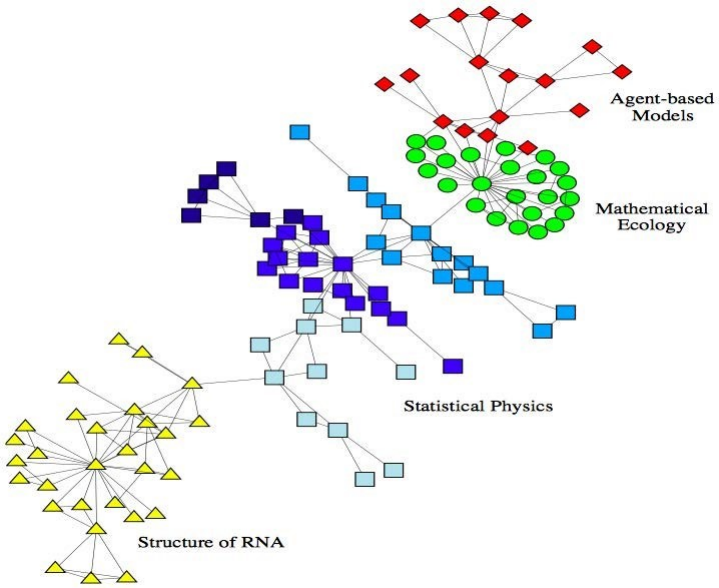

FIGURE 1.3 – Exemple d'un Réseau Social de collaboration entre scientifiques

1.4 Approches de détection de communautés

Il existe des dizaines de méthodes permettant de détecter des communautés dans un graphe. Fortunato dans un article de 2010 recense plus de 50 méthodes [7] ; Rémy Cazabet dans sa thèse indique que selon certains auteurs, il y aurait plus de 250 algorithmes de détection de communautés publiés [6]. Peu importe le nombre exact : ce n'est pas parce qu'un logiciel connu ne propose qu'une seule méthode de détection de communautés que cela autorise á parler des communautés dans un graphe . Surtout que le seul algorithme implémenté par défaut (il existe des mo-

dules additionnels permettant d'en implémenter d'autres) dans ce logiciel est non déterministe : en clair, le relancer plusieurs fois est susceptible de générer des résultats légèrement différents. L'approche traditionnelle est de minimiser le taux de coupe (i.e le nombre d'aretes inter-communautés) en réalisant un partitionnement, avec comme inconvénient principal de devoir choisir en amont le nombre de communautés. Par la suite, une approche alternative visant á trouver des sous-groupes densément connectés s'est répandue, avec comme avantage de pouvoir sélectionner automatiquement le nombre de groupes. Elle comprend notamment les approches de maximisation de modularité, définie ci-dessous. L'ensemble de ces approches sont non supervisées, on n'apprend pas á partir de communautés déjá labellisées.La plupart des algorithmes créent des communautés non recouvrantes ; un sommet ne peut appartenir qu'á une communauté et une seule. Or, les recouvrements de communautés sont fréquents dans les réseaux empiriques. [7]

1.5 Des méthodes et techniques de la détection de communauté

1.5.1 Louvain

La méthode de Louvain [4] implante une méthode d'optimisation gloutonne locale de la modularité. A l'état initial chaque noeud est affecté á une communauté différente des autres. L'algorithme applique ensuite une itération de succession de deux phases :

— Phase d'affectation des noeuds : Pour chaque noeud x on

évalué le gain de la modularité si on le déplace dans la communauté de ses voisins directs. On déplace x dans la communauté du voisin qui maximise le gain de la modularité. Si aucun gain n'est trouvé le noeud reste dans sa communauté ;

— Phase de compression : On compresse le graphe obtenu en remplacant chaque communauté par un seul noeud. Deux noeuds C-x , C-y .

Le poids de lien entre deux communautés est égal á la somme des poids des liens reliant des noeuds de deux communautés. La figure 1.4 illustre l'exécution de ces deux phases dans une double itération. L'algorithme s'arrète s'il a y plus de possibilité de réaffectation de noeuds ou si un maximum de modularité soit atteint. La complexité théorique de l'algorithme n'est pas étudiée , mais d'une manière expérimental, cette complexité est évaluée a O(n log-n) ce qui fait de Louvain la méthode la plus rapide pour l'identification de communautés.

1.5.1.1 L'Algorithme

Algorithme 1 L'algorithme de Louvain

Entrée : Un graphe G = (V ;E)

1 : A répéter jusqu'á l'obtention d'un score local optimal
2 : Phase 1 : partitionner le réseau de manière gloutonne utilisant la modularité
3 : 1) Assigner á chaque noeud une communauté spécifique
4 : 2) Pour chaque noeud i du réseau
Pour chaque voisin j de i, choisir le voisin pour lequel l'assignation du
Noeud i dans une communauté augmenterait le plus la modularité
Répéter le processus jusqu'a ce qu'il n'y ait plus de changement

5 : Phase 2 : Agglomérer les sous-graphes en nouveaux noeuds
6 : 1) Laissons chaque communauté Ci former un nouveau noeud i
7 : 2) Laissons les aretes entre les nouveaux noeuds i et j comme étant la réunion des aretes entre les noeuds qui étaient dans C-i et C-j au sein du graphe précédent .

á l'autre, Il a été montre que l'ordre jouait un role important sur la qualité des communautés détectées.

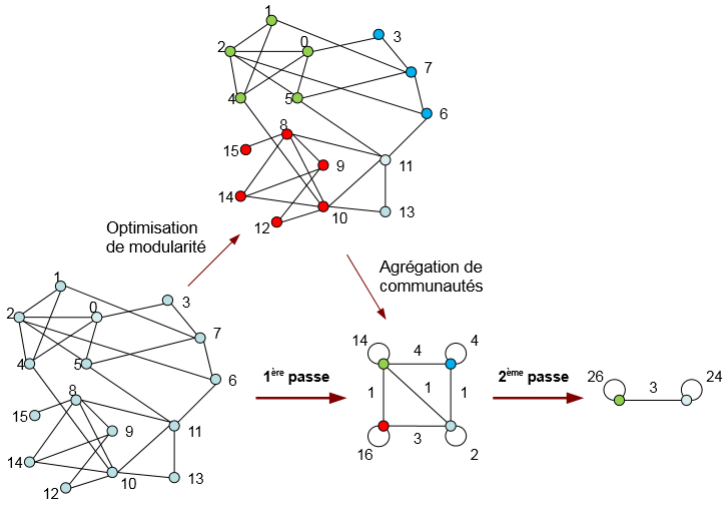

FIGURE 1.4 – Exemple d'application de la méthode de Louvain sur un graphe á 16 sommets [4]

1.5.1.2 Avantages

Cette approche permet d'échapper au défaut de la modularité connu sous le nom de limite de résolution . Si le fait d'associer une partition du premier niveau avec une autre devait engendrer une perte de modularité, cette association ne serait pas réalisée

par la méthode de Louvain. De ce fait, les petites structures dotées d'une bonne modularité ne risquent pas d'etre noyées dans des structures plus grandes et moins significatives. C'est ce qu'il risquerait d'arriver avec un algorithme top-down appliqué á un trop grand graphe.

1.5.1.3 les Inconvénients

Le défaut connu de la méthode provient de la sensibilité du résultat á l'ordre de traitement des sommets. L'ordre dans lequel les sommets sont considérés a une influence sur le partitionnement.

1.5.2 Newman

L'algorithme hiérarchique divisif le plus connu, abrégé souvent par GN pour désigner les auteurs Girvan et Newman, est présenté dans [9]. Il est l'un des algorithmes les plus référencés dans le domaine de la détection de communautés. GN est basé sur une mesure de centralité d'intermédiarité des liens, définie par le nombre de plus courts chemins passant par un lien. Cet algorithme est particulièrement intuitif. La première étape consiste á, pour tous les arcs, calculer une généralisation de la mesure de centralité nommée betweenness définie comme le nombre de plus courts chemins passant par un arc. Vu qu'il existe peu de liens entre les communautés, les auteurs considèrent ces liens comme des ponts de passage permettant de relier rapidement deux communautés différentes. Ces liens se trouvent donc sur un nombre important de plus courts chemins. En effet, l'algorithme GN mesure la centralité d'intermédiarité pour un lien (u, v) en s'inspirant de la centralité d'intermédiarité d'un noeud. L'arc

avec la plus grande betweenness est ensuite retiré du réseau.

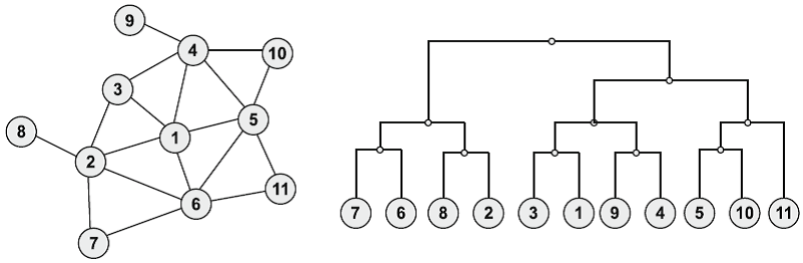

FIGURE 1.5 – Exemple d'un dendrogramme hiérarchique pour Newman

L'algorithme GN est de complexité O(n * m2) avec n le nombre de noeuds et m le nombre de liens du réseau. Cette complexité très élevée rend GN inexploitable pour les réseaux de grande taille. Mais malgré cela, GN a fait l'objet de plusieurs travaux de recherche visant á l'améliorer ou á l'adapter (Wilkinson, Huberman, 2004 ; Massen, Doye, 2005 ; Yoon et al., 2006) grace á ses concepts théoriques robustes et il est également implémenté dans plusieurs outils d'analyse des réseaux complexes (Csardi, Nepusz, 2006 ; Schult, Swart, 2008). Ces deux opérations sont itérées en succession jusqu'á ce que le graphe devienne disjoint, et la procédure entière est répétée sur chaque composante connexe. Une variante de l'algorithme GN est proposée par (Fortunato et al., 2004) et est basée sur une mesure de similarité appelée centralité d'information . Les auteurs définissent l'efficacité de communication S-ij entre deux noeuds i et j du graphe comme étant l'inverse de leur distance d-ij .
algorithme est plus efficace que GN mais moins rapide car il a une complexité de O(n * m^3).

1.5.2.1 L'Algorithme

Algorithme de Girven Newman

Data : a graphe G
Resultat : the hierarchy of nested partitions of V(G)
Go := G ;
Po := V(G) ;
i := 0 ;
c := 0 ;
while E(G_i) $\neq \emptyset do$
$S_i := B(e) : e \in E(G_i)$;
$mx := maxS_i$
$M_i := e \in E(G_i) : B(e) = mx$;
$\bar{E}_i := M_i$;
$\sigma_i := 1$;
$while \mid \bar{E}_i \mid > 1 do$
$\sigma_i := \sigma_i + 1$
$U_{\sigma i} := B(A) : A \subset M_i \wedge \mid A \mid = \sigma_i$;
$gmx := maxU_{\sigma i}$
$\bar{E}_i := A \subset M_i :\mid A \mid = \sigma_i \wedge B(A) = gmx$;
end
$G_i + 1 := G_i \backslash \bar{E}_i$;
$if b_o(G_i + 1) > b_o(G_i) then$
$c := c + 1$;
$P_c := V_1, ..., V_{rc} : < V_1 >, ..., < V_{rc} > areconnectedcomponentsofG_i + 1$;
end
$i := i + 1$;
end
$returnP_i : i = o, ..., c$

1.5.3 k-clique

La définition d'une communauté k –clique est le suivant ; Une communauté k -clique est une union de toutes les k -cliques (sous-graphes complets de taille k) qui peuvent etre atteintes de l'une ou l'autre par une série de k -cliques adjacentes (oú l'adjacence signifie partager k - 1 noeuds). Ces communautés k -cliques ont les propriétés suivantes :

— leur définition est déterministe ;

— le chevauchement est autorisé ;

— chaque communauté identifies un ensemble de noeuds cohésifs.

