الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Détection des communautés par une méthode d'apprentissage automatique

Encadré Par :

Mme. Louafi Wafa

Présenté par :

Boucerredj Nadjoua

Juin 2022

Dédicace

“

Je dédie ce modeste travail A celle qui est dans mon cœur, à celle qui a veillée pour notre confort et sacrifiée beaucoup pour notre réussite,
Ma chère mère, (que dieu me la garde),

A celui qui m'a toujours appris comment réfléchir avant d'agir, à celui qui m'a soutenu tout au long de ma vie scolaire, à celui qui n'a jamais épargner un effort pour mon bien,
Mon cher père (que dieu me le garde),

A mes chères sœurs : Ouahida, Salima,

À toute personne ayant contribué à son aboutissement, Et à tous ceux qui me Connaissent…,

À tous ceux qui me sont chers, à vous tous

Merci.

”

- Nadjoua

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

Je tiens à remercier tout particulièrement ma directrice de mémoire, mon encadrante **Mme. Louafi Wafa**, pour l'aide compétente qu'elle m'a apportée, pour sa patience et son encouragement. Son œil critique m'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Je tiens aussi à adresser mes plus sincères remerciements à **Mr. Ellagoune Salah**, Mr le directeur de l'université 8 mai 1945, pour sa patience, son aide, sa confiance et son soutien inestimable.

J'adresse mes sincères remerciements aux membres de jury, **Mr. GOUASMI NOUREDDINE** et **Mme. ABDELMOUMENE HIBA**, pour leurs efforts de lecture et d'évaluation de ce mémoire.

Aussi, je ne pourrais laisser passer cette occasion sans saluer chaleureusement tous les enseignants du département de l'informatique qui ont veillé à nous accorder une formation de qualité.

Je remercie également **Mr Hallaci Samir**, pour ses suggestions ainsi que ses précieux conseils. Je suis heureuse de lui exprimer ici ma respectueuse reconnaissance.

Pour finir, je souhaite remercier toute personne ayant contribué de prés ou de loin à la réalisation de ce travail.

Résumé

La détection des communautés dans les réseaux joue un rôle essentiel dans la compréhension de leurs structures.

L'application des méthodes d'apprentissage automatique aux tache de détection de communautés dans les réseaux complexes a suscité une attention soutenue ces dernières années, nous proposons dans ce mémoire une nouvelle approche de détection de communautés basée sur k-means avec l'initialisation des noeuds centraux selon leurs densités et leurs degrés, le choix du nombre de communauté k se fait selon la meilleure modularité.

Notre approche est efficace, simple et facile à implémenter. Nous avons comparé notre algorithme avec quelques algorithmes de pointe sur des réseaux synthétiques et des réseaux réels, avec la mesure d'évaluation : La modularité Q et nous obtenons des résultats acceptales.

Mots clés : Détection des communautés, Apprentissage Automatique, densité, modularité, K-Means.

Abstract

Community detection in networks plays an essential role in understanding their structures.

The application of machine learning methods to community detection tasks in complex networks has attracted sustained attention in recent years, we propose in this thesis a new community detection approach based on k-means with the initialization of central nodes according to their densities and their degrees, the choice of the number of community k is made according to the best modularity.

Our approach is efficient, simple and easy to implement. We compared our algorithm with some state-of-the-art algorithms on synthetic networks and real networks, with the evaluation measure : Modularity Q, and we obtain very acceptable results.

Keywords : Community detection, Machine Learning, density, modularity, K-Means.

Table des matières

Table des figures

Liste des tableaux

Liste des Abréviations

ARI	*Adjusted Rand Index*
NMI	*Normalized Mutual Information*
LFR benchmark	*Lancichinetti Fortunato Radicchi benchmark*
RI	*Rand Index*
Q	*Modularity*
PCA	*principal components analysis*
API	*Application Programming Interface*
KNN	*K-Nearest Neighbors*
ML	*Machine Learning*
SL	*Supervised Learning*
UL	*Unsupervised Learning*
NB	*Naïve Bayes*
LR	*Logistic regression*
SVM	*Support Vector Machine*
ANN	*Artificial Neural Network*
SLR	*Simple Linear Regression*
DTR	*Decision Tree Regression*
DTFR	*Decision Tree Forest Regression*
DR	*Dimentiality Reduction*
Q-learning	*Quality-learning*
TDL	*Temporal Difference Learning*
DBSCAN	*Density Based Spatial Clustering of Applications with Noise*
HC	*Hierarchical clustering*

Introduction générale

Contexte

De nos jours, les réseaux sociaux apparaissent comme un nouveau moyen important de communication, ils permettent de représenter les interactions entre les différents individus d'un système, par exemple un échange des photos entre amis, des mails ou des SMS, entre différents groupes d'individus.
La modélisation de ces réseaux par des graphes facilite l'étude et la compréhension de leur structure ; où chaque acteur du réseau social est représenté par un sommet et chaque relation entre deux individus est représentée par une arête. Il est donc possible d'appliquer les concepts de la théorie des graphes aux réseaux sociaux.
Tout réseau social est caractérisé par l'existence des zones plus densément connectées que d'autres. Ces zones correspondent à des groupes de nœuds plus fortement connectés entre eux mais faiblement reliés aux autres. Ces zones sont appelées communautés, dont on peut les définir comme des ensembles de nœuds fortement liés entre eux, et plus faiblement liés avec le reste du réseau, et elles représentent des groupes sociaux (ex. familles, groupes d'amis, etc…).
La détection de communautés dans les réseaux sociaux est une tâche de regroupement des nœuds en plusieurs groupes d'individus ou clusters ayant des intérêts ou des activités communes en se basant sur les méthodes de théorie des graphes ou les algorithmes d'apprentissage automatique. Ces communautés ont souvent, d'une part, des affinités particulières, d'autre part, des caractéristiques similaires, ou encore, des centres d'intérêt commun.
A ce jour, un grand nombre d'algorithmes de détection des communautés pour les réseaux sociaux ont été proposés, notamment les algorithmes de regroupement hiérarchique [8], les algorithmes de propagation d'étiquettes [6], les algorithmes basés sur la densité [46], les algorithmes basés sur la marche aléatoire [119, 96], ou bien les algorithmes de regroupement k-means qui divisent les données en groupes (le nombre de groupes est prédéterminé) sur la base de fonctions d'erreur minimale [65]. l'algorithme de k means se caractérise par un regroupement rapide, une implémentation facile, et une classification efficace dans les ensembles de données à grande échelle. De plus, l'algorithme de clustering k-means présente une faible complexité temporelle par rapport aux méthodes de clustering basées sur la centralité et la similarité.
Pour évaluer la qualité des communautés détectées, nous utilisons la Modularité Q comme une mesure d'évaluation.

Objectifs

L'objectif principal de ce mémoire est de concevoir une approche de détection de communautés dans les réseaux sociaux, facile et efficace, en se basant sur l'algorithme de clustering K-means. L'approche proposée est évaluée sur différents types de réseaux et leur performance est comparée à celle d'autres algorithmes de détection de communautés.

Organisation du mémoire

Ce mémoire est organisé en quatre chapitres :

Le premier chapitre "**La Détection de communautés**" :
Présente des généralités sur la théorie de graphes, la détection des communautés, quelques algorithmes de détection de communautés, définition des réseaux sociaux, ainsi que les mesures d'évaluation.

Le deuxième chapitre "**l'apprentissage automatique** " :
Présente les différentes méthodes et approches existantes de l'apprentissage automatique qui pourraient intervenir dans la détection des communautés, ainsi que des travaux connexes sur la détection de communautés.

Le troisième chapitre "**Conception**" :
Est consacré à la conception de la méthode proposée et son implémentation.

Le quatrième chapitre "**Implémentation**" :
Est consacré à la présentation des outils de programmation utilisés en premier lieu, ensuite la présentation de notre système, puis on présente les résultats du déroulement des algorithmes existants sur les bases de données utilisées. Et enfin, on présente les résultats de déroulement de notre méthode proposée en la comparant avec des méthodes existantes.

Nous clôturons notre travail par une conclusion générale et des perspectives.

Chapitre 1

La Détection de communautés

1.1 Introduction

Beaucoup de systèmes complexes dans divers domaines comme la biologie, l'informatique, la linguistique, le commerce, etc., peuvent être représentés de manière abstraite par des réseaux. Une des caractéristiques communes que l'on retrouve dans de nombreux réseaux concerne l'existence de zones plus densément connectées que d'autres. Ces zones sont habituellement appelées communautés et correspondent intuitivement à des groupes de nœuds plus fortement connectés entre eux qu'avec les autres nœuds du réseau, ce qui signifie que la détection de communautés a pour rôle de classer les membres du réseau en groupes.

Pour faire la détection de communautés dans un réseau donné, nous avons besoin d'une représentation facile et pratique comme la représentation par graphe.

Ce chapitre tente de rapporter l'essentiel concernant la notion de communauté. Mais avant cela, il convient d'introduire au préalable des cas de modélisation par des graphes, ainsi que certaines notions et définitions utiles de la théorie des graphes.

1.2 Quelques notions de la théorie des graphes

La théorie des graphes a vu le jour lors de l'apparition de l'article du mathématicien Leonhard Euler en 1735 [29], qui a traité le fameux problème des sept ponts de Königsberg. Le problème se résume dans la recherche d'un chemin à partir d'un pont donné et en retour de ce même pont en passant une et une seule fois par les autres ponts de la ville. Ce parcours est appelé chemin Eulérien. En réalité, pour représenter mathématiquement un réseau nous faisons souvent référence à cette théorie, où cette représentation considère les entités du réseau comme étant des nœuds et les relations entre celles-ci comme des liens.

1.2.1 Définitions

- **Graphe :** Un graphe est un ensemble $G = (V, E)$ où V (vertex en anglais) désigne les sommets (nœuds) et E (edge en anglais) designe les arêtes qui relient les sommets. (Voir figure 1.1)

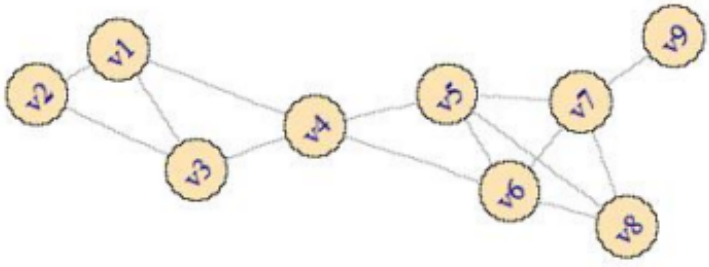

Fig. 1.1 : Représentation d'un graphe à 9 nœuds et 14 arêtes [72]

- **Noeud :** Un nœud (Voir figure 1.2), aussi appelé individu ou acteur est le sommet d'un graphe, il peut avoir des relations avec d'autres individus, Au niveau de la figure 1.2 A, B, C sont des nœuds (acteurs) [62].

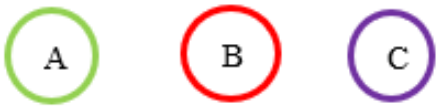

FIG. 1.2 : Représentation des noeuds

- **Lien :** Un lien (ou arête) est une connexion entre deux noeuds.
- **Arc :** Un arc est une arête orientée [7].
- **Nœud chevauchant ou non chevauchant :** Un nœud chevauchant est partagé par plus d'une communauté. Par exemple dans la Figure 1.3, le nœud 3 est chevauchant. Un nœud non chevauchant appartient à une seule communauté [72].

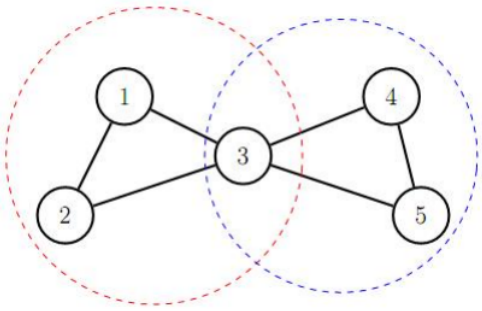

FIG. 1.3 : Représentation d'un noeud chevauchant

- **Degré d'un noeud :** Dans un graphe non-orienté, le degré d'un noeud (sommet) est le nombre d'arêtes incidentes à ce noeud, c'est-à-dire le nombre de noeuds auquel il est relié [62].
- **Taille d'un graphe :** C'est le nombre de lien qu'il contient [27].
- **Ordre d'un graphe :** C'est le nombre de sommets de ce graphe [27].
- **Graphe orienté :** Un graphe orienté est un ensemble de sommets et d'arêtes, chaque arête étant un couple de sommets ordonnés. Ainsi, la relation entre les sommets x et y est différente de celle entre y et x. (Voir figure 1.4)

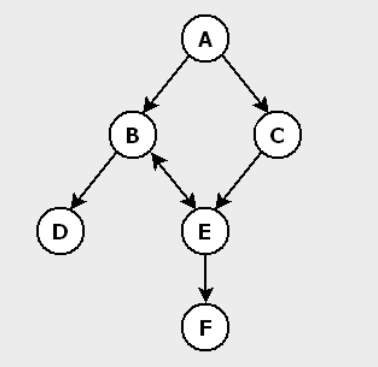

FIG. 1.4 : Exemple de graphe orienté

- **Boucle :** une boucle est une arête reliant un sommet à lui-même.
- **Chemin :** Un chemin entre deux noeuds (sommets) est la succession de noeuds et arêtes qui lient les deux noeuds ; la longueur de ce chemin sera le nombre de liens qu'il comporte.
- **Distance géodésique :** La distance géodésique δ_{ij} représente le nombre d'arêtes du plus court chemin reliant deux sommets (le minimum des longueurs de chemins entre ces deux sommets) [62].
- **Diamètre d'un graphe :** Le diamètre d'un graphe est la plus grande distance géodésique possible entre deux sommets dans le graphe [72].
- **Voisinage :** Deux noeuds liés dans un graphe sont dits voisins l'un à l'autre. Le voisinage Ni d'un nœud correspond à l'ensemble des nœuds qui lui sont directement connectés [72].
- **Graphe connexe :** un graphe est dit connexe s'il existe au moins un chemin entre chaque paire de nœuds. Il n'y a pas de nœuds isolés.
- **Densité d'un graphe :** La densité d'un graphe $G = (V, E)$ est une mesure de sa connectivité, elle indique la quantité de liens au sein d'un réseau ou graphe. Plus un graphe est dense, plus il est connecté [72, 61]. Pour mesurer la densité dans un graphe, on divise le nombre d'arêtes observées (le nombre total de connexions) par le nombre maximal d'arêtes possibles (le nombre de connexions possibles) [61]. Ainsi, elle varie entre 0 pour un graphe vide (les sommets sont isolés) et 1 pour un graphe complet (il y a un lien entre chaque paire de sommets). Elle est calculée [72] par la formule :

$$D = \frac{|E|}{|V|(|V|-1)/2} \tag{1.1}$$

où :
$|E|$: le nombre d'arêtes dans le graphe.
$|V|$: le nombre de sommets dans le graphe.

- **Sous-graphe :** un graphe G2 est un sous graphe du graphe G1 si et seulement si, G2 est composé de certains sommets de G1 et de toutes les arêtes qui les relient dans G1 [72].
- **Clique :** Une clique (Voir figure 1.5) est définie comme un ensemble de nœuds (sommets) qui sont en relation deux à deux. On parle de clique maximale lorsqu'elle n'est pas contenue dans une clique plus grande [6, 69].

  Dans la figure 1.5 les noeuds 5,6,7,8 forment une clique

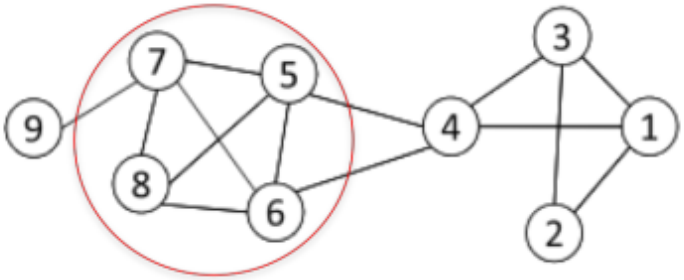

FIG. 1.5 : représentation d'une clique [11]

- **Matrice d'adjacence :** Pour un graphe G d'ordre n non orienté et non pondéré (non valué), la matrice d'adjacence A d'ordre (n×n) est une représentation matricielle exactement équivalente au graphe G. Dans cette matrice les lignes et les colonnes représentent les sommets (noeuds) du graphe [6, 69]. Cette matrice (n×n) est binaire, c'est-à-dire :

$$A_{ij} = \begin{cases} 1, & \text{s'il existe un lien entre les sommets i et j .} \\ 0, & \text{Autrement.} \end{cases} \tag{1.2}$$

  Par exemple : la matrice d'adjacence pour le graphe 1.1 est représentée dans la tableau 1.1 :

A	v1	v2	v3	v4	v5	v6	v7	v8	v9
v1	0	**1**	**1**	**1**	0	0	0	0	0
v2	**1**	0	**1**	0	0	0	0	0	0
v3	**1**	**1**	0	**1**	0	0	0	0	0
v4	**1**	0	**1**	0	**1**	**1**	0	0	0
v5	0	**1**	**1**	**1**	0	0	0	0	0
v5	0	0	0	**1**	0	**1**	**1**	**1**	0
v6	0	0	0	**1**	**1**	0	**1**	**1**	0
v7	0	0	0	0	**1**	**1**	0	**1**	**1**
v8	0	0	0	0	**1**	**1**	**1**	0	0
v9	0	0	0	0	0	0	**1**	0	0

Tab. 1.1 : Matrice d'adjacence pour le graphe 1.1

1.2.2 Mesures de similarités, de distances et de centralités d'un graphe

Mesures de similarités

Soient deux nœuds i et j avec N_i et N_j, représentent l'ensemble des voisins de i et l'ensemble des voisins de j respectivement.
On peux calculer la similarité entre deux noeuds par plusieurs méthodes dont on peux citer :

1. **l'indice de Jaccard :** Pour deux nœuds i et j l'indice de Jaccard est défini par [42] :

$$Jaccard(i,j) = \frac{|N_i \cap N_J|}{|N_i \cup N_j|} \tag{1.3}$$

où :
N_i est l'ensemble des voisins de i.
N_j est l'ensemble des voisins de j.
$|N_i \cap N_J|$ est le nombre de voisins communs des deux noeuds i et j.
$|N_i \cup N_j|$ est le nombre total des voisins deux noeuds i et j.

2. **La similarité Cosinus :** elle est mesurée à l'aide de la formule suivante [105] :

$$Cos(i,j) = \frac{|N_i \cap N_J|}{\sqrt{|N_i * N_J|}} \tag{1.4}$$

où :
$|N_i \cap N_J|$ est le nombre de voisins communs des deux noeuds i et j.
$|N_i * N_J|$ est la multiplication des voisins de i et de j.

Mesures de distances

1. **La Distance de Hamming :** elle est calculée par [38] :

$$Dham(i,j) = 1 - \frac{|N_i \cap N_J|}{n} \tag{1.5}$$

où :
n est le nombre de sommets (noeuds) du graphe.
$|N_i \cap N_J|$ est le nombre de voisins communs des deux noeuds i et j.

2. **La Distance euclidienne :** elle calcule la racine carrée de la somme des différences au carré entre les coordonnées d'une paire d'objets (une paire de noeuds) [11], elle est calculée par [24] :

$$DistE(i,j) = \sqrt{\sum_{k=1}^{n} (i_k - j_k)^2} \tag{1.6}$$

où :
i,j : représente les deux noeuds i et j.
n est le nombre de sommets (noeuds) du graphe.

Mesure de centralité

Afin d'analyser les positions des individus relativement aux autres dans un graphe, des mesures de centralités sont utilisées pour les caractériser. Plusieurs types de centralités ont été définis dont on peut citer quelques uns :

1. **La centralité de degré** : L'indicateur de centralité le plus simple est la centralité de degré. La centralité de degré d'un nœud est tout simplement son degré , c'est-à-dire le nombre de liens qui lui est associé (nombre de voisins adjacents) [117] ;

2. **La centralité de proximité :** Elle indique si le sommet est situé à proximité de l'ensemble des sommets du graphe et s'il peut rapidement interagir avec ces sommets, elle est calculée comme 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 [117, 33], elle s'écrit formellement :

$$C_c(V) = \frac{1}{\sum_{u \in V \setminus \{v\}} d_G(u,v)} \tag{1.7}$$

où $d_G(u,v)$ la distance entre les sommets u et v, telle que le nombre d'arêtes dans le plus court chemin entre deux sommets ou la somme des valuations de ces arêtes pour les graphes valués.

3. **La Centralité intermédiare :** La centralité d'intermédiarité est l'un des concepts les plus importants. Il mesure l'utilité du sommet dans la transmission de l'information au sein du réseau (graphe). Le sommet joue un rôle central si beaucoup de plus courts chemins entre deux sommets doivent emprunter ce sommet. Elle est égale au nombre de fois que ce sommet est sur le chemin le plus court entre deux autres nœuds quelconques du graphe [61, 7], elle s'écrit :

$$C_B(V) = \sum_{i,ji\neq j} \frac{\sigma_{ij}(v)}{\sigma_{ij}} \tag{1.8}$$

Avec $\sigma_{ij}(v)$ le nombre de chemins entre i et j qui passent par v. La centralité $C_B(V)$ d'un nœud est élévée, si la plupart des communiacations dans le graphe passent par lui.

4. **La Centralité PageRank :** La mesure PageRank a été introduite à la fin des années 90 par les deux informaticiens Brin et Page [13] pour classer les pages web, Elle simule le comportement des utilisateurs lorsqu'ils naviguent sur le Web pour classer les pages, où les pages sont des nœuds de graphe et les hyperliens des arêtes. Le PageRank indique l'importance des nœuds en partant du principe que l'importance d'un nœud est la somme attendue de l'importance de tous les nœuds connectés (de ses voisins) et de la direction des arêtes. Sa valeur correspond à la distribution de probabilité de l'accès aléatoire aux nœuds. En théorie des graphes, le PageRank calcule récursivement une valeur normalisée et propagée pour chaque nœud d'un graphe [40]. Soit x et p deux nœuds dans un graphe G, le PageRank de x est donné comme suit :

$$PR(x) = (1-c) + c * \sum_{p \in Pnt_{in}(x)} \frac{PR(p)}{|Pnt_{out}(p)|} \tag{1.9}$$

où c est un facteur d'amortissement qui prend sa valeur dans [0, 1] (typiquement 0,85), $Pnt_{in}(x)$ est l'ensemble des nœuds pointant vers x et $Pnt_{out}(p)$ est l'ensemble des nœuds pointés par p et $|Pnt_{out}(p)|$ est sa cardinalité. Le PageRank opère sur le graphe dirigé et sa valeur pour un nœud donné est calculée itérativement sur la base du PageRank des nœuds pointant sur lui [48, 35].

1.3 Communauté

La notion de communautés dans les graphes n'a pas de définition formelle. Cependant, l'existence de zones plus densément connectées que d'autres est le résultat d'une présence de structures de graphes dont les nœuds se sont regroupés en communautés du fait de leur ressemblance ou de leurs intérêts communs [32]. Cette ressemblance ou ce partage peut avoir des interprétations différentes selon la nature et le type du réseau d'interaction considéré (réseaux sociaux, réseaux biologiques, etc). Nous allons donner ici deux définitions des communautés, l'une sémantique et l'autre structurelle [61, 1].

