MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

**Faculté des Sciences Exactes et d'Informatique**

**Département de Mathématiques et informatique**

**Filière : Informatique**

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Ingénierie des Systèmes d'Information**

Présenté par :

**Belhachemia mohammed ibrahim**

# THEME :

# Problème détection de communauté.

Soutenu le :

Devant le jury composé de :

| | | | |
|---|---|---|---|
| Nom et Prénom | Grade | Université de Mostaganem | Président |
| Nom et Prénom | Grade | Université de Mostaganem | Examinateur |
| Fouad henni | A | Université de Mostaganem | Encadreur |
| Menad bekkai mohamed | A | Université de Mostaganem | Co-Encadreur |

Année Universitaire 2020-2021

# Dédicaçasse

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je voudrais tout d'abord adresser toute ma reconnaissance à la directrice de ce mémoire, M. Fouad henni et Co-encadreur M. Bekkai mohamed menad, pour leur patience, leur disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je désire aussi remercier les professeurs de l'université de Mostaganem., qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

Je voudrais exprimer ma reconnaissance envers les amis et collègues qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche.

# Remerciements

Nous tenus à remercier très sincèrement notre grande famille, spécialement nos parents pour leur soutien indéfectible.

Nous adressons nos remerciements aux personnes qui ont aidé dans la réalisation de ce mémoire. En premier lieu, Nous remercions :

M.Fouad henni et M.Bekkai mohamed menade, l'enseignant à l'université de Mostaganem qui, en tant qu'encadreur, nous a guidées dans notre projet et nous a aidé à trouver des solutions pour avancer.

Nos remerciements s'adressent également à tous les enseignants qui nous ont instruits durant nos études à la faculté des sciences de l'université de Mostaganem.

En fin nous tenons à remercier tous nos collègues d'études particulièrement ceux de notre promotion, ainsi que nos familles.

# Liste des figures

# Liste des tableaux

# Liste des abréviations

| Abréviation | Expression Complète | Page |
|---|---|---|
| PXP | Personal Extreme Programming | 27 |
| RUP | Rational Unified Process | 27 |
| IBM | International Business Machines Corporation | 28 |
| CPU | Central Processus Unit | 39 |
| RAM | Random Access Memory | 39 |
| CGI | Common Gateway Interface | 42 |
| OS | Operating System | 42 |
| API | Application Programming Interface | 42 |

# Table des matières

# Introduction générale

La détection de communautés dans l'analyse des réseaux sociaux suscite de plus en plus d'attention dans la communauté scientifique et de nombreuses recherches ont été menées dans ce domaine. De nos jours, les réseaux sociaux sur internet apparaissent comme un nouveau moyen important de communication. Tout d'abord, il faut savoir qu'un réseau social est un ensemble d'identités sociales où des individus sont reliés entre eux par des liens d'interactions sociales, il regroupe de nombreux individus, nommés des utilisateurs. Les réseaux sont souvent modélisés par des graphes, une structure permettant l'encodage de données relationnelles. Quel que soit le domaine applicatif.

Les sites de réseautage social font partie des centres dans le monde d'échange de connaissances, de langues et de gains tribaux qui distinguent l'individu du reste du groupe et lui font se développer et se rapporter à des événements que nous pouvons incorporer sous une forme mathématique en utilisant un des ( *caractéristiques* ) dites théories des graphes, , comme ce qui se passe sur **Facebook**, **Twitter** et **Instagram** à partir de l'échange de messages constitués d'un ensemble de mots-clés, de sémantiques et d'images qui expriment la forme et notre travail de mémoire fin d'étude a pour objectif de présenter et d'étudier quelques méthodes de détection de communautés , en plus l'évaluations avec modèle de données réal, ce rapport de mémoire est structuré comme suit dans

- Le premier et deuxième chapitre, nous traiterons d'identifier les différents modèles de la détection de communauté et les algorithmes choisie respectivement la méthodologie.

- Le troisième et quatrième chapitre, nous aborderons l'étape de la conception à l'aide le **PXP** concernant les diagrammes et identifications les acteurs externes et internes.

- Le cinquième chapitre l'implémentation de l'algorithme avec langage python.

# Chapitre 1 Modèle des données et méthodologie

## 1.1 Introduction

Dans ce chapitre nous allons présenter le contexte des algorithmes de détection de communauté et la méthodologie de réaliser et choisie les modèles de données.

## 1.2 Contexte

Le monde qui nous entoure peut être vu comme un ensemble d'interactions entre éléments, qui peuvent être d'origine naturelle et concrète, par exemple des contacts entre êtres humains, ou bien basées sur des constructions plus abstraites comme les interactions économiques régissant les marchés financiers. Elles peuvent avoir lieu à l'échelle microscopique, telles les interactions entre protéines en permanence à l'œuvre dans notre corps, ou au contraire à une échelle macroscopique à l'instar des interactions gravitationnelles entre corps célestes, Ces interactions ou ensembles d'interactions, appelés systèmes complexes, dans de nombreux domaines scientifiques : sociologie, physique, économie, biologie etc. Si les plus simples ont pu être expliquées au fil du temps, une grande majorité d'entre elles échappe encore à une modélisation qui les rendrait prédictibles.

## 1.3 Problématique

La détection communautaire est devenue au cours de la dernière décennie l'un des problèmes les plus difficiles et les plus étudiés dans l'analyse de réseau complexe, en raison de sa pertinence pour un large éventail d'applications telles que l'étude de l'information et la propagation de l'analyse et benchmarks [1]. Nous avons identifié trois problèmes suivants :

- La taille des modèles.
- Temps de l'exécution.
- Les règles l'opération de benchmark non satisfiable.

## 1.4 Modèle de données

Un réseau complexe peut être modélisé à l'aide d'un graphe tel que les nœuds représentent des individus et les arêtes un certain type d'interaction entre ceux-ci. Les termes {*nœud, sommet*} sont utilisés de façon interchangeable pour désigner strictement le même concept ; de même pour [*lien, arête*] et [*communauté, cluster, module*]. Le terme {*arc*} est réservé aux arêtes orientées, c'est-à-dire aux relations non *réciproques.*

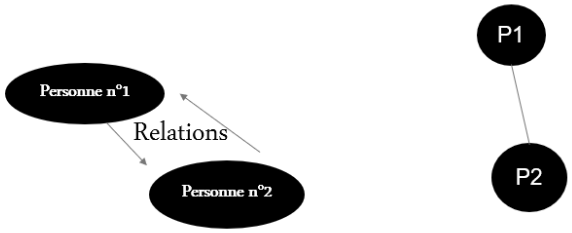


Figure 1 – Représentations avec sommets et les arrêts une relation indirecte.

### 1.4.1 Graphe orienté

Un graphe avec p sommets et q arêtes sera nommée un (*p*, *q*) graphe, Un graphe orienté G = (V, E) est un digraphe tel que (u, v) ∈ E ⇒ (v, u) ∉ E

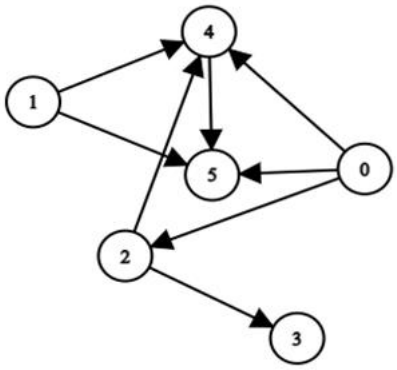


Figure 2 – Graphe orienté avec p=6 et q=8

### 1.4.2 Graphe non orienté

Un graphe non orienté G = (V, E) est un digraphe tel que (u, v) ∈ E ⇒ (v, u) ∈ E

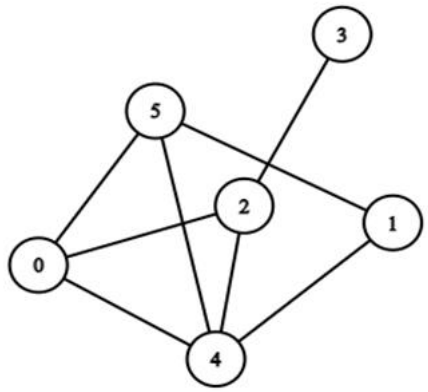


Figure 3 – Graphe non orienté avec p=6 ,q=8

## 1.5 Traduction numérique

Les graphes possédant une structure en communautés plus ou moins naturelles sont engendrés par le protocole de Newman

### 1.5.1 Matrice adjacent graphe orienté

Un graphe peut être représenté à l'aide matrice dite d'adjacence ($A_{ij}$) qui indique les connexions entre les nœuds, dans l'exemple suivant on a une matrice avec **6** sommets et **8** les arêtes.

**Tableau 01 – Matrice adjacente d'un graphe orienté de figure n°1.**

| V | E0 | E1 | E2 | E3 | E4 | E5 |
|---|---|---|---|---|---|---|
| E0 | 0 | 0 | 1 | 0 | 1 | 1 |
| E1 | 0 | 0 | 0 | 0 | 1 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| E2 | 0 | 0 | 0 | 1 | 1 | 0 |
| E3 | 0 | 0 | 0 | 0 | 0 | 0 |
| E4 | 0 | 0 | 0 | 0 | 0 | 1 |
| E5 | 0 | 0 | 0 | 0 | 0 | 0 |

### 1.5.2 Matrice adjacent graphe non orienté

**Tableau 01 – Matrice adjacente d'un graphe non orienté de figure n°3**

| V | E0 | E1 | E2 | E3 | E4 | E5 |
|---|---|---|---|---|---|---|
| E0 | 0 | 0 | 1 | 0 | 1 | 1 |
| E1 | 0 | 0 | 0 | 0 | 1 | 1 |
| E2 | 0 | 0 | 0 | 1 | 1 | 0 |
| E3 | 0 | 0 | 1 | 0 | 0 | 0 |
| E4 | 1 | 1 | 0 | 0 | 0 | 1 |
| E5 | 1 | 1 | 0 | 0 | 1 | 0 |

## 1.6 Notion les graphes statiques et propriétés

Nous allons vous expliquer certains des principes à prendre en compte pour comprendre les algorithmes de détection de communauté, Soit

1. (m, v) une arête reliant les sommets u et v. Si l'arête est non orientée, alors (m, v) = (v, m)
2. G = G (V, E, W) un graphe simple (*sans boucles ni liens multiples*), pondéré et non orienté (*sauf indication contraire*) représentant un réseau où V est l'ensemble des nœuds et E celui des arêtes. , et où W est une matrice $\mathbb{R}^{|v|\times|v|}$, telle que W (u, v) =$W_{uv}$ est le poids de l'arête (m, v) de E. Dans le cas des graphes non-pondérés, w, v = 1, pour chaque lien (m, v) de E. Dans ce cas particulier, le graphe sera noté simplement G = G (V, E)
3. N = |y| le cardinal de l'ensemble V, c'est-à-dire le nombre de sommets du graphe.
4. M = \E\ le cardinal de l'ensemble E, c'est-à-dire le nombre de liens du graphe
5. d(u) = $d_u$ le degré du sommet u de V, c'est-à-dire la somme des poids des arêtes non orientées dont une extrémité pointe le sommet u.
6. $d^{in}$(u) = $d_u^{in}$ le degré entrant du sommet u de V, c'est-à-dire la somme des poids des arcs vers le sommet u.
7. $d^{out}$= $d_u^{out}$le degré sortant du sommet u de V, c'est-à-dire la somme des poids des arcs depuis le sommet u
8. $A \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ la matrice d'adjacence telle que

$$\text{A (u, v)} = a_{uv}\begin{cases}1, si\ (u,v) \in \mathbb{E} \\ 0, \quad \bar{si\ non}\end{cases} \qquad (1.5)$$

- Dans le cas d'un graphe non pondéré, W ≡ A.

## 1.7 Notion les communautés statiques

Soit

- $M = \{C_{1,}C_{2,} \ldots\ldots C_k\}$ une collection de sous-graphes de V telle que $C_i = G\left(V_{C_i}, E_{C_i}, W_{C_i}\right) \in E$ est une communauté de G. Dans le cas de communautés disjointes, $C_i \cap C_j = \emptyset$ pour tout $C_i \neq C_j$, alors que dans le cas de communautés avec recouvrement l'ensemble $C_i \cap C_j$ peut-être non vide pour$C_i \neq C_j$ . Dans les deux cas, un sommet peut être non-affilié, c'est-à-dire m u ∈ G mais$\notin C_i,\ \forall\, i$.

- $N_M = |M|$ le cardinal de l'ensemble c'est-à-dire le nombre de communautés du graphe.
- $n_{C_i} = |V_{C_i}|$ le cardinal de l'ensemble$V_{C_i}$, c'est-à-dire le nombre des sommets du sous-graphe $C_i$.
- $N_{C_i}(u)$ L'ensemble des voisins du sommet $u$ dans $C_i$, c'est-à-dire l'ensemble des nœuds v € $V_{C_i}$ tels que ($u$, $v$)∈ $E_{C_i}$.
- *Com*($u$) un vecteur d'étiquettes du sommet $u$, c'est-à-dire l'affiliation de $u$ à une ou plusieurs communautés.
- $d_{C_i}^{in}$ Le degré interne du nœud $u$ dans la communauté $C_i$ , c'est-à-dire la somme des poids des arêtes ($u$, $v$) de E telles que $u, v \in C_i$

$$d_{C_i}^{in} = \sum_{u \in V_{C_i}} W_{uv}$$

- $d_{C_i}^{out}$ Le degré externe du nœud $u$ dans la communauté $C_i$ , c'est-à-dire la somme des poids des arêtes ($u$, $v$) de E telles que $u, v \notin C_i$

$$d_{C_i}^{out} = \sum_{u \in V_{C_i}} W_{uv}$$

## 1.8 Méthodologie

Le problème de l'identification des données fait partie des étapes les plus complexes auxquelles la plupart des chercheurs sont confrontés en raison de la difficulté et de la taille ,Le secteur privé majeur dans l'étude détection de communauté, qui est une question qui doit choisir les aspects des réseaux dynamiques présents dans les réseaux évolutifs, des solutions proposées sont tout aussi diverses qu'il existe de variantes au problème, nous n'aspirons aucunement à généraliser l'étude des communautés dynamiques, mais bien à confronter un jeu de données réelles à l'état de l'art.

- La première section décrit les caractéristiques d'un réseau traité
- La seconde section propose des modélisations d'un jeu de données réelles satisfaisante les contraintes et les caractéristiques imposées selon les multiples

stratégies de détection de communautés (*statique ou dynamique*) et réseaux évolutifs.

- Troisième section propose les méthodes évolutives et benchmarks pour comparer les algorithmes choisie (*Louvain et Girvan Newman).*

La section algorithmes contient deux parties distinctes expliquant les propriétés et les conditions qui doivent exister pour les appliquer à un graphe (*non orientée*).