1.5.3.1 Conception

Puisqu'elle peut etre décrite comme une chaine de sous-graphes entièrement connectés (c'est-á-dire k -cliques). C'est principalement pour cette raison que nous avons adopté les communautés k -cliques pour analyser la topologie de l'Internet au niveau AS. Sur la base de la définition des communautés k -cliques, nous pouvons prouver que, pour chaque communauté k -clique d'ordre k, communauté i (k), il existe une et une seule communauté k -clique d'ordre k-1 (ou k-1 -clique communauté), communauté j (k - 1), telle que : communauté i (k) communauté j (k - 1)

c'est-á-dire que la communauté i (k) est un sous-graphe de la communauté j (k - 1) [5].

1.5.3.2 L'Algorithme

Algorithme K-Clique
Input : user-training.csv
Output : group - of - users.
Step 1 : initialize mymatrix , sim
Step 2 : for i = 1 to mymatrix do
Step 3 : for j = 1 to mymatrix do
Step 4 : initialize a
Step 5 : for k = 2 to 4 do
Step 6 : if mymatrix [i,k] == mymatrix [j,k]
a[k] = 1
else
a[k] = 0
Step 7 : if (a[2]+a[3]+a[4]==3
sim[i,j]=1
else
sim[i,j]=0
Step 8 : initialize my-network , k4, b , d
Step 9 : for i = 1 to b do
Step 10 : for j=1 to b do d[i,j] = b[i][j] ;

1.5.4 Fluid

1.5.4.1 Conception

L'algorithme Fluid Communautés (FluidC) est un algorithme de détection de communautés basé sur l'idée d'introduire un certain nombre de fluides (c'est-á-dire de communities) dans un environnement non-homogène (c'est-á-dire graphe), où les fluides s'étendent et se poussent les uns les autres influencé par la topologie de l'environnement jusqu'á la stabilité. Une

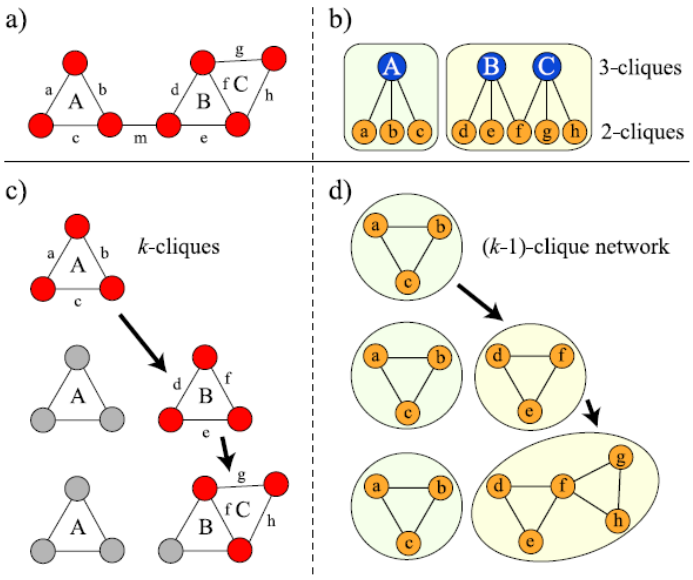

FIGURE 1.6 – Illustration of the algorithm for detecting k-clique communities in a simple example network.

communauté de fluides conque les parties de l'environnement qui ont une topologie favorable (c'est-á-dire, qui sont fortement connectées avec ses sommets) tout en perdant certaines parties au profit d'autres communautés fluid.
De manière significative, lorsqu'une communauté fluide se propage á travers plus de sommets, sa densité diminue, ce qui réduit sa capacité á conquérir et á défendre des sommets.

Considérez un graphe G = (V, E) formé par un ensemble de sommets V et un ensemble d'arètes E. FluidC initialise k-fluid communautés sur k different sommets de V , communautés qui commenceront á s'étendre á travers le graphe. A tout moment, chaque communauté fluide y a une densité totale de 1,0. Lors-

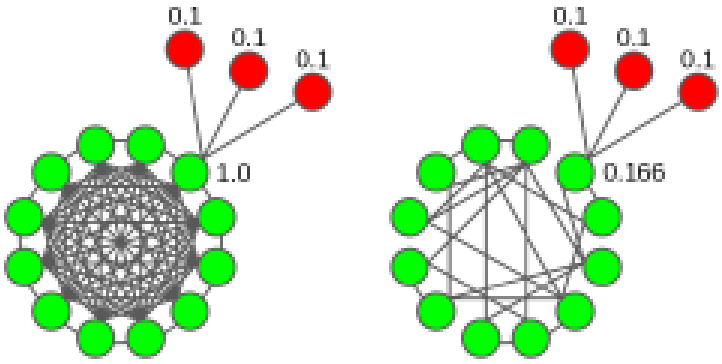

FIGURE 1.7 – Deux cas de règle de mise á jour sur le vert. Celui de gauche appartient á une communauté très bien intriquée et á droite en contient moins bien. La communauté rouge ne conquerra le sommet vert que dans l'exemple du côté droit.

qu'une fluid communauté est compactée en un seul sommet (par exemple, á l'initialisation), ce sommet détient la densité totale de la communauté (c'est-á-dire 1,0), qui est également la densité maximale qu'un sommet unique peut avoir. Lorsqu'une communauté s'étend sur plusieurs sommets, sa densité est répartie uniformément entre les vertices qui la composent [19].

comment travaillé :

Le travail de FluidC fonctionne de la manière de l'approche de propagation introduite par LPA. á chaque étape, FluidC itère sur tous les sommets dans un ordre aléatoire, en mettant á jour la communauté á laquelle appartient chaque sommet en utilisant une règle de mise á jour.

En termes simples, la règle de mise à jour additionne les densités des voisins d'un sommet dans la communauté, y compris

lui-meme,et renvoie la communauté avec la densité maximale. Si la densité maximale est partagée par deux ou plusieurs communautés mais que la communauté précédente du sommet n'en fait pas partie, une communauté aléatoire est choisie. Si la communauté précédente du sommet fait partie de l'ensemble des communautés á densité maximale, le sommet conserve sa communauté précédente.

Remarque :
remarquez que cela garantit qu'aucune communauté ne sera jamais éliminée du graphe, puisque, lorsqu'une communauté est comprimée en un sommet, cette communauté a la densité maximale possible pour la règle de mise á jour de ce sommet (c'est-á-dire 1,0). Formellement, nous défine la règle de mise á jour comme suit.

$$C'_v = argmax_{c \subset C} \sum_{wv, N_v} D_w(C_w, c) \qquad (1.2)$$

L'Algorithme :

Algorithm 1 Fluid Communities

Require: $G = (V, E), num.\ communities\ k > 0, max\_iterations > 0$

$V_{rand} \Leftarrow V$ in random order
for $i \in \{0..k\}$ **do**
  $v \Leftarrow V_{rand}[i]$
  $\mathcal{C}_v \Leftarrow i$
  $\mathcal{D}_v \Leftarrow 1.0$
end for
$converged \Leftarrow False$
while $num\_iterations < max\_iterations$ **and** $converged = False$ **do**
  $converged \Leftarrow True$
  for $v \in V$ in random order **do**
    $v\_new\_community \Leftarrow$ Community Update(G, v)
    if $v\_new\_community \neq \mathcal{C}_v$ **then**
      $v\_old\_community \Leftarrow \mathcal{C}_v$
      $\mathcal{C}_v \Leftarrow v\_new\_community$
      Density Update$(G, v_old_community)$
      Density Update$(G, v_new_community)$
      $converged \Leftarrow False$
    end if
  end for
end while

Algorithm 2 Community Update

Require: G, v

$community\_density[\mathcal{C}_v] \Leftarrow \mathcal{D}_v$
for $w \in Neighbours(v)$ **do**
  if $community\_density[\mathcal{C}_w]$ exists **then**
    $community\_density[\mathcal{C}_w] \Leftarrow community\_density[\mathcal{C}_w] + \mathcal{D}_w$
  else
    $community\_density[\mathcal{C}_w] \Leftarrow \mathcal{D}_w$
  end if
end for
$\mathcal{C}'_v \Leftarrow \mathcal{C}_v$
$max\_density \Leftarrow max(community\_density)$
if $community\_density[\mathcal{C}_v] < max\_density$ **then**
  $\mathcal{C}'_v \Leftarrow rand(community\_density = max\_density)$
end if
return $\mathcal{C}'_v$

Algorithm 3 Density Update

Require: $G, community\_to\_update$

$V_l \Leftarrow v \in V, \mathcal{C}_v = community\_to\_update$
for $v \in V_l$ **do**
  $\mathcal{D}_v \Leftarrow 1.0/|V_l|$
end for

1.6 Conclusion :

détection de communautés, En outre, nous avons donné un petit rappel sur les graphes, Pour achever notre étude nous allons présenter quatre méthodes les plus populaires utilisé pour faire des test et d'évaluation avec des méthodes qui en vue dans la prochaine chapitre.

Chapitre 2

Les Mesures D'évaluation

:

2.1 Introduction

Dans le domaine de la science des réseaux il y a de nombreuse méthodes de détection communautaire et aussi des mesures pour évaluer sa qualité. Dans ce chapitre nous discutant quelque notion des réseaux en général ensuite présenté des quatre mesures nous avons utilisés deux pourévaluer les méthodes qui ont été mentionnés dans le chapitre précédant.

2.2 Les Réseaux

Wasserman décrit un réseau social comme un ensemble fini d'acteurs ainsi que les relations définies entre eux . On pourra prendre l'exemple du réseau social des élèves d'un collège qui ont déjá été dans la mème classe. Les acteurs seront alors tous les élèves du collège. La relation entre deux éléments sera alors "ont déjá été dans la mème classe". Cette définition fait référence de facon directe á la notion mathématique de relation et á celle de graphe utilisée pour représenter le réseau. Avec l'émergence du Web 2.0 et des réseaux numériques, la notion de réseau social a dú ètre généralisée pour tenir compte de caractéristiques décrivant les acteurs du réseau et leurs relations. Ceci a conduit á la définition de la notion de réseau d'information homogènes ou hétérogènes par Han , celle de graphe d'information par Moser et al. ou encore de graphe avec attributs par Zhou [22] .

2.2.1 Rèseaux rèels

La disponibilité d'une partition de référence pour un graphe G peut ètre le résultat d'un des trois processus suivants : Anno-

Résaux	n	m	communautés
club de Karaté de Zachary	34	78	2
Football	115	616	11
Strike	24	38	3
Livres politiques	3892	17262	30
Dauphins	62	159	2

TABLE 2.1 – Quelques réseaux réels souvent utilisés comme un benchmark pour les algorithmes de détection de communautés

tation par un expert : Les graphes pour lesquels des experts ont défini des partitions de références sont souvent des graphes de très petites tailles. Le tableau 2.1 décrit les caractéristiques des principaux réseaux réels annotés par des experts et qui sont souvent utilisés comme un benchmark pour les algorithmes de détection de communautés.