* **Définition sémantique :** Une communauté est un ensemble de nœuds qui partagent les mêmes centres d'intérêt ou ayant le même profil ;

* **Définition structurelle :** Une communauté est un ensemble de nœuds fortement liés entre eux et faiblement liés avec les autres nœuds du graphe.

1.3.1 Structure communautaire

Le groupe de communautés découvert dans le réseau est appelé structure communautaire ou couverture. Elle est représentée comme suit : C = C_1,C_2,C_3,...,C_k où C est la structure de communauté ou la couverture. C_1,C_2, C_k représentent les communautés. La taille de la structure de communauté représentée par $|C|$ indique le nombre de communautés [72]. Par exemple dans la Figure 1.6 les communautés sont : C_1 = (1,6,9,10), C_2 = (2,4,5), C_3 = (3,7,8,11,12) et la structure de communauté est $C = C1, C2, C3$.

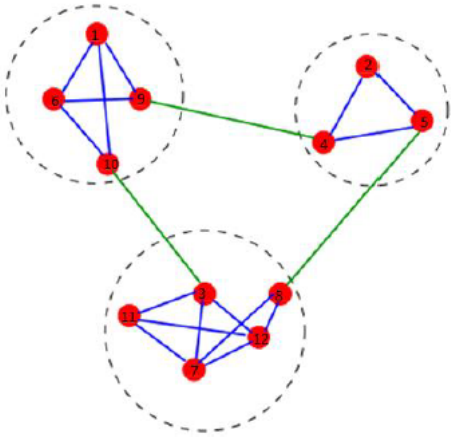

FIG. 1.6 : Structure de communauté dans le réseau [72].

La structure de communauté où certains nœuds sont chevauchants est connue comme structure de communauté chevauchante (voir figure 1.3). La structure de communauté où tous les nœuds sont non chevauchants est connue comme structure de communautés disjointes [1].

La détection des structures communautaires peut être définie par une classification des noeuds du réseau plus densément connectés que d'autres, pour construire des classes connexes d'utilisateurs ayant les mêmes caractéristiques au regard d'une mesure de similarité se référant à des intérêts communs [18, 1].

1.3.2 Objectifs de la détection des communautés :

Parmi les objectifs de cette notion de communautés, on peut citer ce qui suit [27] :

- La détection de communautés a pour objectif de révéler de nouvelles relations et d'extraire de nouvelles propriétés dans un réseau représenté sous forme de graphe, dans le but de comprendre la structure d'un tel réseau, de révéler des informations, etc....
- Identification des acteurs centraux d'un système : dans le domaine de la sécurité informatique, par exemple, il est important de savoir quels sont les nœuds (leaders) les plus importants qui doivent être attaqués (ou mieux sécurisés si on est côté administrateur) pour déstabiliser (ou stabiliser) un système ou un réseau ;
- Fournir un résumé de la structure du réseau ;
- Avoir de nombreuses propriétés sur le réseau comme l'importance d'un acteur donné (son influence, sa popularité, etc.) par rapport aux autres ;
- Étudier la similarité entre les individus d'une même communauté et donc le degré d'interaction et mesurer ensuite la force de la relation entre eux ;
- Extraction des différents profils, centres d'intérêt, sujets d'actualité dont la population en parle ;
- Connaître la tendance politique d'une population, connaître les goûts et les opinions des gens sur les produits proposés sur le marché ;
- La connaissance d'un individu peut induire la connaissance des autres en relation avec lui ou avec le reste du groupe, le cas échéant.
- Mise en place d'une stratégie de marketing (systèmes de recommandation) : à la base de la connaissance de l'ensemble des communautés, nous pouvons connaître les différents profils (ou les intérêts communs regroupant les membres), et ensuite nous pouvons faire une diffusion (publicité, recommandation personnalisée, etc.) d'informations précises à un ensemble bien connu d'utilisateurs ;
- Solutions pour minimiser / maximiser la diffusion ;
- Extraire les connaissances du réseau et mettre en évidence les principales propriétés du réseau.

1.3.3 Applications de la détection des communautes

Plusieurs applications de la détection des communautés existent. On peux citer quelques une [17, 118, 45] :

- **Application dans les systèmes biologiques et les systèmes de santé :** La détection de communautés dans les réseaux biologiques a une signification importante tels que : les réseaux de protéines, les réseaux alimentaires, les réseaux métaboliques, etc.... Dans les réseaux de protéines, elle a été appliquée de manière à détecter les complexes protéiques.
  Dans les systèmes de soins de santé, la détection de communautés peut aider à

analyser la croissance rampante des cellules dans un tissu pulmonaire "cancer des poumons".

- **Application dans la détection de fraudes et d'anomalies :** par exemple, dans les transactions bancaires, les techniques de l'exploration de données (data mining en anglais) telle que la détection de communautés analyse le réseau de transactions en classifiant les transactions en communautés. Si une nouvelle transaction n'appartient à aucune communauté, elle est considérée comme une anomalie ou fraude.

- **Applications scientifiques et académiques :** Des algorithmes de détection de communautés sont utiles dans le domaine de la recherche scientifique. Leur utilité réside dans la capacité de classer les auteurs, leurs publications, les années et les lieux de publications, etc.... Ces algorithmes peuvent prédire de nouvelles relations entre auteurs (collaboration scientifique) et peuvent proposer de nouveaux papiers aux auteurs suivant leur profile. Ce système peut être réduit à un cas d'une simple bibliothèque où nous pouvons proposer des livres aux étudiants suivant leur spécialité, analyser la similarité entre livres et entre étudiants, etc....

1.3.4 Algorithmes de détection de communautés

Il existe plusieurs algorithmes de détection de communautés.

Dans cette section, on présente les algorithmes utilisés pour comparer notre travail.

Girvan et Newman [76]

C'est l'algorithme hiérarchique divisif le plus connu, abrégé souvent par GN pour désigner les auteurs Girvan et Newman, qui a introduit une mesure de centralité appelée centralité d'intermédiarité des liens (Edge-Betweenness Centrality) [74] pour partitionner un graphe. Cette mesure de centralité est définiée comme le nombre de plus courts chemins entre deux nœuds qui passent par une arête. Cet algorithme est particulièrement intuitif. La première étape consiste á calculer cette edge betweenness pour toutes les arêtes du graphe, puis à retirer celle de plus haute betweenness. Ce processus est itéré jusqu'à retirer la dernière arête. Dans une seconde phase, à partir du graphe sans arête, les arêtes sont réintroduites dans l'ordre inverse, ce qui fournit une hiérarchie très fine du réseau, car en ajoutant une arête reliant deux communautés, on ajoute un niveau hiérarchique, les deux communautés étant regroupées dans une super-communauté. [34].
L'idée de cet algorithme est la suivante : si un lien se trouve fréquemment sur les plus courts chemins entre les nœuds du graphe, alors il ne se trouve pas au sein d'une communauté donnée, mais il relie des portions distantes du graphe (des communautés distinctes) [61].

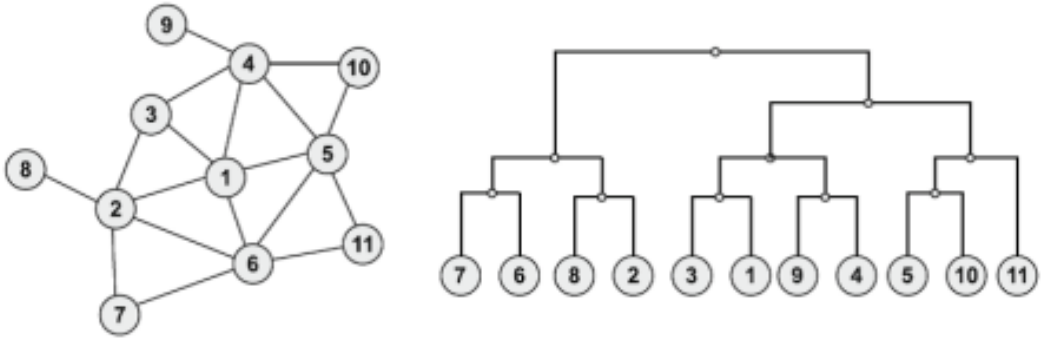

Fig. 1.7 : Exemple d'un dendrogramme hiérarchique pour Newman [106].

L'algorithme de Louvain

C'est une simple méthode hiérarchique ascendante de détection de communautés proposée en 2008 par Blondel et al. de l'université de Louvain [8] , il démarre par l'hypothèse que tout nœud est une communauté, puis il regroupe chaque deux nœuds adjacents dans une même communauté en maximisant la modularité. Il comprend deux phases. Premièrement, il cherche les « petites » communautés en optimisant la modularité de manière locale (séparément, au niveau de chaque communauté). Deuxièmement, il regroupe les nœuds d'une même communauté et construit un nouveau réseau dont les nœuds sont les communautés avec pour poids la somme des poids des arêtes entre les deux communautés.Ces deux phases produisent un nouveau niveau hiérarchique de découpage en communautés. L'algorithme s'arrête lorsqu'aucune fusion de la première phase n'améliore plus la modularité [15, 27].
Cet algorithme est très efficace en termes de temps d'exécution, il permet d'analyser des réseaux typiques de deux millions de nœuds en deux minutes [8], de plus, la qualité de communautés détectées est très bonne [22].
La figure 1.8 suivante nous représente une visualisation des étapes de l'algorithme de Louvain.

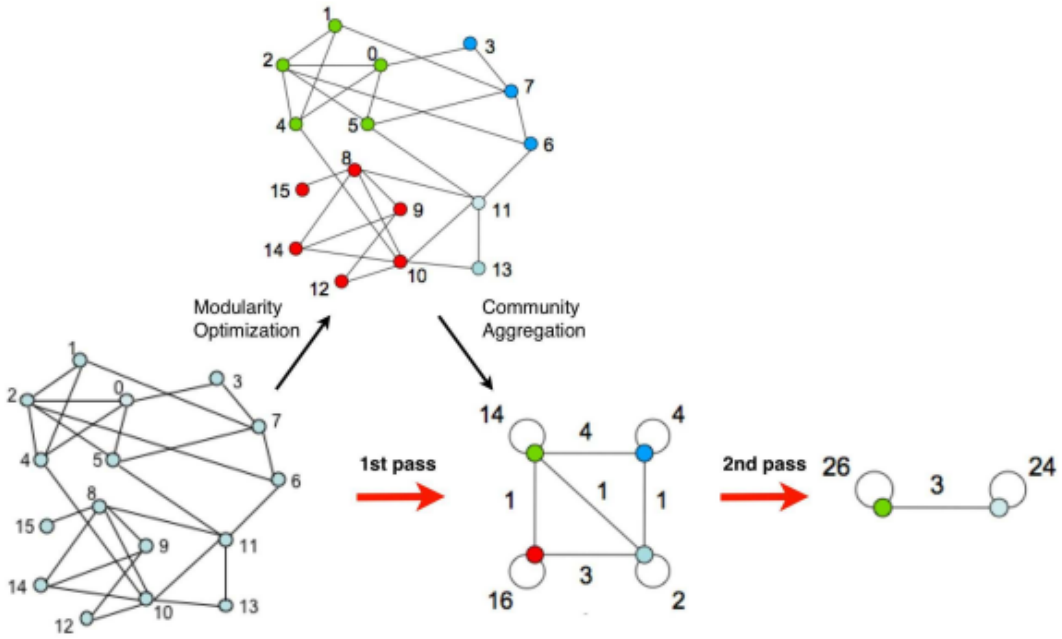

FIG. 1.8 : Visualisation des étapes de l'algorithme de Louvain [8].

Label propagation

L'idée principale de cet algorithme est que chaque sommet (X) du réseau se voit attribuer une étiquette (Label) unique (sa propre communauté) [94], ensuite (X) détermine sa communauté en fonction des étiquettes de ses voisins. (X) fait partie de la communauté qui contient le plus grand nombre de nœuds voisins, Un processus itératif est exécuté afin que les groupes de sommets connectés puissent atteindre un consensus sur une étiquette donnant lieu à une communauté, à la fin du processus de propagation, les nœuds ayant les mêmes étiquettes sont regroupés en une seule communauté [62].

Infomap

l'algorithme Infomap a été créé par Rosvall et Bergstrom (2011) [96]. Cet algorithme fait appel à une équation carte généralisée (generalized map equation). Emprunté de la théorie de l'information, ce concept est utilisé afin d'obtenir une hiérarchie de partitions à niveau variable. Les paramètres de cette équation récursive sont obtenus par marches aléatoires.
cet algorithme est similaire à la méthode Louvain, c'est-à-dire qu'en première phase une liste de sommets triée aléatoirement est parcourue et chaque opération vise à trouver le voisin avec lequel le sommet sélectionné constitue une communauté minimisant la valeur de l'équation de carte jusqu'à ce qu'un minimum soit atteint.
L'algorithme procède alors de façon récursive à une nouvelle phase essentiellement identique mais à un niveau supérieur, soit avec les communautés et sous-communautés. Pour chaque phase, l'ordre aléatoire des sommets ou modules est réeffectué après chaque parcours. Cet algorithme possède l'avantage d'arriver à des partitions hiérarchisées sans avoir à choisir les nombres de communautés ou sous-communautés ni le nombre de niveau de

l'hiérarchie. en plus, Infomap prouve son efficacité sur des grands réseaux par sa rapidité,en lui permettant de les traiter [1].

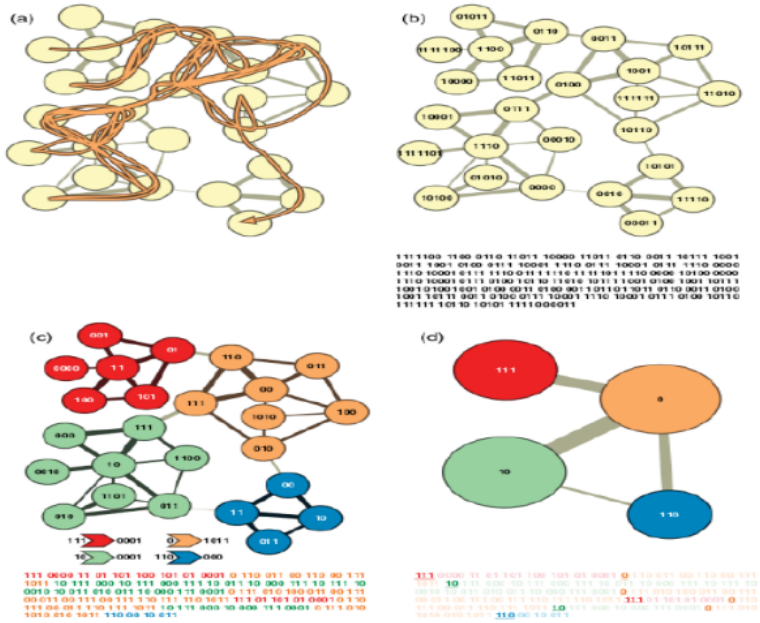

FIG. 1.9 : Illustration de fonctionnement d'Infomap [96].

a) Représentation de parcours d'un marcheur aléatoire que l'on souhaite encoder.

b) Un codage binaire de l'ensemble des parcours avec le moins de caractères possibles.

c) Un codage à deux niveaux (sur les groupes puis sur les nœuds) permettant à un même code d'être utilisé pour des nœuds différents.

d) Une visualisation des groupes sous forme de graphe quotient.

Fast Greedy

Cet algorithme a été développé par Newman et al. [75, 19]. Il est basé sur la modularité et utilise une approche agglomérante hiérarchique. Il est appelé fast greedy car, grâce à une méthode standard, il est significativement plus rapide que les anciens algorithmes [78]. chaque nœud est initialement dans sa propre communauté et ensuite, à chaque étape, l'algorithme regroupe deux communautés afin de de maximiser le gain de modularité.

1.4 Ensembles des données

Un data set ou ensemble de données, est un ensemble de valeurs associées à des variables et à une observation. Ces variables décrivent les attributs d'une unité ou individu. Les data sets peuvent prendre une structure tabulaire, arbre, fichier ou graphe,. . . etc

Dans le domaine de la détection de communauté les data sets sont de deux types : soit des réseaux réels ou bien des réseaux synthétiques (computer-generated benchmarks) [69].

1.4.1 Les réseaux réels

Dans la littérature une collection de data sets réels existent sous forme de graphes. Ces graphes ne sont pas conçus aléatoirement mais à la base de propriétés sociales réelles. En d'autre terme, ces réseaux représentent graphiquement des comportements humains qui peuvent être exploités et analysés.
Plusieurs réseaux réels existent, nous présentons quelques réseaux très utilisés :

1. **Le réseaux Zachary de club de karaté :** Le réseau Zachary de club de karaté [116] , illustré par la Figure 1.10 est un réseau très connu, il est utilisé régulièrement comme référence pour tester les algorithmes de détection de communautés. Il se compose de 34 noeuds, qui représentent les membres d'un club de karaté dans l'université de San Francisco aux Etas Unis, qui ont été observés au cours d'une période de trois ans, et 78 arêtes qui représentaient la connexion et l'interaction entre les membres. Ce réseau comporte deux communautés.

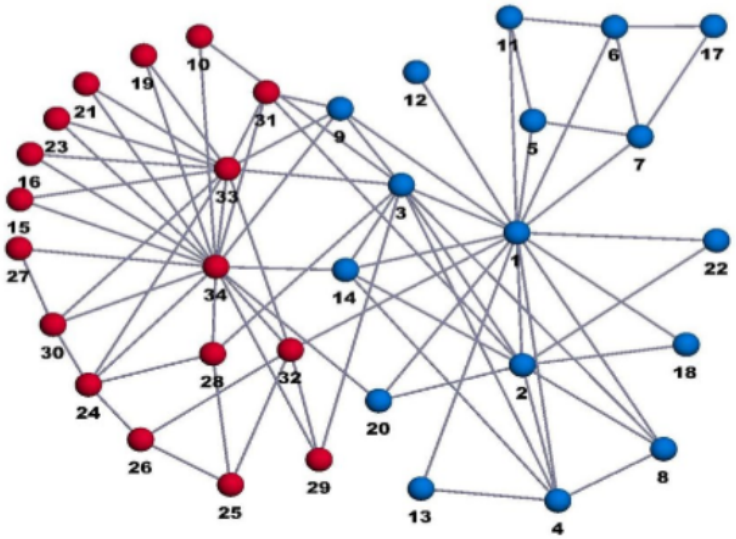

Fig. 1.10 : Le réseau de club de Karaté de Zachary [52].

2. **Réseau des dauphins :** Le réseau des Dolphins[63] a été obtenu par les auteurs qui ont observé le mode de vie des dauphins qui vivent depuis longtemps dans les fjords magiques de la Nouvelle Zélande. C'est un réseau de relations sociales entre dauphins. Les échanges fréquents entre les dauphins constituent une connexion entre différents dauphins, consistant en 62 nœuds et 159 arêtes (Voir figure 1.11). Les nœuds représentent les dauphins et les arêtes représentent les interactions (les associations fréquentes entre chaque deux dauphins) entre les dauphins. Ce réseau comporte deux communautés, mâles et femelles.

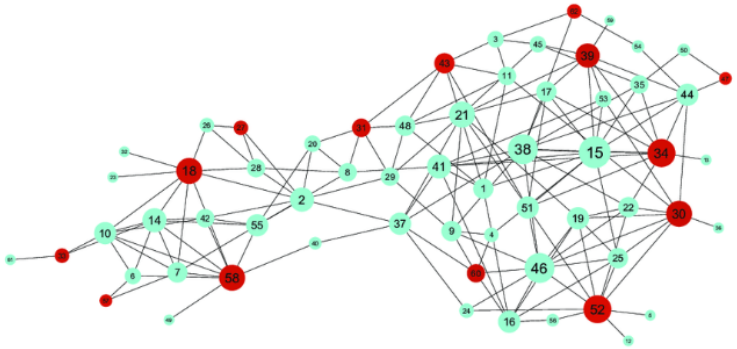

FIG. 1.11 : le réseau dauphins de Lusseau [28].

3. **Réseau de Football Américain (Football Network) :** Un autre exemple de réseau réel étudié dans le cadre de ce mémoire est le réseau de jeux du football américain [79]. Il représente le calendrier des matchs entre des équipes américaines de football durant l'année 2 000. Ce réseau est constitué de douze communautés (Voir figure 1.12), 115 nœuds et 613 liens. Chaque nœud représentait une équipe de football, tandis que chaque arête représentait la relation entre deux équipes.

FIG. 1.12 : La structure communautaire du réseau Football [31].

4. **Réseau de livres politiques américains :** Cet ensemble de données est le réseau de co-achat Amazon avec 105 livres sur la politique américaine. Il y a 441 arêtes (Voir figure 1.13). Les nœuds sont des livres et les arêtes représentent le co-achat de livres par les mêmes acheteurs. Ce réseau comporte 3 communautés, les Démocrates, les Républicains et les neutres [76, 53].

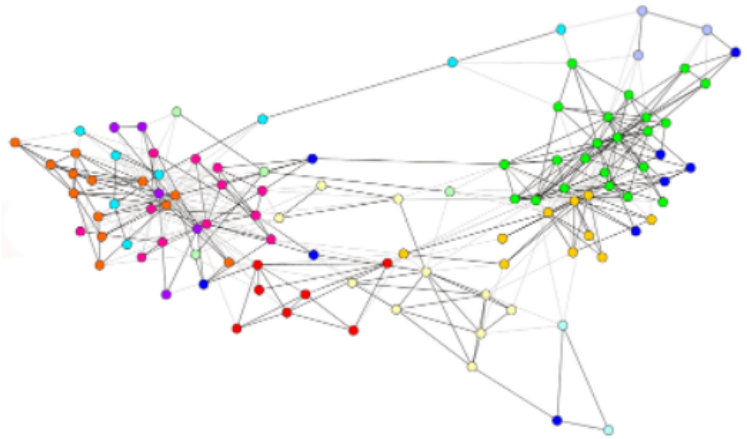

FIG. 1.13 : Réseau de livres politiques [80].

Réseaux	Noeuds	Arêtes	Nombre de communautés
Zachary	34	78	2
Football	115	613	12
Dauphins	62	159	2
Books	105	441	3

TAB. 1.2 : Paramètres des réseaux du monde réel.

1.4.2 Benchmark LFR

Le benchmark LFR est un algorithme qui génère des réseaux de référence (réseaux artificiels) avec des structures communautaires paramétrables [54].
Le réseau de référence LFR proposé par Lancichinetti et al. [54], est l'un des réseaux synthétiques les plus couramment utilisés pour tester les méthodes de détection de communautés dans les graphes.
Il présente des caractéristiques hétérogènes similaires à celles du réseau réel, c'est-à-dire que le degré des nœuds et la taille de la communauté sont répartis suivent une distribution en loi de puissance, avec des exposants différents tau1 et tau2 [113, 69].
Pour générer des LFR_benchmarks, l'algorithme de générateur LFR est le suivant :

LFR_benchmark_graph(n, tau1, tau2, μ, average_degree, min_degree, max_degree, min_community, max_community, tol=1e-07, max_iters, seed)

Cet algorithme se déroule comme suit :

1. Trouver une séquence de degrés avec une distribution en loi de puissance, et une valeur minimale min_degree, qui a un degré moyen approximatif average_degree. Pour cela, il faut soit :

   a) en spécifiant min_degree et non average_degree,

   b) en spécifiant average_degree et non min_degree, auquel cas un degré minimum approprié sera trouvé.

   max_degree peut aussi être spécifié, sinon il sera fixé à n.