## 1.9 Propriétés du réseau évolutif

Dans la partie, nous aborderons les caractéristiques des types de données auxquels nous pouvons appliquer des algorithmes qui dépendent fortement du réseau ou un ensemble des individus et les problèmes auxquels nous pouvons faire face lors de l'application pratique.

### 1.9.1 Réseau social

Les graphes sont à la fois une structure de donnée informatique et un outil mathématique abstrait avec de nombreuses applications par exemple chimie ( ***chemoinformatique*** ) et biochimie ( *génomique* ), génie électrique et théorie des codes, réseaux informatiques, urbains, sociaux ( *Internet, Google, Facebook* ), Chaque type de réseau, qu'il soit biologique, neural, social, d'information, etc., présente une structure propre, que ce soit en rapport à la distribution des degrés des nœuds de son graphe.

### 1.9.2 Réseau temporel

C'est un réseau qui contient et se caractérise par son changement temporel qui est évolutif en termes de composants d'individus dans le temps car il change simultanément par rapport à la base de données utilisée qui est destructible.

### 1.9.3 Réseau hors-ligne

Un jeu de données hors-ligne suppose que toute l'information est disponible avant de modéliser le réseau, c'est-à-dire qu'il s'agit de valeurs historiques connues (*Modèle de données karateclub, Barbell* ).

### 1.9.4 Interactions pondérées

Qu'entretiennent des individus ne sont pas toujours de nature unique. Ainsi, modéliser les liens de façon binaire entre les sommets d'un graphe pourrait masquer une part de l'information sur la force de la relation entre ces individus. Il est alors raisonnable de penser qu'un poids pour chacun des liens peut exprimer la nature de la liaison. Par exemple, ce poids peut représenter la durée ou la répétition de l'interaction [2]

## 1.10 Modélisations d'un réseau social hors ligne sous forme un document TXT ou YML

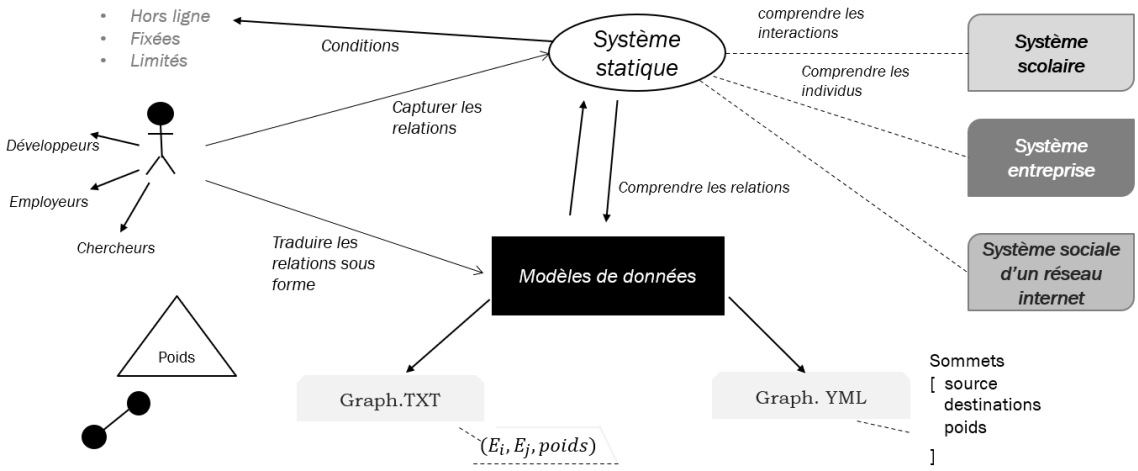


Figure 4 – l'opération de capturer et traduire sous forme YML ou TXT

Nous proposons deux modèles sous forme ( *TXT* ou *YML* ) à l'aide les modèle hors ligne de networks X bibliothèque suivant

1. **Karate club modèle** le club de karaté se compose de méthodes de pointe pour faire un apprentissage non supervisé sur des données structurées en graphes, il d´écrite les relations d'amitiés entre **34** membres d'un club de Karaté observées lors de la gestion d'un conflit durable entre l'entraîneur et l'administrateur du club , on a 34 sommets.
2. **Barbell modèle** le plus couramment utilisé est un graphe à n haltères qui est un graphe simple obtenu en connectant deux copies d'un graphe complet avec n nœuds, dans notre application nous avons choisie **n=5.**

## 1.11 L'étude de l'existence

L'analyse de la société à travers des applications et des algorithmes spécifiques est quelque chose qui doit être exploité pour extraire des résultats efficaces et valorisants car Internet contient beaucoup de code source, mais il souffre de nombreux inconvénients et lacunes qui le rendent incompréhensible par les utilisateurs car la plupart des applications ne contiennent pas d'outils de protection.

- L'absence d'une fonctionnalité qui contribue à organiser les algorithmes en termes d'efficacité.
- L'absence de la fonctionnalité de saisie des informations de manière fluide qui contribue à gagner du temps et à obtenir un résultat précis.
- L'absence benchmarks avec F1 score (*MIN, MAX, MOYENNE*).
- L'absence de fonctionnalité de l'impression pour mieux comprendre les informations et les détails dans le system (*nombre inter communauté)*
- L'absence l'organisation les utilisateurs principales de l'application pour organisée les acteurs dans le système.
- L'absence la politique de la sécurité par exemple le chiffrement et droit accès dans le système.
- L'absence l'outil pour créer les modèles des données ou modifiée les paramètres respectivement le type des modèles choisie.

## 1.12 Conclusion

Dans ce chapitre nous avons conclure que la détection de communautés dans les réseaux sociaux s'appuie sur des algorithmes ou des méthodes issus de domaines de recherche ayant évolué de manière relativement indépendante : la classification automatique à l'aide les modèles de données et la théorie des graphes.

# Chapitre 2 Les méthodes détection de communauté.

## 2.1 Introduction

Dans ce chapitre nous allons présenter les méthodes détection de communauté les plus populaires respectivement les deux types d'agglomération et la division.

## 2.2 Méthodes de détection de communautés

Les méthodes de détection communautaire peuvent être globalement classées en deux types, méthodes d'agglomération et méthodes de division. Dans les méthodes agglomératives (clustering), les arêtes sont ajoutées une par une à un graphe qui ne contient que des nœuds. Les bords sont ajoutés du bord le plus fort au bord le plus faible. Les méthodes de division (*Louvain, Surprise, Leidan*) suivent le contraire des méthodes agglomératives. ici, les arêtes sont supprimées une par une d'un graphique complet.

. <

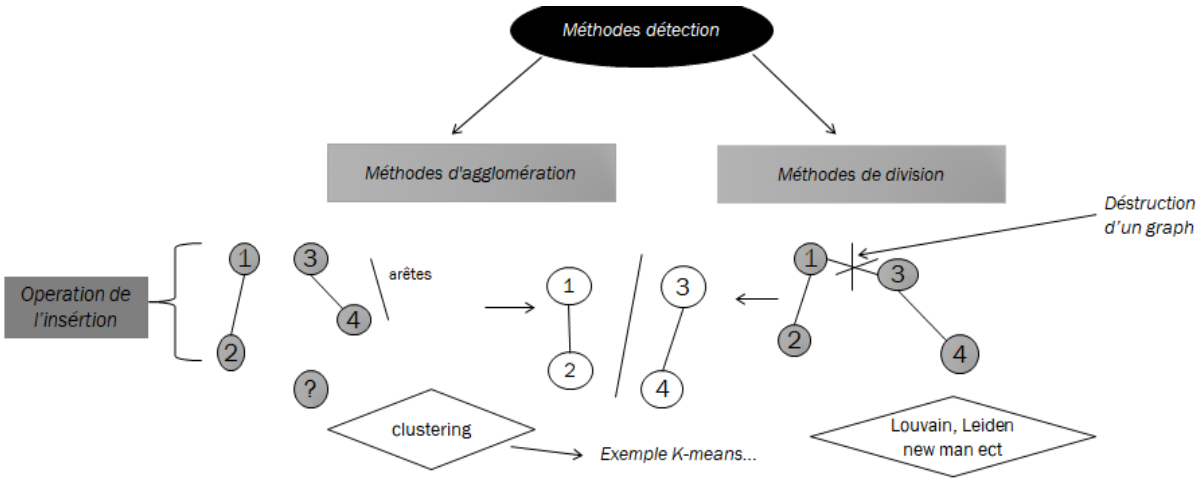


Figure 5 – Méthodes détection de communauté en 02 catégories.

### 2.2.1 Clustering

On peut affirmer que la détection de communauté est similaire au *clustering,* Le *clustering* est une technique d'apprentissage automatique dans laquelle des points de données similaires sont regroupés dans le même cluster en fonction de leurs attributs. Même si le *clustering* peut être appliqué aux réseaux, il s'agit d'un domaine plus large de l'apprentissage automatique non supervisé qui traite de plusieurs types d'attributs. D'autre part, la détende communauté est spécialement conçue pour l'analyse de réseau qui dépend d'un seul type d'attribut appelé arêtes. De plus, les algorithmes de *clustering* ont tendance à séparer les nœuds périphériques uniques des communautés auxquelles ils devraient appartenir. Cependant, les techniques de clustering et de détection de communauté peuvent être appliquées à de nombreux problèmes d'analyse de réseau et peuvent soulever des avantages et des inconvénients différents selon le domaine.

### 2.2.2 Méthode de Louvain

L'algorithme de détection de la communauté de Louvain est une méthode de clustering hiérarchique classée dans le problème NP-difficile. Son temps d'exécution pour trouver des communautés dans de grands graphes est donc un défi. La parallélisassions est une solution efficace pour amortir le temps d'exécution de Louvain concernant le problème de Louvain qui produire des communautés arbitrairement mal connectées.

### 2.2.3 Méthode de Surprise

En raison des limites de la modularité, une mesure basée sur des probabilités classiques connue sous le nom de Surprise a été introduite pour évaluer la qualité d'une partition d'un réseau en communautés. L’algorithme est presque similaire à l'algorithme de détection de la communauté de Louvain sauf qu'il utilise des surprises au lieu de la modularité. Les nœuds sont déplacés d'une communauté à une autre de sorte que les surprises sont améliorées avec gourmandise, cette approche considère la probabilité qu'un lien se trouve au sein d'une communauté, l'utilisation de surprises fonctionne bien dans la limite de nombreuses petites

communautés et l'utilisation de la modularité fonctionne bien dans la limite de quelques grandes communautés.

### 2.2.4 Méthode de Girvan-newman

Pour la détection et l'analyse des structures communautaires, l'algorithme de Girvan-Newman repose sur l'élimination itérative des arêtes qui ont le plus grand nombre de chemins les plus courts entre les nœuds qui les traversent, En supprimant les arêtes du graphique un par un, le réseau se décompose en plus petits morceaux, appelés communautés. L'algorithme a été introduit par Michelle Girvan et Mark Newman

### 2.2.5 Méthode de Leiden

La détection communautaire est souvent utilisée pour comprendre la structure de réseaux vastes et complexes, Nous montrons que Louvain algorithme présente un défaut majeur qui est passé largement inaperçu jusqu'à présent : le Louvain L'algorithme peut produire des communautés arbitrairement mal connectées. Dans le pire des cas, les communautés peuvent même être déconnecté, en particulier lors de l'exécution itérative de l'algorithme. Pour résoudre ce problème, nous introduisons l'algorithme de Leiden. Nous prouvons que l'algorithme de Leiden produit des communautés qui sont assurées d'être connectées. de plus, nous prouvons que, lorsque le Leiden l'algorithme est appliqué de manière itérative, il converge vers une partition dans laquelle tous les sous-ensembles de toutes les communautés sont attribué localement de manière optimale. De plus, en s'appuyant sur une approche de déplacement local rapide, l'algorithme de Leiden s'exécute plus rapidement que l'algorithme de Louvain. les performances de l'algorithme de Leiden pour plusieurs réseaux de référence et du monde réel.

### 2.2.6 Méthode de Walktrap

Cet algorithme a été proposé par *Pon* et *Latapy*, c'est un algorithme de clustering hiérarchique. L'idée de base de cette méthode est que les promenades aléatoires de courte distance ont tendance à rester dans la même communauté, à partir d'une partition totalement non clustérisée, les distances entre tous les nœuds adjacents sont calculées. L'intuition de base de

l'algorithme est que les marches aléatoires sur un graphe / réseau ont tendance à être piégées dans des parties densément connectées correspondant à des communautés, *Walktrap* utilise le résultat de marches aléatoires pour fusionner des communautés distinctes de manière ascendante. La qualité des cloisons peut être évaluée à l'aide de tout critère de qualité disponible. Il peut s'agir soit de modularité comme dans la détection communautaire de Louvain, soit de toute autre mesure.

## 2.3 Comparaison

Grâce à la comparaison des performances et de la fiabilité à l'aide de la figure ci-dessus, on peut trouver une différence entre les différents algorithmes en termes de nombre de nœuds [3] et de liens communs montre l'information mutuelle entre chaque décomposition trouvée et la précédente. qui est assez fréquente, semble indiquer qu'elle est plutôt stable. Fait, si l'on regarde la distance d'édition. Louvain et ***Walktrap*** et ***Fast Greedy***,

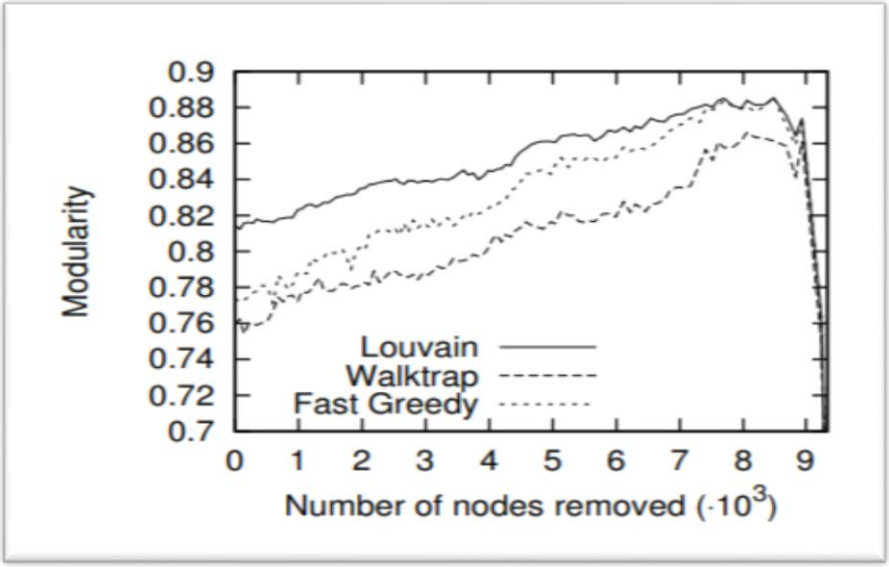


Figure 6 – Représente la modularité par rapport le nombre des nœuds de 03 algorithmes

## 2.4 Conclusion

Dans ce chapitre nous avons présenté quelque algorithme de détection de communauté, on a conclure que il y a différents algorithmes générer différents résultats par rapport la modularité et nombre des individus influent dans l'environnement.