2.2.1.1 Club de karatè de Zachary

La première comparaison a fait sur le club de karaté de Zachary [27].C'est un réseau très populaire et trés utilisé par les algorithmes de détection de communauté qui contient 34 noeuds qui represente membres d'un club de karaté dans une université aux Etats Unis, et 78 aretes.. Il s'agit d'un réseau trés populaire et trés utilisé par plusieurs algorithmes afin de tester leurs performances puisque sa structure de communautés est connue á l'avance.. La figure 2.1 nous montre le rèsultat d'exécution sur le rèseau de Zachary.

2.2.1.2 Les dauphins de Lusseau

La deuxième le réseau utilisé dans notre etudes est de dauphins de Lusseau. Ce réseau contient 62 noeuds et 158 liens. La figure

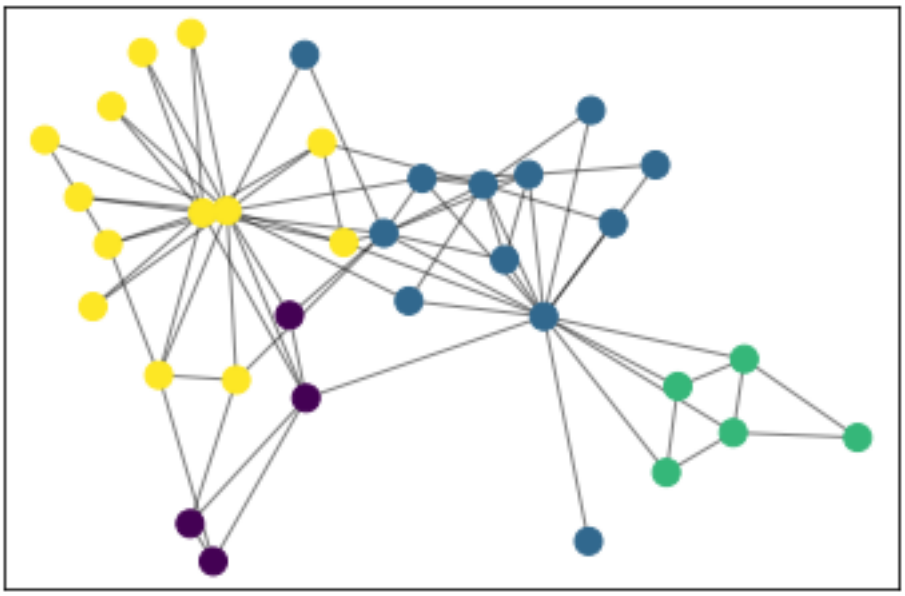

FIGURE 2.1 – le réseau de Zachary.

2.2 nous montre le rèsultat d'exécution sur le rèseau de dauphins

2.2.1.3 Les livres politiques

Un troisième exemple que nous avons traité est le réseau de livres politiques [14]. Il ne s'agit pas de relations entre les etres humains, mais plutot de relations entre l'achat de plusieurs titres sur Amazon. Le réseau est constitué de 3892 livres avec 17262 liens se trouvant entre les livres achetés ensemble. La figure 2.3 nous montre le rèsultat d'exécution sur le rèseau de livres

2.2.2 Benchmark LFR pour tester les algorithmes de détection de communauté :

Principe

Le principe est de générer des graphes artificiels avec des struc-

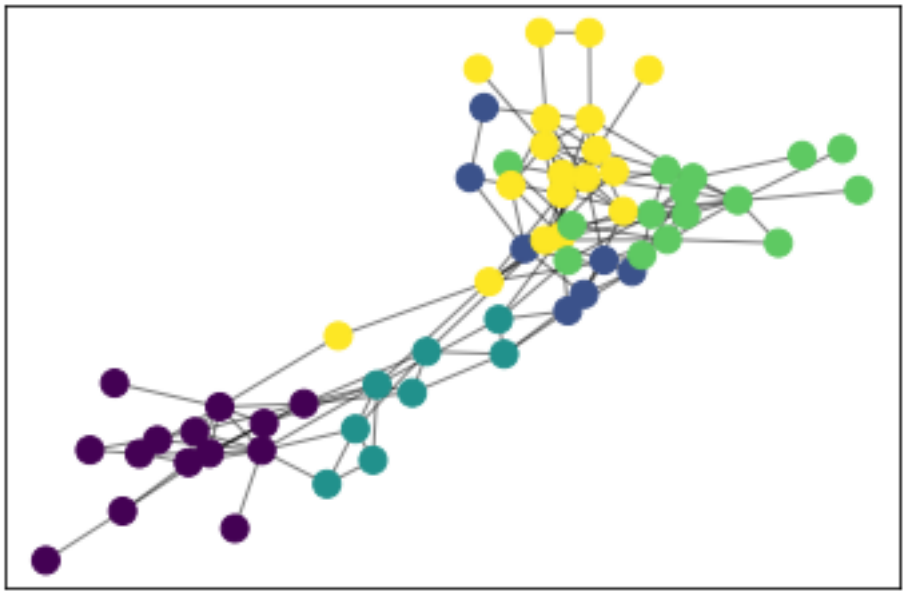

FIGURE 2.2 – le réseau dauphins de Lusseau.

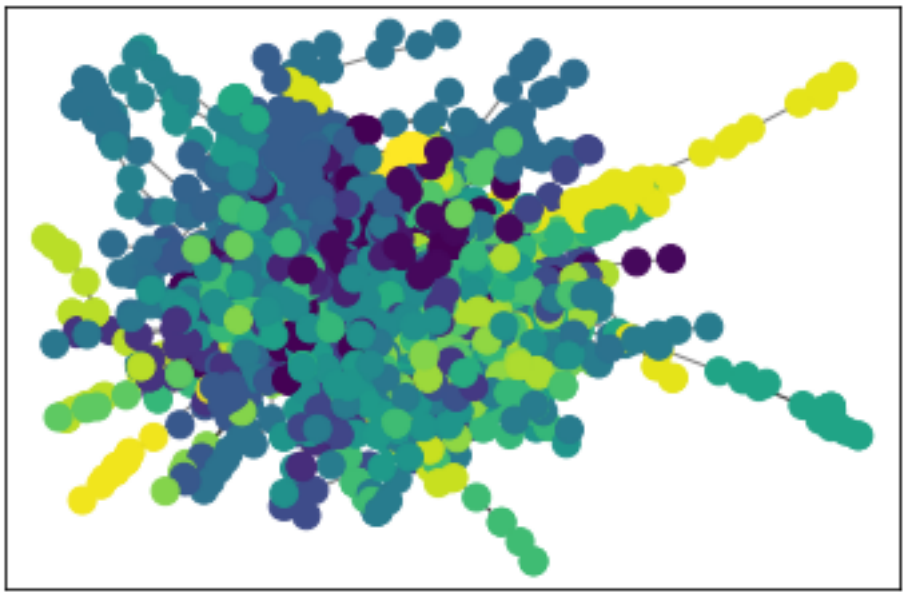

FIGURE 2.3 – le réseau livres politiques.

tures communautaires paramétrables. Le modèle générateur le plus récent est le modèle LFR [15]. Les générateurs sont basés sur la définition de communautés denses qu'on relie entre elles avec une densité paramétrable afin de controler la difficulté de la reconnaissance de la structure communautaire. Cette approche a le mérite de la simplicité et de la possibilité de générer des graphes de différentes tailles et de différents degrés de difficulté á décomposer en communautés. Mais rien ne prouve que les graphes générés par ces modèles sont bien similaires dans leurs mécanismes de formation aux graphes de terrain. L'historique de l'évolution de nos connaissances sur la caractérisation des graphes de terrain ne permet pas de dire qu'on connais déjá la liste ultime de leurs caractéristiques topologiques. Quant á la dynamique de l'évolution de communautés rien n'est encore bien établie de sorte á permettre d'utiliser les actuels modèles génératifs pour la détection de communautés dynamiques.La disponibilité d'une partition de référence permet d'utiliser les différentes mesures de distance entre clusters développées pour l'évaluation des approches de classification non-supervisé (ou clustering) .
Soit V un ensemble de noeuds d'un graphe G. On désigne par R = r-1, . . . , r-n une partition de reference de V .
Soit U = u-1, . . . , u-m une partition calculée par un algorithme de détection de communautés.
Les mesures suivantes sont fréquemment employées pour mesurer la similarité entre deux partitions R et U. [15]
Les Paramètres de LFR :
LFR-benchmark-graph(n,tau1,tau2,mu,average-degree=None,min-degree=None, max-degree=None,min-community=None, max-community=None, tol=1.0e-7, max-iters=500, seed=None,) :

Paramètres	Valeurs
Nombre de sommets	233-22186
Degré maximum	0.1V
Taille maximale de la communauté	0.1V
Degré moyen	20
Exposant de distribution de degrés	-2
Exposant de distribution de la taille de la communauté	-1
Coefficient de mélange	0.03 - 0.75

TABLE 2.2 – Hyperparameters of LFR benchmark

Retourne le graphe de référence LFR. Cet algorithme se déroule comme suit

1. Trouver une séquence de degrés avec une distribution de type loi de puissance, et une valeur minimale "min-degré", qui a un degré moyen approximatif "average-egree". Pour ce faire, on peut soit

   — en spécifiant "min-degré" et non "moyenne-degré",

   — en spécifiant "average-degree" et non "min-degree", auquel cas un degré minimum adéquat sera utilisé.

   degré sera trouvé. "max-degree" peut également ètre spécifié, sinon il sera fixé á "n". Chaque noeud *u* aura mumathrm-deg(u)' arètes le reliant aux noeuds des communautés autres que la sienne et (1 - mu) mathrmdeg(u)' a`etes le reliant aux noeuds de sa propre communauté.

2. Génère des tailles de communauté selon une distribution de loi de puissance avec l'exposant "tau2". Si "min-community"

et "max-community" ne sont pas spécifiés, ils seront sélectionnés pour etre "min-degree" et "max-degree", respectivement. Les tailles des communautés sont générées jusqu'á ce que la somme de leurs tailles soit égale á "n".

3. Chaque noeud se verra attribuer une communauté de facon aléatoire, á condition que la communauté soit suffisamment grande pour que le degré intra-communautaire du noeud, '(1 - mu) mathrmdeg(u)', comme décrit á l'étape 2. Si une communauté devient trop grande, un noeud aléatoire sera sélectionné pour etre réaffecté á une nouvelle communauté, jusqu'á ce que tous les noeuds aient recu une communauté.
4. Chaque noeud *u* ajoute ensuite '(1 - mu) mathrmdeg(u)'des aretes intra-communautaires et 'mu mathrmdeg(u)' des aretes inter-communautaires.

