Table of Contents

[Liste des figures 3](#_Toc531210225)

[Liste des tableaux 3](#_Toc531210226)

[Contexte du projet 4](#_Toc531210227)

[1. Représentation et Génération des réseaux sociaux : 4](#_Toc531210228)

[2. Système à acteurs : 4](#_Toc531210229)

[3. Visualisation : 4](#_Toc531210230)

[Travail réaliser 6](#_Toc531210231)

[1. Réseaux sociaux : 6](#_Toc531210232)

[1.1 Définition : 6](#_Toc531210233)

[1.2 Caractéristiques : 6](#_Toc531210234)

[1.3 Les générateurs de graphe : 7](#_Toc531210235)

[1.4 Le générateur R-MAT : 8](#_Toc531210236)

[2. Système à acteurs : 10](#_Toc531210237)

[2.1 Erlang : 10](#_Toc531210238)

[2.2 Pourquoi Erlang : 11](#_Toc531210239)

[2.3 Taches effectuer : 12](#_Toc531210240)

[2.4 Perspectives : 14](#_Toc531210241)

[3. Visualisation : 14](#_Toc531210242)

[3.1 Python : 14](#_Toc531210243)

[3.2 Bibliothèques utilisés : 14](#_Toc531210244)

[Pandas 14](#_Toc531210245)

[N 15](#_Toc531210246)

[Matplotlib : 15](#_Toc531210247)

[Organisation du travail 16](#_Toc531210248)

# Contexte du projet

Les tendances dans les réseaux sociaux. Ce projet vise à concevoir et réaliser un système d’acteurs qui modélise certaines interactions entre des individus dans un réseau social, grâce à l’analyse de ces interactions nous pourrons observer l’évolution et la propagation des idées dans ce réseau, tout en repérant les acteurs les plus influents.

D’un premier coup le sujet du projet apparait un petit peut vague et plusieurs questions se posent, Comment on va modéliser un réseau social ? comment on va transformer un modèle d’un réseau à un paradigme à acteur pour simuler les interactions ? comment on va simuler les interactions entres les différents acteurs du réseau ?

Alors et pour éclaircir le but du projet on a décidé de diviser le projet en trois sous parties : Représentation et Génération des réseaux sociaux, Système à acteurs et visualisation des résultats.

1. Représentation et Génération des réseaux sociaux :

Un réseau social est mieux représenté par un graphe, ce graphe est constitué d’un ensemble de nœuds N qui symbolise l’ensemble des acteurs sociaux dans le réseau, et un ensemble de liens E, qui peuvent être orienté ou non-orienté, peuvent contenir un poids (*positif/négatif*) pour exprimer les directions et les degrés d’influence entre les acteurs.

Le but est de créer un générateur de diffèrent graphes qui ont des propriétés semblables à celles des réseaux sociaux réels.

Avant d’entamer la génération du réseau, on commence par étudier les caractéristiques et particularités des réseaux sociaux. Ensuite, chercher et étudier les différents modèles existants de génération de graphes et enfin implémenter un algorithme de génération efficace qui approche au mieux la structure d’un réseau social réel.

Comme JAVA est un langage qui nous a suivi tout au long de notre parcours universitaire et vu nos prérequis dans le langage comparé aux autres langages du même -style-, JAVA s’est avéré le choix le plus confortable pour réaliser cette partie, notons aussi la facilité d’utiliser les librairies JSON en JAVA.

1. Système à acteurs :

Le but de cette partie est de prendre les graphes des réseaux sociaux généré dans la première partie « Représentation et Génération des réseaux sociaux », les réaliser dans le paradigme à acteurs, lancer une simulation entre les différents acteurs et garder des traces de cette simulation dans des logs, afin de pouvoir savoir les influenceurs dans le réseau à la fin de la simulation.

Pour ça, notre encadrant Monsieur VARACCA nous a proposé de le réaliser en utilisant **Erlang** qui est un langage puissant en ce qui concerne **la notion de concurrence et des acteurs**.