2. Générer la taille des communautés selon une loi de puissance avec l'exposant "tau2". Si "min_community" et "max_community" ne sont pas spécifiés, ils seront sélectionnés pour être respectivement "min_degree" et "max_degree".

4. Chaque noeud *u* ajoute ensuite (1 - μ) mathrmdeg(u) des arêtes intra-communauté et μ mathrmdeg(u) des arêtes inter-communauté.

Paramètres [113, 106] :

- n : de type int, représente le nombre de noeuds dans le graphe créé.
- tau1 : de type float, représente l' Exposant de la loi de puissance pour la distribution des degrés du graphe créé. Cette valeur doit être strictement supérieure à un.
- tau2 : de type float, représente l'Exposant de la loi de puissance pour la distribution de la taille de la communauté dans le graphe créé. Cette valeur doit être strictement supérieure à un.
- μ : de type float, c'est Le paramètre de mélange (mixing paramater) qui représente la fraction des arêtes inter-communautaires incidentes à chaque noeud, tel que chaque nœud partage une fraction $1 - \mu$ avec les nœuds de la communauté où il appartient et une fraction μ avec les nœuds du reste de graphe.
  Cette valeur doit être dans l'intervalle [0, 1].
  Plus le paramètre de mélange est grand, moins la structure de la communauté est claire.

- average_degree : de type float, c'est le degré moyen désiré des noeuds dans le graphe créé. Cette valeur doit être comprise dans l'intervalle [0, n].
- min_degree : de type int, c'est le degré minimum des noeuds dans le graphe créé. Cette valeur doit être dans l'intervalle [0, n].
- max_degree : de type int, c'est le degré maximum des noeuds dans le graphe créé. S'il n'est pas spécifié, il est fixé à n (le nombre total de noeuds dans le graphe).
- min_community : de type int, c'est la taille minimale des communautés dans le graphe. Si elle n'est pas spécifiée, elle prend la valeur min_degree.
- max_community : de type int, c'est la taille maximale des communautés dans le graphe. Si elle n'est pas spécifiée, la valeur est fixée à n.
- tol : float, c'est la tolérance lors de la comparaison de flottants, en particulier lors de la comparaison de valeurs de degrés moyens.
- max_iters : de type int, c'est le 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, ou None (par défaut), c'est l'Indicateur de l'état de la génération de nombres aléatoires.

1.5 Mesures d'évaluation de la qualité des structures communautaires

Actuellement, il existe de nombreuses mesures pour évaluer l'efficacité des algorithmes de détection de communautés. Parmi ces métriques, on a :

1.5.1 La Modularité (Q)

C'est l'une des métriques fréquemment utilisées pour mesurer la qualité de la détection communautaire des réseaux, elle est proposée par Girven et Newman en 2004 [76]. Les réseaux à forte modularité présentent des connexions denses entre les nœuds au sein des modules, mais des connexions éparses entre les nœuds de différents modules. La modularité (Q) est définie par [57, 115] :

$$Q = \frac{1}{2m} + \sum\nolimits_{ij} \left[A_{ij} - \frac{k_i.k_j}{2m}\right]\delta(c_i, c_j) \tag{1.10}$$

où :
A_{ij} : Représente la matrice d'adjacence du réseau.
c_i : Représente la communauté à laquelle le noeud i est assigné
c : Représente le nombre de communautés.
k_i : Représente le degré du nœud i.
k_j : C'est le degré du nœud j.
m : C'est le nombre de nombre d'arêtes dans un réseau.
Ainsi, la fonction $\delta(c_i, c_j)$ peut être défini comme suit :

$$\delta(c_i, c_j) = \begin{cases} 1, & \text{Si le nœud i et le nœud j sont dans la même communauté,} \\ 0, & \text{sinon.} \end{cases} \tag{1.11}$$

La valeur de Q est comprise entre -1 et $+1$, plus la valeur est proche de 1, plus la force de la structure communautaire dans le réseau est élevée, plus la qualité de la détection de communautés est bonne [115].

1.5.2 l'information mutuelle normalisée (NMI)

C'est une mesure de similarité qui estime la similarité entre deux partitions A et B ; où A représente la partition réelle du réseau et B la partition détectée par les algorithmes expérimentaux de détection de communautés. Elle est fondée sur la théorie de l'information [26].

Pour deux partitions A et B d'un réseau, la valeur de NMI est calculée par l'équation (1.12) [57] :

$$NMI(A, B) = \frac{-2\sum\nolimits_{i=1}^{C_A}\sum\nolimits_{j=1}^{C_B} N_{ij} log(\frac{N_{ij}N}{N_i N_j})}{\sum\nolimits_{i=1}^{C_A} N_i log\frac{N_i}{N} + \sum\nolimits_{j=1}^{C_B} N_j log\frac{N_j}{N}} \tag{1.12}$$

où :
A : représente la partition réelle du réseau.
B : la partition découverte par les algorithmes de détection de communautés.
C_A : représente le nombre de communautés dans la partition A.
C_B : désigne le nombre de communautés dans la partition B.
N : représente le nombre total de nœuds dans le réseau.
N_{ij} : représente le nombre de nœuds identiques dans la communauté i de la partition A et la jème communauté dans la partition B.
N_i : c'est le nombre de nœuds dans la communauté réelle i (la somme de la ligne i de la matrice N_ij).
N_j : C'est le nombre de noeuds dans la communauté calculée j (la somme de la colonne j).

La valeur de NMI peut varier entre 0 et 1. Plus la valeur de NMI est proche de 1, plus les deux partitions sont similaires.
En d'autre terme, lorsque deux partitions A et B sont complètement différentes, alors $NMI(A, B) = 0$.
Si NMI prend sa valeur maximale qui est égale à 1, alors, la partition A correspond exactement à la partition B.

1.5.3 La F-mesure :

aussi appelée F-score, qui est une moyenne harmonique pondérée de la precision et le rappel. Elle est souvent utilisée pour évaluer la qualité du modèle de classification[60, 25]. La définition de la F-mesure du cluster A avec le cluster B est donnée par l'équation(1.13) :

$$F(A, B) = \frac{2 * P(A, B) * R(A, B)}{P(A, B) + R(A, B)} \tag{1.13}$$

où P(A,B) est la précision et R(A,B) est le taux de rappel, avec [11] :

$$P(A, B) = \frac{n_{A,B}}{n_B} \tag{1.14}$$

Où n_{AB} est le nombre des nœuds de la partition A assigné à la partition B et n_B est le nombre des nœuds de la partition B.

$$R(A, B) = \frac{n_{A,B}}{n_A} \tag{1.15}$$

Où n_{AB} est le nombre des nœuds de la partition A assigné à la partition B et n_A est le nombre des nœuds de la partition A.

1.5.4 L'indice de Rand (RI) :

Une autre mesure de concordance (de similarité) entre deux partitions est l'indice de Rand. Celui ci est simplement le taux de paires d'éléments qui sont en accord, c'est-à-dire associés à la fois dans $P1$ et $P2$ ou séparés à la fois dans $P1$ et $P2$ [60, 20].

Ainsi, L'indice de Rand associé à deux partitions P_1 et P_2 est défini par l'équation (1.16) :

$$RI(P_1, P_2) = \frac{A + B}{A + B + C + D} \tag{1.16}$$

Où :

A : nombre de paires d'éléments classés ensemble à la fois dans P_1 et P_2
B : nombre de paires d'éléments présents dans des classes différentes dans P_1 et P_2
C : nombre de paires d'éléments présents dans des classes différentes dans P_1 mais présents dans une même classe dans P_2.
D : nombre de paires d'éléments présents dans une même classe dans P_1 mais présents dans des classes différentes dans P_2.
Il varie entre 0 et 1 et prend la valeur maximale en cas de concordance des partitions.

1.5.5 Indice de Rand ajusté (ARI)

Dans la mesure où l'indice de Rand varie de façon très importante sur deux partitions tirées au hasard, ce que Vinh appelle la "constant baseline property", une version "ajustée" a été créée dont l'espérance est nulle lorsque les partitions sont sélectionnées au hasard [41]. L'indice de Rand ajusté (ARI, pour Adjusted Rand Index) corrige cet effet en normalisant l'indice de Rand (RI), c'est une adaptation de l'indice de Rand conçue pour être insensible au nombre de classes [20, 102]. L'ARI est définie par l'équation (1.17) :

$$ARI(P_1, P_2) = \frac{RI(P_1, P_2) - E(RI(P_1, P_2))}{max(RI(P_1, P_2)) - E(RI(P_1, P_2))} \tag{1.17}$$

où $E(RI(P_1, P_2))$ est l'espérance de la valeur de l'indice de Rand, autrement dit l'indice obtenu en partitionnant les données au hasard. Si les partitions sont identiques, l'ARI vaut 1. L'ARI est une mesure corrigée pour la chance : l'espérance de l'ARI de deux partitions tirées aléatoirement vaut 0 [20].

On va comparer notre travail avec les autres travaux par la mesure d'évaluation Q (la Modularité) .

1.6 Conclusion

La présentation générale menée au cours de ce chapitre sur les communautés portant sur la définition et la détection des communautés, quelques définitions de la théorie de graphe. Nous avons présenté aussi les mesures d'évaluation de la détection de communautés, par exemple la modularité Q, la NMI, la F-mesure.

Dans le chapitre suivant, nous présenterons l'apprentissge automatique, ses principaux types ainsi qu'une présentation détaillée de l'algorithme qu'on va utiliser dans notre approche "l'algorithme **K-Means**".

Chapitre 2

L'apprentissage automatique

2.1 Introduction

Récemment, les réseaux sociaux font partie de notre vie quotidienne, par exemple Facebook, Twitter et Instagrame,…etc. Un réseau social est un ensemble d'individus reliés par différents types de liens : amitié, fraternité, profession, etc. L'interaction entre individus se fait par envoi de messages et de partage de photos, etc. Pour comprendre cette interaction et la structure de ces réseaux, une analyse intéressante a été faite ; cette analyse est la détection des communautés.

l'application d'algorithmes de l' apprentissage automatique dans les tâches de détection de communautés dans les réseaux a attiré une grande attention ces dernières années.

Dans ce chapitre, nous nous intéressons à l'apprentissages automatique, sa définition, ses différents types, ainsi que des définitions de certains types. Nous achevons ce chapitre par une explication détaillée de l'algorithme utilisé dans l'approche proposée, qui est l'algorithme **K-means**.

2.2 Méthodes d'Apprentissage Automatique

L'apprentissage automatique ou Machine Learning (ML), est une science qui consiste à développer des algorithmes d'apprentissage, qui apprennent à résoudre une tâche spécifique sans être programmés explicitement [98].

L'apprentissage automatique est un type de l'intelligence artificielle qui représente la capacité d'un logiciel à apprendre puis à reconnaître des modèles complexes de données, tout comme un être humain ; plus la quantité des données collectées est grande, plus la machine décuple ses compétences .

On peut classer les méthodes d'apprentissage automatique en plusieurs catégories : l'apprentissage supervisé, l'apprentissage non supervisé, et l'apprentissage par renforcement [99] (Voir figure 2.1).

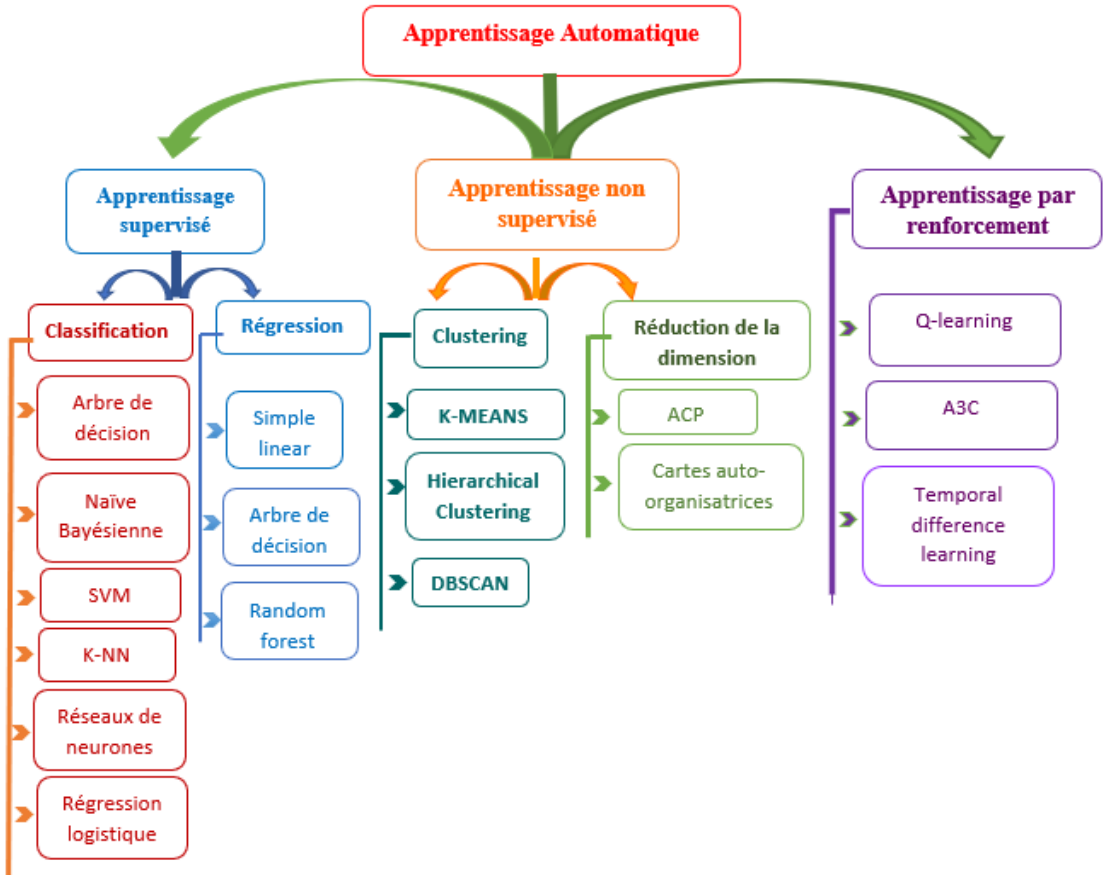

FIG. 2.1 : Les méthodes d'apprentissage automatique

2.2.1 L'apprentissage supervisé

L'apprentissage supervisé (SL) est un algorithme d'apprentissage automatique permettant d'acquérir des informations sur la relation entrée-sortie d'un système sur la base d'un ensemble donné d'échantillons d'apprentissage d'entrée-sortie étiquetées. Autrement dit, c'est un système qui fournit à la fois les données en entrée et les données attendues en sortie. Les données en entrée et en sortie sont étiquetées afin d'établir une base d'apprentissage pour le traitement ultérieur des données [23, 55].

L'objectif de l'apprentissage supervisé est de construire un système artificiel capable d'apprendre la correspondance entre l'entrée et la sortie, et de prédire la sortie du système en fonction de nouvelles entrées [59].

L'apprentissage supervisé est généralement effectué dans le contexte de **la classification** et de **la régression** [59].

Classification

La classification est un processus consistant à trouver une fonction qui aide à diviser l'ensemble de données en classes en fonction de différents paramètres. Dans la classification, un programme informatique est entraîné sur l'ensemble de données d'entraînement et, sur la base de cet entraînement, il catégorise les données en différentes classes [16]. La tâche de l'algorithme de classification est de trouver la fonction de mappage pour mapper

l'entrée (x) à la sortie discrète (y).

Les algorithmes de classification peuvent être divisés en plusieurs types, on peut citer quelques-uns [73] :

- Arbre de décision (DT).
- Naïve Bayesienne (NB).
- Machine à vecteurs de support (SVM).
- Les K-voisins les plus proches (KNN).
- Réseau neuronal artificiel (ANN).
- Régression logistique (LR).

Parmi ces algorithmes, on définit :

Arbre de décision :

L'arbre de décision (Decision Tree ou DT) est une technique de base de l'analyse de données [16, 95]. Il a été utilisé dans plusieurs domaines comme l'apprentissage automatique, la reconnaissance de forme, le diagnostic médical, etc.

Dans sa forme la plus simple, un arbre de décision est un type d'organigramme qui montre un chemin clair vers une décision. En termes d'analyse de données, il s'agit d'un type d'algorithme qui inclut des instructions de contrôle (conditionnelles) pour classer les données.

Un arbre de décision commence à un point unique (nœud) qui se ramifie ensuite dans deux directions ou plus. Chaque direction générale offre différents résultats possibles, incorporant une variété de décisions et d'événements aléatoires jusqu'à ce qu'un résultat final soit atteint [97]. Lorsqu'ils sont montrés visuellement, leur apparence ressemble à un arbre, d'où son nom.

La Classification naïve bayésienne :

La classification naïve bayésienne (NB) est une autre méthode d'apprentissage supervisé, elle est listée parmi les modèles probabilistes les plus connus [95]. Il s'agit d'un type de classification simple (" naïf") reposant sur le théorème de Bayes [21].

Les algorithmes Naïves Bayes sont utilisés dans la catégorisation et la classification de documents. Cette technique permet d'estimer la probabilité de chaque classe parmi les exemples trouvés, et donner une étiquette à la classe la plus probable [95, 16].

Machines à vecteurs de support :

la machine à vecteurs support (Support Vector Machine ou SVM), est un algorithme d'apprentissage automatique supervisé utilisé en particulier pour les problèmes de classification.

Cet algorithme place toutes les données sous forme de points dans une zone de n dimensions de caractéristiques et les classifier avec l'hyperplan qui différencie les classes [44, 77].

Il est très efficace pour traiter les données en masse, grâce aux vecteurs de support pour la prédiction [67, 16].

Les k-plus proches voisins :

La méthode des k-plus proches voisins (KNN) est une approche de classification supervisée, elle consiste à déterminer pour chaque nouvel individu que l'on veut classer, la liste des k plus proches voisins parmi les individus déjà classé [100, 107]. L'individu est affecté à la classe qui contient le plus d'individus parmi ces k plus proches voisins. Cette méthode nécessite de choisir une distances (la distance Euclidienne, la distance de Manhattan, distance de Jaccard, etc), et donc le nombre k de voisins à prendre en compte [16].

L'inconvénient majeur de cette approche est qu'elle est incapable d'analyser des données volumineuses pour un ensemble d'apprentissage qui nécessite un large espace de stockage[16, 67].

Les Réseaux de neurones artificiels :

Les réseaux de neurones artificiels (ANN) sont l'un des principaux outils les plus utilisés dans l'apprentissage automatique. Ce sont des structures inspirées des circuits nerveux du système nerveux humain et se composent d'un groupe d'unités informatiques interconnectées appelées neurone artificiel [43].

Un neurone artificiel consiste en une fonction de transfert avec plusieurs entrées pour une seule sortie. Dans un réseau, ces neurones sont connectés entre eux, la sortie d'un neurone pouvant être connectée à un ou plusieurs autres neurones. Chaque neurone envoie et reçoit des impulsions d'autres neurones, et ces changements sont utilisées pour prédire la classe des données en entrée [97].

Un réseau de neurones très commun est le perceptron multicouche [114]. Dans un perceptron, les neurones sont organisés en couches (une ou plusieurs) où chaque couche prend en entrée l'ensemble des sorties de la couche précédente (Voir figure 2.2). Pour entraîner un réseau de neurones, on ajuste les poids associés à chaque connexion entre deux neurones [114, 97].

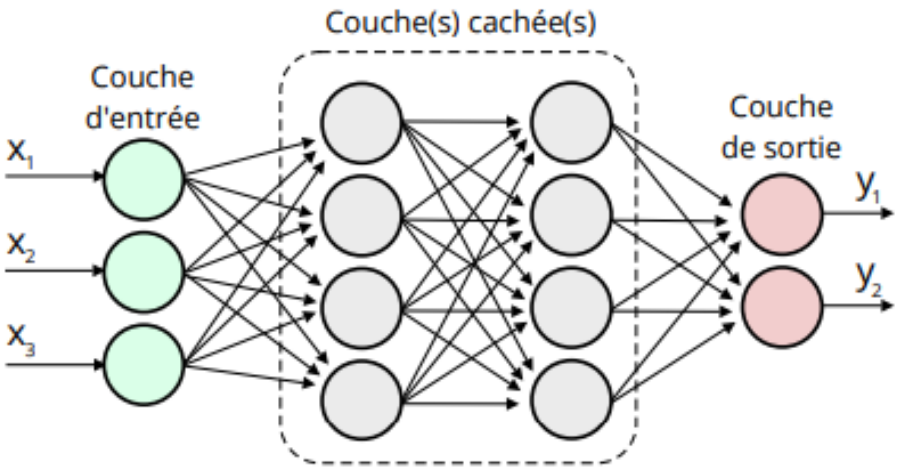

Fig. 2.2 : Un exemple de réseau de neurones avec 2 couches cachées

La Régression logistique :

La régression logistique (RL) est l'une des plus importantes techniques statistiques et d'exploration de données employées par les statisticiens et les chercheurs pour l'analyse et la classification des ensembles de données à réponse binaire. Certains des principaux avantages de la LR sont qu'elle peut naturellement fournir des probabilités et s'étendre aux problèmes de classification multi-classes. Un autre avantage est que la plupart des méthodes utilisées dans l'analyse des modèles LR suivent les mêmes principes que ceux utilisés dans la régression linéaire [64].

La Régression

La régression est un processus de recherche des corrélations entre les variables dépendantes et indépendantes. Il aide à prédire les variables continues telles que la prévision des tendances du marché, la prévision des prix des maisons, etc. La tâche de l'algorithme de régression est de trouver la fonction de mappage pour mapper la variable d'entrée (x) à la variable de sortie continue (y) [108]. En d'autres termes, la variable de sortie correspondante n'est pas une classe mais **une valeur mathématique (valeur réelle)** ; un pourcentage ou une valeur absolue. Par exemple, une probabilité pour une machine de tomber en panne (15 % ,20 %, etc.) ou le prix de vente idéal d'un appartement en fonction de critères comme la surface, le quartier, etc.

Les algorithmes les plus courants de la régression sont [73] :

* La Régression linéaire simple (SLR).
* La Régression de l'arbre de décision (DTR).
* La Régression de foret d'arbres décisionnels (DTFR).

La Régression linéaire simple :

La régression linéaire simple (SLR) est un type d'algorithmes de régression qui modélise la relation entre une variable dépendante et une seule variable indépendante. La relation montrée par un modèle de régression linéaire simple est linéaire ou une ligne droite inclinée, d'où son nom de régression linéaire simple. Le point clé de la régression linéaire simple est que la variable dépendante doit être une valeur continue (réelle). Cependant, la variable indépendante peut être mesurée sur des valeurs continues ou catégorielles [68].

Régression de l'arbre de décision :

La régression par arbre de décision (DTR) est une variante de l'arbre de décision qui peut être utilisée pour approximer des fonctions réelles telles que les proportions de classe.

La construction d'un arbre de régression est également basée sur le partitionnement récursif binaire, qui est un processus itératif qui divise les données en partitions. Au départ, tous les échantillons d'apprentissage sont utilisés pour déterminer la structure de l'arbre. Ensuite, l'algorithme décompose les données en utilisant toutes les divisions binaires possibles et sélectionne la division qui divise les données en deux parties de telle sorte qu'elle minimise la somme des écarts quadratiques par rapport à la moyenne dans les parties séparées.