Parametres :
n : int Nombre de noeuds dans le graphe crée.
tau1 : float Exposant de loi de puissance pour la distribution des degrés du graphe crée. tau1 $>$ 1.
tau2 : float Exposant de loi de puissance pour la distribution de la taille de la communauté dans le graphe crée. Tau2 $>$1.
mu : float Fraction des aretes intra-communautaires incidentes á chaque noeud. mu [0, 1].
average-degree : float Degré moyen souhaité des noeuds dans le graphe crée. average-degree [0, *n*]. Un seul de ces "min-degree" doit etre spécifié, sinon un :exc :'NetworkXError' est émis.
min-degree : int Degré minimum des noeuds dans le graphe crée. Cette valeur doit etre dans l'intervalle [0, *n*]. Cette valeur doit etre comprise dans l'intervalle [0, *n*]. Il faut spécifier exactement l'une de ces deux valeurs et "average-degree", sinon un :exc :'NetworkXError' est généré.

max-degree : int Degré maximum des noeuds dans le graphe crée. S'il n'est pas spécifié, il est fixé á "n", le nombre total de noeuds dans le graphe.
min-community : int Taille minimale des communautés dans le graphe. Si elle n'est pas spécifiée, elle est fixée á "min-degree".
max-community : int Taille maximale des communautés dans le graphe. Si elle n'est pas spécifiée, elle est fixée á "n", le nombre total de noeuds dans le graphe.
tol : float Tolérance lors de la comparaison des flottants, en particulier lors de la comparaison des valeurs moyennes des degrés.
max-iters : int Nombre maximum d'itérations pour essayer de créer la taille des communautés, la distribution des degrés, et les affiliations des communautés.
seed : integer, random-state, or None (default) Indicateur de l'état de génération des nombres aléatoires.
Retours — G : graphe NetworkXLe
graphe de référence LFR généré selon les paramètres spécifiés. Chaque noeud du graphe possède un attribut de noeud "community" qui stocke la communauté (c'est-á-dire l'ensemble des noeuds) qui l'inclut.
La modularité ; le F1-score ; la pureté ; la conductance ; l'indice de Rand ajustée ; le NMI Pour apprécier la qualité de partitionnement des algorithmes de détection de communautés, des mesures existent fondées sur la comparaison entre partition réelle et partition trouvée par l'algorithme, il s'agit des mesures supervisées, des mesures fondées sur le nombre de liens a l'intérieur des communautés et á l'extérieur, ou sur la comparaison de structures avec un graphe aléatoire sont utilisées, il s'agit dans ce cas des mesures non supervisées.

2.3 Les méthodes d'évaluation de la qualité des structures communautaires

2.3.1 L'indice de Rand

Cette mesure, est basée sur le comptage de nombre d'accords entre deux partitions sur l'appartenance communautaire de chaque pairs de noeuds. Considérons deux partitions P1 et P2 d'un mème ensemble S-v . L'idée pour évaluer la ressemblance entre deux partitions consiste á mesurer le taux de bonnes assignations de ces paires d'observations. Pour chaque couple d'observations , on peut compter quatre types d'assignations possibles :

— N-11 le nombre de paires de noeuds classées ensemble selon P1 et P2

— N-10 le nombre de paires de noeuds classées ensemble selon P1 et séparées selon P2

— N-01 le nombre de paires de noeuds séparées selon P1 et classées ensemble selon P2

— N-00 le nombre de paires de noeuds séparées a la fois dans P1 et dans P2.

L'indice de Rand est donne par la formule suivante :

$$Rand(P1, P2) = \frac{N_{11} + N_{00}}{N_{11} + N_{10} + N_{01} + N_{00}} \tag{2.1}$$

Cette mesure varie entre 0 et 1 et prend la valeur maximale en cas de parfaite correspondance. Cependant, cet indice ne demande pas d'établir un appariement entres les classes réelles et celles estimées, car seule est prise en compte la classification

commune des différentes paires d'éléments dans les deux partitions.Cela induit deux problèmes majeurs.
Premièrement, l'indice est fonde exclusivement sur les paires d'objets, et non les objets eux-mèmes.
Deuxièmement, une inversion de classification pour deux éléments dans deux parties de petite taille sera moins pénalisée que si elle se produit dans des parties de plus grande taille.
De plus, l'indice de Rand varie beaucoup entre deux partitions tirées au hasard. C'est pourquoi l'indice de Rand ajustée (ARI) (Hubert et Arabie (1985)) fut proposé dont l'espérance est nulle lorsque les partitions sont tirées aléatoirement.Il a pour forme :

$$ARI(P1, P2) = \frac{2(N_{11}N_{00} - N_{01}N_{10})}{(N_{01} + N_{00})(N_{11} + N_{01}) + (N_{00} + N_{10})(N_{10} + N_{11})} \quad (2.2)$$

Sa valeur est comprise entre 0 et 1. Elle prend la valeur 1 lorsque les deux partitions sont identiques. Si la valeur est proche de 0, les deux partitions sont très différentes.

L'Algorithme

Algorithm de l'indice de Rand

```
import numpy as np
from scipy.misc import comb

def rand-index-score(clusters, classes) :

tp-plus-fp = comb(np.bincount(clusters), 2).sum()
tp-plus-fn = comb(np.bincount(classes), 2).sum()
```

A = np.c-[(clusters, classes)]
tp = sum(comb(np.bincount(A[A[:, 0] == i, 1]), 2).sum()
for i in set(clusters))
fp = tp-plus-fp - tp
fn = tp-plus-fn - tp
tn = comb(len(A), 2) - tp - fp - fn
return (tp + tn) / (tp + fp + fn + tn)

---

exemple de resultat :
In [1] : clusters
Out[1] : [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

In [2] : classes
Out[2] : [0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 2, 1, 0, 2, 2, 2, 0]

In [3] : rand-index-score(clusters, classes)
Out[3] : 0.67647058823529416

2.3.2 Mesure F1

La F-mesure de [20] est un indicateur de synthèse communément utilisé pour évaluer les algorithmes de classification de données textuelles, á partir de la précision et du rappel. Elle est utilisée indifféremment pour les classifications et les catégorisations.
Cependant, si vous ètes profondément intéressé par la définition de la mesure F, vous devez recapituler les définitions des moyennes arithmtéique et harmonique.

2.3.2.1 Arithmétiques et harmonique signification

L'arithmétique A (une moyenne au sens habituel du terme) et la harmonique H sont définies comme suit :

$$A = \frac{1}{n} \times \sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + x_2 + ... + x_n) \tag{2.3}$$

$$H = \frac{n}{(\sum_{i=1}^{n} * \frac{1}{x_i})} = \frac{n}{(\frac{1}{x_i} + ... + \frac{1}{x_n})}. \tag{2.4}$$

Si x1= P et x2 = R , A et B sera :

$$A = \frac{1}{2}(P + R) \tag{2.5}$$

$$H = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{2.6}$$

La moyenne harmonique est plus intuitive que la moyenne arithmétique lorsqu'on calculer une moyenne de rapports.

$$mesureF\frac{((1 + \beta^2) * Precision * Rappel)}{((\beta^2 * Precision) + Rappel)}, avec\beta^2 = 1 \tag{2.7}$$

La F-mesure correspond á une moyenne harmonique de la précision et du rappel. e paramètre B permet de pondérer la précision ou le rappel et vaut généralement 1, [10]. La mesure devient :

$$\frac{(2 * Precision * Rappel)}{(Precision + Rappel)} \tag{2.8}$$

la figure 2.4 explique sa evénement en détail :

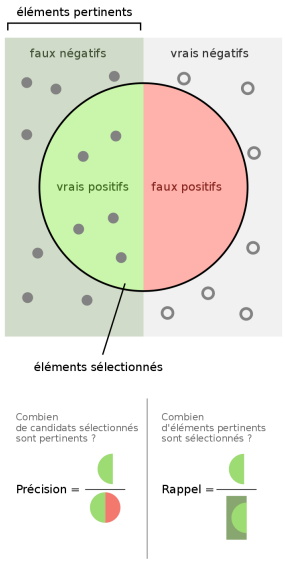

FIGURE 2.4 – Précision et rappel (recall)

L'avantage de ce choix est que lorsque la précision est égale au rappel, on obtient :
Précision = Rappel = F1-mesure.
Ceci facilite la lecture et on recherche a'a maximiser la F-mesure en maximisant simultanément la précision et le rappel. Considérons la partition trouvée par l'algorithme de détection de communautés P1 = c1 , c2 , ..., ct et la partition représentant l'ensemble des communautés de vérité de terrains P2 = c1 , ..., ck . On note que le nombre de communautés peut être différent entre les deux partitions. La moyenne du F-mesure que nous utilisons est celle concernant les communautés trouvées par un algorithme ou nous comparons chacune d'entre elles avec la communauté la plus similaire parmi les communautés de la partition de vérité de terrain, c'est-á-dire, avec la communauté de vérité de terrain partageant le plus de noeuds en commun. Nous avons ainsi :

$$\bar{F}(P_1, P_2) = \frac{1}{t}\sum_{i=1}^{t} max_{j\in[1,k]} F_1(C_i, c'_i) \tag{2.9}$$

Cette valeur illustre á quel point chaque communauté que l'algorithme a trouvée est similaire aux communautés de la partition de vérité de terrain, en regardant les communautés entre les deux partitions partageant le plus de noeuds en commun. Une valeur proche de 0 signifie que les deux partitions sont totalement différentes, sans aucune similarité entre communautés détectées et réelles alors qu'une valeur proche de 1 signifie que les deux partitions sont très semblables avec les mèmes noeuds á l'intérieur des mèmes communautés.

L'Algorithme

F-mesure Algorithme

Input : matrix P and probability p(Y=0)
define matrix W with elemnts given by Eq. 7 ;
compute F= PW
for k = 1 to m do
solve the inner optimization problem(3) that can be reformulated as :

$$h^(k)^+ = argmax_h \in H_k 2\sum_{i=1}^{m} h_i f_i k \tag{2.10}$$

by setting $h_i = 1 for top k elements in the k - th column of matrix F, and h_i = 0 for the rest$;
$store a value of$

$E_y \sim p(Y)[F(y, h^(k)^*)]2 \sum_{i=1}^{m} h_i^(k)^* f_i k(2.11)$
end for
for k= 0 take $h^(k)^+ = 0, and E_y(Y)[F(y,0) = p(Y = 0);$
$solvetheouteroptimizitionproblem(4);$
$h*_F = argmax_h \in h^(0) * + ...h^(m) * E_y \sim p(Y)[F(y,h)];$
$returnh *_F and E_y \sim p(Y)[F(y,h)];$

Le problème que pose cette méthode est qu'elle ne permet pas la différenciation des erreurs et reste sensible á la distribution des classes car basée sur la précision et le rappel.

2.3.3 L'information mutuelle normalisée (NMI) :

C'est une mesure qui permet de représenter le degré de dépendance entre deux partitions P1 et P2. Elle est fondée sur la théorie de l'information. La probabilité qu'un noeud choisi au hasard dans une partition P1 appartienne á la communauté k est P(k) =nk/n ou nk est le nombre de noeuds dans la communauté k et n est le nombre total de noeuds du système [16].
. L'entropie de Shannon, en utilisant la distance de Kullback-Leibler, est définie comme :

$$H(P1) = -\sum_{k=1}^{|P1|}(\frac{n_k}{n} log_2 \frac{n_k}{n}) \tag{2.12}$$

Ou P1 est le nombre de communautés dans la partition P1. L'entropie de Shannon de la partition P1 représenté la quantité d'information fournie par cette mème partition sur la topologie du graphe en termes de communautés.

L'information mutuelle I (P1 ; P2) évalue le niveau d'interdépendance entre deux partitions d'un graphe. Il est possible de définir une matrice de confusion pour les partitions P1 et P2 en identifiant combien de noeuds nij de la communauté i de la partition P1 sont dans la communauté j de la partition P2.
L'information mutuelle est :

$$I(P1; P2) = \sum_{j=1}^{P2} \frac{n_i j}{n} log_2(\frac{n_i j}{n_i n_j} \tag{2.13}$$

Ou ni est le nombre de noeuds de la communauté i de la partition P1 et nj est le nombre de noeuds de la communauté j de la partition P2. La variation de l'information V (P1 ; P2) est V (P1 ; P2) = H (P1) + H (P2) - 2I (P1 ; P2) qui mesure la "distance" de l'information entre les deux partitions P1 et P2. Pour l'obtention d'une valeur comprise entre 0 et 1, [16]
, on définit l'information mutuelle normalisée comme étant :

$$NMI(P1, P2) = \frac{I(P1, P2)}{\sqrt{H(P1) + H(P2)}} \tag{2.14}$$

Sa valeur peut varier entre 0 et 1. Plus la valeur est proche de 1, plus les deux partitions sont identiques.