# Chapitre 3 Les algorithmes d'un modèle statique et analyse les besoins

## 3.1 Introduction

Dans ce chapitre nous avons identifié les modèle statique et analyse les besoins fonctionnels et non fonctionnels de notre applications respectivement les objectifs et l'algorithme proposées.

## 3.2 Les algorithmes proposés et les conditions nécessaires pour réaliser la détection de communauté.

Nous avons proposé deux algorithmes pour faire la traduction algorithmique et application en programme avec langage python, les deux algorithmes sont la méthode Girvan new man et Louvain.

### 3.2.1 Girvan new man

La méthode de Girvan-Newman est une méthode de division talque le poids de sommets est le nombre de chemins les plus courts passant à travers l'arrêts (Graph non orientée) ou l'arc (graph orientée). Cette valeur s'appelle " *Edge between us* " et c'est une généralisation de Edge betweennus sommets centraux qui déterminent l'influence des sommets sur les autres sommets de réseau. L'entre-deux ( *Edge between us* ) sommets est le nombre de chemins passant par le sommet, par conséquent, l'arrête l'entre-deux est le nombre de chemins les plus courts passant à travers les extrémités de l'arrête [4]

#### 3.2.1.1 Les équations numériques

- La modularité

$$Q = \sum_i (e_{ii} - a_i)^2 = tr(e) - \|e^2\| \quad (1.13.1)$$

### 3.2.2 Calculer la valeur Edge betweennus d'un graphe non orientée

Un graphe non orienté G = (V, E) est un digraphe tel que (u, v) ∈ E ⇒ (v, u) ∈ E, on a p=6 et q=9 respectivement p le nombre de sommet et q le nombre des arrêts.

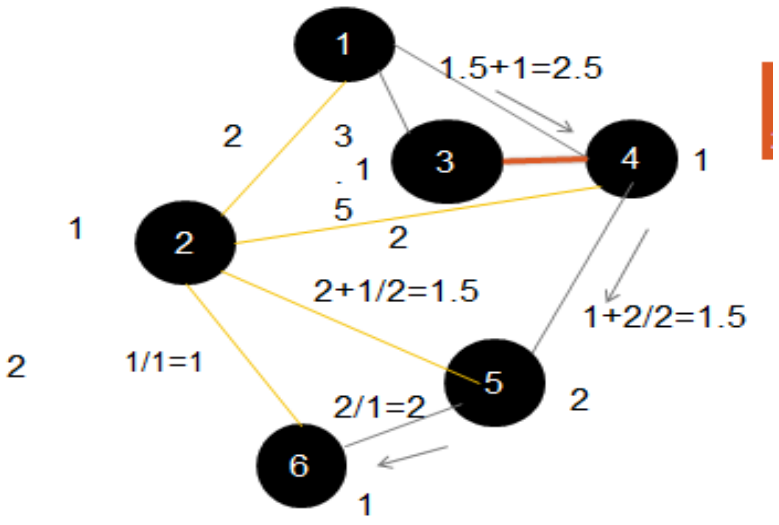


Figure 7 – L'opération de calculer *Edge Betweennes* d'un graph avec p=6, q=9.

### 3.2.3 Principe et les étapes de l'algorithme Girvan new man

L'algorithme fonctionne sur l'invariant de boucle suivant

1. Calculez « *Betweenness* » de toutes les arêtes.
2. Suppression les nœuds avec la plus grande « *Edge betweenness*» et modularité après le retrait des nœuds.
3. Recalculer « *Edge betweenness*» et la modularité après le retrait des nœuds.
4. Répétez jusqu'à ce qu'il ne reste plus de nœuds.
5. Le maximum de toutes les modularités calculées donnera les communautés optimales du réseau.

### 3.2.3.1 La compléxité de l'algorithme

- **Best case** $O(n^2m)$
- **Worst case** $O(n^2m)$

## 3.2.4 Algorithme

*Entrées* $\boldsymbol{G^0=(V^0, E^0)}$ *// Graphe non orienté,* $V^0$ *(représente le sommet) et* $E^0$ *(représente les arêtes) (Modèle de données initial karate_club sous forme txt)*

$\boldsymbol{Adj(v)}$ *// La matrice adjacent talque v* $\in V$.

$\boldsymbol{Q_{0(C)}}$ *// La modularité de la communité égale zero .*

*Sorties* $\boldsymbol{Q}$ *//Queue avec les valeurs de l'edge betweenness*

$\boldsymbol{L}$ *//List continent plus courts chemins.*

***Début***

$i \leftarrow 1$ *//Pour le sommet n° i (Sommet initial jusqu' i < V totale i.e le nombre de sommet totale)*

$j \leftarrow 1$ *//Pour le sommet n° j (Sommet différents j ≠i)*

$d_i \leftarrow 0$ *// La distance par rapport le sommet initial.*

$b_i \leftarrow 0$ *//Le nombre les plus cours chemin par rapport la source et n'importe quel sommet dans le graph.*

$W_i \leftarrow 1$ *// Le nombre le plus court chemin associé sommet i talque i=1 alors W(S) =1*

$afficher("Le\ nombre\ de\ sommets\ Totale :" + V)$ *//Le nombre de sommet totale (positions de de la dernière sommet)*

*//1ère Boucle de l'algorithme pour réaliser l'opération détection le nombre plus courts chemin.*

$Tanque\ i <> j\ et\ i < V$

$W_v \leftarrow 0$ *// Le nombre plus courts chemin égale la valeur 0*

$b_v \leftarrow 0$

$d_j \leftarrow inf$ *// Le nombre inf représente la distance inférieure*

$i \leftarrow i + 1$ *// Indice pour l'opération de l'incrémentation*

$Q \leftarrow \{s\}$ *// Création Queue (***Q***)*

$L \leftarrow \{s\}$ *// Création d'une liste(***L***)*

*//La matrice Adjacente*

*//La partie n°2 pour ordonner les plus courts chemins dans une Liste (L)*

*Tanque Q< >0*

$i \leftarrow Q$

*Pour* $j \in Adj(i)$

*Si* $d_j$ *inf alors*

$d_j \leftarrow d_i + 1$

$W_j \leftarrow W_i$

$j \rightarrow Q$

$j \rightarrow L$

*Fin Si*

*Si* $d_j$ *< > inf alors*

$d_j \leftarrow d_i + 1$

$W_j \leftarrow W_i + W_j$

*Fin Si*

*Si* $d_j$ *< > inf alors*

*Afficher ("problème dans l'opération de la détection ")*

*Fin Si*

*Fin tanque*

*//La partie n°2 pour calculer edge betweenus à l'aide de la distance et la matrice adjacent*

*Tanque L<> 0*

$i \leftarrow L$

$somme_j \leftarrow 0$

$\sigma_{ij} \leftarrow somme_j + 1$

*Pour* $j \in Adj(i)$

*Si* $d_i < d_j$ *alors*

$b_i \leftarrow 1 + somme_j$

*Fin si*

*Si* $d_i > d_j$ *alors*

$\sigma_{ij} \leftarrow W_j / W_i * b_i$

*Fin*

*Fin si*

*Fin pour*

*Fin Tanque*

***Fin***

### 3.2.5 Louvain

Louvain est une méthode heuristique basée sur l'optimisation de la modularité et elle est prouvée pour être rapide et évolutif sur les réseaux à grande échelle. L'optimisation des modularités fait en deux étapes [6], classé dans les algorithmes de ***"Optimisation gloutonne[1] "***, L'algorithme de détection de communauté de Louvain a été proposé à l'origine en 2008 comme méthode de déploiement rapide de communauté pour les grands réseaux. Cette approche est basée sur la modularité, qui tente de maximiser la différence entre le nombre réel d'arêtes dans une communauté et le nombre attendu d'arêtes dans la communauté. Cependant, l'optimisation de la modularité dans un réseau est "***NP-difficile"***, il faut donc utiliser l'heuristique[1]

- **Étape n°1** l'algorithme attribue une communauté différente à chaque nœud du réseau, pour chaque nœud, il a considéré les voisins et évalué le gain de modularité en supprimant le nœud particulier de la communauté actuelle et en le plaçant dans la communauté voisine. Le nœud sera placé dans la communauté voisine si le gain est positif et maximisez.
- **Étape n°2** dans la deuxième phase, l'algorithme construit un nouveau réseau en considérant les communautés trouvées dans la première phase comme des nœuds. Une fois la deuxième phase terminée, l'algorithme réappliquera la première phase au réseau résultant.

### 3.2.6 Les équations numériques et partie algorithmique

[1] L'heuristique est un algorithme qui permet rapidement dénouer des difficultés complexe d'optimisation, sans concevoir une modélisation formelle.

### 3.2.6.1 Les équations numériques :

- **La modularité** est une valeur de type flottant représente la qualité de la communauté, représente par la formule suivante

$$M = 1/2m \sum_{i,j}(A_{ij} - P_{ij})\delta(C_i, C_j) = 1/2m \sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\ \delta(C_i, C_j) \qquad (2.3.4)$$

$\boldsymbol{A_{ij}}$ *Représente la matrice adjacent associé par un graph avec N sommet telque 'i' inférieure ou égale à la valeur N, 'j'représente* ***les sommets***.

$\boldsymbol{k_i}$ *Représente* ***Le sommet n° i***.

$\boldsymbol{C_i}$ *Représente la communauté n° i.*

- **Optimisation la modularité** est une valeur représente l'optimisation de chaque modularité de manière aléatoire.

$$\Delta M = \left[\frac{\sum_{in} + 2K_{in}}{2m} - \left(\frac{\sum_{tot} + K_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right) - \left(\frac{K_i}{2m}\right)^2\right] \qquad (2.3.4)$$

- Pour améliorer la formule.

$$\Delta M = \left[\frac{K_{i,n}}{m} - \frac{2\sum_{tot} Ki}{(2m)^2}\right] \Leftrightarrow \Delta M_{m=} K_{i,n} - \frac{\sum_{tot} K_i}{2m} \qquad (2.3.4)$$

## 3.2.7 Principe et les étapes de l'algorithme Louvain avec détails

L'algorithme fonctionne sur l'invariant de boucle suivant

1. Création une matrice adjacente à l'aide d'un graph sans boucle et bien formaliser.
2. Création des partitions avec les sommets voisine.
3. Application de la formule M pour calculer la modularité ***Q***.
4. Pour commencer, chaque nœud est affecté à une communauté différente oucloison, Le nombre de partitions est égal au nombre de nœuds **N** .
5. Pour chaque voisin **j** du nœud **i**, on vérifie si la modularité globale augmente en déplaçant i de sa partition vers la partition **j** .

6. L'étape **n°2** est répétée pour tous les nœuds de manière séquentielle. C'est appelé une itération. L'itération est également répétée jusqu'à ce qu'aucune amélioration dela modularité.
7. L'idée est d'atteindre un maximum local de modularité après laquelle aucune augmentation supplémentaire de la modularité n'est possible. Veuillez noter qu'un nœud peut être et probablement visité plus d'une fois pour évaluer le changement de modularité en déplaçant ses voisins vers des partitions. [7]

### 3.2.8 Algorithme

*Entrées* $G^0=(V^0, E^0)$ *// Graphe non orienté,* $V^0$ *(représente le sommet) et* $E^0$ *(représente les arêtes) (Modèle de données initial karate club sous form txt)*

$\mathcal{Q}_{0(C)}$ *// La modularité de la communité égale zéro*

*Sortie* $\boldsymbol{G^k=(V^k, E^k)}$

***M*** *//Résultat de modularité.*

***Mod*** *// Valeur représente la modularité.*

***δMod***

$\boldsymbol{K_i}$ *//La somme des poids de tout arrêts avec sommet i.*

$\boldsymbol{K_{i,j}}$*// Le poids de l'arrêts entre sommet I avec sommet j.*

$m = \sum ki, j, (i, j) \in E^0$

***Debut***

$k \leftarrow 0$ *//Nombre de l'itérations*

$N \leftarrow V$ *//Nombre de repetition ou le nombre de sommet*

*Tanque k < V*

*Pour* $i \in V^K$

$M_i^k \leftarrow \{ \ i \ \}$

$Mod_{new} \leftarrow Mod\ (M)$

*Fin*

$Mod \leftarrow Mod_{new}$

*//Ordonner les sommets de manière aléatoire*

*Pour* $i \in V^k$

*best_communauté* $\leftarrow M_i^k$

*best_increase* $\leftarrow 0$

*sommetot* $\leftarrow 0$

*somme* $\leftarrow 0$

*Pour* $M^t \in C^k$

$M_i^k = M_i^k \leftarrow \{i\}$

$i \leftarrow i + 1$

$sommetot^{M_i^K} \leftarrow somme_{\alpha \in M_i^k} K_\alpha - K_i$

$sommetot^{M_i^K} \leftarrow somme_{\alpha \in M_i'^k} K_\alpha - K_i$

*Si* $\delta Mod_{M_i^k \to M'^k_i} >$ *best_increase alors*

*best_increase* $\leftarrow \delta Mod_{M_i^k \to M'^k_i}$

*best_communauté* $\leftarrow M'^k_i$

$M_i^k \leftarrow M'^k_i \cup \{i\}$

*Fin Si*

*Si* $\delta Mod_{M_i^k \to M'^k_i} <=$ *best_increase alors*

$M'^k_i \leftarrow M'^k_i \cup \{i\}$

*Fin si*

*Fin pour*

*Fin pour*

*//l'opération de calcul de la modularité avec n fois*

*Tanque i>N //nombre de sommets*

$Mod_{new} \leftarrow$ *Mod(M) // l'équations numéro n°3 dans le document*

*Fin pour*

*Fin pour*

*Fin Tanque*

*Si* $Mod_{new}$ *= Mod(M) alors // Condition l'égalité de la modularité.*

*Break ;*

*Fin si*

$Mod \leftarrow Mod_{new}$

*//Combiner les communautés*

$V^{k+1} \leftarrow C^k$

$E^{k+1} \leftarrow e\ (C_u^k, C_v^k)$

$G^{k+1} = (V^{k+1}, E^{k+1})$

$k \leftarrow k+1$

***Fin***

#### 3.2.8.1 La compléxité de algorithme

- **Best case** $O(nlogn)$ **.**
- **Worst case** $O(nlogn)$ .