Le processus de division est ensuite appliqué à chacune des nouvelles branches. Le processus se poursuit jusqu'à ce que chaque nœud atteigne une taille de nœud minimale spécifiée par l'utilisateur (c'est-à-dire le nombre d'échantillons d'apprentissage au niveau du nœud) et devienne un nœud terminal [111].

Régression de la foret d'arbres décisionnels :

La forêt aléatoire (DTFR) est une méthode dans laquelle nous considérons la prédiction de plusieurs arbres de régression décisionnelle. On identifie n, où n est le nombre de régresseurs d'arbre de décision à créer. La valeur moyenne de chaque branche est attribuée à chaque nœud feuille dans l'arbre de décision [16].

Remarque : La principale différence entre les algorithmes de régression et de classification est que les algorithmes de régression sont utilisés pour **prédire les valeurs** continues telles que le prix, le salaire, l'âge, etc. et les algorithmes de classification sont utilisés pour **prédire(classer) les valeurs discrètes** telles que masculin ou féminin, vrai ou faux, Spam ou non Spam, etc.

2.2.2 Apprentissage non supervisé

L'apprentissage non supervisé (UL) est un type d'apprentissage automatique qui consiste à ne disposer que de données d'entrée et pas de variables de sortie correspondantes. Dans ce type d'apprentissage, les algorithmes sont laissés à leurs propres mécanismes pour découvrir et présenter la structure intéressante des données ; il n'y a pas de réponse correcte ni d'enseignant [4].

Les réponses que l'on cherche à prédire ne sont pas disponibles dans les jeux de données. Ici, l'algorithme utilise un jeu de données non étiquetées. On demande alors à la machine de créer ses propres réponses. Elle propose ainsi des réponses à partir d'analyses et de groupement de données.

L'apprentissage non supervisé comprend deux catégories d'algorithmes : Algorithmes de Regroupement (Clustering en anglais) et les algorithmes de Réduction de la dimentialité (DR) [9].

Regroupement (Clustering)

Le regroupement (clustering) consiste à séparer ou à diviser un ensemble de données en un certain nombre de classes (clusters), en fonction de leurs similarités [12], de sorte que les objets de données de la même classe soient aussi similaires que possible, et que ceux des différentes classes soient aussi dissemblables que possible. Autrement dit, après le regroupement (clustering), des données similaires sont regroupées et les données différentes sont séparées. Contrairement aux tâches de classification, les classes (groupes) ne sont pas connus par avance [66].

C'est un type d'apprentissage automatique non supervisé car ici nous n'avons pas de variable cible et nous ne regroupons les points de données qu'en fonction de leurs similitudes [39, 99].

parmi les applications du clustering, on peut citer :

- Segmentation de la clientèle ;
- Système de recommandation ;
- Segmentation d'image ;
- Clustering de documents.

Il existe plusieurs algorithmes de clustering, les plus répandus sont :

- Le Clustering DBSCAN.
- Le Regroupement hiérarchique (Hierarchical clustering).
- Le Clustering K-means.(expliqué dans la section 2.3)

L'algorithme de culstering DBSCAN

L'algorithme de regroupement spatial basé sur la densité d'applications avec bruit (DBSCAN) est un algorithme non supervisé très connu en matière de clustering. Il a été proposé par Martin Ester et al. en 1996 [30] .

C'est un algorithme de clustering basé sur la densité (La densité d'un objet peut être calculée par le nombre d'objets proche de celui-ci) qui fonctionne sur l'hypothèse que les clusters sont des régions denses dans l'espace séparées par des régions de densité inférieure [66].

DBSCAN recherche les objets principaux, c'est à dire les objets qui ont des voisinages denses. Il relie les objets centraux et leurs voisins pour former des régions denses appelés clusters. Il regroupe les points de données densément connectés en un seul cluster. Il peut identifier les clusters dans de grands ensembles de données spatiales en examinant la densité locale des points de données. La caractéristique la plus intéressante du clustering DBSCAN est qu'il est robuste aux valeurs aberrantes. Il ne nécessite pas non plus que le nombre de clusters soit indiqué à l'avance [101], contrairement aux K-means, où nous devons spécifier le nombre de centroïdes.

L'algorithme DBSCAN ne nécessite que deux paramètres : epsilon et minPoints. Epsilon est le rayon du cercle à créer autour de chaque point de données pour vérifier la densité et minPoints est le nombre minimum de points de données requis à l'intérieur de ce cercle pour que ce point de données soit classé comme point central [101]. Dans les dimensions supérieures, le cercle devient hypersphère, epsilon devient le rayon de cette hypersphère et minPoints est le nombre minimum de points de données requis à l'intérieur de cette hypersphère [101].

Le Clustering hiérarchique

L'algorithme de clustering hiérarchique (HC) est un autre algorithme d'apprentissage automatique non supervisé, qui est utilisé pour regrouper les points de données non étiquetés ayant des caractéristiques similaires dans un cluster [39].
L'avantage de cette méthode est qu'elle n'est soumise à aucune initialisation particulière de paramètre(s) ce qui la rend déterministe, et en outre, que le nombre de classe n'a pas à être fixé a priori [10].
Dans cet algorithme, nous développons la hiérarchie des clusters sous la forme d'un arbre, et cette structure en forme d'arbre est connue sous le nom de dendrogramme.

La technique de clustering hiérarchique a deux approches [71] :

1. **Approche agglomérative :** Dans les algorithmes hiérarchiques agglomératifs, chaque point de données est traité comme un seul cluster, puis le processus fusionne ou agglomère successivement les paires de clusters (approche ascendante). La hiérarchie des clusters est représentée sous la forme d'un dendrogramme ou d'une arborescence [10].

2. **Approche divisive :** Dans cette approche, tous les les points de données (individus) sont considèrés comme une seule classe au début, et le processus de clustering divise successivement les classes en classes plus raffinées (approche descendante). Le processus marche jusqu'à ce que chaque classe contienne un seul point ou bien si l'on atteint un nombre de classes désiré [71].

La Réduction de la dimensionnalité

Cette tâche consiste à compresser les données d'entrée sur un espace de représentation plus petit. Outre le simple gain en terme de place, cette réduction est également intéressante pour représenter les données.

Réduire la dimensionnalité permet d'éviter les problématiques liées à la malédiction de la dimension [39]. Cette malédiction fait référence aux problématiques que l'on rencontre sur des données de grandes dimensions. Par exemple, la recherche de plus proches voisins obtient de très mauvais résultats sur des données avec beaucoup de dimensions [47].

Cette méthode a plusieurs types, on peut citer quelques uns :
Analyse en composantes principales :

L' analyse en composantes principales (PCA), permet d'analyser et de visualiser un jeu de données contenant des individus décrits par plusieurs variables quantitatives [14]. C'est une méthode statistique qui permet d'explorer des données dites multivariées (données avec plusieurs variables). Chaque variable pourrait être considérée comme une dimension différente.

L'analyse en composantes principales est utilisée pour extraire et de visualiser les informations importantes contenues dans une table de données multivariées. L'PCA synthétise cette information en seulement quelques nouvelles variables appelées composantes principales. Ces nouvelles variables correspondent à une combinaison linéaire des variables

originels. Le nombre de composantes principales est inférieur ou égal au nombre de variables d'origine.

En d'autres termes, l'PCA réduit les dimensions d'une donnée multivariée à deux ou trois composantes principales, qui peuvent être visualisées graphiquement, tout en conservant le plus d'informations possibles de l'ensemble original, et donc en perdant le moins possible d'information.

Les cartes auto-organisatrices :

Les cartes auto-organisatrices (self organizing cards en anglais), permettent de projeter de manière non-linéaire les individus de l'ensemble d'apprentissage sur un espace topologique structuré appelé carte. Cet espace, est généralement défini par un maillage régulier en deux dimensions où chaque nœud peut être associé à une position dans l'espace des descripteurs. Cette méthode permet de représenter de manière pertinente les données en deux dimensions et est donc souvent utilisée en analyse de données exploratoire [51].

2.2.3 Apprentissage par renforcement :

l' Apprentissage par renforcement (Reinforcement Learning ou RL) est généralement utilisé pour apprendre à une machine à exécuter une séquence d'étapes, il est différent de l'apprentissage supervisé et non supervisé. Les scientifiques programment un algorithme pour effectuer une tâche, en lui donnant des indices positifs ou négatifs au fur et à mesure qu'il travaille sur la façon d'effectuer la tâche. Le programmeur fixe les règles pour les récompenses, mais laisse à l'algorithme le soin de décider lui-même des étapes à suivre pour maximiser la récompense, et donc accomplir la tâche [5, 49].

Nous citons dans cette partie deux méthodes d'apprentissage par renforcement :
Q-learning : L'apprentissage par Q-learning est une forme d'apprentissage par renforcement sans modèle. Il peut également être considéré comme une méthode de programmation dynamique (PD) asynchrone.

Il permet aux agents d'apprendre à agir de manière optimale dans des domaines markoviens en expérimentant les conséquences de leurs actions, sans qu'ils aient à construire des cartes de ces domaines [110].
L'apprentissage par différence temporelle : L'apprentissage par différence temporelle (TDL), est une technique d'apprentissage non supervisée qui est très couramment utilisée dans l'apprentissage par renforcement dans le but de prédire la récompense totale attendue à l'avenir.

Il peut être utilisé pour prédire d'autres quantités. C'est essentiellement un moyen d'apprendre à prédire une quantité qui dépend des valeurs futures d'un signal donné [103].

2.3 Le Culstering K-means

2.3.1 Définition

Le clustering K-moyenne ou K-means a été proposé en premier lieu par J.MacQuenn [65]. C'est l'un des algorithmes de clustering (de partitionnement) les plus connus et les plus utilisés, c'est l'algorithme qu'on va l'utiliser dans notre travail.

Il représente la plus basique forme des algorithmes de partitionnement basé sur les centroïdes. Cet algorithme vise à réduire la distance entre les points de données et leurs centroïdes au sein d'un cluster.

Il est considéré parmi les algorithmes de clustering les plus simples de l'apprentissage automatique qui peut être utilisé pour reconnaître automatiquement des groupes d'objets similaires dans la base de données (L'algorithme segmente les données (objets) en K groupes ou clusters prédéfini, spécifié par l'utilisateur de façon à ce que les données similaires soient dans le même groupe) [104].

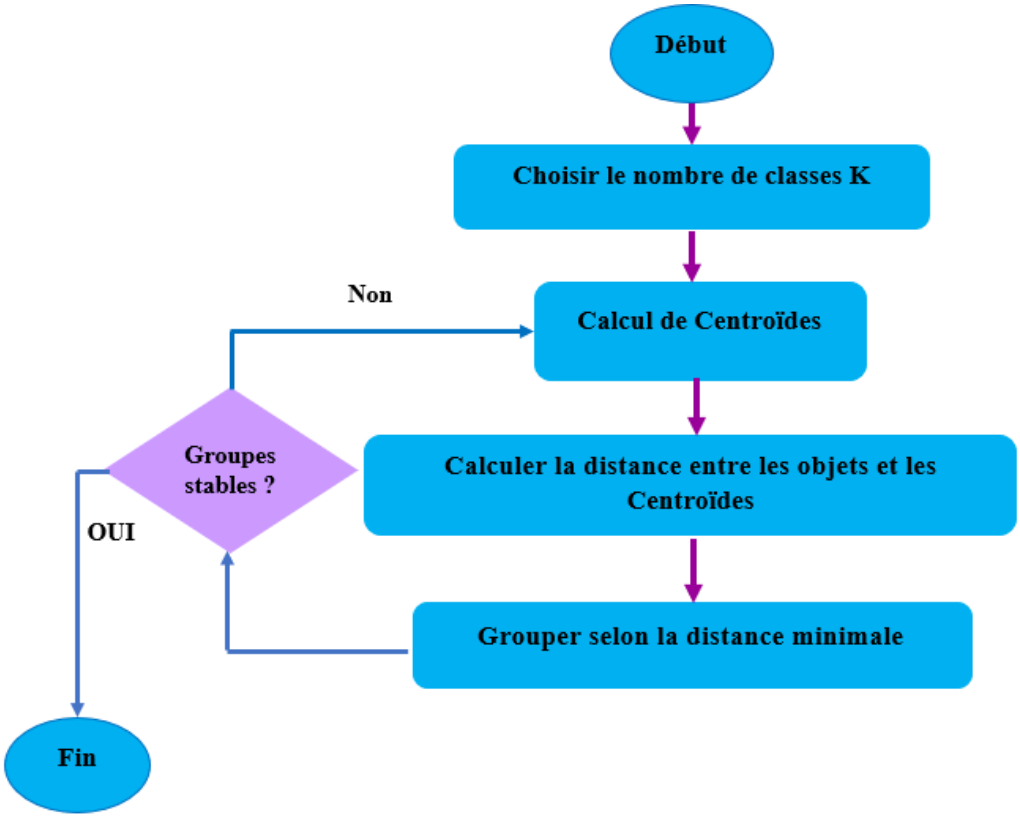

Fig. 2.3 : Organigramme de l'algorithme K-means [70].

Il est sans aucun doute la méthode de partitionnement la plus connue et la plus utilisée dans divers domaines d'application scientifiques et industrielles (Il peut être utilisé par exemple pour la détection et le filtrage du spam, l'identification des fausses nouvelles, etc.). Ce succès est dû au fait que cet algorithme présente un rapport cout/efficacité avantageux [11].

L'algorithme K-means est rapide, facile à implémenter et peut traiter de grandes

quantités de données.

La première étape importante de l'algorithme consiste à choisir (sélectionner) aléatoirement k objets qui représentent les centroïdes initiaux. Ensuite, l'algorithme continue à lire chaque objet de l'ensemble des données et l'affecte au cluster le plus proche. Il existe plusieurs méthodes pour mesurer la distance entre l'objet et le centroïde, mais la plus populaire est la distance euclidienne. Les centroïdes des clusters nouvellement formés sont toujours recalculés. Ce processus est itéré jusqu'à ce que plus aucun changement ne soit effectué [104, 3].

2.3.2 Les étapes de K-Means

L'algorithme k-Means est expliqué dans le pseudo-code suivant [104, 112, 50] :

* **Étape 1 :** Choisir le nombre de clusters (k).
* **Étape 2 :** Choisir au hasard k points et les définir comme centroïde (Sélectionnez K points aléatoires dans les données en tant que centroïdes : Ensuite, nous sélectionnons au hasard le centroïde pour chaque cluster).
* **Étape 3 :** Calculer la distance entre chaque objet et tous les centroïdes (en utilisant la méthode euclidienne par exemple).
* **Étape 4 :** Attribuer (Affecter) chaque objet au centroïde du cluster le plus proche.
* **Étape 5 :** Recalculez les centroïdes des clusters nouvellement formés. Le calcul des centroïdes pour les clusters est effectué en prenant la MOYENNE de tous les points de données qui appartiennent à chaque cluster.
* **Étape 6:** Répétez les étapes 3 à 5 jusqu'à la convergence de l'algorithme ou jusqu'à atteindre un nombre maximal d'itérations. Après quelques itérations, l'algorithme trouve un découpage stable du jeu de données : on dit que l'algorithme a convergé.

La figure 2.4 illustre l'algorithme k-means sur un ensemble de quatre points a, b, c et d, qui doivent être classés en 2 classes (clusters). On remarque sur cet exemple que bien qu'à l'initialisation les centres de classes sont mal repartis, l'algorithme a convergé en retrouvant les "vraies" classes.

(1) On dispose de 4 points à classer en deux classes.

(2) À l'initialisation, deux de ces points sont choisis aléatoirement comme centre de classe.

(3) Deux classes sont créées en regroupant les autres points en fonction du centre de classe le plus proche.

(4) On définit les nouveaux centres de classe comme étant le barycentre des classes nouvellement créées.

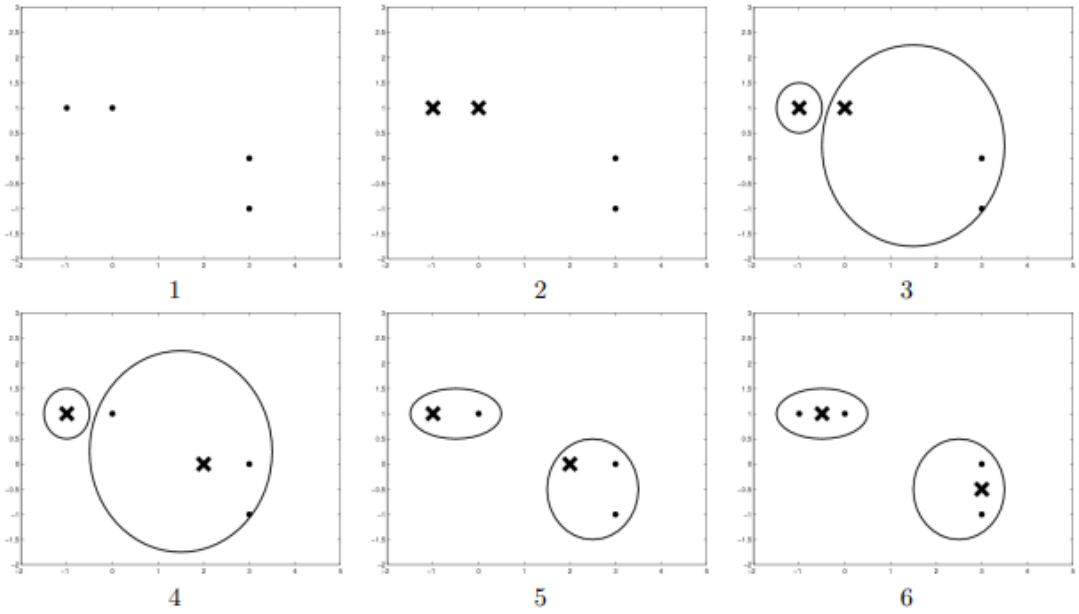

FIG. 2.4 : Illustration de l'algorithme K-means [56].

(5) On regroupe à nouveau les points.

(6) On définit les nouveaux centres de classes. À l'étape suivante rien ne change, l'algorithme s'arrête.

2.3.3 Critères d'arrêt pour le clustering K-Means

Il existe essentiellement trois critères d'arrêt qui peuvent être adoptés pour arrêter l'algorithme K-means [10] :

1. **Les centroïdes des amas nouvellement formés ne changent pas :** Nous pouvons arrêter l'algorithme si les centroïdes des clusters nouvellement formés ne changent pas. Même après plusieurs itérations, si nous obtenons les mêmes centroïdes pour tous les clusters, nous pouvons dire que l'algorithme n'apprend aucun nouveau modèle et c'est un signe pour arrêter l'apprentissage ;
2. **Les points restent dans le même cluster :** Un autre signe clair que nous devrions arrêter le processus d'entraînement si les points restent dans le même cluster même après avoir entraîné l'algorithme pour plusieurs itérations ;
3. **Le nombre maximum d'itérations est atteint :** Enfin, on peut arrêter l'entraînement si le nombre maximum d'itérations est atteint. Supposons que nous ayons défini le nombre d'itérations sur 100. Le processus se répétera pendant 100 itérations avant de s'arrêter.

2.3.4 Avantages de l'algorithme K-Means

On peut citer les avantages suivants [11, 112] :

- Il est simple, facile à comprendre et à implémenter.
- Il est applicable à des données de grandes tailles : K-means convient à un grand nombre d'ensembles de données.
- Insensible à l'ordre des données.
- Un objet peut être affecté à une classe au cours d'une itération puis changer de classe à l'itération suivante, ce qui n'est pas possible avec la classification ascendante hiérarchique pour laquelle une affectation est irréversible.
- il a un faible coût de calcul (rapide) : Comparée à l'utilisation d'autres méthodes de classification, une technique de classification k-means est rapide et efficace en termes de coût de calcul, en effet sa complexité est $(O(i * K * n * m))$ avec i le nombre d'itérations, k le nombre de clusters, n le nombre d'observations et m la dimension de l'espace de définition des observations.

2.3.5 Inconvénients de l'algorithme K-Means

Parmi les inconvénients majeurs de l'algorithme k-means, on peut citer les points suivants [112, 50] :

- Le nombre de clusters doit être fixé à l'avance.
- Les centres des clusters, mis à part des centres initiaux, sont des objets inexistants puisqu'ils correspondent à des moyennes calculées sur un sous-ensemble d'observations à chaque itération.
- Le résultat final dépend fortement du choix des centroides initiaux.
- Le nombre de classes est un paramètre de l'algorithme. Un bon choix du nombre k est nécessaire, car un mauvais choix de k produit de mauvais résultats (Une forte influence des valeurs aberrantes sur les résultats).
- Il n'est pas applicable aux données non numériques.
- Il est difficile de prévoir les valeurs k ou le nombre de clusters. Il est également difficile de comparer la qualité des clusters produits.

2.4 L'apprentissage automatique et la détection de communautés

Aujourd'hui, un grand nombre d'algorithmes de l'apprentissage automatique et surtout les algorithmes de l'apprentissage automatique non supervisés ont été utilisé pour la détection de communautés.

Li et Al, 2016 [58] ont proposé un algorithme amélioré de détection de communauté basé sur l'Analyse en Composante Principal (PCA) et l'algorithme k-means. Les résultats de la simulation montrent que l'algorithme proposé peut détecter les communautés avec plus de précision dans les réseaux complexes.

Gujral et Al, 2019 [36] ont proposé un algorithme d'apprentissage automatique efficace de détection de communautés représenté par HACD (Hierarchical Agglomerative Community Detection), qui combine l'information locale d'un graphe avec la propagation de l'appartenance, et ils ont montré l'efficacité de cet algorithme dans la détection de communautés.

Huan Li, Ruisheng Zhang, Zhili Zhao et Xin Liu,2021 [57] ont présenté et évalué une nouvelle perspective pour la détection de communautés basé sur un modèle d'apprentissage automatique , représenté par un algorithme amélioré de propagation des étiquettes nommé LPA-MNI (en combinant la fonction de modularité et l'importance du nœud avec la LPA d'origine). Ils ont prouvé que cet algorithme a une meilleure précision, une modularité plus élevée et un nombre de communautés plus raisonnable par rapport aux autres algorithmes de comparaison.

Aftab et Al, 2021 [2] ont proposé un autre algorithme d'apprentissage automatique qui permet la détection de communautés dans les réseaux sociaux. Ils ont proposé un cadre de regroupement basé sur la communauté qui est utilisé pour identifier les utilisateurs ayant des intérêts similaires. Pour atteindre cet objectif, ils ont proposé un cadre hybride avec une combinaison de minibatch K-means et DBSCAN (l'algorithme Hybrid DBSCAN). Ce cadre s'adapte bien à la taille de l'ensemble de données.

2.4.1 Les traveaux de détection des communautés avec K-Means

Malgré que le problème de la détection de communauté dans les réseaux est un sujet relativement récent, mais il a très rapidement conduit à une grande quantité de travaux, nous citons parmi eux :

Le travail de Jinfang Sheng et al [102] qui a été fait en 2020, où les chercheurs ont proposés une nouvelle approche de détection de communauté basée sur l'attraction des nœuds internes, appelée IACD (Internode Attraction Community Detection en anglais). l'idée principale de cet algorithme consiste en trois étapes : représentation de l'importance des nœuds, sélection de paires de nœuds attractifs et division de la communauté.