Parametres	labels-true : int tableau, forme = [n-echantillons] Un regroupement des donnees en sous-ensembles disjoints. labels-predint : int de type tableau de forme (n-echantillons,) Un regroupement des donnees en sous-ensembles disjoints. contingence{ndarray, sparse matrix} de forme (n-classes-true, n-classes-pred), defaut=Non . une matrice de contingence donnee par la fonction contingency-matrix. Si la valeur est Non, elle sera calculee, sinon la valeur donnee est utilisee, avec labels-true et labels-pred ignores.
Return	Mi : flottant Information mutuelle, une valeur non-negative

TABLE 2.3 – Hyperparameters of NMI

L'Algorithme

NMI Algorithme

Input : Graph real and results of detection community method
Output : number integer
def H(labels) :
labels$_u nique$ = $np.unique(labels) num_l abels_u nique$ = $len(labels_u nique) data_l en$ = $len(labels) Nprob$ = $[None] * num_l abels_u nique$
$fori, class_i nenumerate(labels_u nique)$: $count$ = $(labels == class_) .sum() prob[i]$ = $count/data_l enH$ = $-1 * sum(prob * np.log2(prob))$
$returnH$
$defI(Y, C)$:
$sum_H = 0 C_u nique = np.unique(C)$
$forcluster_n ameinC_u nique$: $index$ = $np.where(C == cluster_n ame) cluster_d ata$ = $Y[index] prob_c luster$ =

$(C == cluster_name).sum()/len(C)h = prob_cluster * H(cluster_data)sum_H = sum_H + hi = H(Y) - sum_H$
$returni$
$defNMI(Y, C) :$
$nmi = (2 * I(Y, C))/(H(Y) + H(C))$
$returnnmi$

2.3.4 La Modularité

Cette méthode consistent á optimiser une quantité appelée modularité qui représente, pour une partition donnée, le nombre d'arètes supplémentaires á l'intérieur des groupes par rapport la situation ou les arètes sont réparties aléatoirement en conservant les degrés de chaque point. Plus la modularité est grande, plus la structure du graphe est organisée, par opposition á un graphe aléatoire. [24]
L'expression générale de la modularité, qui consiste en une somme sur les couples de points i et j du graphe, chacun appartenant á un bloc $C_i et C_j est$:

$$Q = \frac{1}{2m} + \sum_i j(A_i j - \frac{k_i k_j}{2m})\sigma(C_i, C_j) \qquad (2.15)$$

Optimiser la modularité directement sur toutes les partitions possibles du graphe est difficile computationellement ; pour cette raison, l'heuristique la plus utilisée est celle dite de Louvain Cette dernière consiste en le processus suivant :

— On part de la partition du graphe en singletons.

— Pour chaque point, on considère si le mettre dans la communauté d'un de ses voisins augmente la modularité ; si oui, on

le met dans celle du voisin qui maximise la modularité. On passe ensuite au point suivant dans une liste arbitraire.

— Une fois la liste parcourue, on considère s'il existe un changement de ce type qui augmente la modularité. Si oui, on recommence.

Les avantages de cette approche (nommément, son efficacité sur de très grands réseaux, la possibilité de réeffectuer un clustering du graphe induit par les communautés afin d'avoir un clustering hiérarchique á plusieurs niveaux (figure 2.3), et son succès sur les réseaux á ground truth de test habituels) en ont fait l'algorithme standard de recherche de communautés.[25]

Cependant, la modularité souffre d'un problème de résolu-

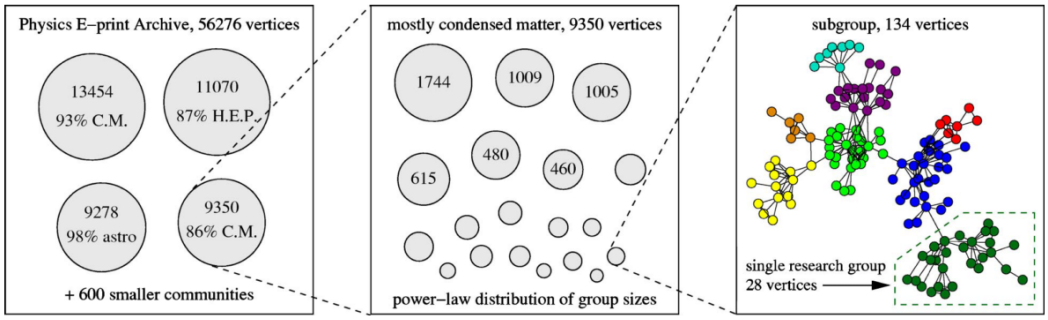

FIGURE 2.5 – Clustering hiérarchique d'un réseau de collaboration physiciens

tion dú au choix de P : dans les cas oú celui-ci est uniformément petit devant 1, par exemple si pour la plupart des points i du graphe ki « m dans le modèle de configuration, la modularité perd en précision et est parfois plus haute pour des associations de communautés qui sont contre-productives.

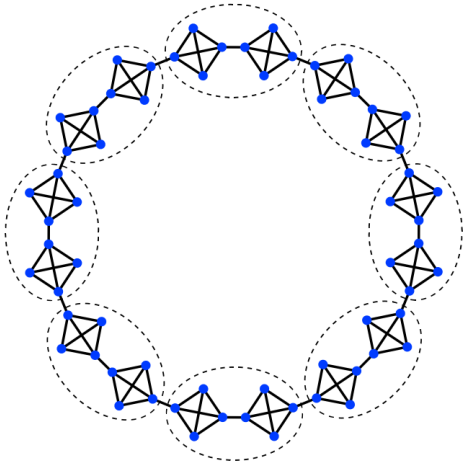

FIGURE 2.6 – Clustering hiérarchique d'un réseau de collaboration physiciens

Algorithme de Modularité

Input : Graph real and list of detection community method
Output : number integer
def d(G,u) :
ml=0 for i in l : ml+=d(G,i) m=float(ml)
return m
def inter(a,b) :
da=0 for i in range(len(a)) : if a[i] in b : da=da+1 return da
def Lst(node,l,G)
gg=list(G.neighbors(node)) s=inter(gg,l) return s
def Ls(l,G) :
dw=0 for i in l : dw+=Lst(i,l,G) w=float(dw) return w
def modularty(s, G) :
if G.is$_d$*irected*() :

$raiseTypeError("BadGtype, useonlynondirectedG")$
$m = len(G.edges)$
$m = float(m)$
$if m == 0.0 :$
$raiseValueError("AGwithoutlinkhasanundefinedmodularity")$
$q = 0.0$
$foriinrange(len(s)) :$
$f = s[i]$
$q = ((Ls(f, G))/(2.0 * m)) - (ds(G, f)/(2.0 * m)) * *2$
$q+ = q$
$returnq$

2.4 Comparaison entre les mesures :

Dans la littérature de la détection de communautés, les mesures les plus utilisées sont les suivantes :

— la modularité

— le F-mesure

— l'indice de Rand

— le NMI

Ce sont les mesures que nous utiliserons car elles mettent en oeuvre les principales caractéristiques des communautés, á savoir densité, nombre de liens sortants et graphe aléatoire.
De plus, elles permettent d'établir des études comparatives dans la mesure oú il s'agit des mesures les plus utilisées dans la littérature de la détection de communautés.

Mesures	Reference	caracteristique	Supervise/ non supervise
La modularite	Newman et Girvan (2004)	Maximise la densite des communautés a travers le modele nul	non
NMI	Ana et Jain (2003)	Quantite mesurant la dependance statistique entre deux partitions	oui
L'indice de Rand	Rand (1971)	Comptage sur les paires d'objets pareillement classees	oui
F-mesure	Van Rijsbergen (1979)	Combine la precision et le rappel et leur moyenne harmonique	oui

TABLE 2.4 – comparaion entre les populaire mesure

2.5 Conclusion

Dans ce chapitre, nous avons présenté les expérimentations que nous avons mené afin d'évaluer les méthode de détection de communauté , ensuite nous détaillons troi resau reél et une artificiel ,puis la présentation de deux mesures qui sont le F-mesures et L'indice de Rand bravement avec essentiel information et nous avons présenté deux mesure essentiel dans notre travail la modularité et le NMI qui on va faire une comparaison entre eux dans la suite chapitre.

Chapitre 3

Outils et Méthodologie

:

3.1 Introduction

Dans ce chapitre, nous allons montrer la méthodologie de notre travaille et on montrant les environnements de développement. Nous présentons en premier lieu l'environnement de développement matériel. Ensuite, nous allons définir les logiciels et les outils utilisées pour implémenter notre application et comment les installer. Nous terminerons ce chapitre par une discussion sur les critè qui nous avons basé pour faire les comparison souhaitée dans notre etude.

3.2 Les outils de développement

3.2.1 Les outils materiels

Toutes nos installations et nos tests seront réalisés sur une machine physique (un micro-ordinateur) dont la configuration est la suivantes :

— Processeur : Intel(R) CoreTM i5-5200U CPU @ 2.20 GHz

— Mémoire Installée(RAM) : 4 Go

— Type de système : système d'exploitation windows8 64 bits

Notre application a été développée sous un système d'exploitation Windows 8 de 64 bits où on a utilisé le langage de programmation python 3.8 sous Anaconda.

3.2.2 Les outils logiciels

3.2.2.1 Langage de programmation

Le langage utilisé pour le codage de notre application est principalement python 3.8.