1. Visualisation :

Cette partie consiste à analyser les logs sauvegarder lors de la simulation

# Travail réalisé

1. Réseaux sociaux :

Récemment, les réseaux sociaux ont attiré beaucoup d’attention à cause de leurs étonnantes régularités. Le World Wide Web, la topologie d’internet et les réseaux sociaux en ligne suivent des lois et des structure surprenantes, l’analyse de ces réseaux nous permettent d’observer et suivre le flux de données ainsi que les changements que ces derniers apportent sur le réseau, comme la propagation des virus ou encore la détection les communautés et les changements d’opinions dans un réseau social.

### 1.1 Définition :

En sciences humaines et sociales, un réseau social désigne un agencement des liens entre des individus et/ou des organisations, constituant un groupement qui a un sens. (1, s.d.)

Cette définition est aussi appliquée sur les réseaux sociaux en ligne qui sont des application web dites aussi « service de réseautage social en ligne » qui permettent aux utilisateurs de communiquer et de créer des liens d’amitié, les individus et les organisations sont représentés par les utilisateurs de l’application.

Nous nous intéresserons dans ce qui suit par les réseaux sociaux en ligne statique, qui sont réseaux déjà formés. La création de nouveaux liens n’est pas considérée.

### 1.2 Caractéristiques :

* **Niveau microscopique :**

La première question qu’on se pose dans un réseau social est : quelle est le degré d’influence d’un individu précis dans le réseau ?

Pour répondre à cette question, il faut calculer le nombre de connections (liens) de cet individus (dans le cas où le graphe est orienté on calcule les liens sortants), cette mesure s’appelle **centralité**.

La centralité d’un nœud exprime le pouvoir et l’influence d’un acteur. Ces propriétés sont fondamentales dans les structures sociales, en d’autres termes, plus l’acteurs a de liens plus il a de pouvoir et influence dans le réseau.

Malheureusement, en réalité il n’existe pas une réponse exacte et final pour savoir le degré d’influence d’un acteur, mais la plupart des chercheurs utilisent différentes mesures incluent :

* Le degré de centralité.
* Centralité de proximité.
* Centralité d’intermédiarité.
* Centralité de prestige. (Eigenvector)

Le degré de centralité nous indique à quel point un nœud est intégré dans le réseau ainsi une haute connectivité implique un pouvoir mais aussi une vulnérabilité tout dépend de la nature de l’information diffusée dans le réseau. Un nœud avec une haute connectivité s’appelle un **Hub.**

Le **degré de centralité** seul n’est pas une mesure suffisante pour savoir l’importance d’un nœud dans un réseau et peut être souvent trompeuse, nous aurons besoin d’autres métriques pour la soutenir comme la **centralité de proximité** qui représente la distance moyenne du nœud avec les autres acteurs du réseau, la **centralité d’intermédiarité** qui consiste à repérer l’ensemble des plus courts chemins entre toute paire de points passant par un sommet donné : il permet de mettre en évidence les points les plus centraux pour la circulation de l’information (2, s.d.) et la **centralité de prestige** d’un nœud qui est la somme des centralités des nœuds avec lesquels il est connecté, en d’autres termes, plus un nœud est lié avec des nœuds importants plus il est prestigieux. Ces mesures regroupées peuvent nous aider à bien repérer les nœuds influents dans un réseau social.

Le **phénomène du petit monde** est aussi l’une des propriétés principales dans les réseaux sociaux, ce phénomène dit que la distance entre n’importe quels deux acteurs dans le réseau est strictement inférieure ou égale à 6, on l’appelle aussi le concept des **six degrés de séparation**.

* **Niveau macroscopique :**

La **densité** de réseau est un facteur déterminant dans la composition de tout réseau,   
avec la densité, nous demandons simplement combien de connexions y a-t-il par rapport au nombre maximal de connexions possible, c’est cette mesure et la **distance moyenne** entre les acteurs du réseau qui nous montre le degré de cohésion et interdépendance d’un réseau et qui distingue le cas un simple groupe d’acteurs et le cas d’un réseau d’acteurs.

Ces deux mesures nous permettent aussi de découper notre réseau en sous-réseau que nous appelons des **clusters** ou **communautés**, car la densité dans un cluster est élevée, ce qui rends les distances moyenne entre les acteurs petits par rapport aux acteurs des autres clusters.