Tout d'abord, ils ont calculé tour à tour les influences IF (Node Influence) et GIF (Global Influence) de chaque nœud. Ensuite, ils calculent le nœud le plus attractif de chaque nœud pour former des paires de nœuds attractifs. Enfin, faire le tableau bidimensionnel et sortir le résultat.
Cet algorithme part de la perspective des nœuds importants du réseau complexe et se réfère à la relation gravitationnelle entre deux objets en physique pour représenter les forces entre les nœuds dans l'ensemble de données du réseau, puis effectue la détection de communauté. Grâce à des expériences sur un grand nombre d'ensembles de données du monde réel et de réseaux synthétiques, il est démontré que l'algorithme IACD peut

rapidement et précisément diviser la structure de la communauté, et qu'il est supérieur à certains algorithmes classiques et à des algorithmes récemment proposés.

Mustafa Hajij, Eyad Said et Robert [37] ont utilisé le vecteur PageRank comme une mesure de centralité pour trouver les centroides initiaux sur un graphe afin de les utiliser comme un nombre prédéfinie k dans l'algorithme de clustering k-means. Ils ont montré que cet algorithme est efficace et présente plusieurs avantages principaux. Premièrement, le vecteur PageRank peut être défini pour des graphes dirigés et non dirigés. Deuxièmement, le fait que le vecteur PageRank ait été conçu pour être calculé sur des graphes massifs apporte une vitesse supplémentaire. Troisièmement, cet algorithme peut être facilement généralisé aux espaces métriques, ce qui le rend applicable à d'autres domaines. En plus, c'est un algorithme simple et façile.

Huan Li et al. [57] ont proposé un algorithme amélioré qui est l'algorithme LPA-MNI pour la détection de la structure communautaire dans des réseaux complexes. L'idée de base de LPA-MNI est de combiner la modularité et l'importance des nœuds pour surmonter le caractère aléatoire du LPA. Tout d'abord, LPA-MNI utilise des procédures d'optimisation de la modularité pour identifier les communautés initiales. Deuxièmement, tous les nœuds d'une communauté initiale se voient attribuer la même étiquette. Par la suite, dans les processus itératifs de propagation des étiquettes, LPA-MNI détermine l'ordre des nœuds de mise à jour des étiquettes en fonction de l'ordre décroissant de l'importance des nœuds. Lorsque plusieurs étiquettes sont contenues par le nombre maximal de nœuds, LPA-MNI calcule l'importance de chaque nœud, puis sélectionne l'étiquette du nœud le plus influent à mettre à jour.

Pour mesurer les performances de l'algorithm LPA-MNI, 12 réseaux du monde réel (Karate, Dolphins, Football, Riskmap, Lesmis, Jazz, PolBlogs, Yeast, Ca-Hep, Astro-ph, Cond-mat, Cond-mat2005) et deux types de réseaux synthétiques sont utilisés et les résultats sont comparés à d'autres méthodes de pointe, à savoir Fastgreedy, LPA, Leading eigenvector, Walktrap , un algorithme LPA amélioré NIBLPA et EdMot. Les résultats expérimentaux montrent que l'algorithme LPA-MNI peut obtenir des résultats plus proches de la partition réelle dans les réseaux avec une structure communautaire connue. De plus, LPA-MNI peut atteindre une modularité plus élevée dans les réseaux dont la structure communautaire est inconnue. En outre, les comparaisons indiquent que LPA-MNI a une meilleure stabilité que LPA et peut efficacement surmonter le caractère aléatoire et l'inexactitude de l'algorithme LPA.

Yan Yuan et al.[115] ont basé sur la maximisation de l'influence des noeuds pour sélectionner le nœud le plus influent parmi tous les nœuds d'un réseau social comme nœud actif initial et d'affecter le nombre maximal de nœuds.

Dans cette étude, un algorithme de maximisation de l'influence basée sur la détection communautaire est proposé qui est l'algorithme EdgeBetweenness. Cet algorithme utilise l'algorithme K-means pour diviser la communauté et selon la modularité, le résultat optimal de segmentation communautaire est choisi. Après avoir divisé la communauté, ils ont calculé l'intermédiarité de l'arête (edge betweenness) de chaque communauté, certains nœuds sont sélectionnés comme nœuds importants. Les nœuds importants de chaque

communauté constituent l'ensemble des nœuds de départ utilisés dans l'algorithme de maximisation de l'influence. Enfin, le modèle IC (independent cascade) est utilisé pour les simulations afin de maximiser la propagation de l'influence. Les résultats expérimentaux montrent que l'influence de l'algorithme est meilleure que celle de certains algorithmes classiques, et la complexité de l'algorithme est faible, et il est adapté aux réseaux complexes à grande échelle.

Alexandre Vilcek [109] a proposé une nouvelle approche pour effectuer le clustering de graphes pour la détection de communauté de réseau qui est l' algorithme Deep K-Means, cet algorithme est entièrement basée sur K-Means.

Il a proposé dans cette approche de remplacer l'étape de décomposition des vecteurs propres dans le clustering spectral, par un pipeline d'auto-encodeur multicouche implémenté en utilisant uniquement l'algorithme de clustering K-Means de manière récursive, ce qui facilite la tâche de regroupement des données en communautés.

Il a montré à travers des expériences utilisant plusieurs ensembles de données de tailles variées que l'algorithme proposé surpasse le clustering spectral traditionnel en termes de précision mesurée par l'information mutuelle normalisée entre les communautés découvertes par l'algorithme et les communautés de vérité de terrain associées aux ensembles de données respectifs.

L'inconvénient de l'approche proposée est son temps d'exécution, qui augmente plus rapidement que le clustering spectral à mesure que les jeux de données augmentent. La manière dont l'algorithme a été implémenté nécessite la connaissance préalable du nombre de communautés à trouver. Ce n'est pas un problème pour les réseaux avec des communautés de vérité de terrain, comme ceux testés dans ce travail, mais ce n'est pas le cas pour la majorité des problèmes pratiques. Il serait intéressant d'incorporer une technique pour choisir automatiquement le nombre de communautés, comme l'optimisation de la fonction de modularité.

2.5 Conclusion

Dans ce chapitre, on a défini l'apprentissage automatique, ses différents types ainsi que des définitions de certains types , on a détailler l'algorithmes que nous allons utiliser dans notre travail qui est l'algorithme K-Means, ainsi que des travaux connexes sur la détection des communautés.

Le chapitre suivant est consacré à la conception de l'approche proposée.

Chapitre 3

Conception

3.1 Introduction

Nous proposons dans ce chapitre une approche basée sur les techniques d'apprentissage automatique pour détecter les communautés disjointes dans les réseaux sociaux qui est un problème de classification non supervisé sur les graphes. Notre proposition est fortement lié à l'influence des sommets, que nous avons choisir selon le produit de la densité et le degré des nœuds, nous utilisons les noeuds les plus influencés comme des centres initiaux de l'algorithme de clustering K-means ; le choix du K se fait selon la modularité maximale, la méthode proposée est applicable sur des réseaux non orientés et non pondérés. ce chapitre est organisé comme suit : nous présentons d'abord les objectifs de notre système, ensuite, nous décrivons l'architecture du système, les étapes de notre conception, ainsi que les algorithmes proposés, nous achevons ce chapitre par une conclusion.

3.2 Objectifs du système

L'objectif principal de ce travail est de réaliser un système de détection de communautés disjointes dans les réseaux sociaux et complexes en se basant sur la méthode d'apprentissage automatique k-means avec des centroïdes Présélectionnés selon le degré et la densité des noeuds afin d'atteindre les principaux objectifs :

* Détection efficace et rapide des communautés.
* Réduire le nombre d'itération dans le processus de clustering k-means et éviter les trop courtes distances entre les centroïdes initiaux.
* Choisir le nombre de classes k de k-means en fonction de la meilleure modularité.
* Eviter le problème de la sélection aléatoire des centres de clusters initiaux dans les algorithmes de clustering k-means conventionnels, de sorte que les nœuds isolés ne seront pas sélectionnés comme centres de clusters initiaux.
* Faire une méthode simple, facile à implémenter , et applicable aux grands réseaux.

3.3 Architecture du système

Notre application se compose de trois parties principales :

1. La génération des données réels et artificiels.
2. La précision des centroïdes initiaux.
3. L'application de l'algorithme k-means itérativement, en augmentant le k à chaque fois, et on calcule la modularité pour chaque k, si la modularité est déminuer ; on arrète et on prend la détection précédante.

L'architecture globale de notre système est présentée dans le schéma suivant (voir 3.1) :

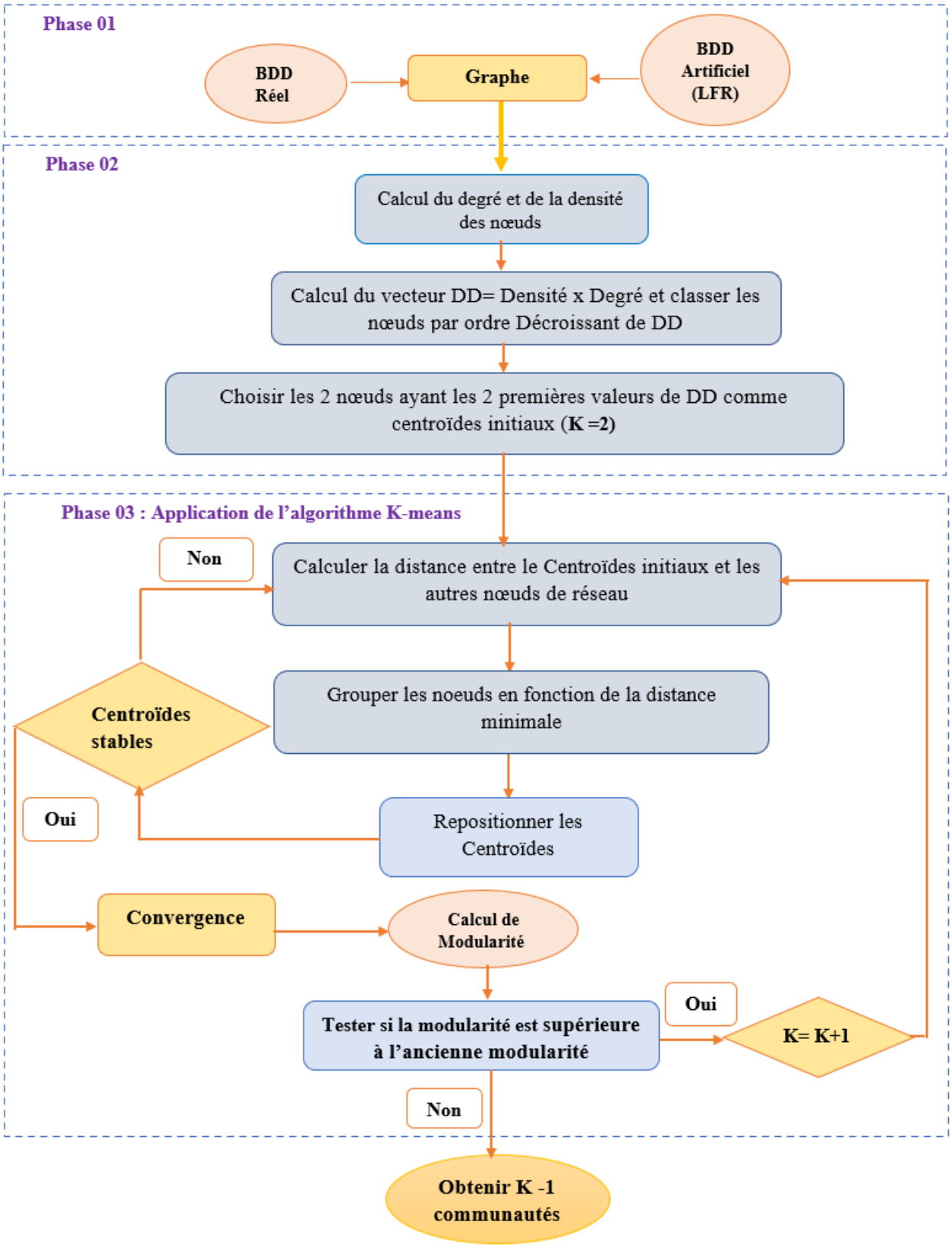

FIG. 3.1 : Architecture générale du système.

3.4 Conception détaillée de l'approche proposée

Notre approche est basée sur le produit de densité et le degré des nœuds, l'algorithme conventionnel de clustering k-means et le critère de meilleure modularité. Dans cet algorithme, les centres initiaux des clusters sont sélectionnés sur la base d'une combinaison du degré, de la densité, et de produit de ces deux derniers afin d'éviter la sélection de nœuds isolés, ce qui améliore l'efficacité de la détection des communautés.

Notre algorithme se déroule en trois phases :

3.4.1 La première phase

Cette phase consiste à importer une base de données réelle ou générer une base de données artificielle (LFR_Benchmark) et la transformer en graphe.

3.4.2 La deuxième phase

Cette phase se résume en trois étapes :

1. Calculer le degré et la densité des noeuds.
2. Calculer le produit de la densité et du degré des nœuds (DD), et classer les noeuds du réseau par ordre décroissant du produit de la densité et du degré des nœuds (CDD).
3. Sélectionner les deux premiers éléments de CDD comme des noeuds initiaux dans l'algorithme de clustering k-means (CC).

3.4.3 La troisième phase

Cette phase se déroule en sept étapes :

1. Calculer la distance euclidienne du vecteur de similarité de Jaccard des nœuds aux centroïdes CC.
2. Classer les nœuds dans le cluster dont le centre de clustering a la distance la plus courte par rapport à ce nœud. De cette façon, K clusters (Cluster (1), Cluster (2), ... , Cluster (K)) sont générés.
3. Recalculer le centre de clustering de chaque cluster (j) et le définir comme un nouveau centre de clustering (Cj).
4. Si les centres de clustering restent inchanger, l'itération est terminée, sinon retournez à l'étape 1 de la troisième phase.
5. calculer la modularité du résultat finale selon l'équation 1.10.

6. Si cette modularité est supérieur à la précédente, on incrémente k de 1 et mettre en CC les (k+1) premiers valeurs de CDD comme des centroïdes et revenir à l'étape 1; sinon l'algorithme se termine et on choisit les (k-1) clusters précédents.

7. Exporter les (K-1) communautés (Com (1), Com (2), ... , Com (K-1)).

3.5 Algorithme de notre approche

Entrées : G = (V, E) : le graphe initial.
Sorties : C_1, C_2, C_3, ..., C_{k-1} : les communautés.

3.5.1 Algorithme de la première phase

Les fonctions

1. Reseau_real() : on peux générer un graphe G par l'importation d'un réseau réel (exemple : le réseau dauphin).

2. Reseau_artificiel(n,mu) : une fonction qui permet de générer un graphe G de réseau synthétique (LFR Benchmark) à partir des deux paramètres n et *mu*.
   n : représente le nombre de noeuds.
   mu qui représente le paramètre de mélange.

3. Affiche_graph() : une fonction qui permet d'afficher le graphe G.

Pseudo code

Algorithme 1.1 : Reseau_real

```
1. Début :
2.      Lire 'txt'.txt
3.      Retour G
4. Fin.
```

Algorithme 1.2 : Reseau_Artificiel

```
Entree : n,mu
1. Début :
2.      Generer_LFR(n,mu)
3.      Retour G
4. Fin.
```

Algorithme 1.3 : Affiche_graphe

1. Début :
2. Dessiner_graphe(G)
3. G.visible
4. Fin.

3.5.2 Algorithme de la deuxième phase

Les fonctions

1. Degre_noeud() : une fonction qui permet de générer la liste de degrés des noeuds du graphe G.

2. Densite_noeud() : une fonction qui permet de calculer la densité (voir l'équation 1.1) des noeuds du graphe G.

3. trier (l) : une fonction qui permet de trier la liste l par un ordre décroissant.

4. premier(l,n) : une fonction qui permet de prendre les n premier éléments de la liste l.

Pseudo code

Algorithme 2 : Trouver_les_centroides

Début :
1. Pour i allant de 1 à Taille(liste_des_noeuds) faire
2. D(i)=densite_noeud(i)
3. Dg(i)=degre_noeud(i)
4. DD(i)=D(i)*Dg(i)
5. Fin Pour
6. CDD=trier(DD)
8. CC=premiers(CDD,2)
9. Fin.

3.5.3 Algorithme de la troisième phase

Entrée : Vecteur CC qui contiens les k nœuds centraux de clustering.
Sortie : k, Cluster ().

Les fonctions

1. Calcul_dist(c,n1) : une fonction qui permet de calculer la distance entre les centroïdes et le reste des noeuds du graphe G.

2. grouper(n,C) : une fonction qui permet de grouper les noeuds du graphe G en fonction de la distance minimale de jacd_sim.

3. nouveaucentre() : fonction de mise à jour des centres de clusters.

4. modularite(C) : fonction de calcul de la modularité pour choisir le nombre k des centres de clusters.

Pseudo code

Algorithme 3

Début :
Entrée : cencroïde CI
1.x=premier(CI,2)
2.Tant que (x<>CID) Faire
3. Pour nœuds dans G
4. Pour j appartient à CI
5. $CDD = Calcul\_dist(j, nud)$
6. $GCD = Grouper(j, nuds)$
7. Fin Pour ;
8. Fin Pour ;
9. $CID = Calcul(nouveau\_centre)$
10.Fin Tant que ;
11.$Mod1 = modularite(G)$
12. k=2
13.Répétez
13. $Mod = Mod1$
14. $y = premier(CI, k)$
15. $Mod2 = modularite(G)$
16. $k = k + 1$
17.jusqu'à $(Mod2 < Mod)$
18.$generer(k - 1, communautes)$
19. Fin.

3.6 Un exemple illustratif

Pour montrer l'efficacité de notre approche, on utilise l'exemple suivant :

Notre exemple est un graphe avec 9 nœuds et 14 arrêtes, ce graphe est disjoint et non orienté.

Tous les schémas qu'on va utiliser au cours de ces exemples sont de notre application.

Notre exemple se déroule en trois phases.

3.6.1 La première phase

Dans cette phase, on va importer un réseau de nœuds et 14 arrêtes, et on va le transformer en graphe en utilisant la matrice d'adjacense, voir equation 1.2.

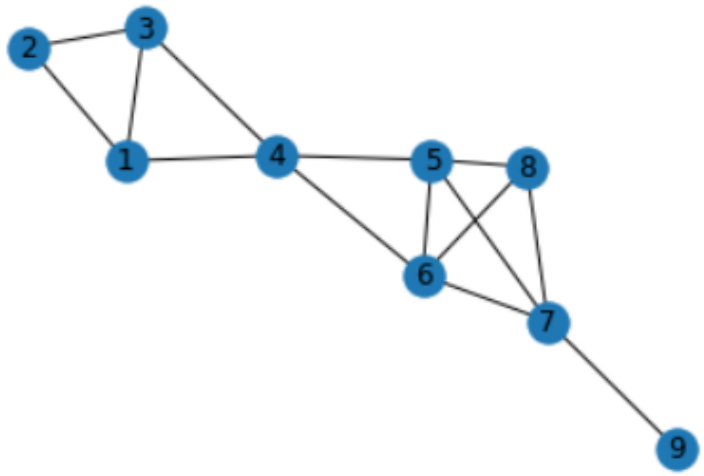

Fig. 3.2 : Représentation graphique du graphe de cet exemple.

3.6.2 La deuxième phase

Dans cette phase, notre algorithme calcule le degré(D), la densité (DEN) et le produit de degré et de densité (DD).

Le tableau 3.1 illustre les résultats de cette étape.

Nœuds	Degré	Densité	Degré x Densité
1	3	0.533	1.599
2	2	0.833	1.667
3	3	0.533	1.599
4	4	0.464	1.856
5	4	0.428	1.712
6	4	0.428	1.712
7	4	0.600	2.400
8	3	0.600	1.800
9	1	0.700	0.700

Tab. 3.1 : Résultats des degré, Densité, densité x degré.

Ensuite, Puis il ordonne tous les nœuds du réseau par ordre décroissant de DD, dans le cas où il trouve 2 nœuds avec la même valeur de DD, il classe ces nœuds par ordre croissant de leur numéro (choisi la première).

Le tableau 3.2 illustre les résultats de cette étape.

Classement	Nœuds	Degré x Densité
1	7	2.400
2	4	1.856
3	8	1.800
4	5	1.712
5	6	1.712
6	2	1.667
7	1	1.599
8	3	1.599
9	9	0.700

Tab. 3.2 : Classement des nœuds par ordre décroissant de DD.

3.6.3 La troisième phase

Dans cette phase, on initialise k à 2, puis on calcule la modularité des communautés obtenues ; puis on fait une boucle de l'incrémentation de k par 1 et on recalcule la modularité à chaque fois, et on la compare avec les modularités passées ; dès qu'on obtient la meilleure modularité, on fixe k à cette valeur et on sort de la boucle. Finalement, on obtient les communautés finales.

Avec **k=2**, on a obtenu :
Les noeuds centraux initiaux ['7', '4'].
Les noeuds centraux après l'apprentissage ['7', '3'].
Algorithme converge après 2 itirations.
modularité1= 0.34693877551020413.
communauté 1 = ['5', '6', '7', '8', '9'].
communauté 2 = ['2', '1', '3', '4'].

Avec **k=3**, on a obtenu :
Les noeuds centraux initiaux ['7', '4', '8'].
Les noeuds centraux après l'apprentissage ['7', '3', '8'].
Algorithme converge après 2 itirations.
modularité2= -0.022959183673469385 .
communauté 1 = ['5', '6', '7', '9'].
mmunauté 2 = ['2', '1', '3', '4'].
communauté 3 = ['8'].

On remarque que **(modularité1 > modularité2)**, on prend k=2 et le résultat final

est 2 communautés ; communauté 1 = ['5', '6', '7', '8', '9'] et communauté 2 = ['2', '1', '3', '4'], avec une modularité = modularité1 = 0.34693877551020413.
le schema 3.3 montre un affichage graphique des communautés détectées.

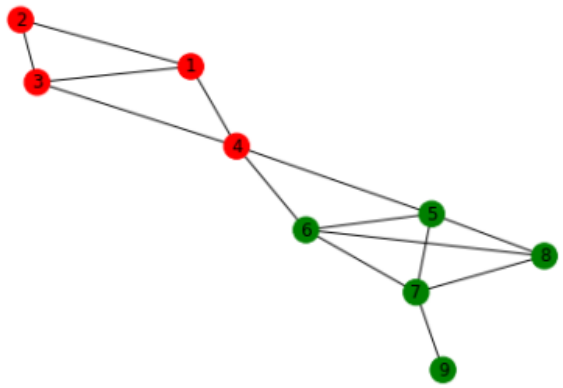

FIG. 3.3 : Représentation graphique des communautés détectées de cet exemple.

3.7 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode non supervisée de détection de communautés disjointes dans les réseaux avec un exemple illustratif détaillé où nous avons montré que notre méthode est capable de détecter facilement les communautés disjointes, aussi les communautés trouvées par notre méthode restant stables dans plusieurs exécutions sur le même réseau à cause du bon choix initial des centres de clusters initiaux.

Dans ce qui suit, on va définir l'environnement de travail matériel et logiciel, implémenter notre approche et comparer ses résultats avec les algorithmes décrits dans la section en utilisant trois mesures d'évaluations, la modularité Q, la NMI, et le ARI.

Chapitre 4

Implémentation

4.1 Introduction

Dans ce chapitre, nous présentons les outils et les langages utilisés pour implémenter la méthode proposée dans le chapitre précédent. Par la suite, nous montrons les résultats obtenus après la comparaison de notre méthode avec quelques algorithmes de pointe.