Python est un langage de programmation interprété, de haut niveau et polyvalent. Crée par Guido van Rossum et publiée pour la première fois en 1991, la philosophie de conception de Python met l'accent sur la lisibilité du code avec son utilisation notable d'espaces blancs importants. Ses constructions de langage et son approche orientée objet visent á aider les programmeurs á écrire un code clair et logique pour les projets á petite et grande échelle. Python est typé dynamiquement. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation structurée (en particulier, procédurale), orientée objet et fonctionnelle. Python est souvent décrit comme un langage piles incluses en raison de sa bibliothèque standard complète [python – Dave Kuhlman]. L'installation de python est gratuite et facile, il suffit de le télécharger dans le site : https ://www.python.org/downloads/

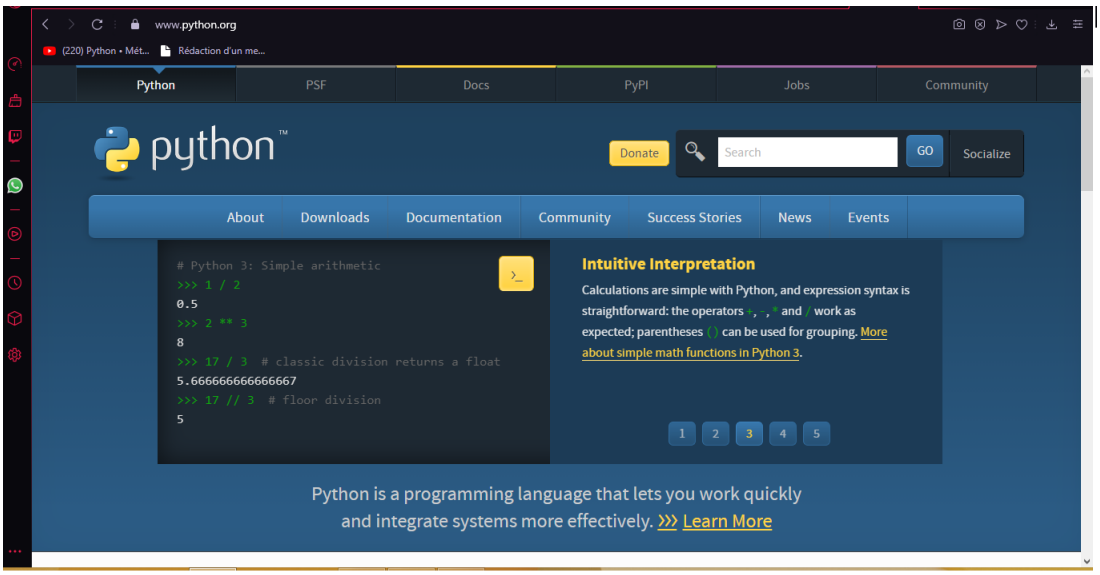

FIGURE 3.1 – Le site d'installation de python

3.2.2.2 Plateforme et IDE

Après l'installation de Python, on passe á l'installation de la plateforme avec tous les logiciels qu'on a besoin, pour ce but on a choisir Anaconda.
est l'une des nombreuses plates-formes open source qui facilitent l'utilisation de langages de programmation open source (R, Python) pour le traitement de données á grande échelle, l'analyse prédictive et le calcul scientifique. La communauté de recherche environnementale peut choisir d'adapter l'utilisation des langages de programmation R ou Python pour analyser les problèmes de science des données sur la plate-forme Anaconda. Téléchargez et installez la dernière version appropriée de la plate-forme Anaconda basée sur le système d'exploitation de l'ordinateur de l'utilisateur et la dernière version de Python á partir du site Web d'Anaconda : https ://www.anaconda.com/products/individual

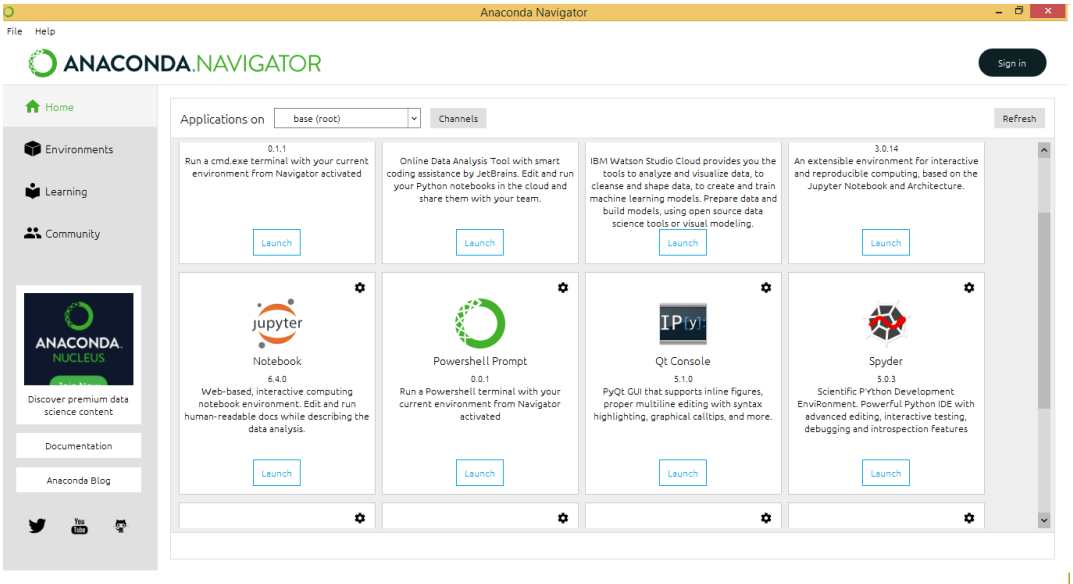

FIGURE 3.2 – Anaconda Navigator.

est un IDE multiplateforme open source pour la science des données qui fournit un environnement de développement scientifique Python qui facilite les fonctionnalités avancées d'édition, de test interactif, de débogage et d'introspection sans utiliser les commandes de ligne de commande qui se trouve dans la barre des taches [11].

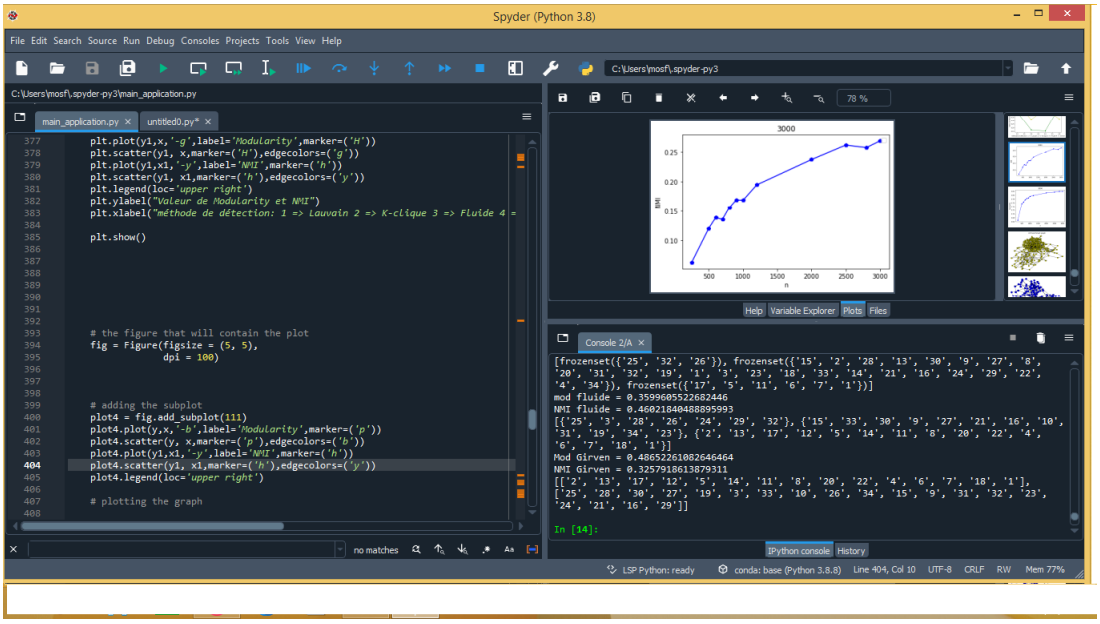

FIGURE 3.3 – l'IDE Spyder.

3.2.2.3 Les bibliothèques

Python est un langage de programmation tres riche avec ses bibliothèques ; dans notre application nous l'avons utilise et nous avons également profité de ses fonctions prédéfinies. Nous citons quelques-unes des bibliotheques utilisees dans ce travail.

— networkx : est un package de langage Python pour l'exploration et l'analyse des réseaux.
— community : pour générée les algorithmes de detection des communautés.
— sklearn.metrics.cluster : pour les machines learning
— tkinter : Pour les interfaces graphiques.
— matplotlib : Pour le tracage des graphes.

3.3 Architecture de notre application

Notre application compose de quatre parties principales :

1. Génération des données réels et artificiels
2. Application des algorithme de détection des communautés sur ces graphes
3. Évaluation les résultats par NMI et Mdularité
4. Comparaison entre les methodes d'évaluation.

L'architecture de notre application est présentée dans le schéma ci-dessus.

3.3.1 Génération des données

dans notre ètude , nous avanos travaillé sur deux types de réseaux réels et artificiels :

— Pour les réseaux réels : Karté club, dauphin de lusseau et les livres politiques, (nous les discuterons en dètail dans le chapite 2).
— Pour les réseaux artificiells, nous avons choisir le benchmark LFR(discuté en détaille dans le chapitre 02).

Le résultats est un graphe qui est l'entré de la phase suivante.

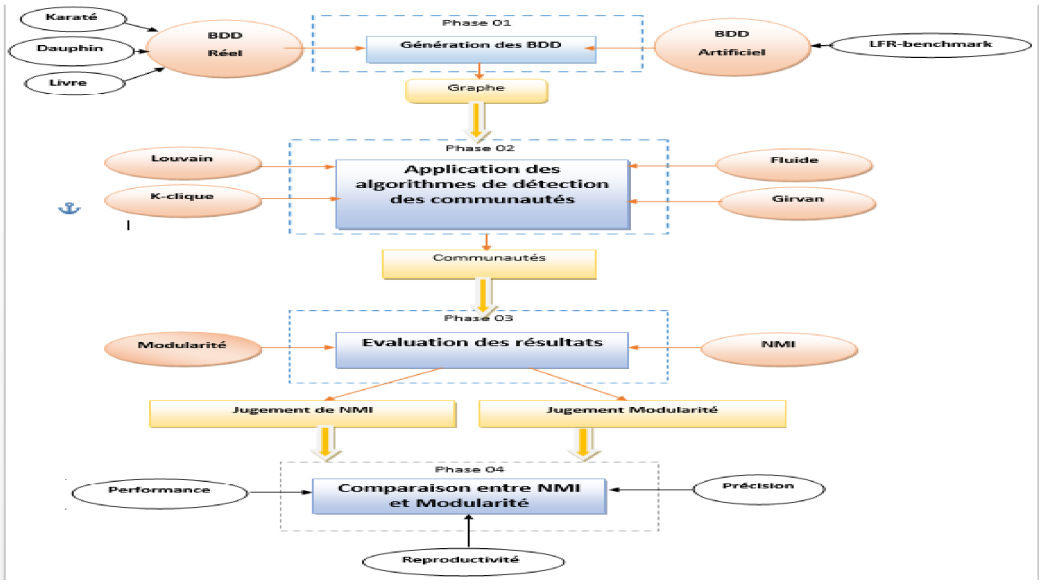

FIGURE 3.4 – Schéma d'architecture générale du système.

3.3.2 Les algorithmes de détection des communautés

Pour détecter les communautés, nous avons appliquons les quatre algorithmes les plus populaires : Louvain ,K-clique, Fluide et Girven (aussi nous les avons dèjà expliquè dans la première chapitre), on a appliquè les mèthodes sur les diffèrates rèseaux pour trouver les communautés résultantes qui sont le sujets d'évaluation de la phase suivante.

3.3.3 Les métoudes d'évaluation

Pour jujer la qualité des communautés trouvé, nous avons appliquées deux mésures d'évaluations : NMI et Modularité(nous les avons dèjà expliquè dans le deuxième chapitre), on a les appliquè ces sur les dètections pour évaluer les résultantes qui sont le fruit q'on á besoin de lui de le début pour faire la comparaison.

3.3.4 Comparaison entre NMI et Modularité

Cette phase est ce qui nous intéresse plus dans le cadre de ce travail, mais on a obligé de faire tous les étapes précedantes pour arriver ici, pour faire une comparaison efficace et determiné nous avons utilisée trois critères principales qui sont : la précision , la perfermance et la reproductivité

3.3.4.1 Prècision

La prècision est la qualitè globale d'une mesure ou d'un instrument, elle dépend beucoup plus aux nombre du noeuds.

3.3.4.2 Perfarmance

La performance d'un algorithme est définie théoriquement par sa complexité, et en réalité par le temps qu'il prend pour retourner un résultat.