### 3.2.9 Les conditions nécessaires pour appliquer les algorithmes

A travers les des études, nous avons constaté que des conditions doivent être fournies pour appliquer des algorithmes

- La taille de donnée est limitée.
- Un graph non orienté avec poids et bien formalisée i.e. sans boucle et mal connecté.

## 3.3 Justification choix de l'algorithme

Pour différents types et échelles de réseaux complexes, les chercheurs ont proposé de nombreux algorithmes de détection. Ces algorithmes peuvent être grossièrement divisés en algorithme d'optimisation de modularité, algorithme de percolation clique, algorithme de propagation d'étiquette et algorithme de partitionnement hiérarchique, A travers des comparaisons et des conclusions basées sur plusieurs aspects, deux algorithmes ont été retenus,

chacun avec des avantages différents et communs, voici quelques justifications qui ont été retenues dans notre sélection parmi de nombreux algorithmes.

1. La rapidité concernant temps de l'exécution.
2. Populaire, connus et utilisées dans les réseaux internet.
3. Basée sur la modularité.
4. Efficacité concernant le nombre de communauté par rapport le nombre d'individus spécifiquement les modèles avec des millions personnes ou des objets et les interactions.
5. La comparaison et benchmarks à l'aide les modèles données réelles.
6. La simplicité concernant la partie applications et la traduction algorithmiques et les modèles intégrées dans langage python pour traduire les relations et application numériques.

## 3.4 Benchmarks et évaluation de Louvain et Girvan Newman

L'évaluation d'un algorithme de détection de communauté est une tâche complexe en raison de l'absence d'une définition partagée et universellement acceptée de la communauté (l'évaluation des résultats fournis par un algorithme de détection communautaire est l'un des les tâches les plus difficiles de l'analyse de réseau complexe, car il n'y a pas définition universellement acceptée de ce qu'est une communauté). Dans la littérature, l'un des moyens les plus courants d'évaluer les performances d'un algorithme de détection de communauté est de *comparer sa sortie avec des communautés* de vérité terrain données en utilisant des *métriques coûteuses* en calcul (c'est-à-dire des informations mutuelles normalisées). Nous avons utilisé la méthode de **F1** (un *score F1)* moyen qui capture le niveau d'approximation atteint par les partitions de réseau obtenues grâce à des algorithmes de **découverte de communauté et méthodes de benchmarks**

La figure suivante, représente la partie algorithmique et leurs interactions avec **l'application de détection de communauté**.

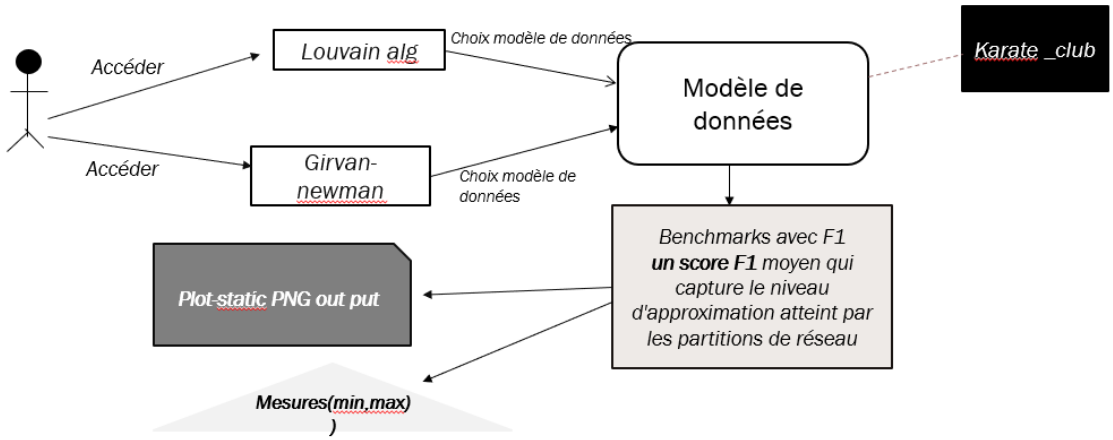


Figure – 8 L'opération de benchmarks et l'algorithme Louvain et Girvan-new man.

Évaluer la qualité des algorithmes de détection communautaire est une tâche difficile en particulier car le problème lui-même est mal pose, chaque algorithme optimise une qualité différente métrique introduisant sa propre définition de communauté.

## 3.5 Analyse des besoins de l'application

### 3.5.1 Déterminer les besoins fonctionnels et non fonctionnels

C'est la première étape de la conception, Elle consiste à analyser la situation pour tenir compte des contraintes, des risques et de tout autre élément pertinent et assurer un logiciel ou un processus répondant aux besoins du client.

### 3.5.2 Besoins functionals

Un besoin fonctionnel spécifie l'action qu'un système doit être capable d'effectuer, hors contrainte physique : besoin spécifiant un comportement d'entrée/sortie d'un système, Dans ce contexte notre application de la détection de communauté, implémente les fonctionnalités suivantes :

- L'application doit fournir la **fonction d'ajout d'utilisateur afin de réduire la possibilité d'entrer** et d'utiliser l'application (*droit d'accès et le rôle de l'utilisateur*).

- **Produire des images révisables et ajouter des informations et des notes utiles** pour déterminer les relations sociales entre les individus, en conjonction avec la qualité des informations ou des bibliothèques disponibles dans l'étude des sociétés (*détection de communautés*), opérations d'extraction des images (*la possibilité d'extraire un type spécifique d'images et de documents et de les placer dans une base de données spécifique*).

- **La multiplicité des algorithmes** et méthodes disponibles et capables d'étudier la communauté (*application riche par rapport aux algorithmes disponibles en ligne*).

- L'application contribue à **ajouter une fonctionnalité qui facilite à l'utilisateur l'analyse des images e**t des nœuds avec différents interactions et couleurs.

- L'application offre la possibilité **de stocker les données comme des remarques**.

### 3.5.3 Besoins non fonctionnels

Il s'agit des besoins qui caractérisent le système. Notre application doit répondre aux critères suivants :

- **Ergonomie et souplesse** l'application doit offrir une interface conviviale et ergonomique Exploitable par l'utilisateur en envisageant toutes les interactions possibles à l'écran tenu.
- **Rapidité l'application** doit optimiser les traitements pour avoir un temps de génération de Schéma raisonnable.
- **Efficacité** l'application doit être fonctionnelle indépendamment de toutes circonstances pouvant entourer l'utilisateur
- **Maintenabilité et l'extensibilité** le code de l'application doit être lisible et compréhensible Afin d'assurer son état évolutif et extensible par rapport aux besoins de marché.
- **La stabilité** l'application doit être très stable durant son exécution.
- **La sécurité** l'application les données doivent être sécurises pour éviter toutes utilisations Non permises.

- **La compatibilité** l'application doit être compatible avec plusieurs types d'ordinateurs.
- **La généricité** consiste à définir des algorithmes identiques opérant sur des données de types différents
- **La modularité** Le développement du code des modules peut être attribuée à des (groupes des personnes différentes, qui effectuent leurs tests unitaires et similaires de façons indépendamment.

## 3.6 Objective de projet

Afin de mettre en œuvre les étapes de l'industrie du programme, nous devons définir les objectifs du projet qui donnent une vision future du résultat final

- Pour les chercheurs, faciliter et fluidifier l'accès les graphes et obtenir les liens communs.
- Améliorer le suivi des données sortantes et identifié les liens dans les graphes complexe.
- Réaliser une meilleure conservation des données afin de pourvoir faire des études par la suite et avoir toutes sortes de statistiques et benchmark.
- Prouver qu'il existe un moyen d'améliorer la gestion des données avec théories des graphes.
- Améliorer la détection de communauté avec les algorithmes proposés «*Louvain*» et « *Girvan new man*».
- Etablir un control d'accès aux données des chercheurs et générer les mots de passe afin d'augmenter la sécurité du système.
- Développer la fonctionnalité de manipulation d'images et acquérir une compréhension plus précise des sociétés en identifiant des points communs et d'harmonie.
- Faciliter l'application pratique des algorithmes d'identification communautaire à l'aide d'un modèle simple avec **N** sommet inférieure à la valeur **10** ou un modèle génératif comme un simple document sous forme « *Txt » ou « YML »,*

## 3.7 Identification les principales des utilisateurs

Nous avons dans notre application 03 catégories principales d'utilisateurs

- **Les chercheurs** il peut travailler sur le suivi et l'analyse de la communauté en entrant certaines données telles que le nombre d'utilisateurs dans l'organisation ou tout groupe d'individus au sein d'un système commun, virtualités l'image avec un éditeur (*la communauté image éditeur*).
- **Les développeurs** il peut effectuer plusieurs rôles tels que la surveillance et l'analyse des données pour chaque communauté, créations les modèles de données avec un éditeur de Txt intégrer (*La communauté détection système*).
- **Les employeurs** il peut analyser la communauté, extraire des points communs et ajouter des commentaires qui sont stockés dans une base de données, imprimer les informations nécessaires, imprimer les résultats (*Employeur système).*

## 3.8 Conclusion

Dans ce chapitre nous avons détails les algorithmes détection de la communauté proposé pour générer un détection de communauté et l'objective de notre projet fin d'étude respectivement les utilisateurs principale de l'application. .

# Chapitre 4 La méthodologie de la conception et les diagrammes.

## 4.1 Introduction

Au cours de ce chapitre, nous allons détailler la conception de notre système, nous avons utilisé la méthode « *RUP* » et méthode adaptive « *PXP* » au cours de notre conception et nous allons à présent détailler les différents diagrammes, en respectant les étapes d'extraction des informations et en les classant sous une forme spécifique et facilement compréhensible, nous allons élaborer les diagrammes de classes de conception et de séquence de cas d'utilisation, d'activité, flux de données, le diagramme qui suit va nous montrer les classes qui apparaissent dans le cas d'utilisation « *Détection communauté système* ».

## 4.2 Méthodologie avec PXP et RUP

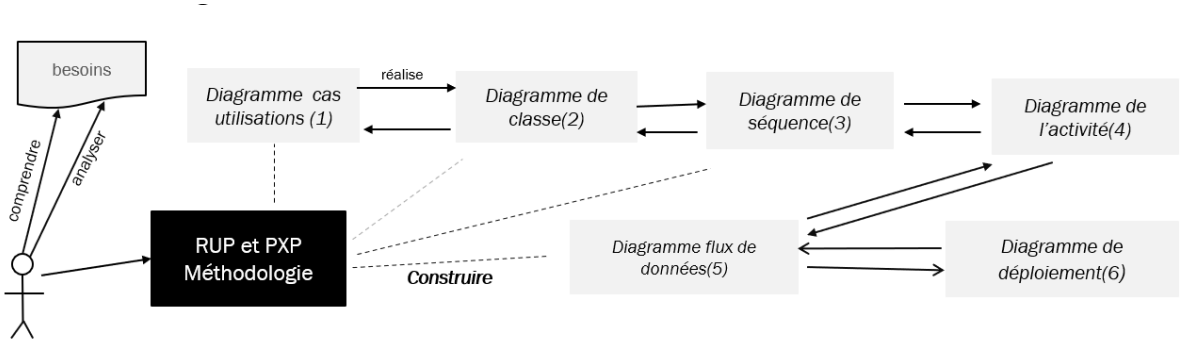


Figure 9 – nombre de diagramme à l'aide RUP et la méthode adaptive PXP.

### 4.2.1 Méthode Personal Extreme Programming « PXP »

Est un processus de développement logiciel pour une équipe composée d'une seule personne. Il est basé sur les valeurs de *l'Extreme Programming* **(XP)** c'est-à-dire *la simplicité*, *la communication*, le retour d'expérience et le courage. Il fonctionne en conservant les aspects importants de **XP** et en affinant les valeurs afin qu'elles puissent s'intégrer dans une situation de programmeur isolé. **PXP** peut encore être affiné et amélioré. Il est dans la tradition des praticiens

**XP** de faire varier **XP** pour englober tout ce qui fonctionne. Nous espérons que **PXP** héritera également de ces racines pragmatiques. Abandonner les principes **XP** comme la programmation en binôme n'est pas nécessairement une tragédie. Nous croyons toujours que suivre strictement **XP** est un moyen plus efficace de poursuivre des projets multi-personnes. Mais nous sommes également convaincus que de nombreuses pratiques et méthodes **XP** peuvent être appliquées au travail individuel.

### 4.2.2 Méthode « RUP »

Est le guide ultime pour l'attribution des tâches et des responsabilités au sein d'une organisation de développement et le développement de logiciels de haute qualité qui répondent aux besoins et aux exigences de ses utilisateurs. Il a été initialement créé par Rational Software Corporation qu'IBM a racheté en 2003. [7]

#### 4.2.2.1 Utilisateur interne et externe

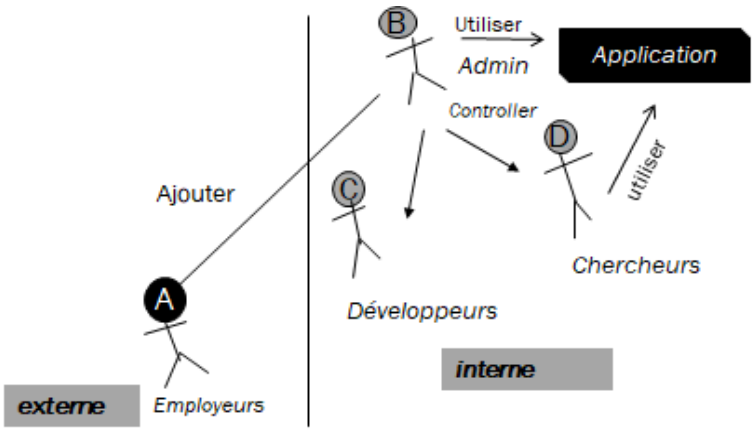


Figure 10 – System utilisateurs et le système détection de communauté

Nous avons dans notre application « **03 »** catégories d'utilisateurs principales

- **Les acteurs internes**

1. Les chercheurs comprennent de nombreuses personnes intéressées par la recherche sur la société et incluent des étudiants et des professeurs**.**

2. Les développeurs comprennent les informaticiens, les ingénieures de l'informatique.

- **Les acteurs externes par rapport l'objectivent de notre système**

1. Les employeurs comprennent service scolaire, entreprise.

Les acteurs interagissent dans nos applications sont les employeurs et les chercheurs et les développeurs.