4.2 Environnement de travail

4.2.1 Environnement matériel

Toutes nos installations et nos tests seront réalisés sur un un micro-ordinateur qui possède les caractéristiques suivantes :

- Processeur : Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz ;
- Mémoire Installée (RAM) : 4 Go ;
- Type de système : système d'exploitation windows10 64 bits.
  Notre application a été développée sous un système d'exploitation Windows 10 de 64 bits où on a utilisé le langage de programmation python 3.8 sous Anaconda.

4.2.2 Environnement logiciel

Nous avons utilisé le langage de programmation Python la version 3.8.12. Python est un langage de programmation interprété, interactif, orienté objet et de haut niveau [89]. Il a été créé par Guido van Rossum et publiée pour la première fois en 1991.

* **Python est interprété :** Python est traité au moment de l'exécution par l'interpréteur. Nous n'avons pas besoin de compiler notre programme avant de l'exécuter. Ceci est similaire à PERL et PHP.
* **Python est interactif :** Nous pouvons réellement nous installer à une invite Python et interagir directement avec l'interpréteur pour écrire nos programmes.
* **Python est orienté objet :** Python prend en charge le style ou la technique de programmation orienté objet qui encapsule le code dans des objets.

L'installation de python est gratuite et facile, il suffit de le télécharger depuis le site : https://www.python.org/downloads/ (Voir figure 4.1).

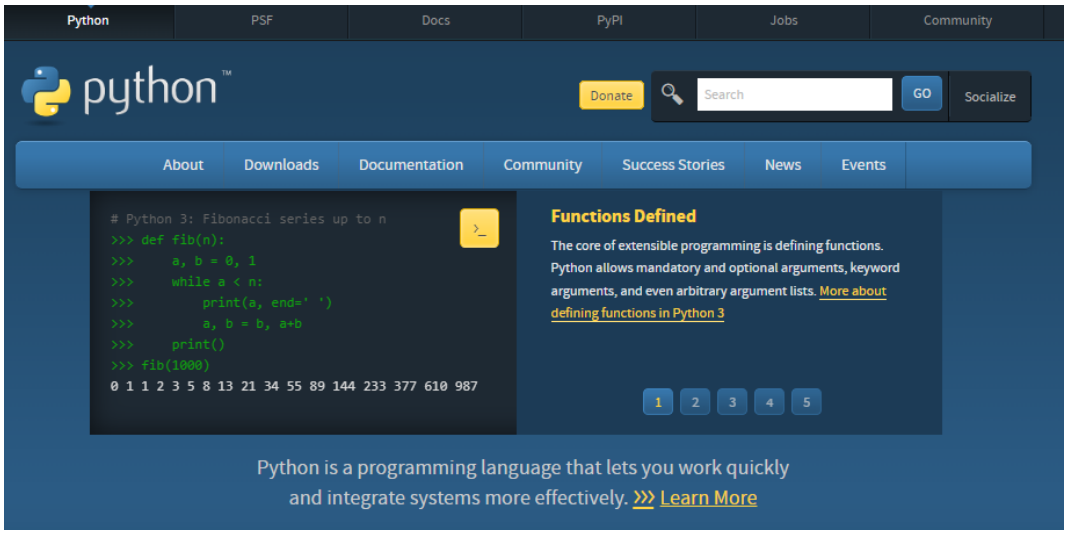

Fig. 4.1 : Le site d'installation de python

4.2.3 Plateforme et IDE

Nous avons utilisé la Plateforme Anaconda car c'est un distributeur libre et open source du langage de programmation Python appliqué au développement d'applications dédiées à la science de données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique)
Pour téléchargez et installez la 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/distribution

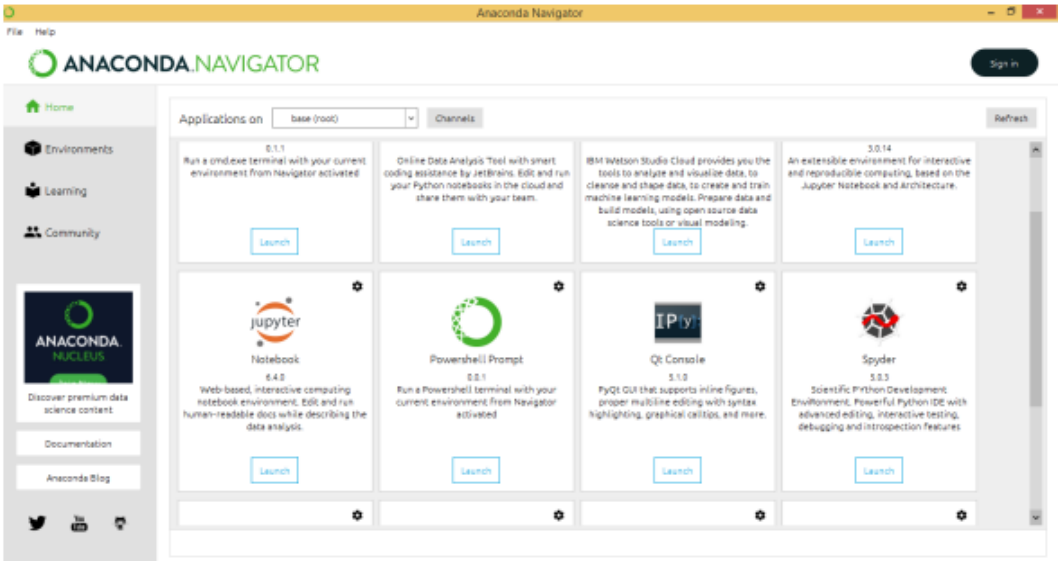

Fig. 4.2 : Anaconda Navigator

Nous avons utilisé l'environnement de développement (IDE) Spyder. C'est un environnement de développement pour Python libre et multiplateforme (Windows, Mac OS, GNU/Linux) qui contient plusieurs bibliothèques d'usage scientifique : Matplotlib, NumPy, SciPy et IPython.

La figure (4.3) présente le logo de l'environnement de développement Spyder :

FIG. 4.3 : Logo de l'environnement de développement Spyder.

4.2.4 Bibliothèques utilisées

Une bibliothèque est un ensemble de fonctions prédéfinies. Celles-ci sont regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Python est un langage de programmation très riche avec ses bibliothèques.
On a utilisé plusieurs paquets (bibliothèques) dans ce travail qui sont :

* **networkx :** c'est une bibliothèque Python pour l'étude des graphes et des réseaux, elle fournit des classes pour les objets graphiques, des générateurs pour créer des graphiques standard, des algorithmes pour analyser les réseaux résultants et certains outils de dessin de base [86].
* **Community :** c'est une bibliothèque Python utilisée pour détecter la structure de la communauté dans les graphes. cette bibliothèque implémente et visualise les algorithmes de détection des communautés [81].
* **NumPy :** est une bibliothèque Python fondamentale pour le calcul scientifique, spécialisée dans la manipulation des tableaux (array), essentiellement les vecteurs et les matrices [87].
* **math :** Ce module donne accès aux fonctions mathématiques [84].
* **tkinter :** c'est un module intégré à la bibliothèque standard de Python, permettant de créer des **interfaces graphiques**, par exemple : des fenêtres, des widgets (boutons, zones de texte, cases à cocher, …), des évènements (clavier, souris, …) [93].
* **Matplotlib :** c'est une bibliothèque complète du langage de programmation Python destinée à tracer et à visualiser des données sous formes de graphiques (elle permet de créer des visualisations statiques, animées et interactives en Python) [85].
* **Itertools :** c'est un module de Python qui fournit diverses fonctions qui fonctionnent sur des itérateurs pour produire des itérateurs complexes (il fournit des fonctions créant des itérateurs pour un bouclage efficace) [83].
* **random :** C'est un module de Python qui implémente des générateurs de nombres pseudo - aléatoires pour diverses distributions [90].

* **sklearn :** Sklearn (Scikit-learn) est la principale bibliothèque d'outils dédiés au machine learning et à la data-science dans l'univers Python [91].

* **datetime :** c'est un module de Python qui fournit des classes pour manipuler les dates et les heures [82].

* **operator :** Le module operator de python exporte un ensemble de fonctions efficaces correspondant aux opérateurs intrinsèques de Python. Par exemple, $operator.add(x, y)$ est équivalent à l'expression $x + y$ [88].

* **sys :** C'est un module Python qui donne accès à certaines variables utilisées ou maintenues par l'interpréteur et à des fonctions qui interagissent fortement avec l'interpréteur (Paramètres et fonctions spécifiques au système) [92].

Pour l'installation des packages nécessaires il suffit de taper les codes ci-dessous dans Anaconda Powershell Prompt :

NetworkX : !pip install Networkx.

Community : !pip install community.

Python-louvain : !pip install python-louvain.

Matplotlib : !pip install matplotlib.

Le téléchargement des packages nécessite l'accès à l'internet.

4.2.5 Présentation du système

Dans cette section, nous allons présenter quelques modules de notre application. La figure 4.4 ci-dessous montre l'interface principale de notre application, elle est composée de huit parties principales :

1. Importation des données réels.
2. Génération des données artificielles (synthétiques).
3. Affichage du graphe des données à traiter.
4. Application de l'algorithme de la méthode proposée sur ces données.
5. Application de l'algorithme K-means.
6. Comparaison de notre algorithme avec quelques algorithmes de détection de communautés.
7. Représentation graphique de la modularité du réseaux de karaty dans divers algorithmes.
8. Représentation graphique des modularités des réseaux LFR utilisés dans deux algorithmes (K-means et notre algorithme) (voir Figure 4.19).

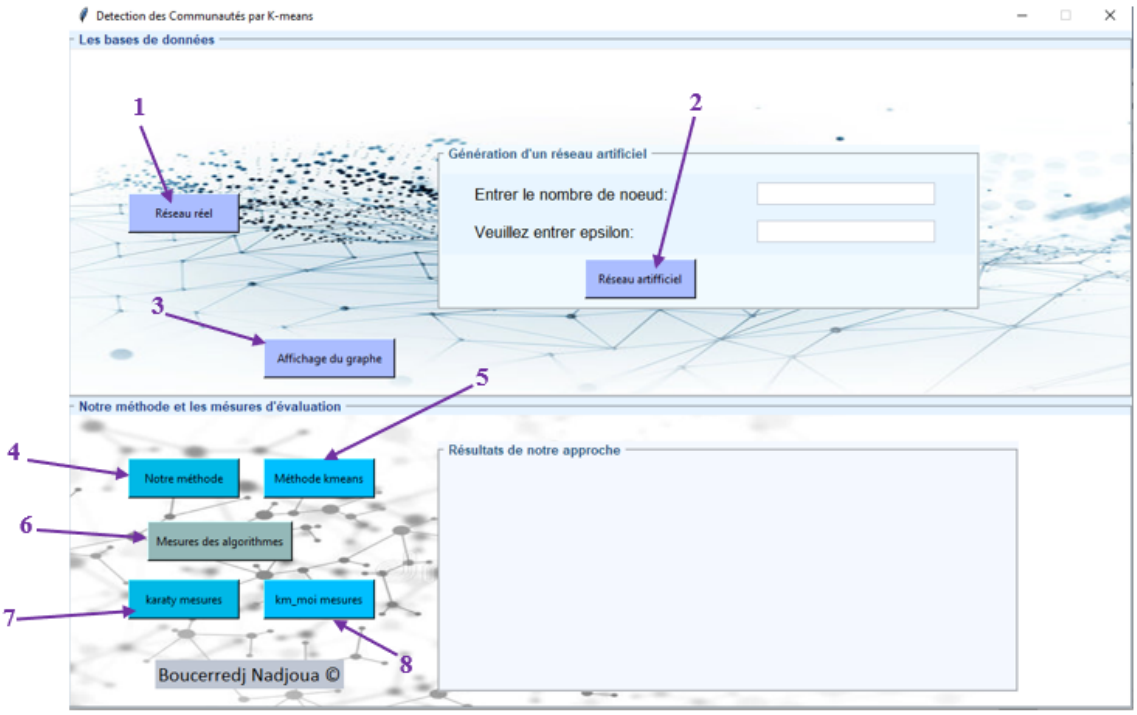

FIG. 4.4 : Interface générale de l'application.

Dans cette section, nous expliquons quelques parties.

1. importation des données réels : En cliquant sur le bouton Réseau réel une fenêtre s'affiche (voir figure 4.5), qui permet de choisir un fichier texte .txt qui représente la base de données réelle.

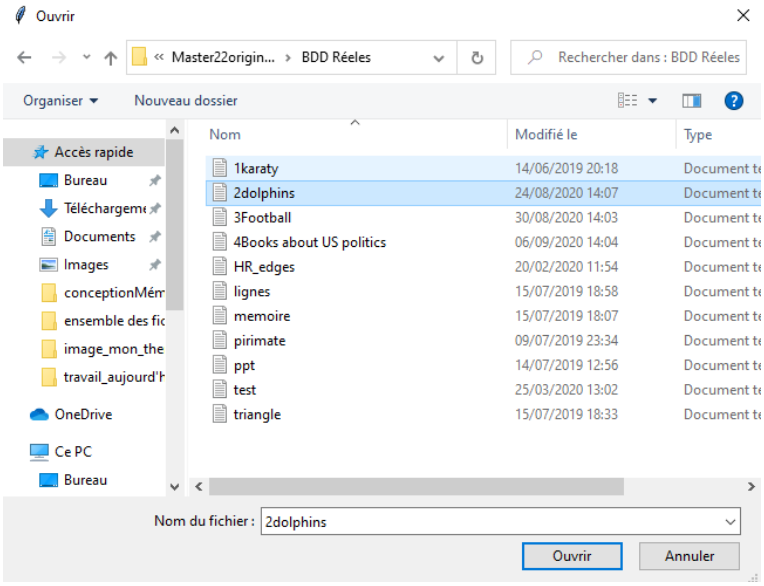

FIG. 4.5 : Importation des bases de données.

2. Génération des données artificielles (synthétiques) : L'utilisateur doit saisir d'abords

les deux paramètres N et Epsilon (voir figure 4.6), puis il clique sur le bouton Réseau Artificiel afin de générer le réseau synthétique.

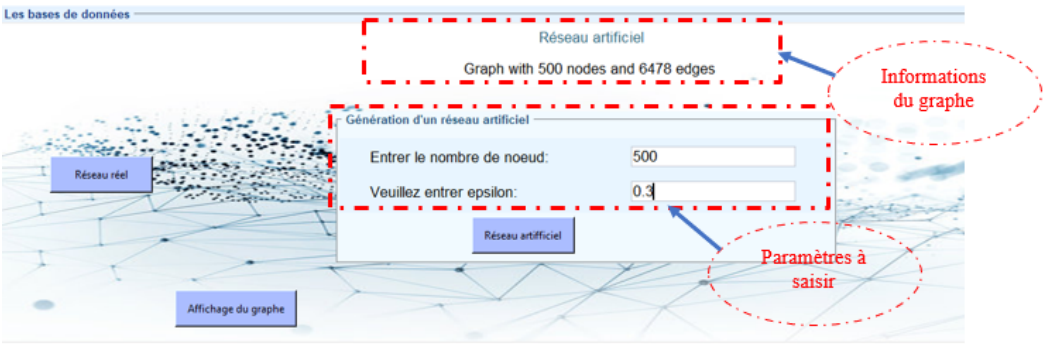

FIG. 4.6 : Importation des bases de données.

3. Affichage du graphe des données à traiter : En cliquant sur le bouton Affichage du graphe, une représentation graphique du réseau s'affiche (voir figure 4.7).

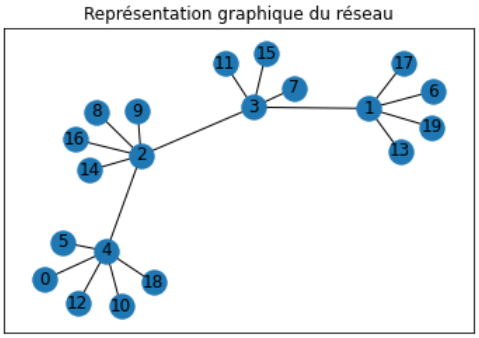

FIG. 4.7 : Représentation graphique du réseau de Pirimate.

4. Application de l'algorithme de la méthode proposée sur ces données : En cliquant sur le bouton Notre méthode, deux fenêtre s'affichent ; une donne la représentation graphique (voir figure 4.8) et l'autre donne les centroïdes initiaux et finaux ainsi que la modularité du réseau (par exemple le réseau Pirimate) après l'application de notre algorithme sur ce réseau (le réseau Pirimate, est un réseau de 20 nœuds et 19 arrêtes) (voir figure 4.9).

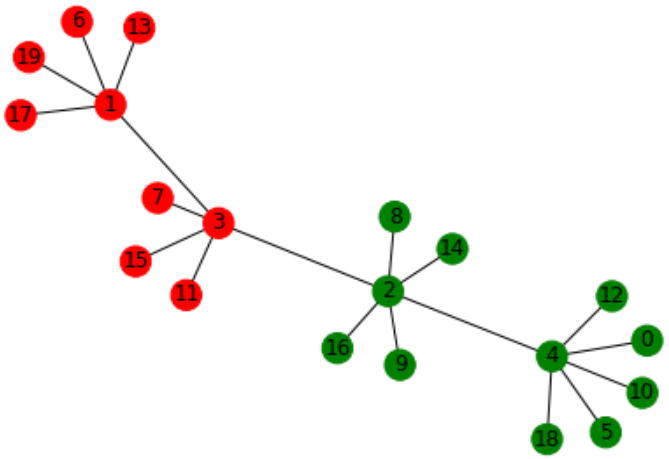

FIG. 4.8 : Représentation graphique de l'exécution de notre l'algorithme sur le réseau de Perimate.

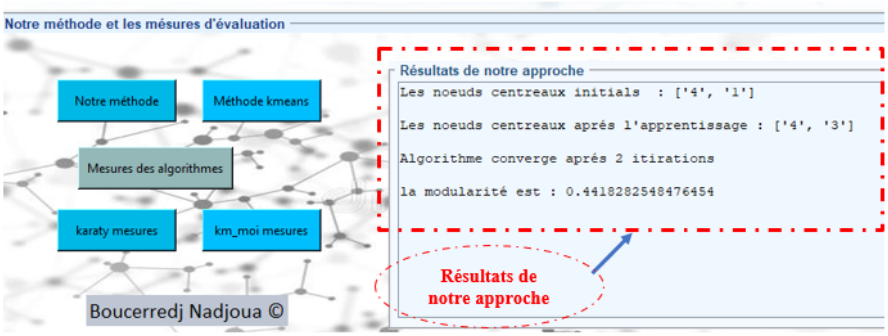

FIG. 4.9 : Résultat de l'exécution de notre l'algorithme sur le réseau de Pirimate.

5. Application de l'algorithme K-means.

6. Comparaison de notre algorithme avec quelques algorithmes de détection de communautés : En cliquant sur le bouton Notre méthode, une fenêtre s'affiche qui donne la comparaison entre le résultat de notre algorithme et les résultats des algorithmes décrits dans la section 1.3.4).

4.3 Résultats expérimentaux et analyse

Pour qu'un algorithme de détection de communautés soit considéré comme performant, il faut que les communautés qu'il trouve soient pertinentes. Dans notre travail, la performance de l'algorithme proposé a été évaluée par une mesure d'évaluation : la modularité Q (voir la section 1.5.1), en utilisant quatre réseaux du monde réel (karaty, dolphin, livres politiques, Football américain) et des réseaux LFR, et en comparant nos résultats avec ceux des algorithmes décrits dans la section (1.3.4), soit Newman [76], Label Propagation [94] et l'agorithme de Louvain [8], fast greedy [75, 19] et Infomap [96].

On a choisi le k selon le critère de maximisation de modularité, c'est à dire, on teste

plusieurs fois la valeur de k et on calcule la modularité à chaque fois, et dés qu'on obtient une valeur de modularité inférieur à la valeur ancienne, on s'arrête ici. Et ça représente le critère de fin d'itération dans l'algorithme proposé.

Par plusieurs essais, on a conclu que pour une meilleure modularité la valeur de k est 2 (voir tableau 4.1).

k	Zachary	Dauphins	Polbooks	Football
2	0.313	0.316	0.416	0.219
3	0.118	0.251	0.048	0.1279

Tab. 4.1 : valeurs de modularités des réseaux selon K .

4.3.1 Expériences sur les réseaux du monde réel

Nous avons utilisé les quatre réseaux du monde réel mentionnés ci-dessus (voir section 1.4.1) pour vérifier l'efficacité de notre algorithme.

Club de karaté de Zachary

La première comparaison est faite sur le réseau de club de karaté de Zachary [116]. C'est un réseau très populaire et très utilisé par les algorithmes de détection de communauté afin de tester leurs performances puisque sa structure de communautés est connue à l'avance. Il contient 34 nœuds et 78 arêtes. Ce réseau comporte deux communautés. La figure ci-dessous nous montre le résultat d'exécution de notre algorithme sur le réseau de Zachary. L'algorithme proposé a divisé ce réseau en deux communautés. Les deux communautés sont représentées par différentes couleurs.

Cette structure dégagée est très proche de la structure communautaire réelle, Donc nous pouvons constater que notre algorithme est performant sur le réseau du club de karaté de Zachary.

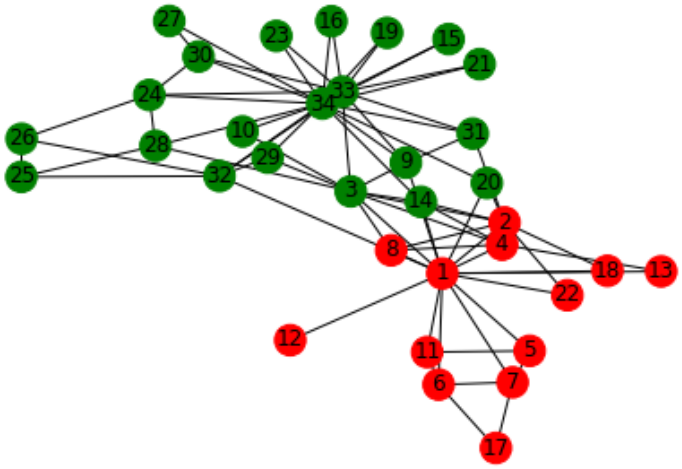

FIG. 4.10 : Structure de communautés trouvées par la méthode proposée sur le réseau de Zachary (2 communautés).

la figure 4.11 montre la Modularité du réseau de Zachary pour differents algorithmes.

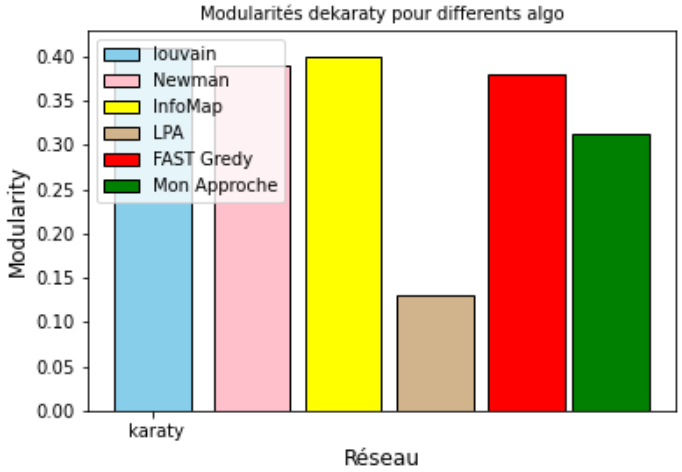

FIG. 4.11 : Modularité du réseau de Zachary pour differents algorithmes.