3.3.4.3 Reproductivité

pour la reproductivité nous avons applique le NMI et la modularité sur la meme resultat (communaute detecte) du reseau soit reel apré lapplication des methodes de detection ou bien artificiel qui est crée les communautés elle meme et voir les resultats.
dans le prochaine chapitre nous présentons les résultat obtenu des application de trois critere et en fin dégagé quelle est la meilleure mesure .

3.4 Conclusion

Nous avons présenté dans ce chapitre une vue générale sur notre environnement de travail. Nous avons commencé par la présentation des outils que nous avons utilisé du début jusquá la fin de nos expérimentations et notre environnement de test (matériel et logiciel).ensuite nous avons expliqué notre architecture d'application de début á l'acces au comparaison requise ,ainsi présenté á quelle base notre etude construit , apré faire les essayes et l'application des méthode existe déjá, nous dégageons ce critère. Nous estimons que ces informations sont trés utiles pour tout chercheur qui veut reproduire ou prendre nos résultat comme référence. Le dernier chapitre sera réservé pour la présentation et l'interprétation des résultats.

Chapitre 4

RÉSULTATS Et INTERPRÉTATION

:

4.1 Introduction

Dans ce chapitre, nous allons montrer l'implémentation de notre système .nous présentons les résultat obtenu apré les tests et les différantes figures qui represente notre étude comparative entre les mesure utilisée pour evalue les méthode de detection Nous terminerons ce chapitre par une discussion ces résultats on comparant les résultats des deux méthodes NMI et modularité a base les critère montionée dans le chapitre précédant et dire brevement lequel avantagée et moins des inconveninents a l'autre et en fin donnée notre humble avis apres cette étude realisée .

4.2 Présentation du système

Dans cette section on va présenter quelque module de notre application. ou il y'a deux espace pour choisir les base de donnée deja existe et importer pour va traiter ou l'utilisateur peut crée reseau artificiel par le LFR et exécuter le .
La figure 4.1 nous avons montre comment l'utilisateur peut saisir les données á traiter et choisir entre exécuter le ou bien crée artificielement .

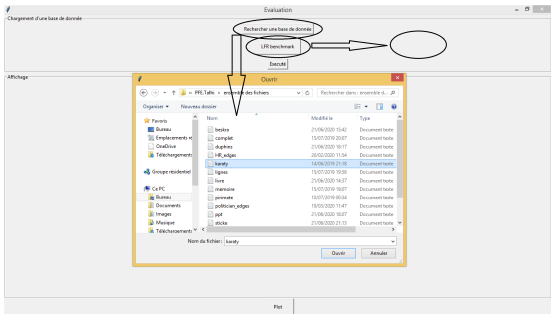

FIGURE 4.1 – Fenetre Réserveé Pour L'utilisateur

4.2.1 intégration de LFR

nous avons integré le réseau artificiel LFR-benchmark par la fonction suivant :
def genrate-lfr-graph(size=250) :
params = "n" :n, "tau1" :2, "tau2" :1.1, "mu" :0.1,
"min-degree" :20, "max$_d egree$" : 50

G = LFR-benchmark-graph(params["n"], params["tau1"],
params["tau2"], params["mu"],
min-degree=params["min-degree"],
max-degree=params["max-degree"],
max-iters=5000, seed = 10,)
communities = frozenset(G.nodes[v]['community']) for v in G
communities=list(communities)
partition = community-louvain.best-partition(G)
c1,c2=clusters-classes-from-frozensets(communities)
classes = np.array(list(partition.values()))
La figure ci-dessus nous montre la graphe obtenu après choisir la création artificiel LFR-benchmark.

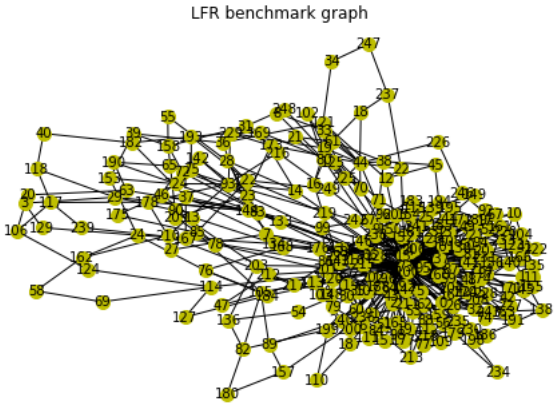

FIGURE 4.2 – LFR-benchmark graphe

4.2.2 intégration de base de donnée réel

pour faire nous tests d'une autres facon nous utilisons des base de donnée réel avec la figure ci-dessus nous montre la graphe obtenu après choisir l'importation d'une base triangle déjá existe .

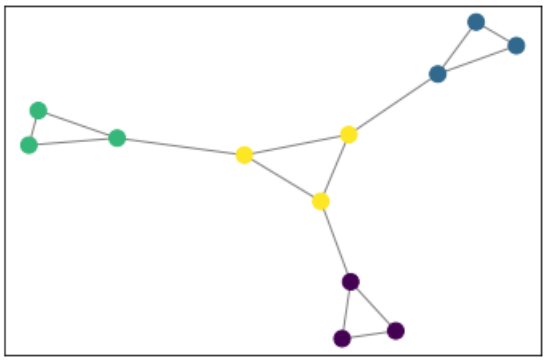

FIGURE 4.3 – Triangle Graph

4.2.3 intégration des méthodes de détection

comme nous l'avons mentionné précédement dans le premier chapitre les quatre méthodes , nous allons présente le résultat de chaque méthode sur le reseau réel Karaté comme exemple .
Louvain :
'2' : 0, '1' : 0, '3' : 0, '4' : 0, '5' : 2, '6' : 2, '7' : 2, '8' : 0, '9' : 3, '10' : 0, '11' : 2, '12' : 0, '13' : 0, '14' : 0, '17' : 2, '18' : 0, '20' : 0, '22' : 0, '26' : 1, '24' : 3, '25' : 1, '28' : 3, '29' : 1, '30' : 3, '27' : 3, '31' : 3, '32' : 1, '33' : 3, '15' : 3, '16' : 3, '19' : 3, '21' : 3, '23' : 3, '34' : 3
K-clique :
frozenset('26', '32', '25'), frozenset('32', '29', '3', '27', '4', '24', '33', '20', '15', '8', '16', '21', '19', '9', '23', '14', '13', '18', '28', '31', '34', '2', '1', '30', '22'), frozenset('7', '5', '11', '17', '6', '1')

Fluide :
('10', '26', '3', '28', '31', '25', '9'), ('32', '29', '27', '33', '24', '34', '16', '15', '21', '19', '30', '23'), ('12', '13', '7', '18', '4', '11', '5', '17', '22', '6', '8', '20', '2', '1', '14')

Girven :
('12', '13', '7', '18', '4', '5', '11', '17', '14', '6', '8', '20', '2', '1', '22'),('26', '32', '27', '33', '24', '21', '19', '25', '15', '10', '29', '3', '16', '9', '28', '31', '34', '23', '30')
apré l'étape précédent nous passerons aux application de notre mesures.

Taille de resaux	NMI obtenu
250	0.06254293209886343
500	0.1202227034360495
600	0.13811749390564945
700	0.13554260875803928
800	0.16152141557541408
900	0.16887783089431305
1000	0.16841895492848735
1200	0.1944839573488305
2000	0.23725542580715894
2500	0.2616547478528476
2800	0.25734699349636364
3000	0.2665607940753393

TABLE 4.1 – Resultat de NMI sur LFR-benchmark

4.3 Tests

Dans cette partie, on va discuter les tests que nous avons faits avant de passer au version finale qui est la comparaison entre la Modularité et le NMI .

4.3.1 Test sur les réseaux artificiel

Au début, Nous avons fait des tests sur une base de données artificielle de differante taille avec nombre des noeuds dans l'intervale [250, 500, 600, 700, 800, 900, 1000, 1200, 2000, 2500, 2800, 3000] ,
Le résultat l'application de Nmi et modularité sur le graphe créez est représenté dans le tableaux 4.1 .

Taille de resaux	NMI obtenu
250	0.6671179448341837
500	0.7548981841617095
600	0.7636423804228172
700	0.7752521874279785
800	0.7820184521258323
900	0.7854496778466563
1000	0.7930479778189816
1200	0.7975923714277313
2000	0.8128003859019381
2500	0.8156246298039609
2800	0.8190277512892269
3000	0.819118130826028

TABLE 4.2 – Resultat de Modularité sur LFR-benchmark

La figure 4.4 contient un graphe qui montre le changement de NMI en terme de taille de resaux artificiel LFR .

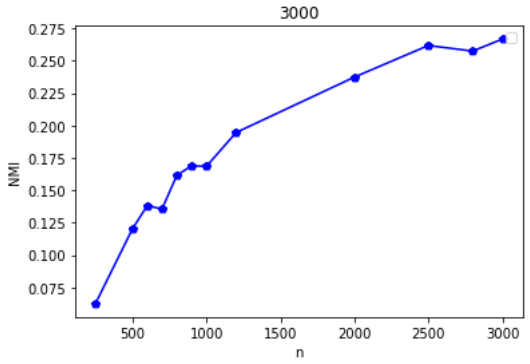

FIGURE 4.4 – NMI en terme différantes sizes de LFR

deuxiement on fait l'appliquation de modularité sur les meme données précédentes et presente les resultat dans le tableau 4.2 :

la mème chose pour la modularité La figure 4.5 contient un graphe qui montre le changement de Modularité en terme de taille de resaux artificiel LFR

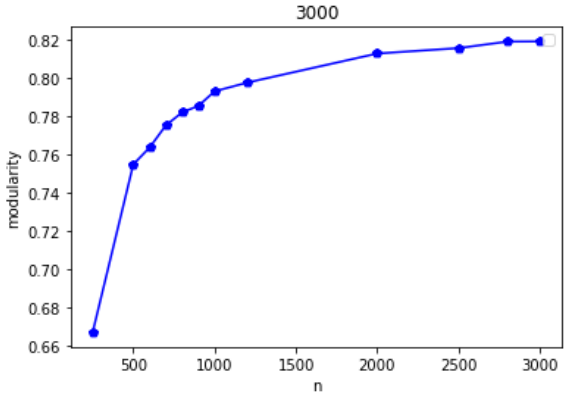

FIGURE 4.5 – Modularité en terme différantes sizes de LFR

apre les deux tests fait nous allons faire une comparaison entre les deux methode , le graphe suivant montre les differante existe.

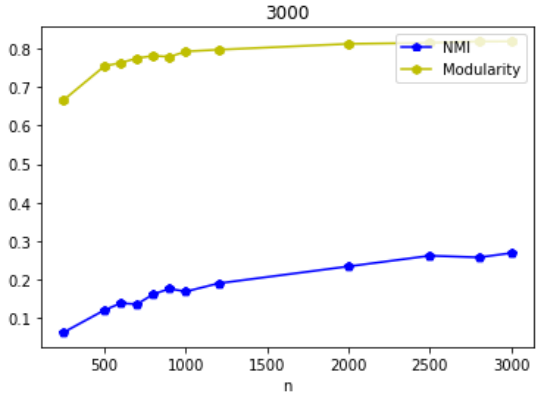

FIGURE 4.6 – NMI vs Modularité

par cette comparaison, nous montrons que la modularité plus fort

Méthode	NMI obtenu	Modularity obtenu
Louvain	0.5449398643703183	0.4197896120973044
K-clique	0.3172648424908407	0.0650887573964497
Fluide	0.4605960216566614	0.34582511505588426
Girven	0.3257918613879311	0.48652261082646464

TABLE 4.3 – Resultat de Modularité et NMI avec les méthode choisi

que le NMI avec les plus grand réseau de taille 3000 noeuds, apré terminé cette comparaison passons a les tests et comparaison avec les méthode de détection dans les réseau réel de grande taille et de petit taille .