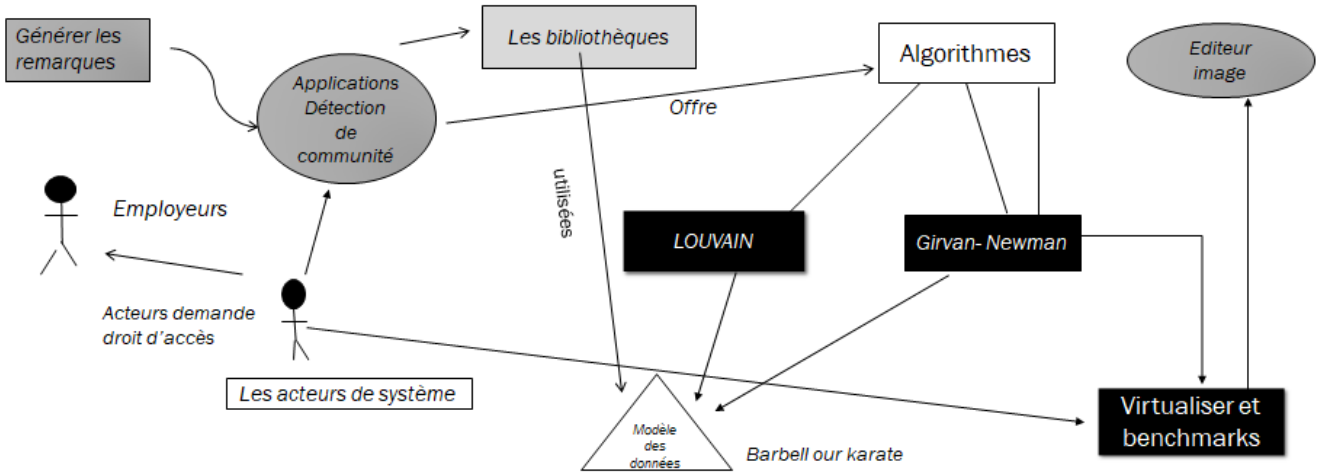


Figure 11 – Les acteurs et l'opération de détection de communauté.

## 4.3 Les diagrammes graphiques

### 4.3.1 Diagramme de cas d'utilisation

Définir les exigences fonctionnelles et opérationnelles du système (*produit*) en définissant un scénario d'utilisation convenu entre l'utilisateur final.

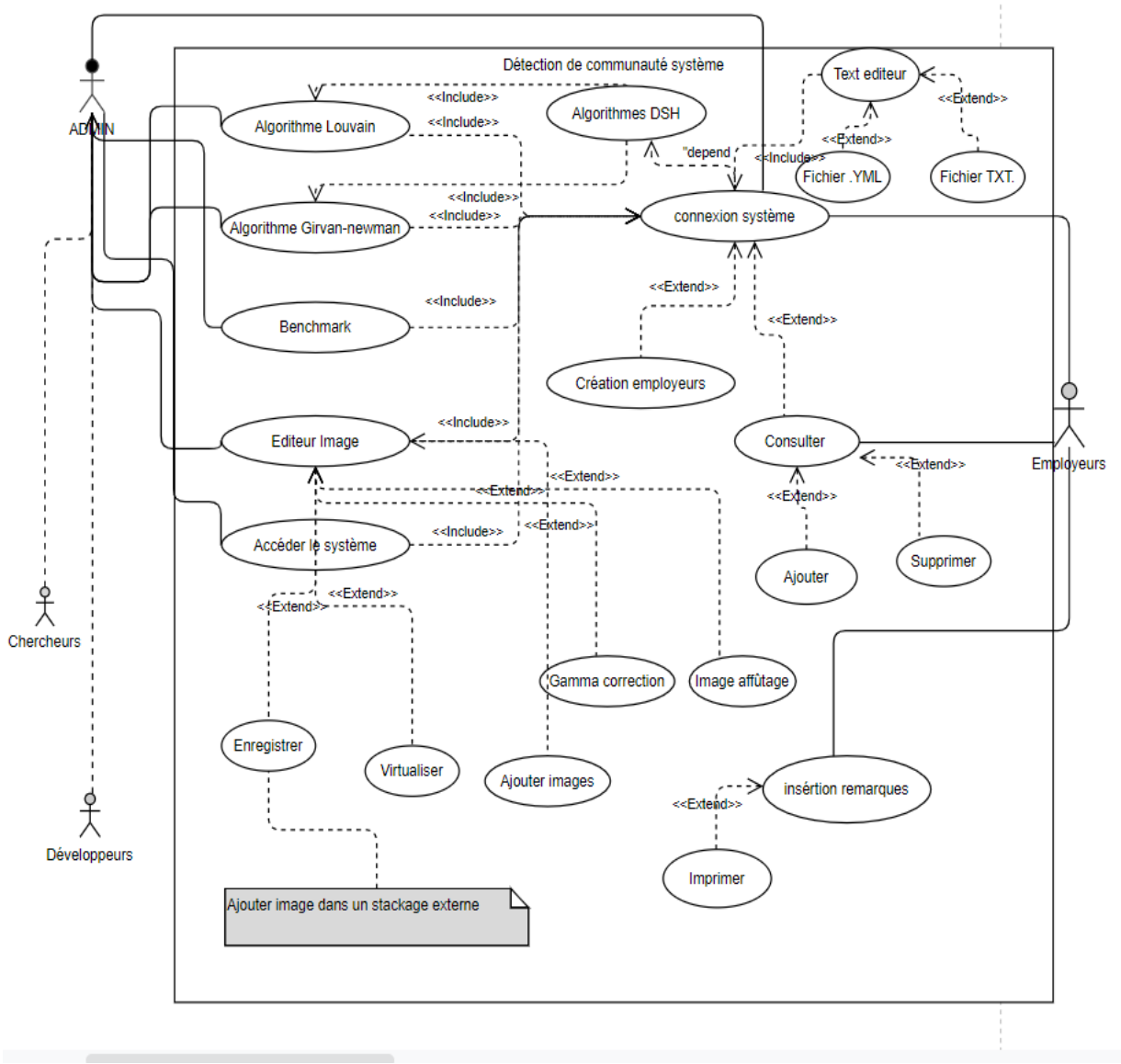


Figure 12 – Diagramme de cas d'utilisation.

### 4.3.2 Diagramme de classe

Le diagramme de classe permet d'exprimer comment les objets et les classes vont être définies, ainsi que les relations qui existent entre les différentes classes.

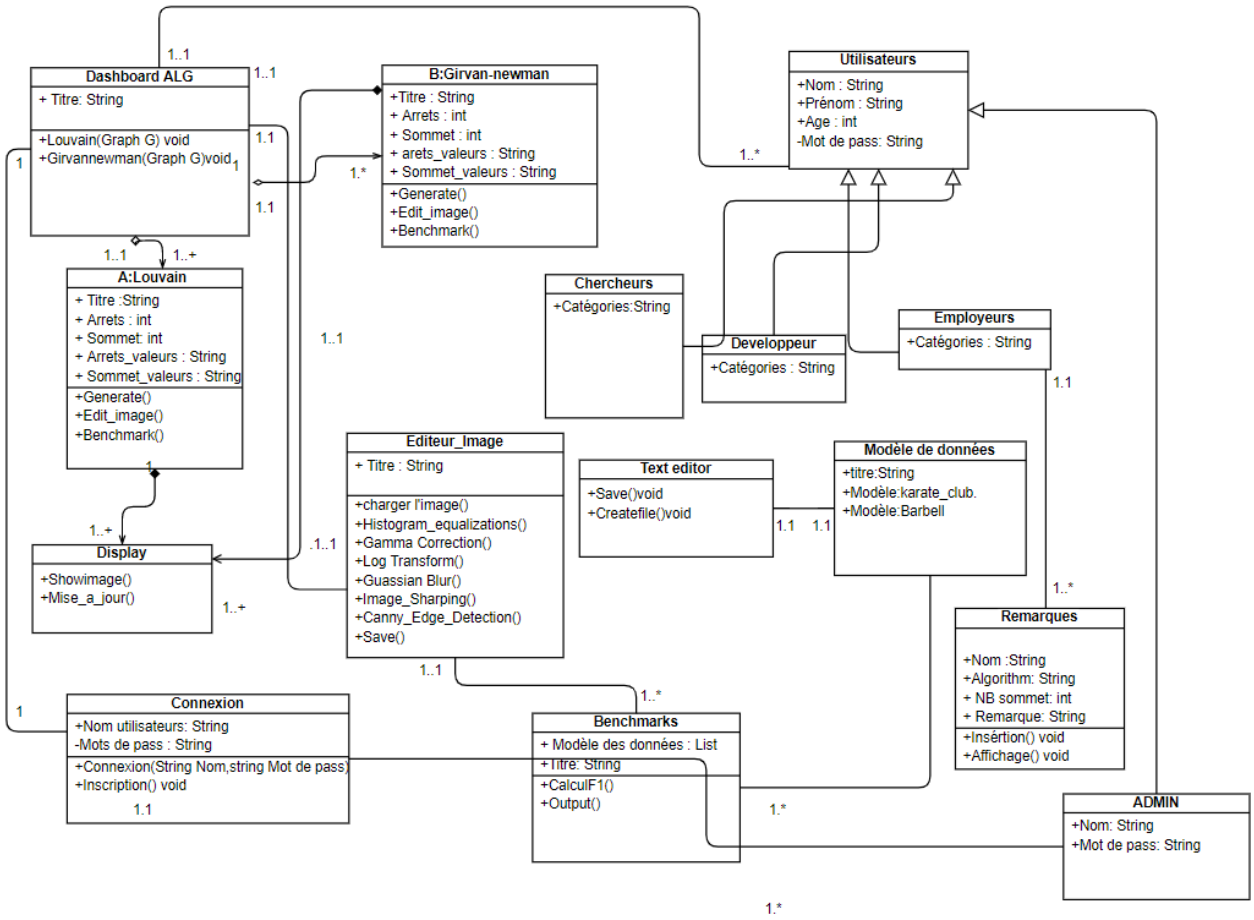


Figure 13 – Diagramme de classe.

### 4.3.3 Diagramme de séquence

Le diagramme de séquence contribue à définir les relations entre l'utilisateur et l'application dans un ordre précis et compréhensible, afin de répondre au scénario de chaque fonctionnalité présente dans l'application.

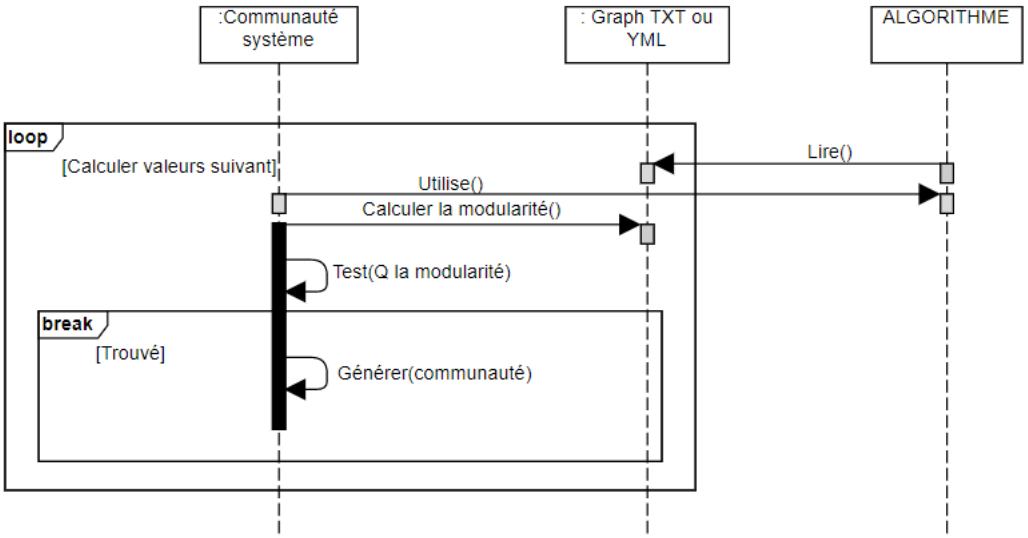


Figure 14 – Diagramme de séquence.

### 4.3.4 Diagramme d'activité

Un diagramme d'activités est une représentation graphique qui montre le travail d'activités et de procédures qui doivent être respectées simultanément et progressivement tout en conservant la caractéristique de la répétition.

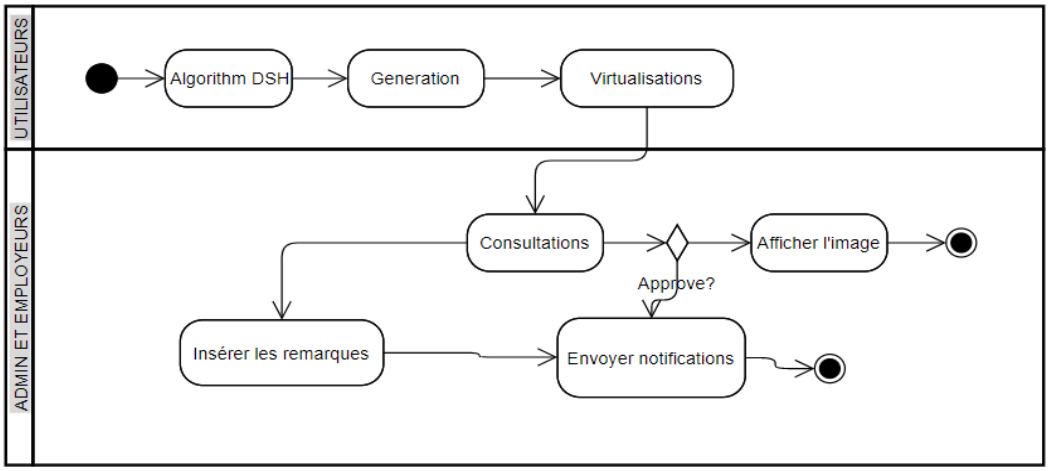


Figure 15 – Diagramme d'activité.

### 4.3.5 Diagramme de déploiement

Il clarifie les caractéristiques physiques qui doivent être démontrées dans les opérations qui ont lieu au sein de l'application, telles que les appareils utilisés et les outils qui sont saisis afin de garantir que l'application fonctionne sous une forme intégrée.