Les dauphins de Lusseau

Le deuxième exemple à traiter est le réseau de dauphins de Lusseau [63]. Ce réseau contient 62 nœuds et 159 liens. Il est constitué de deux communautés.

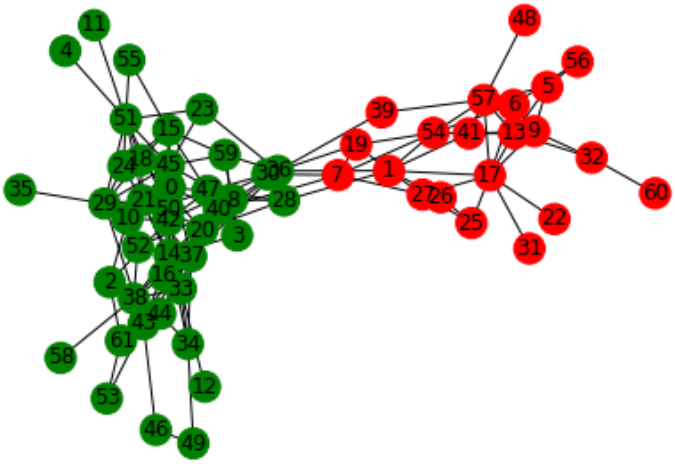

FIG. 4.12 : les communautés détectées par notre algorithme pour le réseau de dauphins de Lusseau (2 communautés).

L'algorithme proposé a divisé ce réseau en deux communautés. Les deux communautés sont représentées par différentes couleurs.

Cette structure dégagée est très proche de la structure communautaire réelle, Donc nous pouvons constater que notre algorithme est performant sur le réseau du club des dauphins de Lusseau.

la figure 4.13 montre la Modularité du réseau de Dauphins pour differents algorithmes.

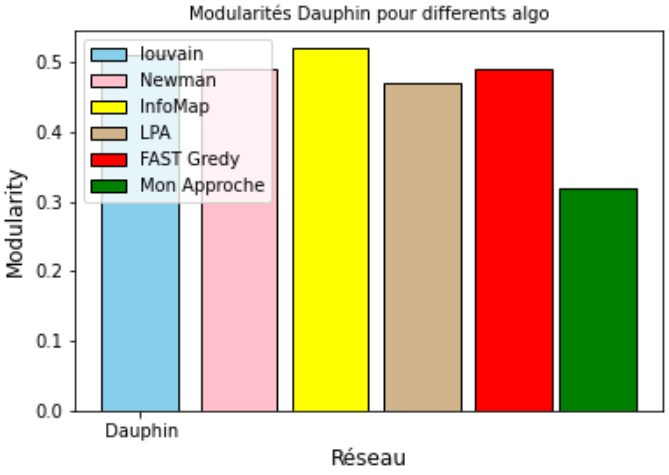

FIG. 4.13 : Modularité du réseau de Zachary pour differents algorithmes.

Les livres politiques

Le troisième exemple à traiter est le réseau de livres politiques [76, 53]. Ce réseau est constitué de 105 livres avec 441 liens. Il a initialement trois groupes de livres (libéraux, conservateurs et neutres).

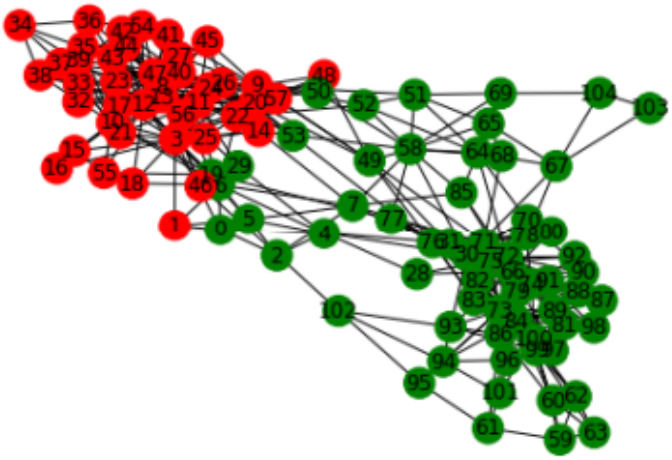

FIG. 4.14 : Structure de communautés trouvées par notre algorithme pour le réseau de livres politiques (2 communautés).

L'algorithme proposé a divisé ce réseau en deux communautés. Les deux communautés sont représentées par différentes couleurs.

Cette structure dégagée est proche de la structure communautaire réelle, Donc nous pouvons constater que notre algorithme est performant sur le réseau de livres politiques.

la figure 4.15 montre la Modularité du réseau de livres politiques pour differents algorithmes.

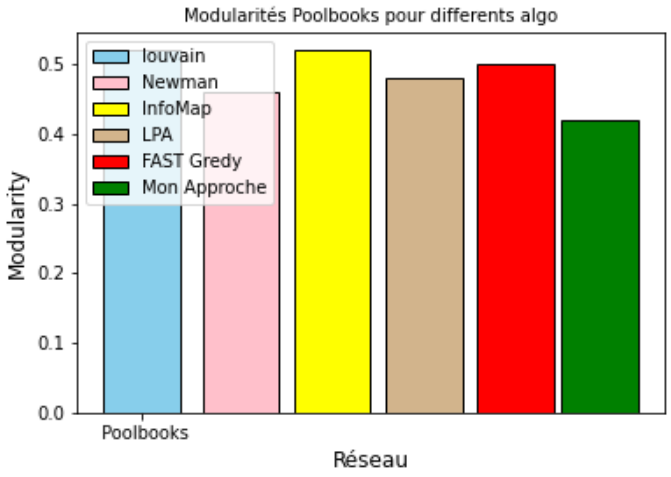

FIG. 4.15 : Modularité du réseau de Zachary pour differents algorithmes.

Football américain

L'autre exemple de réseau réel étudié dans le cadre de ce mémoire est le réseau de jeux de football américain (American football games) [79]. Ce réseau est constitué de 115 nœuds et 613 liens.

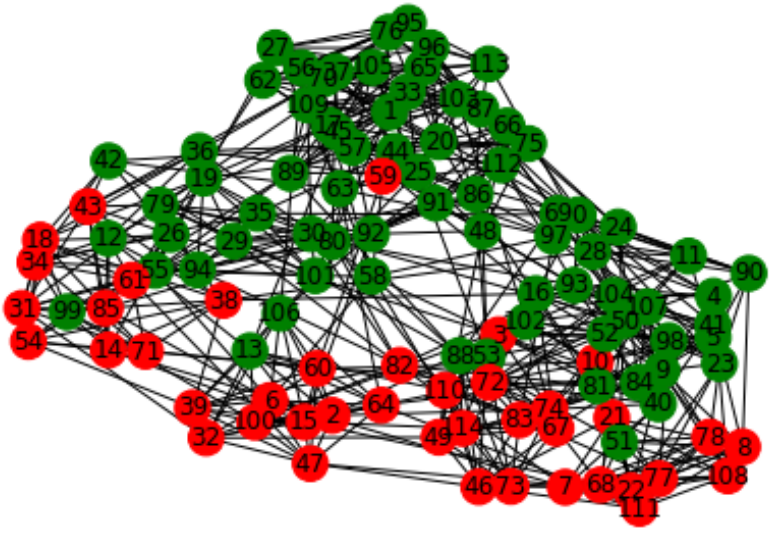

FIG. 4.16 : Structure de communautés trouvée par notre méthode pour le réseau du Football américain (2 communautés).

L'algorithme proposé a divisé ce réseau en deux communautés. Les deux communautés sont représentées par différentes couleurs.

la figure 4.17 montre la Modularité du réseau de Football pour differents algorithmes.

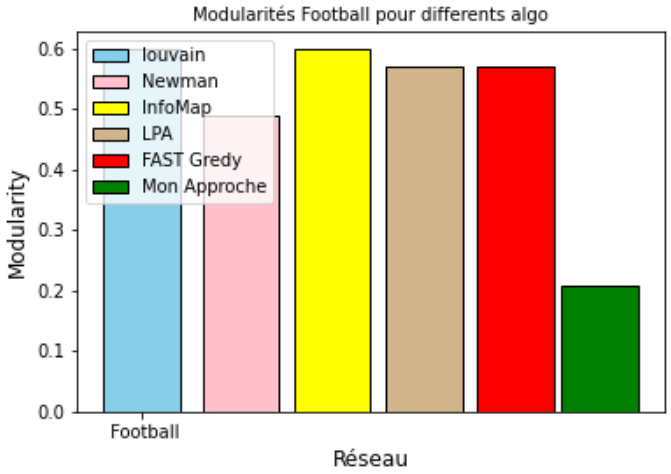

FIG. 4.17 : Modularité du réseau de Zachary pour differents algorithmes.

Nous avons comparé les résultats de notre algorithme avec celles de l'algorithme Newman [76], Label Propagation [94] et l'agorithme de Louvain [8], Fast Greedy [75, 19] et Infomap [96].

Les résultats sont présentés dans le tableau 4.2.

Algorithmes	Louvain	Newman	infomap	LPA	F.Greedy	Notre Algorithme
Karaty	0.41	0.39	0.40	0.132	0.38	0.313
dolphins	0.51	0.49	0.52	0.47	0.49	0.317
Football	0.60	0.49	0.60	0.52	0.57	0.219
polbooks	0.52	0.46	0.52	0.48	0.50	0.422

Tab. 4.2 : Valeurs de modularités des differents réseaux.

En regardant les valeurs de la modularité Q figurant dans le tableau 4.2, la méthode proposée a donné de résultats acceptables sur la plupart des réseaux (voir Figure 4.18), et elle a des résultats meilleures que l'algorithme LPA dans le réseau de karaty.

Malgré les résultats des autres algorithmes sont un peu meilleures que les résultats de notre algorithmes pour les réseaux réels (de petite taille), d'après les expériences qu'on a fait dans les algorithmes utilisés ici sur ces réseaux, on a remarqué que notre algorithme est stable, il donne les mêmes résultats après chaque exécution, par contre les autres algorithmes ne sont pas stables, ils donnent des résultats différents des Centroïdes, des communautés trouvées, de modularités, et de temps d'exécution, après chaque exécution. Ce qui rend notre algorithme plus stable que les autres algorithmes.

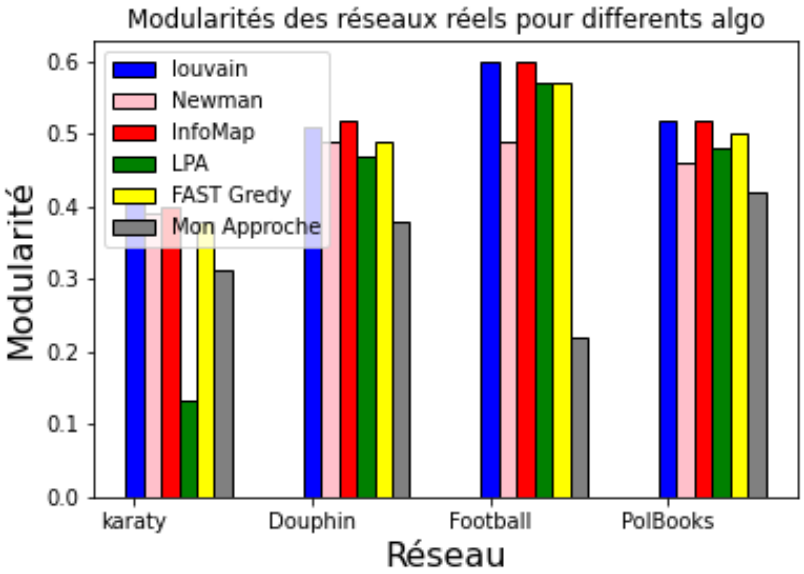

Fig. 4.18 : Modularités des réseaux réels pour differents algorithmes.

4.3.2 Expériences sur les réseaux LFR Benchmark

Nous avons testé notre algorithme sur six réseaux de référence LFR.

Le programme de génération LFR fournit un riche ensemble de paramètres à travers lesquels la topologie du réseau peut être contrôlée, y compris la taille du réseau N le degré moyen (average degree) (k) , le degré maximum $Max(k)$, la taille maximale de la communauté, $Max(c)$ et les paramètres de mélange μ. Les degrés des nœuds sont régis

par des lois de puissance avec des exposants de $tau1$ et $tau2$. Les caractéristiques des réseaux générés sont présentées dans le tableau suivant :

Réseaux	N	$tau1$	$tau2$	$Max(c)$	(k)	$Max(k)$	μ	**Nbr Liens**
LFR1	400	3	1.5	50	20	50	0.3	5580
LFR2	500	3	1.5	50	20	50	0.1	6194
LFR3	500	3	1.5	50	20	50	0.7	6561
LFR4	600	3	1.5	50	20	50	0.5	7427
LFR5	700	3	1.5	50	20	50	0.7	8639
LFR6	800	3	1.5	50	20	50	0.5	9856

Tab. 4.3 : Paramètres des réseaux LFR utilisés.

Nous avons utilisé les réseaux LFR1, LFR2 et LFR3, LFR4,LFR5,LFR6 pour tester notre algorithme et l'algorithme k-means traditionnel décrit dans la section 2.3.

On a choisi le k selon le critère de maximisation de modularité, c'est à dire, on teste plusieurs fois la valeur de k et on calcule la modularité à chaque fois, et dés qu'on obtient une valeur de modularité inférieur à la valeur ancienne, on Arrête ici. Et ça représente le critère de fin d'itération dans l'algorithme proposé.

Par plusieurs essais, on a conclu que pour une meilleure modularité la valeur de k est 2.

Comme les résultats de l'algorithme k-means sont différents à chaque fois, nous avons pris la moyenne des résultats des 10 itérations de six réseaux ci-dessus.

Les résultats sont présentés dans le tableau 4.4 et shématisés dans la Figure 4.19.

Réseaux	**LFR1**	**LFR2**	**LFR3**	**LFR4**	**LFR5**	**LFR6**
Notre Algo	0.0791	0.102	0.0294	0.0475	0.0176	0.041
K-Means	0.013	0.0786	0.0072	0.007	0.0089	0.0086

Tab. 4.4 : Valeurs de modularités des réseaux synthétiques dans les deux algorithmes (k-means et notre algorithme).

le tableau 4.4 et la figure 4.19 montrent les résultats de notre algorithme et de l'algorithme k-means sur les réseaux LFR1, LFR2 et LFR3, LFR4, LFR5, LFR6 ; les résultats de l'algorithme proposé sont meilleurs sur tous les réseaux LFR.

L'algorithme proposé était stable sur ces réseaux (LFR1, LFR2 et LFR3, LFR4, LFR5, LFR6), et il n'y a pas de différence significative dans la performance du réseau avec différents nombres de nœuds et d'échelles de communauté.

On a augmenté le parametre de mélange μ (LFR3, LFR4, LFR5, LFR6) et on a remarqué que notre algorithme reste stable, Cela signifie que notre algorithme est stable et performant dans les réseaux **dense**.

Notre algorithme converge après 2 itérations dans (LFR1, LFR3, LFR4, LFR5, LFR6), une itération dans (LFR2) mais l'algorithme k-means converge après trois itérations dans LFR2, LFR3, LFR5, LFR6 et deux itérations dans LFR1, et 3 itérations dans LFR4 donc

le temps d'exécution est minimale pour notre algorithme, cela signifie que la complexité de notre algorithme est meilleur que l'algorithme k-means traditionnel.

L'algorithme k-means n'est pas stable, à chaque exécution donne un résultat différent de Centroïdes, de nombre de communautés trouvées, de modularité, et de temps d'éxécution, par contre notre algorithme donne toujours le même résultat après chaque exécution, ce qui le rend plus stable que l'algorithme k-means traditionnel.

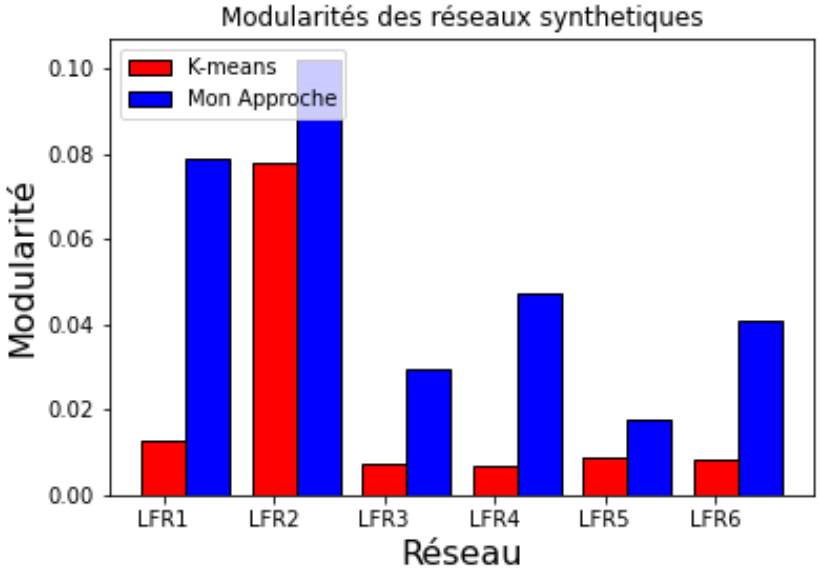

Fig. 4.19 : Représentation graphique des modularités des réseaux LFR utilisés de l'algorithme K-means et notre algorithme.

4.4 Temps d'exécution

En ce qui concerne le temps d'exécution, la méthode proposée a fait preuve de sa performance. En effet, elle est toujours rapide avec des réseaux réel qu'on a utilisé, elle est égale à 0.0126 s pour le réseau de karaty, 0.07 s pour le réseau de Daulphin, 0.25 s pour Poolbooks et 0.51 s pour Football ; donc elle est toujours rapide pour les réseaux de petite taille.

Nous avons aussi fait des tests pour les réseaux de grandes tailles, et on a comparé notre méthode avec l'algorithme K-means. Les résultats de temps d'exécution sont présentés dans le tableau 4.5 :

Réseaux	LFR1	LFR2	LFR3	LFR4	LFR5	LFR6
Notre Algo	10.314 (s)	16.99 (s)	16.77 (s)	25.03 (s)	28.60 (s)	38.25 (s)
K-Means	26.09 (s)	34.92 (s)	36.59 (s)	38.97 (s)	51.55 (s)	83.03 (s)

Tab. 4.5 : Valeurs de temps d'exécution des réseaux synthétiques en seconde.

D'après les résultats obtenus (tableau 4.5), en augmentant le nombre de nœuds (entre

400 pour LFR1 et 800 pour LFR6) et le nombre de liens (entre 5580 pour LFR1 et 9856 pour LFR6), on remarque que notre algorithme est toujours plus rapide que l'algorithme K-means dans les grands réseaux (voir Figure 4.20).

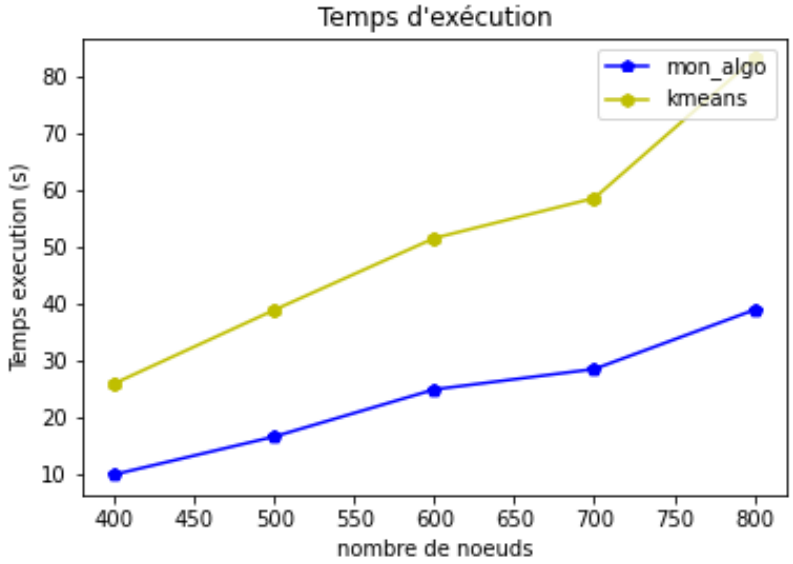

Fig. 4.20 : Représentation graphique du temps d'exécution des réseaux LFR utilisés de l'algorithme K-means et notre algorithme.

4.5 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation de notre approche, nous avons commencé par la présentation de l'environnement matériel et logiciel du travail réalisé, ensuite nous avons présenté notre implémentation.

Aprés, nous avons fait des tests sur notre méthode et nous avons comparé notre méthode avec d'autres algorithmes.

Enfin nous avons discuté sur les résultats obtenus, et le temps d'exécution.

Conclusion et perspectives

Conclusion générale

La détection de communautés est un domaine qui est encore dans une phase d'exploration, et pour lequel il faudra encore attendre quelques années avant d'arriver à un stade de maturation. Cette relative jeunesse du domaine a, d'une part, représenté un challenge et, d'autre part, a été un facteur de motivation important.

Le travail exposé dans ce mémoire s'intéresse à la détection de communauté disjointes dans les réseaux.

Dans le premier chapitre, nous avons présenté quelques définitions essentielles dans la détection des communautés, la structure communautaire, les réseaux sociaux réels et synthétiques, ainsi que quelques algorithmes de la détection de communauté.

Dans le deuxième chapitre, nous avons étudié l'apprentissage automatique, ses différents types ainsi que la définition de l'algorithme utilisé dans ce travail "K-means" , puis nous décrivons les travaux les plus récents dans le domaine de détection des communautés par K-means.

Après avoir introduit les notions principales de l'étude, dans le troisième chapitre, nous avons présenté une nouvelle approche de détection de communautés disjointes basée sur le clustering k-means qui se concentre principalement sur la densité et le degré des nœuds.

Dans le dernier chapitre, on a implémenté l'algorithme proposé, et d'après nos expériences, nous avons conclu que notre algorithme peut bien sélectionner les centres de regroupement, empêchant ainsi la sélection de centres de regroupement initiaux qui sont trop proches les uns des autres, et réduire les temps d'itération dans le processus de clustering. En d'autres termes, nous avons proposé une solution au problème d'initialisation des centroïdes, cette méthode est applicable avec tout algorithme nécessitant des valeurs de départ des centroïdes.

L'algorithme proposé a été testé sur des réseaux du monde réel, ainsi que des réseaux artificiels, et il a montré des résultats satisfaisants.

C'est un algorithme simple, facile à comprendre. Il est performant et il fonctionne bien sur les réseaux du monde réel et sur différents types de réseaux synthétiques. Les résultats de partition sont acceptables, et les communautés trouvées restent **stables** dans plusieurs exécutions sur le même réseau, la mise en œuvre de la méthode proposée est très facile.

L'algorithme proposé est rapide, et il présente une faible complexité temporelle et un résultat stable.

Perspectives

Pour rendre notre approche encore plus compétitive, certaines améliorations peuvent être réalisées dans le futur.

Notre première perspective consiste à appliquer le schéma proposé à d'autres algorithmes de Clustering autres que l'algorithme de Clustering K-means, afin d'observer les améliorations qu'on peut tirer à partir de notre approche et de profiter de ses avantages.

Notre deuxième perspective concerne à étendre notre approche afin de supporter les communautés pondérées.

Une autre idée consisterait à rendre notre approche applicable dans des graphes hétérogènes de différents types de nœuds et de liens.