4.3.2 Test sur les réseaux Réel

nous commencons avec le réseau de karaty club qui est les caractiristique de :
Number of nodes : 34,
Number of edges : 78
Average degree : 4.5882 [12] .
nous applicons les methode de detection suivant : louvain , k-clique , fluide et girven newman , les resultat obtenus presente dans le tableau 4.3 : apre trouvé ces resultat nous compairent entre les deux mesure et affiche dans le graphe suivant :

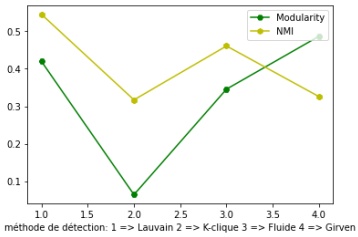

FIGURE 4.7 – NMI vs Modularité avec les quatre methode applique au karaty club

Méthode	NMI obtenu	Modularity obtenu
Louvain	06049466717635833	0.8711485924380034
K-clique	0.7000314368593882	0.00034744837249727536
Fluide	0.22452574457752153	0.45803065021431677
Girven	0.023714626766648567	0.01729105603212118

TABLE 4.4 – NMI et Modularité de méthode applique sur livre

apre ces test nous avons degagé que le NMI travail mieux que la modularité avec petit réseau sauf avec la méthode girven .
par la suite nous allons augmenter le volume de reseau reel avec l'utilisation de reseau livre politique qui se compose de :
Number of nodes : 3892
Number of edges : 17262
Average degree : 8.8705 [8]
les résultat obtenus présenté dans le tableau 4.4 : la meme chose que le cas précedent , nous présntons les résultat pour meilleur vesion de différances remarquées dans le graphe de comparaison suivant suite des resultat obtenus dans la figure4.9 montrons que

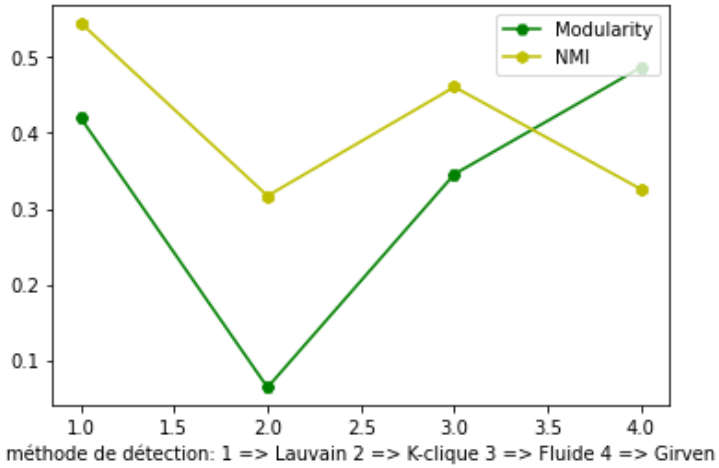

FIGURE 4.8 – NMI vs Modularité avec les quatre méthode applique au livre

Méthode	NMI obtenu	Modularity obtenu
Louvain	0.52459307227155	0.5165438231052716
K-clique	0.4290477201293705	0.2642204774875821
Fluide	0.412991634521872	0.26724483255888476
Girven	0.26856901009486356	0.41135734072022156

TABLE 4.5 – NMI et Modularité de méthode applique sur dauphin

la modularité evalue bien que le NMI dans la plus part méthode sauf dans k-clique , et de cela ont confirmé ces résultats les résultat precedents extrais de l'application des deux mesure sur le reseau artificiel de grand taille, la modularité et fonctionne mieux que NMI dans les reseau complexe.
troisièmement nous presentons les tests appliquées au réseau de dauphin , qui consitent en les données suivantes :
Number of nodes : 62
Number of edges : 158
Average degree : 5.0968 . [1]
le resultat obtenu est dans le tableau 4.5 : pour plus de précisions et une bonne analuse plus efficace nous présentent dans un graphe suivant : les resultat sur un petit reseau qui sont le dauphin , le NMI montrer la superiorite de la modularité et confeme les resultat precedent sur karaty club sauf au cas de girven .

4.4 Avantages et Inconvénients

En plus de la facilité de l'implémentation, nous avons comparé entre deux mesure d'evalution des métohdes de détection des communautés populaires et LFR-benchmark méthode permis de crée reseau artificiel avec des communauté sont crée automatiquement.

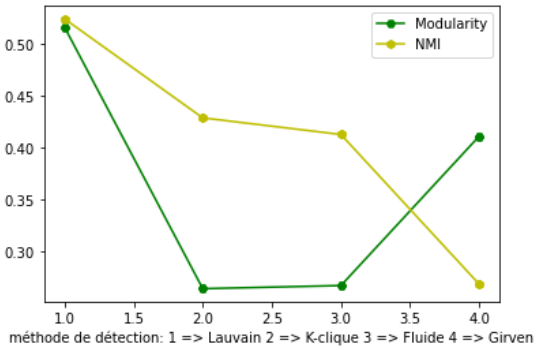

FIGURE 4.9 – NMI vs Modularité avec les quatre méthode applique au Dauphin

Les résultats de comparaison dans notre étude très riche information et des test faite au plusieurs cas.
Concernant le NMI sont restent stable dans reseau artificiel tendis que aux réseau réel elle change sa valeur a chaque execution, d'autre part le NMI besion d'une résultat préexistants pour faire le comparaison , d'un autre coté le NMI plus faible dans grand graphe et plus efficace dans le petit .
concernant la modularité restent stable dans les deux cas soit reéel ou artificiel , la modularité n'a pas besoin des résultat préexistants , d'autre part et dans la grand graphe la modularité fonctionner plus efficacement que dans la petit graphe . En ce qui concerne le temps d'exécution, notre algorithme a fait preuve de sa performance par rapport aux d'autres algorithme de détection de communautés.

4.5 Conclusion

Dans ce chapitre, nous avons tout d'abord présenté les outils que nous avons utilisés pour faire etude comparative . Ensuite, nous avons exposé les résultats des expérimentations que nous avons réalisées aprés l'analyse des test que nous avons fait avant d'obtenir les meilleurs mesure a travere le dégagement des avantage et des inconvènients pour les deux mesure .

Conclusion Generale

Ce mémoire avait comme objectif etude comparative des méthodes d'évalution de la qualité des structure communautaires qui on choisi deux mesure qui sont la modularité et le NMI. Pour cela, le premier chapitre a passé en revue les approches de l'état de l'art pour la détection des communautés et Les mesures d'évaluation. Nous commencant l'état de l'art avec une représentation des communautés et quelques définitions essentielles dans la détection des communautés. Ensuite nous avons abordé les différentes catégories des algorithmes de la détection de communauté en décrivant les travaux les plus récents dans le domaine. Nous avons présenté quatre mesure d'évalution parmi eux, nous en avons choisi deux et les avons expliqués en détail . Notre etude est simple et facile á comprendre, ou nous avons mene une serie des tests pour montrer les différances dans l'application de chacune des deux méthodes et les points de similitude avec l'extraction des forces et des faiblesses pour chacune d'entre elles, D'après les tests que nous avons réalisés sur des réseaux artificiels et d'autres réseaux réels et de lapplication de les mesures sur les quatre methode de la detection de communaute , notre etude montre que parmi les deux mesure on peut dire prudement que la modularité et avantagée par rapport le NMI car la dernière mesure necessite toujours des resultats réel prealables pour faire la comparaison et appliquer sa propre

équation , autre inconvénient dans le réseau artificiel LFR le NMI calcule a chaque fois execution différent résultat
Nos buts de future sont :

— faire d'hybridation entre les deux methodes modularité et NMI pour mieux résultat d'évaluation moin de des inconvenients et renforcement de la mesure faible .

Bibliographie

[1] Jean-Philippe Attal and Maria Malek. Un nouvel algorithme de propagation de labels avec barrages. In *Journées réseaux sociaux et IA*, 2015.

[2] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439) :509–512, 1999.

[3] Niko Beerenwinkel, Barbara Schmidt, Hauke Walter, Rolf Kaiser, Thomas Lengauer, Daniel Hoffmann, Klaus Korn, and Joachim Selbig. Diversity and complexity of hiv-1 drug resistance : a bioinformatics approach to predicting phenotype from genotype. *Proceedings of the National Academy of Sciences*, 99(12) :8271–8276, 2002.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics : theory and experiment*, 2008(10) :P10008, 2008.

[5] Luís Cavique, Armando B Mendes, and Jorge MA Santos. An algorithm to discover the k-clique cover in networks. In *Portuguese Conference on Artificial Intelligence*, pages 363–373. Springer, 2009.

[6] Rémy Cazabet. *Détection de communautés dynamiques dans des réseaux temporels*. PhD thesis, Université Paul Sabatier-Toulouse III, 2013.

[7] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5) :75–174, 2010.

[8] Olivier Gach. *Algorithmes mémétiques de détection de communautés dans les réseaux complexes : techniques palliatives de la limite de résolution.* PhD thesis, Université du Maine, 2013.

[9] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12) :7821–7826, 2002.

[10] Bouzgarne Itri, Youssfi Mohamed, Qbadou Mohammed, and Bouattane Omar. Performance comparative study of machine learning algorithms for automobile insurance fraud detection. In *2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS)*, pages 1–4. IEEE, 2019.

[11] Akhil Kadiyala and Ashok Kumar. Applications of python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy*, 36(6) :1580–1586, 2017.

[12] Rushed Kanawati. Détection de communautés dans les grands graphes d'interactions (multiplexes) : état de l'art. 2013.

[13] Robert V Kozinets. E-tribalized marketing ? : The strategic implications of virtual communities of consumption. *European management journal*, 17(3) :252–264, 1999.

[14] Valdis Krebs. Proxy networks. analyzing one network to reveal another. *Bulletin de méthodologie sociologique. Bulletin of sociological methodology*, (79) :61–70, 2003.

[15] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4) :046110, 2008.

[16] Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv :1110.2515*, 2011.

[17] Mark EJ Newman. Detecting community structure in networks. *The European physical journal B*, 38(2) :321–330, 2004.

[18] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113, 2004.

[19] Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities : A competitive, scalable and diverse community detection algorithm. In *International Conference on Complex Networks and their Applications*, pages 229–240. Springer, 2017.

[20] David MW Powers. Evaluation : from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv :2010.16061*, 2020.

[21] Caetano Traina Jr, Agma J Traina, and Christos Faloutsos. Distance exponent : A new concept for selectivity estimation in metric trees. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1999.

[22] Stanley Wasserman, Katherine Faust, et al. Social network analysis : Methods and applications. 1994.

[23] Robin J Wilson. *Introduction to graph theory*. Pearson Education India, 1979.

[24] Zhao Yang, René Algesheimer, and Claudio J Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6(1) :1–18, 2016.

[25] Jeongah Yoon, Anselm Blumer, and Kyongbum Lee. An algorithm for modularity analysis of directed and weighted biological networks based on edge-betweenness centrality. *Bioinformatics*, 22(24) :3106–3108, 2006.

[26] Xuemei You, Yinghong Ma, and Zhiyuan Liu. A three-stage algorithm on community detection in social networks. *Knowledge-Based Systems*, 187 :104822, 2020.

[27] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4) :452–473, 1977.