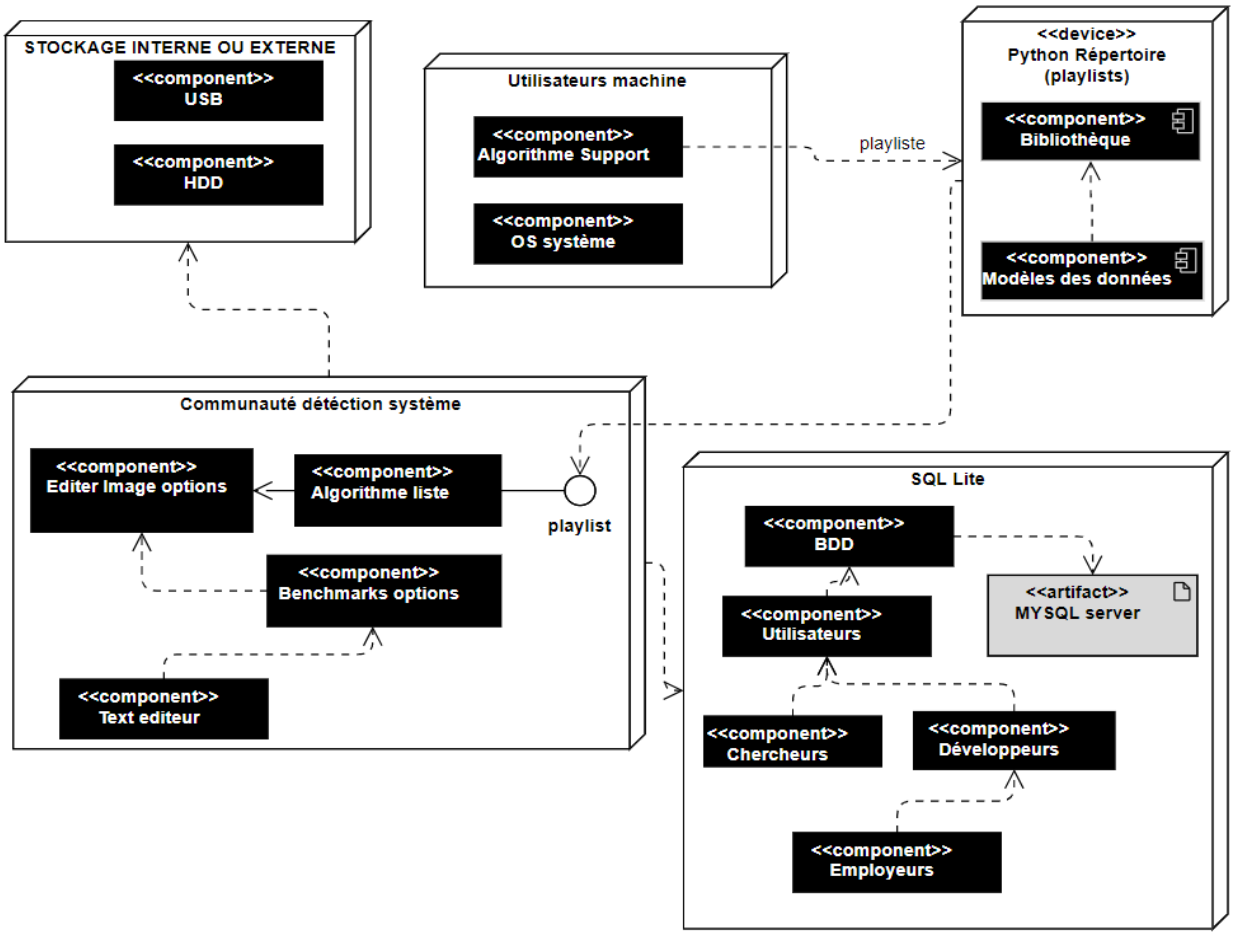


Figure 16 – Diagramme de déploiement.

### 4.3.6 Diagramme flux de données

Le diagramme flux de données montre comment les informations circulent et changent au sein de l'application sur les sorties et les entrées de chaque entité et ne spécifie pas le flux du contrôleur.

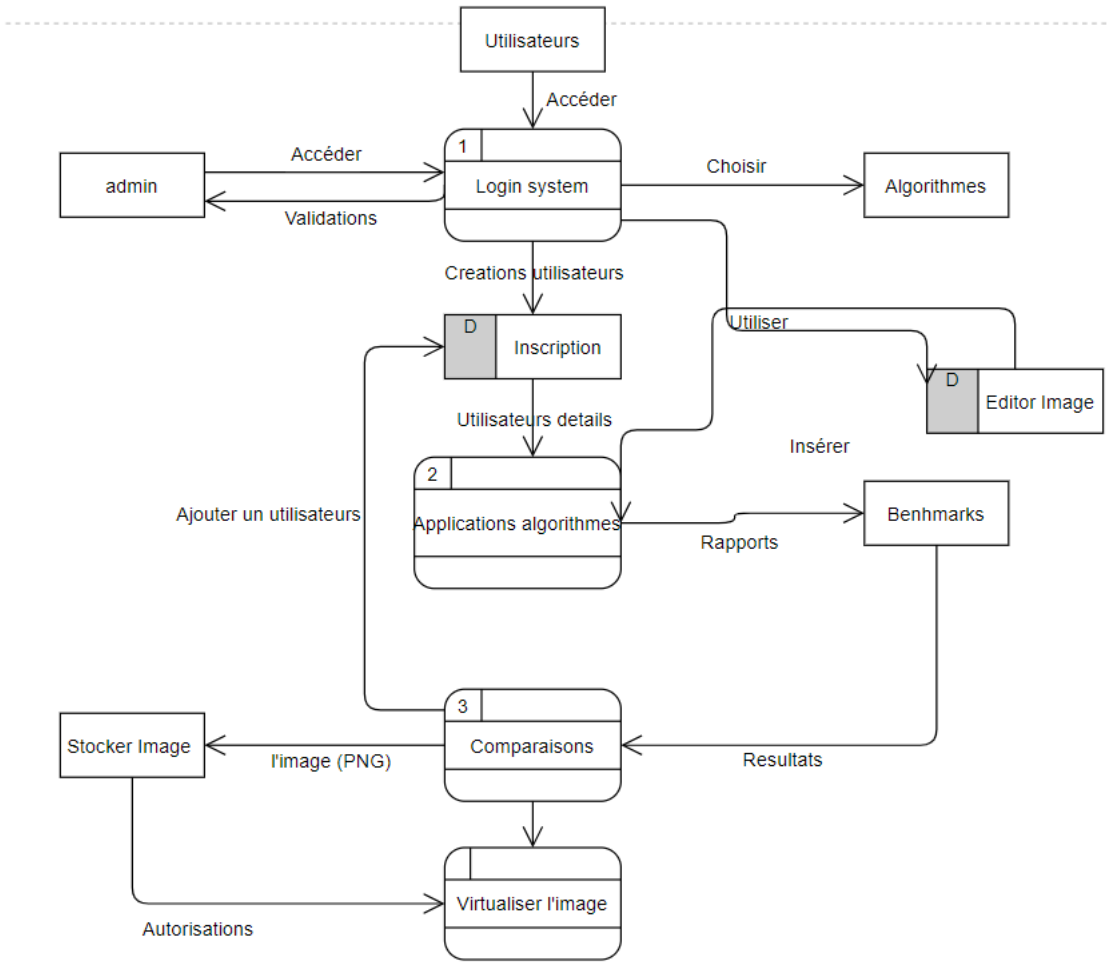


Figure 17 – Diagramme flux de données.

## 4.4 Les opérations et utilisateurs de chaque itération

Grâce au diagramme, nous pouvons apprendre les caractéristiques avec lesquelles l'application fonctionne et étudier les réactions que l'utilisateur effectue lors de l'échange d'informations qui se caractérisée par son mouvement dans l'environnement de l'application, l'une des choses les plus importantes qui peuvent aboutir à l'obtention d'un une meilleure compréhension du fonctionnement du programme et de l'environnement théorique pour répondre aux désirs de l'utilisateur.

### 4.4.1 Diagramme de séquence

- **Enter ()** un processus basé sur un mot de passe qui donne accès à l'application
- **Click_connexion ()** la fonctionnalité permet d'accéder à l'application en vérifiant les informations saisies par l'utilisateur dans la base de données.
- **Access ()** contribue à la saisie et au passage au panneau utilisateur et à l'accès aux informations.
- **Find Community ()** c'est un processus qui contribue à l'étude et à l'analyse d'une base de données en distinguant deux types d'algorithmes, une fonctionnalité accessible à tous les utilisateurs.
- **Benchmarks ()** un processus qui compare le fonctionnement des algorithmes disponibles pour l'utilisateur est détermine lesquels sont les plus efficaces et les plus fiables.
- **GetEdgesdetails ()** un processus qui capture les informations nécessaires pour représenter le poids d'une personne ou un objet.
- **DisplayGraphdetails()** un processus pour afficher un graph avec un nombre de n inférieure 10.
- **Show succès message()** Il s'agit d'une alerte indiquant que l'opération s'est terminée avec succès et que les relations ont été extraites à l'aide des algorithmes disponibles dans le panneau de l'utilisateur ( *Girvan new man et Louvain algorithme*).

## 4.5 Conclusion

En clair, comme nous venons de le voir, ce chapitre était consacré, à la conception des diagrammes, des cas d'utilisation qui nous ont permis de dégager l'architecture générales de notre application représentée ici par ce diagramme de déploiement qui nous renseigne sur la relation homme Machine de notre application qui sera détaillée dans le prochain chapitre.

# Chapitre 5 L'implémentation de l'application et les algorithmes.

## 5.1 Introduction

Dans ce chapitre nous allons présenter notre environnement de travail (*le côté matériel et logiciel*), ainsi que des captures d'écran pour montrer les fonctionnalités de notre application.

## 5.2 Langage de programmation

Nous avons choisi langage python, est un langage de programmation généraliste interprété de haut niveau, et par rapport notre projet de la détection de communauté car les modèles et algo rithmes bénéfice avec les points suivants

1. La simplicité de traduction numérique à l'aide de la théorie des graphes.
2. Python est populaire et largement utilisé, il vous offre donc un meilleur support communautaire.
3. Le code Python peut s'exécuter sur n'importe quelle machine, que ce soit Linux, Mac ou Windows.
4. La génération de documentation, les tests unitaires, les navigateurs Web, le filetage, les bases de données, le CGI, le courrier électronique, la manipulation d'images.
5. Python permet aux développeurs de créer des fonctions avec moins de lignes de code.

## 5.3 Environnement de développement

### 5.3.1 Environnement logiciel

Nous avons utilisé pour le développement de notre application le langage python sous *Python IDLE*, ainsi que le *SQLITE* (système de gestion de base données).

#### 5.3.1.1 SQLITE SQLite

Est un système de gestion de base de données relationnelle (*SGBDR*) contenu dans une bibliothèque C. Contrairement à la plupart des autres systèmes de gestion de base de données, *SQLite* n'est pas un moteur de base de données client-serveur mais est intégré au programme final.

1. La lecture et l'écriture à partir d'une base de données *SQLite* sont plus rapides que la lecture et l'écriture de fichiers directement à partir du disque. L'application charge uniquement les données dont elle a besoin, plutôt que de lire.
2. L'intégralité du fichier et de conserver une analyse complète en mémoire. Les petites modifications écrasent uniquement les parties du fichier qui changent, ce qui réduit le temps d'écriture et l'usure des disques *SSD*.
3. Les problèmes de performances peuvent souvent être résolus, même plus tard dans le cycle de développement, à l'aide de *CREATE INDEX*, ce qui permet d'éviter des efforts coûteux de refonte, de réécriture et de nouveau test.

#### 5.3.1.2 Visual Paradigme

Est un logiciel de création de diagrammes dans le cadre de la programmation. Tout en un, il possède plusieurs options permettant une large possibilité de modélisation en *UML*. Il offre de nombreux outils pour créer différents types de schémas comme les diagrammes d'exigences et de cas d'utilisation, permet de générer des codes sources en divers langages comme le Java et python.

#### 5.3.1.3 Python IDLE

Est un environnement de développement intégré pour Python, qui a été fourni avec l'implémentation par défaut du langage depuis l'invention de langage *2008*. Il est fourni en tant que partie facultative de l'empaquetage Python avec de nombreuses distributions Linux. Il est entièrement écrit en Python et dans la boîte à outils ***Tkinter*** *GU* [8].

### 5.3.2 Les bibliothèques utilisées dans l'application de détection de communauté

- **From tkinter import *** ce *framework* fournit aux utilisateurs Python un moyen simple de créer des éléments d'interface graphique à l'aide des widgets trouvés dans la boîte à outils *Tk*. Les widgets Tk peuvent être utilisés pour construire des boutons, des menus, des champs de données, etc. dans une application Python.
- **Import random** ce module implémente des générateurs de nombres pseudo-aléatoires pour diverses distributions.
- **Import time** la fonction renvoie le nombre de secondes écoulées depuis l'époque, pour afficher le temps à l'aide l'horloge intégrée dans votre os (soit Linux, Windows).
- **Import os** Le module OS de Python fournit des fonctions pour interagir avec le système d'exploitation. Le système d'exploitation fait partie des modules utilitaires standard de Python. Ce module un moyen portable d'utiliser les fonctionnalités dépendantes du système d'exploitation.
- **Import tkMessageBox** le module *« tkMessageBox »* est utilisé pour afficher des boîtes de messages dans vos applications, Ce module fournit un certain nombre de fonctions que vous pouvez utiliser pour afficher un message approprié.
- **Import networkx** est une bibliothèque Python pour l'étude des graphes et des réseaux.
- **Import NumPy** est un package de traitement de tableau à usage général. Il fournit un objet tableau multidimensionnel hautes performances et des outils pour travailler avec ces tableaux. C'est le package fondamental pour le calcul scientifique avec Python.
- **Import win32api** ce module contient des constantes liées à la programmation Win32. Il ne fait pas partie de la version Python 2.6, mais devrait faire partie du téléchargement du projet pywin32.
- **Import win32print** un module encapsulant l'**API** d'impression Windows.
- **Import traceback** un traceback est un rapport contenant les appels de fonction effectués dans votre code à un moment précis, c'est-à-dire que lorsque vous

obtenez une erreur, il est recommandé de le tracer en arrière (traceback). Chaque fois que le code obtient une exception, le traceback donnera les informations sur ce qui s'est mal passé dans le code

- **From tkinter.filedialog import askopenfilename** fonction pour afficher une boîte de dialogue d'ouverture de fichier qui permet aux utilisateurs de sélectionner un fichier. Utilisez la fonction *askopenfilenames()* pour afficher une boîte de dialogue d'ouverture de fichier qui permet aux utilisateurs de sélectionner plusieurs fichiers
- **Import PySpark** PySpark est une interface pour Apache Spark en Python. Il vous permet non seulement d'écrire des applications Spark à l'aide d'API Python, mais fournit également le shell PySpark pour analyser de manière interactive vos données dans un environnement distribué. PySpark prend en charge la plupart des fonctionnalités de Spark telles que Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) et Spark Core.