Finalement, il est souhaitable d'introduire la notion de chevauchement de communautés. Cela permet à notre approche de supporter le partitionnement flou où un nœud peut appartenir à plusieurs communautés.

Bibliographie

[1] NEDIOUI MED ABDELHAMID. “Fouille et apprentissage automatique dans les réseaux sociaux dynamique”. Mém. de mast. 2015.

[2] Huma Aftab et al. “Hybrid DBSCAN based Community Detection for Edge Caching in Social Media Applications”. In : juill. 2021. doi : 10.1109/IWCMC51323.2021.9498609.

[3] Mohiuddin Ahmed, Raihan Seraj et Syed Mohammed Shamsul Islam. “The k-means algorithm : A comprehensive survey and performance evaluation”. In : *Electronics* 9.8 (2020), p. 1295.

[4] Mohamed Alloghani et al. “A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science”. In : jan. 2020, p. 3-21. isbn : 978-3-030-22474-5. doi : 10.1007/978-3-030-22475-2_1.

[5] Alex M Andrew. “REINFORCEMENT LEARNING : AN INTRODUCTION by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning series, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii+ 322 pp, ISBN 0-262-19398-1,(hardback,£ 31.95).” In : *Robotica* 17.2 (1999), p. 229-235.

[6] Michael J Barber et John W Clark. “Detecting network communities by propagating labels under constraints”. In : *Physical Review E* 80.2 (2009), p. 026129.

[7] Safaa Belloussaief. “Désintégration d’un réseau social au sein de ses communautés”. Thèse de doct. Université du Québec en Outaouais, 2020.

[8] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In : *Journal of statistical mechanics : theory and experiment* 2008.10 (2008), P10008.

[9] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[10] Mounzer Boubou. “Contribution aux méthodes de classification non supervisée via des approches prétopologiques et d’agrégation d’opinions”. Thèse de doct. Université Claude Bernard-Lyon I, 2007.

[11] Abdelhak Bousbaci. “Algorithmes parallèles de clustering de données”. Thèse de doct. 2019.

[12] Max Bramer. *Clustering*. Springer, 2007.

[13] Sergey Brin et Lawrence Page. “The anatomy of a large-scale hypertextual web search engine”. In : *Computer networks and ISDN systems* 30.1-7 (1998), p. 107-117.

[14] Fred B Bryant et Paul R Yarnold. “Principal-components analysis and exploratory and confirmatory factor analysis.” In : (1995).

[15] Maël Canu. “Détection de communautés orientée sommet pour des réseaux mobiles opportunistes sociaux”. Thèse de doct. Université Pierre et Marie Curie-Paris VI, 2017.

[16] MAROUA CHEMLAL. “Détection des sites d’hameçonnage pour assurer la sécurité sur Internet”. In : (2020).

[17] Bolin Chen et al. “Identifying protein complexes and functional modules—from static PPI networks to dynamic PPI networks”. In : *Briefings in bioinformatics* 15.2 (2014), p. 177-194.

[18] Nadia Chouchani. “Une approche de détection des communautés d’intérêt dans les réseaux sociaux : application à la génération d’IHM personnalisées”. Thèse de doct. Université Polytechnique des Hauts-de-France, 2018.

[19] Aaron Clauset, Mark EJ Newman et Cristopher Moore. “Finding community structure in very large networks”. In : *Physical review E* 70.6 (2004), p. 066111.

[20] David Combe. “Détection de communautés dans les réseaux d’information utilisant liens et attributs.(Community detection in information networks using links and attributes).” Thèse de doct. Jean Monnet University, Saint-% C3% 89tienne, France, 2013.

[21] Jerome Cornfield. “Bayes theorem”. In : *Revue de l’Institut International de Statistique* (1967), p. 34-49.

[22] Michel Crampes et Michel Plantié. “Partition et recouvrement de communautés dans les graphes bipartis, unipartis et orientés”. In : *IC-24èmes Journées francophones d’Ingénierie des Connaissances*. 2013.

[23] Pádraig Cunningham, Matthieu Cord et Sarah Jane Delany. “Supervised learning”. In : *Machine learning techniques for multimedia*. Springer, 2008, p. 21-49.

[24] Per-Erik Danielsson. “Euclidean distance mapping”. In : *Computer Graphics and image processing* 14.3 (1980), p. 227-248.

[25] Maximilien Danisch. “Mesures de proximité appliquées à la détection de communautés dans les grands graphes de terrain”. Thèse de doct. Université Pierre et Marie Curie-Paris VI, 2015.

[26] Leon Danon et al. “Comparing community structure identification”. In : *Journal of Statistical Mechanics : Theory and Experiment* 2005.09 (sept. 2005), P09008-P09008. doi : 10.1088/1742-5468/2005/09/p09008.

[27] Rachid Djerbi. “Détection de communautés dans les réseaux sociaux”. Thèse de doct. University M’Hamed Bougara of Boumerdes, 2021.

[29] François Dubois. “Les ponts de Königsberg”. In : (2015).

[30] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In : *kdd*. T. 96. 34. 1996, p. 226-231.

[32] Santo Fortunato. “Community detection in graphs”. In : *Physics reports* 486.3-5 (2010), p. 75-174.

[33] Linton C Freeman. “Centrality in social networks conceptual clarification”. In : *Social networks* 1.3 (1978), p. 215-239.

[34] Olivier Gach. “Algorithmes mémétiques de détection de communautés dans les réseaux complexes : techniques palliatives de la limite de résolution”. Thèse de doct. Université du Maine, 2013.

[35] Vince Grolmusz. “A note on the pagerank of undirected graphs”. In : *Information Processing Letters* 115.6-8 (2015), p. 633-634.

[36] Ekta Gujral et al. “Hacd : Hierarchical Agglomerative Community Detection In Social Networks”. In : *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2019, p. 1-6.

[37] Mustafa Hajij, Eyad Said et Robert Todd. “PageRank and The K-Means Clustering Algorithm”. In : *arXiv preprint arXiv :2005.04774* (2020).

[38] Richard W Hamming. “Error detecting and error correcting codes”. In : *The Bell system technical journal* 29.2 (1950), p. 147-160.

[39] Corentin Hardy. “Contribution au développement de l’apprentissage profond dans les systèmes distribués”. Thèse de doct. Rennes 1, 2019.

[40] Khadidja Henni, Neila Mezghani et Charles Gouin-Vallerand. “Unsupervised graph-based feature selection via subspace and pagerank centrality”. In : *Expert Systems with Applications* 114 (2018), p. 46-53.

[41] Lawrence Hubert et Phipps Arabie. “Comparing partitions”. In : *Journal of classification* 2.1 (1985), p. 193-218.

[42] Paul Jaccard. “Étude comparative de la distribution florale dans une portion des Alpes et des Jura”. In : *Bull Soc Vaudoise Sci Nat* 37 (1901), p. 547-579.

[43] Anil K Jain, Jianchang Mao et K Moidin Mohiuddin. “Artificial neural networks : A tutorial”. In : *Computer* 29.3 (1996), p. 31-44.

[44] Vikramaditya Jakkula. “Tutorial on support vector machine (svm)”. In : *School of EECS, Washington State University* 37.2.5 (2006), p. 3.

[45] Muhammad Aqib Javed et al. “Community detection in networks : A multidisciplinary review”. In : *Journal of Network and Computer Applications* 108 (2018), p. 87-111.

[46] Hong Jin, Shuliang Wang et Chenyang Li. “Community detection in complex networks by density-based clustering”. In : *Physica A : Statistical Mechanics and its Applications* 392.19 (2013), p. 4606-4618.

[47] Kevin Beyer Jonathan et al. “When Is” Nearest Neighbor” Meaningful ?” In : *In Int. Conf. on Database Theory*. Citeseer. 1999.

[48] Arun Kadavankandy et al. “PageRank in undirected random graphs”. In : *CoRR, abs/1511.04925* (2015).

[49] Leslie Pack Kaelbling, Michael L Littman et Andrew W Moore. “Reinforcement learning : A survey”. In : *Journal of artificial intelligence research* 4 (1996), p. 237-285.

[50] Nadjia Khatir et Safia Nait-bahloul. “Multi-criteria-based fusion for clustering texts and images case study on Flickr”. In : *Kybernetes* (2018).

[51] Teuvo KOHONEN. "Self-organized formation of topologically correct feature maps". In : *Biological cybernetics* 43.1 (1982), p. 59-69.

[53] Valdis KREBS. "Proxy networks. analyzing one network to reveal another". In : *Bulletin de méthodologie sociologique. Bulletin of sociological methodology* 79 (2003), p. 61-70.

[54] Andrea LANCICHINETTI, Santo FORTUNATO et Filippo RADICCHI. "Benchmark graphs for testing community detection algorithms". In : *Physical review E* 78.4 (2008), p. 046110.

[55] Erik G LEARNED-MILLER. "Introduction to supervised learning". In : *I : Department of Computer Science, University of Massachusetts* (2014), p. 3.

[56] E LEBARBIER et T MARY-HUARD. "Classification non supervisée". In : *Polycopié de cours AgroParisTech* (2008).

[57] Huan LI et al. "LPA-MNI : an improved label propagation algorithm based on modularity and node importance for community detection". In : *Entropy* 23.5 (2021), p. 497.

[58] L LI et al. "Community detection algorithm based on local expansion k-means". In : *Neural network world* 26.6 (2016), p. 589.

[59] Bing LIU. "Supervised learning". In : *Web data mining*. Springer, 2011, p. 63-132.

[60] Chuang LIU, Yingkui DU et Jiahao LEI. "A SOM-based membrane optimization algorithm for community detection". In : *Entropy* 21.5 (2019), p. 533.

[61] Nassira LOGRADA. "La détection de communautés dans les réseaux sociaux". Thèse de doct. UNIVERSITE MOHAMED BOUDIAF-M'SILA, 2019.

[62] Noor elHouda LOUAFI. "Analyse des réseaux sociaux et détection de communautés". Thèse de doct. Université 8 mai 1945 - GUELMA, 2019.

[63] David LUSSEAU et al. "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations". In : *Behavioral Ecology and Sociobiology* 54.4 (2003), p. 396-405.

[64] Maher MAALOUF. "Logistic regression in data analysis : an overview". In : *International Journal of Data Analysis Techniques and Strategies* 3.3 (2011), p. 281-299.

[65] James MACQUEEN et al. "Some methods for classification and analysis of multivariate observations". In : *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. T. 1. 14. Oakland, CA, USA. 1967, p. 281-297.

[66] T Soni MADHULATHA. "An overview on clustering methods". In : *arXiv preprint arXiv :1205.1117* (2012).

[67] Mohammad Saeid MAHDAVINEJAD et al. "Machine learning for Internet of Things data analysis : A survey". In : *Digital Communications and Networks* 4.3 (2018), p. 161-175.

[68] Dastan MAULUD et Adnan M ABDULAZEEZ. "A review on linear regression comprehensive in machine learning". In : *Journal of Applied Science and Technology Trends* 1.4 (2020), p. 140-147.

[69] Imane Messaoudi. “Détection de communautés dans les réseaux sociaux”. Thèse de doct. 2021.

[70] Mawloud Mosbah. “Mesures de Distance dans le Contexte de la Recherche d’Images par le Contenu (CBIR)”. Thèse de doct. Mai 2017.

[71] Fionn Murtagh et Pedro Contreras. “Algorithms for hierarchical clustering : an overview”. In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 2.1 (2012), p. 86-97.

[72] Merazka Mustapha. “Détection de communautés dans les réseaux sociaux”. Thèse de doct. Université Abderahmane MIRA de Bejaia, 2014.

[73] Vladimir Nasteski. “An overview of the supervised machine learning methods”. In : *Horizons. b* 4 (2017), p. 51-62.

[74] MED ABDELHAMID NEDIOUI. “Fouille et apprentissage automatique dans les réseaux sociaux dynamique”. Thèse de doct. Université Mohamed Khider-Biskra, 2015.

[75] Mark EJ Newman. “Fast algorithm for detecting community structure in networks”. In : *Physical review E* 69.6 (2004), p. 066133.

[76] Mark EJ Newman et Michelle Girvan. “Finding and evaluating community structure in networks”. In : *Physical review E* 69.2 (2004), p. 026113.

[77] William S Noble. “What is a support vector machine ?” In : *Nature biotechnology* 24.12 (2006), p. 1565-1567.

[78] Günce Keziban Orman et Vincent Labatut. “A comparison of community detection algorithms on artificial networks”. In : *International conference on discovery science*. Springer. 2009, p. 242-256.

[79] Juyong Park et Mark EJ Newman. “A network-based ranking system for US college football”. In : *Journal of Statistical Mechanics : Theory and Experiment* 2005.10 (2005), P10014.

[94] Usha Nandini Raghavan, Réka Albert et Soundar Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In : *Physical review E* 76.3 (2007), p. 036106.

[95] Lior Rokach et Oded Maimon. “Decision Trees”. In : jan. 2005, p. 165-192. doi : 10.1007/0-387-25465-X_9.

[96] Martin Rosvall et Carl T Bergstrom. “Maps of random walks on complex networks reveal community structure”. In : *Proceedings of the national academy of sciences* 105.4 (2008), p. 1118-1123.

[97] Gilles Roudiere. “Détection d’attaques sur les équipements d’accès à Internet”. Thèse de doct. INSA de Toulouse, 2018.

[98] Jean-Francis Roy. “Apprentissage automatique avec garanties de généralisation à l’aide de méthodes d’ensemble maximisant le désaccord”. In : (2018).

[99] Alassane Samba. “Science des données au service des réseaux d’opérateur : proposition de cas d’utilisation, d’outils et de moyens de déploiement”. Thèse de doct. Oct. 2018.

[100] Hanan Samet. "K-nearest neighbor finding using MaxNearestDist". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2007), p. 243-252.

[101] Neil Scicluna et Christos-Savvas Bouganis. "ARC 2014: a multidimensional FPGA-based parallel DBSCAN architecture". In : *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 9.1 (2015), p. 1-15.

[102] Jinfang Sheng et al. "Research on community detection in complex networks based on internode attraction". In : *Entropy* 22.12 (2020), p. 1383.

[103] Richard S Sutton. "Learning to predict by the methods of temporal differences". In : *Machine learning* 3.1 (1988), p. 9-44.

[104] Iwan Syarif, Adam Prugel-Bennett et Gary Wills. "Unsupervised clustering approach for network anomaly detection". In : *International conference on networked digital technologies*. Springer. 2012, p. 135-145.

[105] Raphaël Tackx. "Analyse de la structure communautaire des réseaux bipartis". Thèse de doct. Sorbonne université, 2018.

[106] FAYSSAL TALHI. "Étude comparative des méthodes d'évaluation de la qualité des structures communautaires". In : (2021).

[107] BOUDHEB TARIK et al. "Privacy Preserving Classification of Biomedical Data". Thèse de doct. 2019.

[108] Shahadat Uddin et al. "Comparing different supervised machine learning algorithms for disease prediction". In : *BMC medical informatics and decision making* 19.1 (2019), p. 1-16.

[109] Alexandre Vilcek. "Deep Learning with K-Means Applied to Community Detection in Networks". In : *CS224W Project Report* (2014).

[110] Christopher JCH Watkins et Peter Dayan. "Q-learning". In : *Machine learning* 8.3 (1992), p. 279-292.

[111] Min Xu et al. "Decision tree regression for soft classification of remote sensing data". In : *Remote Sensing of Environment* 97.3 (2005), p. 322-336.

[112] Ahmed Yahi. "Clustering des données de puces à ADN". Thèse de doct. UNIVERSITE MOHAMED BOUDIAF-M'SILA, 2019.

[113] Zhao Yang, Juan I Perotti et Claudio J Tessone. "Hierarchical benchmark graphs for testing community detection algorithms". In : *Physical review E* 96.5 (2017), p. 052311.

[114] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[115] Yan Yuan et al. "An influence maximisation algorithm based on community detection". In : *International Journal of Computational Science and Engineering* 22.1 (2020), p. 1-14.

[116] Wayne W Zachary. "An information flow model for conflict and fission in small groups". In : *Journal of anthropological research* 33.4 (1977), p. 452-473.

[117] Junlong Zhang et Yu Luo. “Degree centrality, betweenness centrality, and closeness centrality in social network”. In : *Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017)*. T. 132. 2017, p. 300-303.

[118] Yun Zhang. “Cluster Analysis and Network Community Detection with Application to Neuroscience”. Thèse de doct. University of Pittsburgh, 2017.

[119] Haijun Zhou. “Distance, dissimilarity index, and network community structure”. In : *Physical review e* 67.6 (2003), p. 061901.

Webographie

[28] DOPHINN. *dop networks*. 2022. URL : https://www.researchgate.net/figure/Graph-of-the-dolphin-network-where-the-red-nodes-are-selected-according-to-Algorithm-1_fig1_324859794 (visité le 14/04/2022).

[31] FOOTB. *foot networks*. 2022. URL : https://www.researchgate.net/figure/The-community-structure-of-Football-network-a-The-real-community-structure-b-The_fig2_356752926 (visité le 14/04/2022).

[52] KONECT. *sum networks*. 2022. URL : http://konect.cc/networks/ (visité le 14/04/2022).

[80] POLITICALB. *politic networks*. 2022. URL : https://www.researchgate.net/figure/Political-Books-Network_fig4_261339381 (visité le 14/04/2022).

[81] PYTHON. *community Python*. 2022. URL : https://pypi.org/project/community/ (visité le 09/04/2022).

[82] PYTHON. *datetime Python*. 2022. URL : https://docs.python.org/3/library/datetime.html (visité le 09/04/2022).

[83] PYTHON. *itertools Python*. 2022. URL : https://docs.python.org/3/library/itertools.html (visité le 09/04/2022).

[84] PYTHON. *Math Python*. 2022. URL : https://docs.python.org/3/library/math.html (visité le 09/04/2022).

[85] PYTHON. *matplotlib Python*. 2022. URL : https://matplotlib.org/ (visité le 09/04/2022).

[86] PYTHON. *networkx Python*. 2022. URL : https://networkx.org/documentation/stable/reference/introduction.html (visité le 09/04/2022).

[87] PYTHON. *NumPy Python*. 2022. URL : https://www.datacamp.com/community/tutorials/python-numpy-tutorial (visité le 09/04/2022).

[88] PYTHON. *operator Python*. 2022. URL : https://docs.python.org/3/library/operator.html (visité le 09/04/2022).

[89] PYTHON. *Python Tutorial*. 2022. URL : https://www.tutorialspoint.com/python/index.htm (visité le 03/04/2022).

[90] PYTHON. *random Python*. 2022. URL : https://docs.python.org/3/library/random.html (visité le 09/04/2022).

[91] PYTHON. *sklearn Python*. 2022. URL : https://scikit-learn.org/stable/ (visité le 09/04/2022).

[92] PYTHON. *sys Python*. 2022. URL : https://docs.python.org/3/library/sys.html (visité le 09/04/2022).

[93] PYTHON. *tkinter Python*. 2022. URL : https://docs.python.org/3/library/tkinter.html (visité le 09/04/2022).

Annexes

Annexe A

Définitions

A.1 Apprentissage par renforcement

A.1.1 A3C (Asynchronous Actor-Critic Agent algorithm)

L'algorithme A3C est l'un des algorithmes de pointe de l'Apprentissage par renforcement, il peut être bénéfique dans les expériences qui impliquent une optimisation globale du réseau avec différents environnements en parallèle à des fins de généralisation. Voici la magie derrière : Asynchrone représente qu'un seul réseau neuronal interagit avec un seul environnement. Au contraire, dans ce cas, nous avons un réseau mondial avec plusieurs agents ayant leur propre ensemble de paramètres. Il crée la situation de chaque agent en interagissant avec son environnement et en récoltant l'expérience d'apprentissage différente et unique pour la formation globale. Cela traite également en partie de la corrélation des échantillons de l'Apprentissage par renforcement, un gros problème pour les réseaux de neurones, qui sont optimisés en supposant que les échantillons d'entrée sont indépendants les uns des autres (ce qui n'est pas possible dans les jeux). Dans ce cas, nous avons un réseau mondial avec plusieurs agents ayant leur propre ensemble de paramètres.

Actor-Critic représente deux réseaux de neurones - Acteur et Critique. L'objectif de l'Acteur est d'optimiser la politique "Comment agir ? ", et le Critique vise à optimiser la valeur "Quelle est la qualité de l'action ?".

Ainsi, cela crée une situation complémentaire pour qu'un agent acquière la meilleure expérience d'apprentissage rapide.

Annexe B

Quelques résultats de notre approche

Exemple 1

Soit un réseau test1 de 97 nœuds et 133 arrêtes.

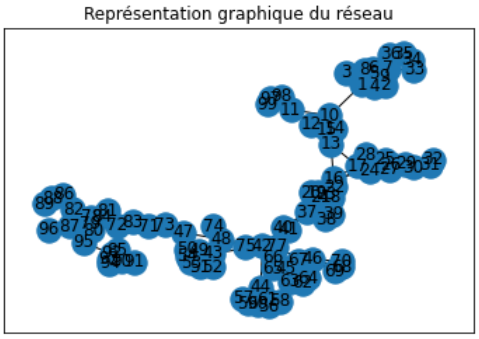

Fig. B.1 : Représentation graphique du réseau de test1.

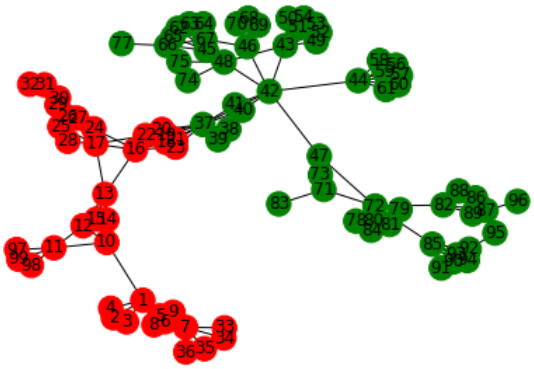

Fig. B.2 : Résultat de notre approche sur le réseau test1.

Exemple 2

Soit un réseau test2 de 17 nœuds et 17 arrêtes.

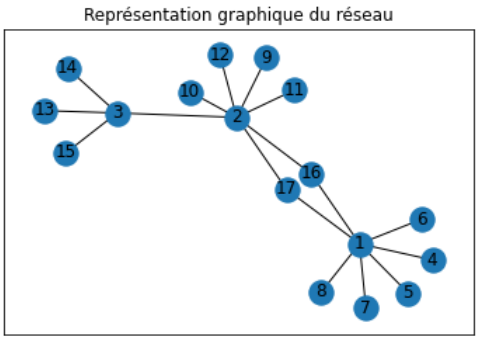

Fig. B.3 : Représentation graphique du réseau de test2.

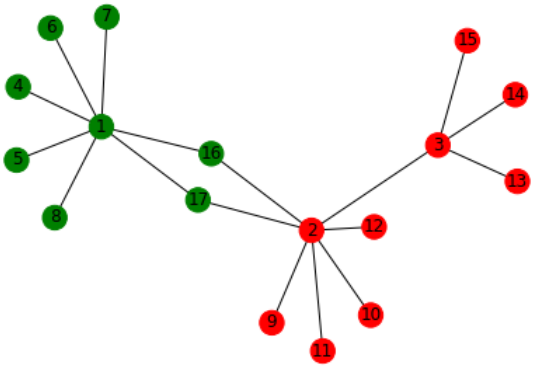

Fig. B.4 : Résultat de notre approche sur le réseau test2.