Le **degré de distribution** est une autre mesure pour mieux analyser la structure globale d’un réseau, dans le cas de la plupart des réseaux sociaux réel, le degré de distribution est très asymétrique, la plupart des acteurs auront peu de liens alors qu’une petite fraction (*mais significative*) d’acteurs auront un nombre très grand de liens.

### 1.3 Les générateurs de graphe :

Il existe deux approches pour créer un générateur de réseau social :

D’un graphe aléatoire à réseau social : cette approche consiste à générer un graphe complètement aléatoire, puis chercher à transformer le graphe résultant en réseau social en modifiant sa structure par la création de nouveaux liens, cette création est guidée par des lois respectées par les réseaux sociaux principalement le degré de distribution, ce modèle est appelé Barabasi-Albert model (3, s.d.).

Les générateurs de Kronecker : l’idée générale dit que tout réseau peut être "compressé" en une très petite génération de graphes, par exemple, la réplication de ce graphe de façons fractale créera un réseau avec les mêmes propriétés que le réseau d'origine sur une plus grande échelle. En d’autres termes au lieu de créer un graphe de nœuds, on créer un graphe de graphe (4, s.d.).

Ces générateurs suivent certaines lois guidées par des probabilités qui leurs permettent de créer des graphes très semblables à des réseaux sociaux réels.

Après avoir exploré pleins de modèles de génération de graphes, le générateur R-MAT s’est avéré très simple à implémenter et en même temps très efficace concernant la génération des réseaux sociaux de grande taille.

### 1.4 Le générateur R-MAT :

R-MAT signifie Recursive MATrix, c’est un modèle de générateur de graphe introduit par Chakrabarti, Faloutsos, and Zhan (5, s.d.), il est capable de générer des graphes de très grande taille et de différentes structures seulement en modifiant les paramètres d’entrées qui sont le nombre de nœuds, le nombre de liens et une matrice .

**Quelques notations :**

Avant de commencer d’expliquer le fonctionnement du générateur nous allons besoin de quelques notations : notons le graphe généré où est l’ensemble de nœuds et est l’ensemble de liens, le graphe peu être orienté ou pas, les deux choix son possible tout dépends du type du réseau social. Le nombre de nœuds est noté par =numéroté de et le nombre de lien est noté par .

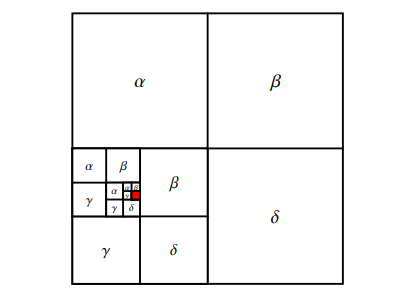
La matrice d’adjacence est notée , l’entrée dans la matrice correspond au lien entre les nœuds et , la valeur de cette entrée peut être 0 qui signifie qu’il n’existe pas de lien entre les nœuds, ou bien une valeur non nulle qui signifie qu’il y a un lien entre et ayant le poids .

**Fonctionnement :**

Le générateur opère par subdiviser la matrice d’adjacence du graphe en quatre partitions de tailles égales, puis distribuer les liens au sein de ces partitions avec des probabilités inégales.

La distribution sera guidée par quatre paramètres non nuls où (La matrice), en initialisant toutes les entrées de la matrice d’adjacence par 0, l’algorithme place un lien dans cette matrice en choisissant une des quatre partitions avec les probabilités respectivement, the quadrant choisi sera ensuite subdivisé à son tour en quatre partition plus petites, la procédure est répéter jusqu’à aboutir à une partition sélectionnée de taille , où on va incrémenter la valeur de cette entrée. (5, s.d.)

La figure suivante présente un exemple de sélection d’une case de la matrice d’adjacence :

 Figure 1 : création d'un lien avec R-MAT

**Implémentation :**

L’implémentation de cet algorithme est très simple, il faut juste choisir convenablement les structure de données :

Pour créer un lien entre deux nœuds il faut faire décomposition sur la matrice d’adjacence basé sur les probabilités , pour chaque décomposition, on tire un nombre aléatoire suivant la loi uniformeentre pour sélectionne l’un des quadrants selon ces probabilités.