## 5.4 Fonctionnement de l'application

### 5.4.1 Connexion détection communauté système

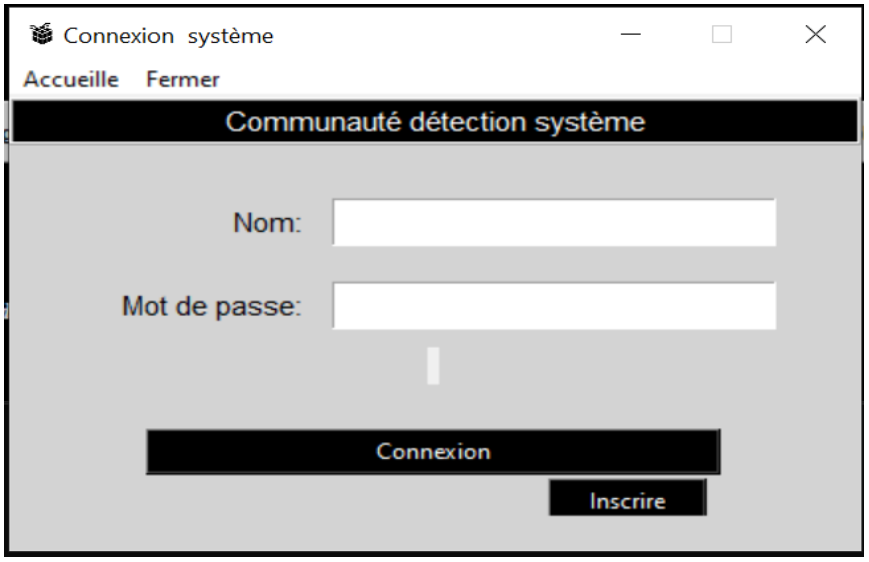


Figure 18 – L'interface droite d'accès système

La figure 18 montre la fenêtre *connexion système* ou l'utilisateur d'entrer avec son nom d'utilisateur et son mot de passe, *L'interface droite d'accès* que demander aux utilisateurs de se connecter présente presque toujours des avantages à la fois pour l'entreprise et pour l'utilisateur. De la sécurité aux préoccupations d'entreprise en passant par le support client, la puissance de la connexion de notre application de détection de communauté

### 5.4.2 Dashboard system ou la fenêtre d'accueil du système

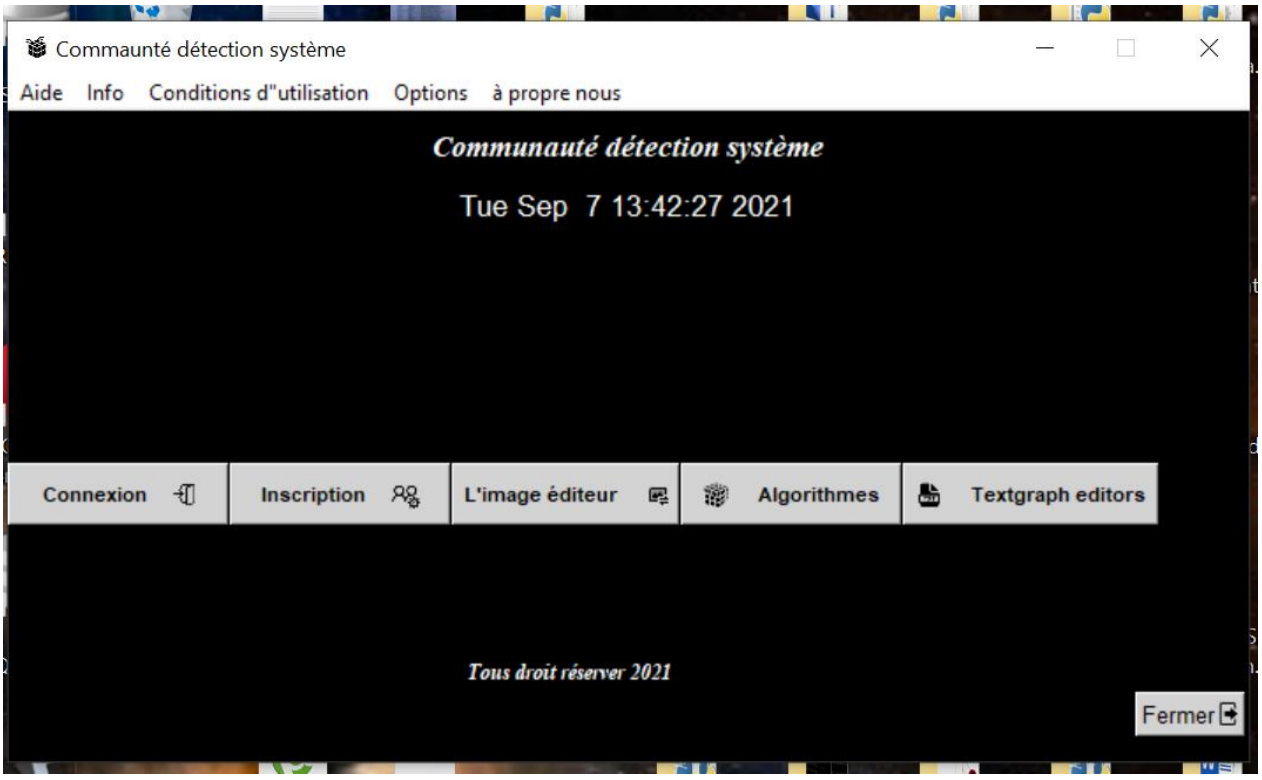


Figure 19 – Montre la fenêtre d'accueil du système détection de communauté

La figure 19 montre la fenêtre d'accueil du système détection de communauté, le tableau de bord sont fondamentaux pour toute application construite ou inclut des données, ce qui est aujourd'hui presque toutes les applications

### 5.4.3 La fenêtre choix d'utilisateurs

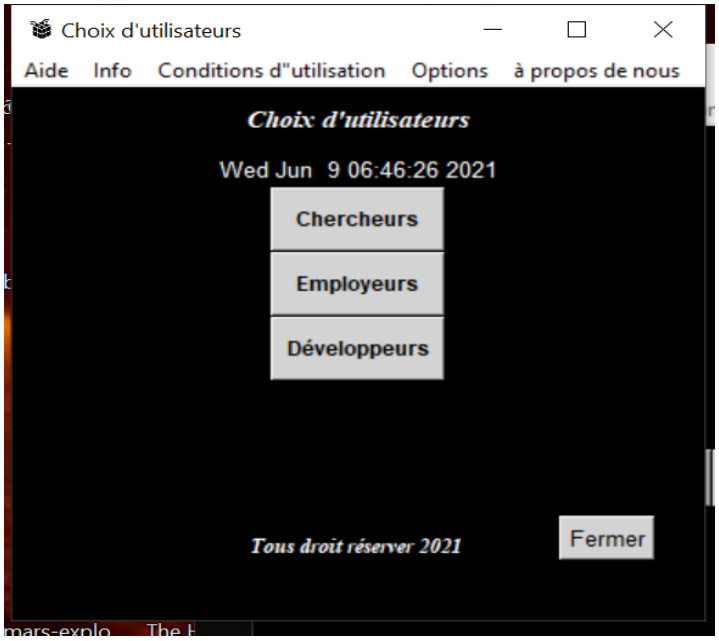


Figure 20 – Choix d'utilisateurs.

La figure 20 montre la fenêtre choix d'utilisateurs en 03 catégorie chercheurs, employeurs, développeurs.

### 5.4.4 Les employeurs enregistrement fenêtrent

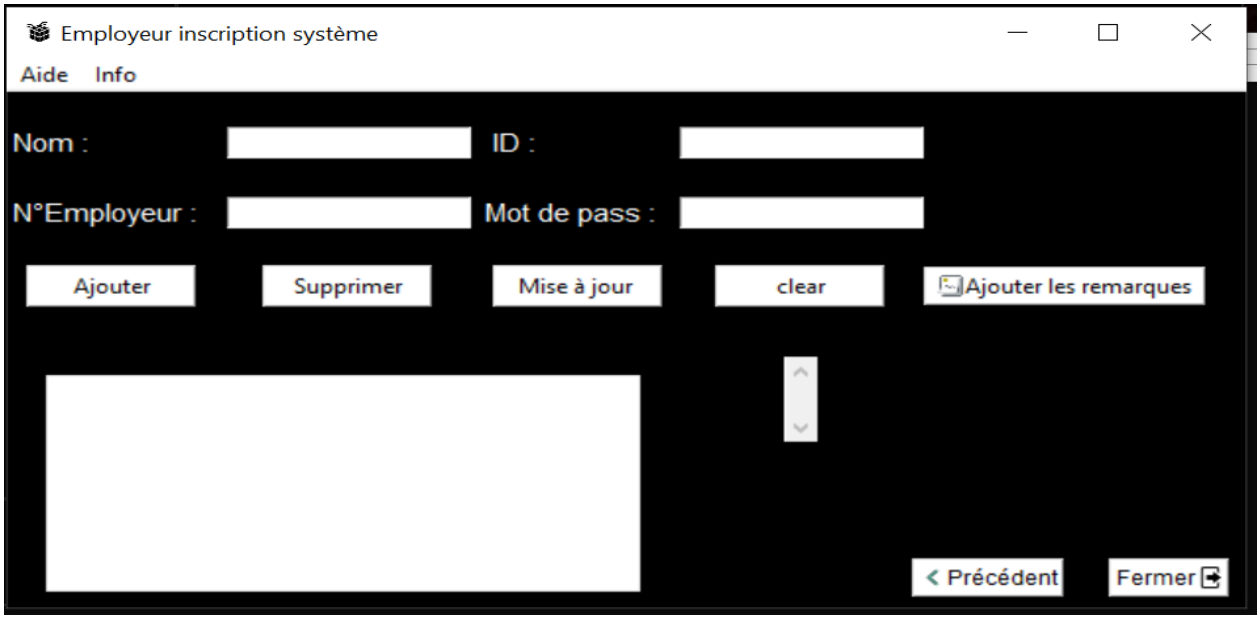


Figure 21 – Enregistrement les employeurs

La figure 21 montre la fenêtre de l'inscription les utilisateurs de type employeurs sur l'application de détection de communauté.

### 5.4.5 Louvain et Girvan new-man algorithme détection de communauté


Figure 22 – Louvain et Girvan new-man algorithme Dashboard.

La figure 22 montre la fenêtre de l'application Louvain algorithme sur les graphes avec un nombre de sommets inférieure à M et les arêtes inférieure à la valeur N.

### 5.4.6 Ajouter les remarques par rapport l'algorithme et nombre noauds

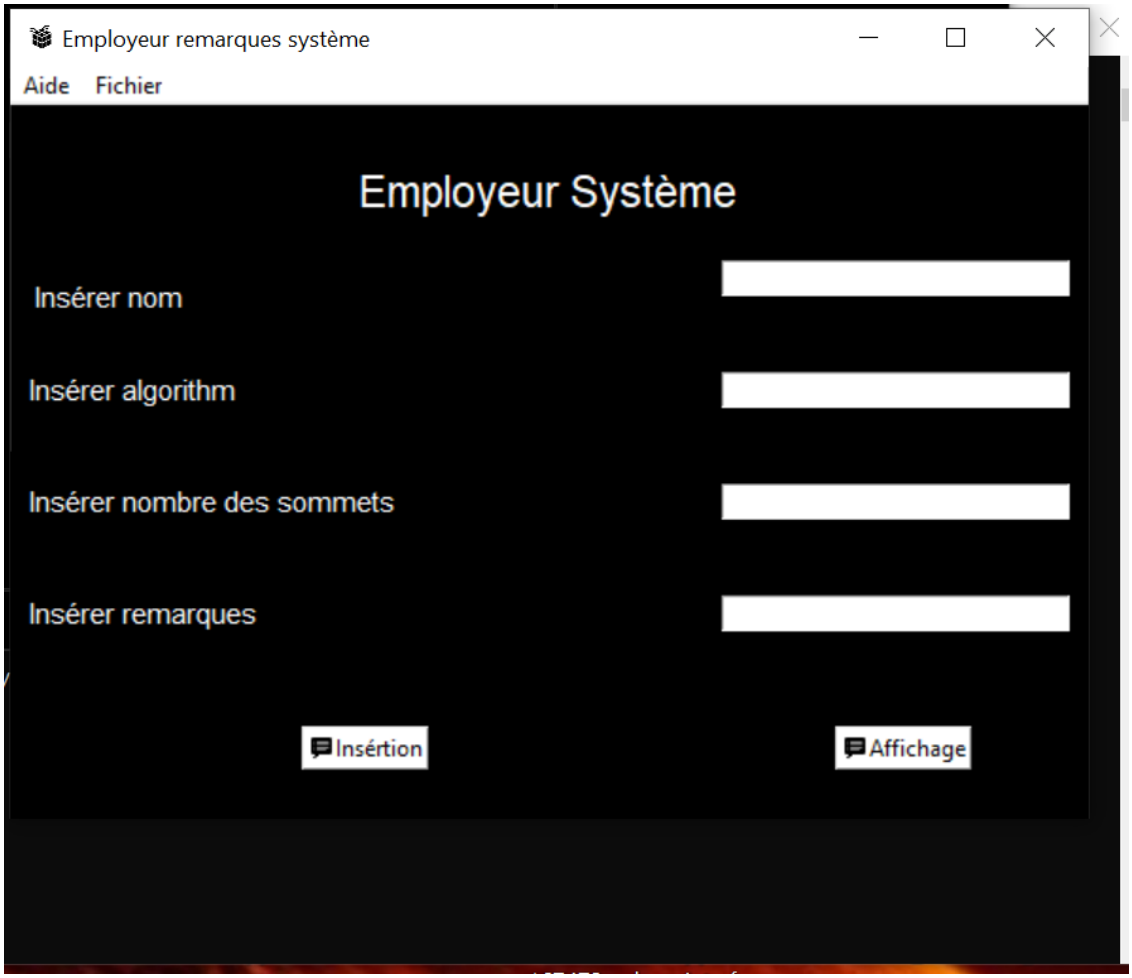


Figure 23 – Ajouter les remarques

La figure 23 montre la fenêtre de l'ajoute les remarques ou l'utilisateur de type employeurs peut insérer les remarques dans la base ***SQL***.

### 5.4.7 Fenêtre inscription les chercheurs

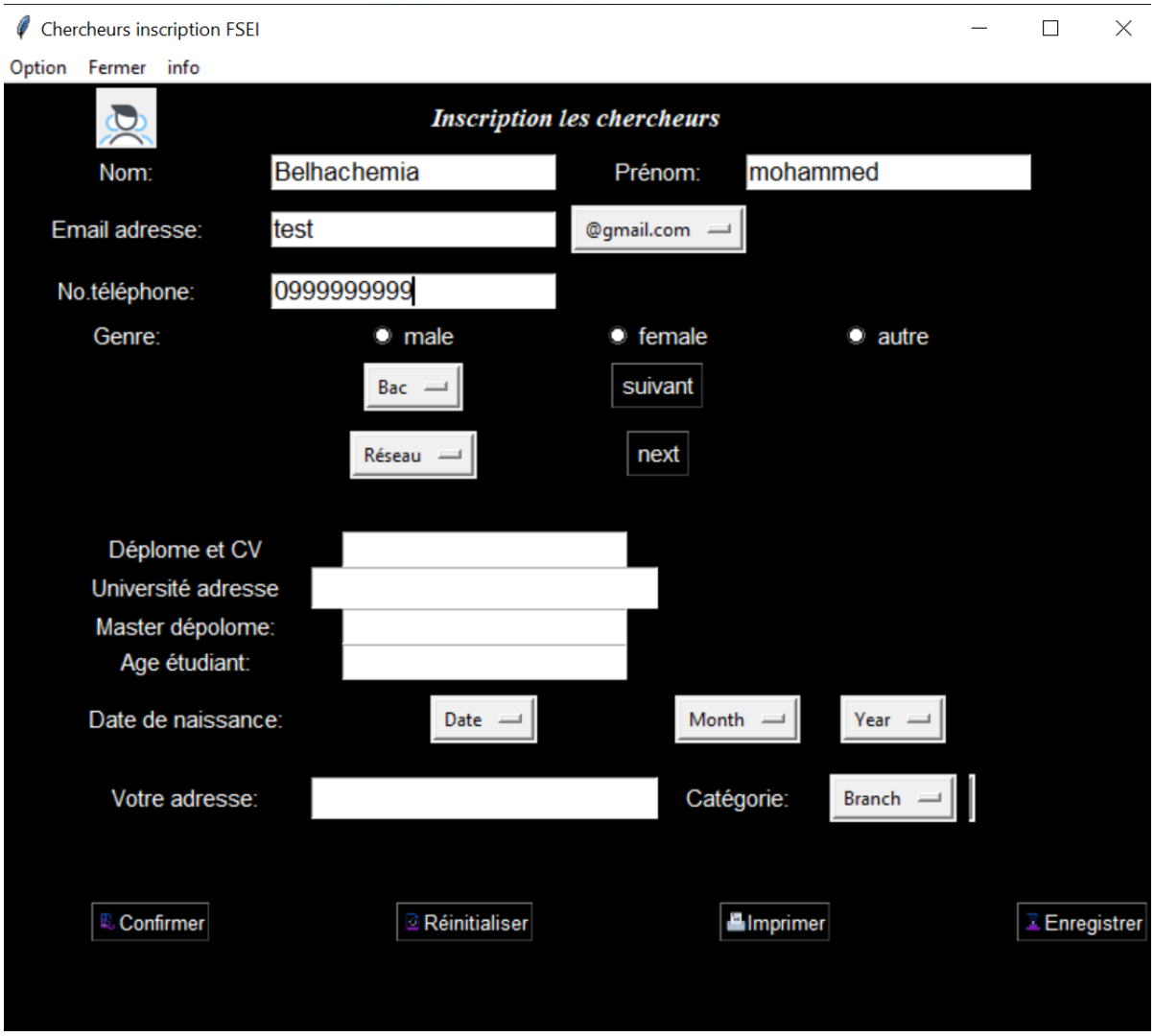


Figure 24 – Inscription les chercheurs.

La figure 24 montre la fenêtre de l'inscription les utilisateurs de type chercheurs sur l'application de détection de communauté.

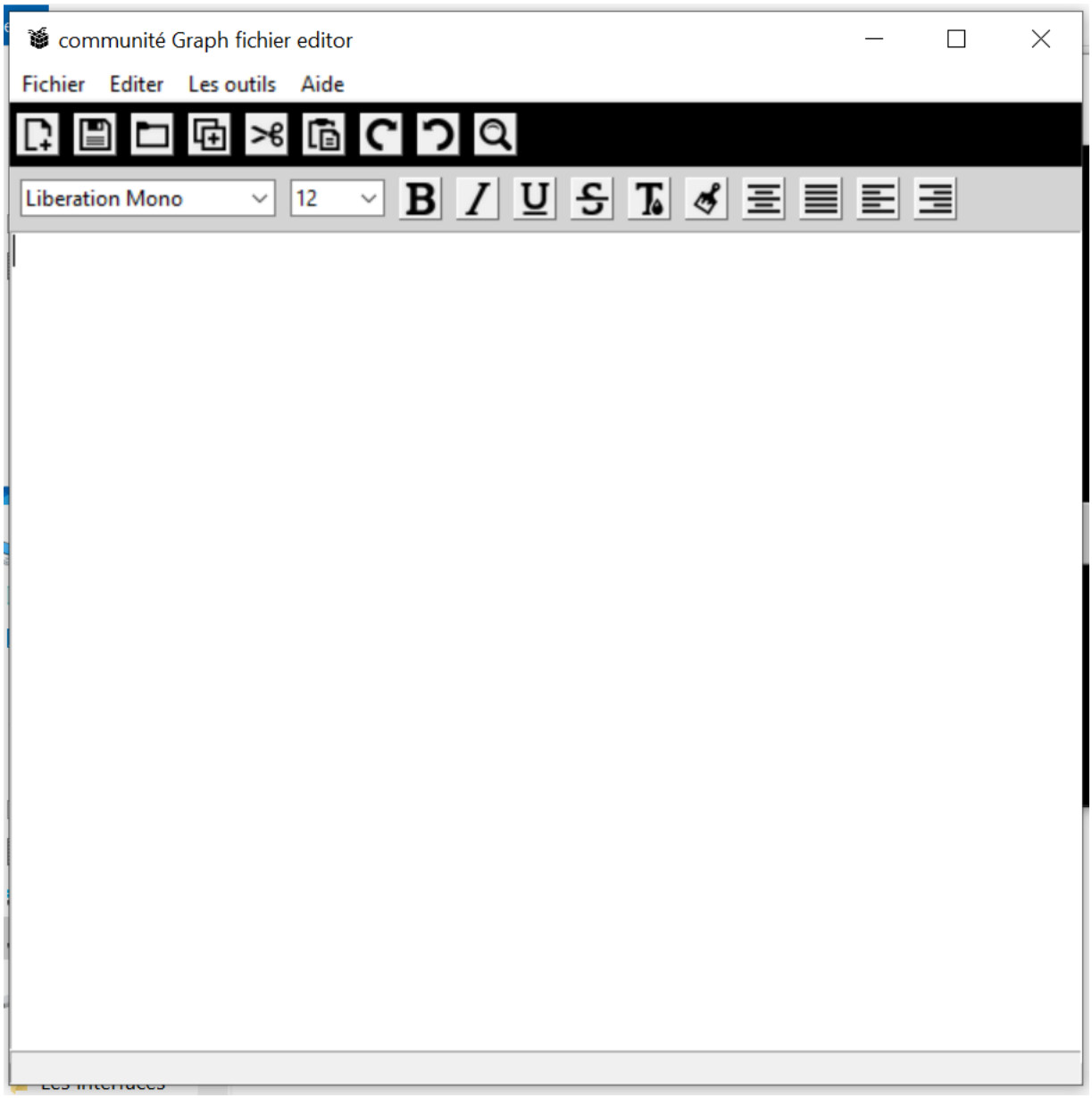


Figure 25 – Graph texte éditeur

La figure 25 montre la fenêtre de l'insertion, modifications sur les modèles donnés sous forme T XT.

### 5.4.8 Algorithmes avec les modèles proposées


Figure 26 – Applications les algorithmes sur les modèles de donnée.

La figure 26 montre la fenêtre de l'inscription les utilisateurs de type employeurs sur l'application de détection de communauté

### 5.4.9 Fenêtre représente les algorithmes Louvain et Girvan-new man

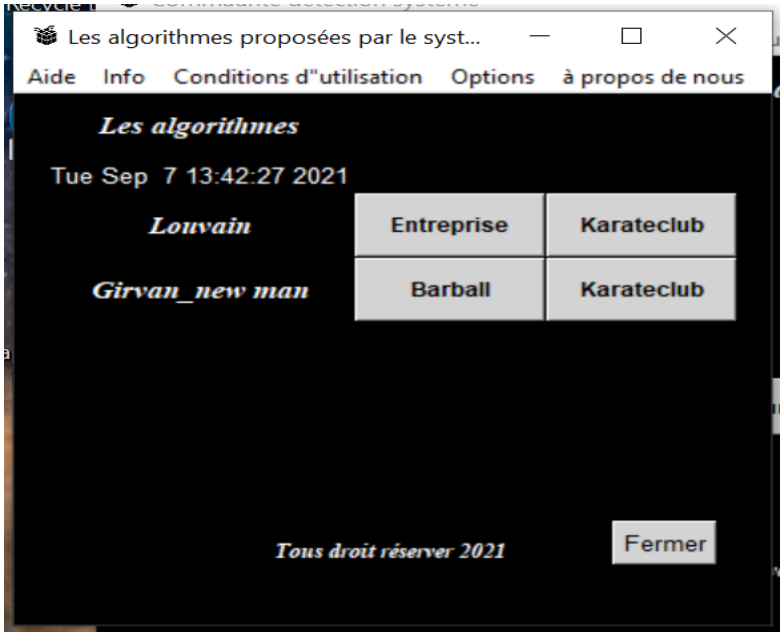


Figure 67 les algorithmes Louvain et Girvan new man

### 5.4.10 Imprimer les données


Figure 28 – Imprimer les données

La figure 28 montre la fenêtre de l'impression les données ou l'utilisateur de type chercheur, développeurs, employeurs peut imprimer différents types documents (*WORD, IMAGE, TXT*).

### 5.4.11 Système de l'affichage les remarques

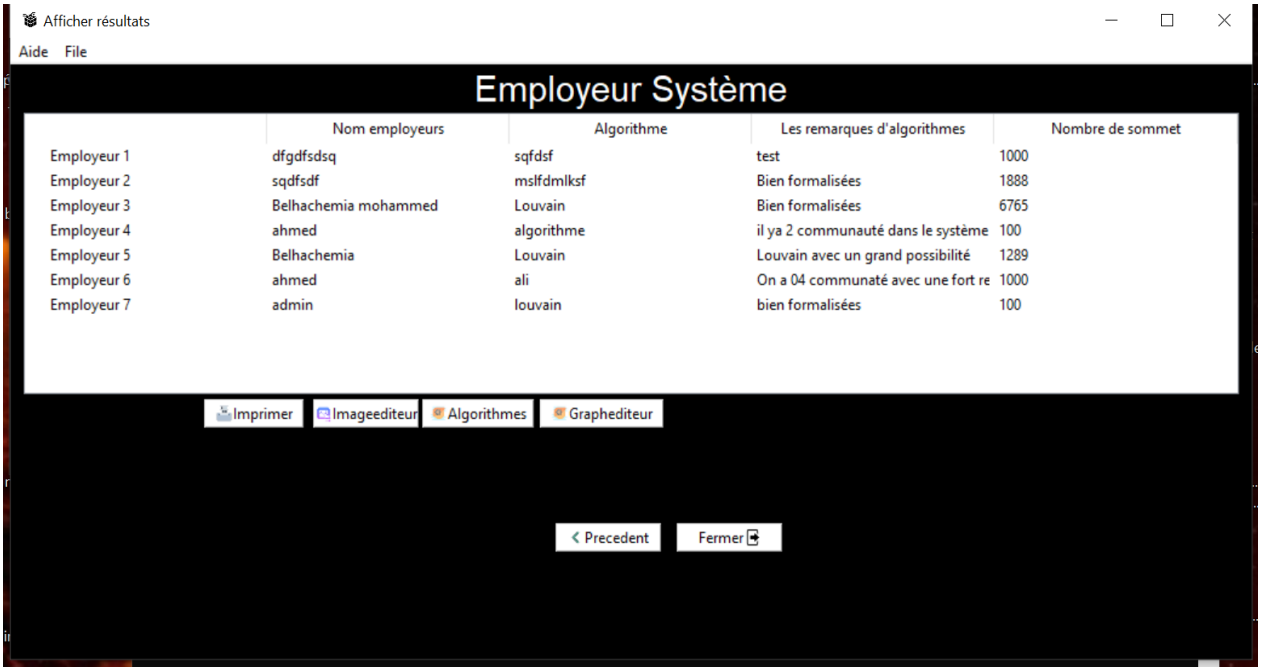


Figure 29 – Affichage les remarques à l'aide les algorithmes.

La figure 29 montre la fenêtre du système *« insertion les remarques à l'aide l'observation des no mbres de nœuds et communauté ».*

### 5.4.12 Inscription les développeurs


Figure 30 – inscription les développeurs

La figure 30 montre la fenêtre l'interface de l'inscription les développeurs.

## 5.5 Conclusion

Dans ce chapitre nous avons présenté les différents outils que nous avons utilisés lors du développement de notre application, ainsi que des captures montants les différentes fonctionnalités de l'application que nous avons développées.

.

# Conclusion générale

Dans ce mémoire, nous avons effectué une étude de l'état de l'art liée à la problématique d'analyse de réseaux hors ligne et aux méthodes de détection de communautés. ceci nous a permis d'adopter un modèle de représentation des réseaux hors ligne général et flexible pour pouvoir étudier tout type de relation et manipuler un réseau de grande dimension, et après avoir défini la notion de communauté de façon claire, nous avons étudié plusieurs solutions de détection de communautés en prenant en compte la complexité en temps et la structure dynamique des graphes. les solutions proposées ont été confrontées aux approches existantes en utilisant des jeux de données réal.

# Bibliographies

**Livre,monographie**

[1] S. Bhat and M. Abulaish, "*Overlapping social network communities and viral marketing," in International Symposium on Computational and Business Intelligence, Aug 2013, pp. 243– 246. 5. X. Wu and Z. Liu, "How community structure influences epidemic spread in social networks," Physica A: Statistical Mechanics and its Applications,* vol. 387, pp. 623–630, 2008.

[2] M. E. Newman, *« Analysis of weighted networks* », Physical Review E, vol. 70, no. 5, p. 056131,2004.

[5] *A novel approach to evaluate community detection algorithms on ground truth Giulio Rossetti,* Luca Pappalardo, and Salvatore Rinzivillo.

[8] *Modélisarion objet avec UML,* Pierre-Alain Muller.

[9] *Apprendre à programmer avec Python 3* ,Gérard Swinnen.

**Article d'actes de conférence**

[3] Thomas Aynaud *, détection des communautés dans les réseaux dynamiques cnrs, lip6 Avril 2010 , Université Pierre et Marie Curie,* article.

[4] Tanja *Vojkovic Community structure in networks : Girvan-Newman algorithm improvement*

[6] Luis Rita ,*Louvain Algorithm* apr-2020

[7] Abhishek Mishra , Demystifying Louvain's Algorithm and its implementations in GPU

**Source code de la partie l'implémentation avec langage python**

L'implémentation de F1 benchmarks *https://github.com/GiulioRossetti/f1-communitie.*