Les nœuds sont représentés par une suite binaire.

A une itération , on associe la décomposition au bits des nœuds (partant de gauche à droite), si on a choisi le quadrant de , on met le bit de i à 0 et de j à 0, si on a choisi le quadrant de , on met le bit de i à 0 et de j à 1, si on a choisi le quadrant de , on met le bit de i à 1 et de j à 0 et si on a choisi le quadrant de , on met le bit de i à 1 et de j à 1 (5, s.d.).

**Exemple :**

Tableau 1 : Exemple

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Itération | 0 | 1 | 2 | 3 | 4 |
| Quadrant choisi (probabilité) |  | Quadrant | Quadrant | Quadrant | Quadrant |
| Bits de i | \*\*\*\* | 0\*\*\* | 01\*\* | 010\* | 0100 |
| Bits de j | \*\*\*\* | 0\*\*\* | 00\*\* | 001\* | 0010 |

**Complexité :**

Notre algorithme génère M liens aléatoires suivant le modèle r-mat expliqué précédemment qui nécessite itérations pour avoir tous les bits du nœud source et destination de chaque lien, en conséquence, la complexité du générateur est de .

**Expérimentations :**

Pour pouvoir générer un graphe similaire à un réseau social, il faut bien choisir les paramètres d’entrées de l’algorithme, en particulier les probabilités .

Nous avons décidé de vérifier deux caractéristiques pour juger si le graphe correspond à un réseau social, la loi de distribution et la propriété du petit monde.

Après une phase d’expérimentation et en variant les différents paramètres d’entrées de l’algorithme, nous avons remarqué que la loi de distribution est fortement influencée par les valeurs de , nous avons distingué quelques cas :

, sont égaux : dans ce cas chaque probabilité aura la valeur 0.25, le fonctionnement de l’algorithme aura le même comportement d’un algorithme complétement aléatoire, vue que les quadrants seront choisis avec des probabilités égales, la procédure de décomposition sera équivalente à celle du choix aléatoire d’une case de la matrice…

, avec cette configuration la loi de distribution est très similaire à celle des réseaux sociaux réels.

En ce qui concerne la propriété du petit monde, il est clair que la densité du réseau est l’un des facteurs influents principaux, ça veut dire que plus le rapport est grand plus la probabilité que le graphe généré vérifie la propriété du petit monde, l’algorithme de vérification reste à implémenté.

**Perspectives :**

* Vérifier la propriété du petit monde.
* Optimiser le code déjà écrit.
* Trouver les différentes configurations des probabilités.
* Détecter les acteurs influents dans le réseau avant de lancer la simulation, pour pouvoir les observer de près.
* Explorer d’autres modèles de génération de graphe.

## Système à acteurs :

### 2.1 Erlang :

Erlang est un langage de programmation de la compagnie Ericsson dont le but premier était de développer le monde des télécommunications. Son nom a une origine double : il s’agit à la fois d’une référence au mathématicien danois Agner Krarup Erlang, et d’une abréviation de « Ericsson Language ». Premièrement intégré dans les équipements d’Ericsson, en tant que langage propriétaire, il est ensuite devenu open source en 1998 avec la distribution OTP *(Open Telecom Platform)*. Cette dernière représente le [framework](https://fr.wikipedia.org/wiki/Framework) standard de l'univers Erlang, la plupart des programmeurs l'utilisant comme base d'[interopérabilité](https://fr.wikipedia.org/wiki/Interop%C3%A9rabilit%C3%A9). Il peut s'interfacer avec d'autres langages comme [Java](https://fr.wikipedia.org/wiki/Java_(langage)) ou [C++](https://fr.wikipedia.org/wiki/C_plus_plus).

Erlang supporte plusieurs paradigmes : concurrence, [temps réel](https://fr.wikipedia.org/wiki/Syst%C3%A8me_temps_r%C3%A9el) et distribution. C’est un [langage fonctionnel](https://fr.wikipedia.org/wiki/Programmation_fonctionnelle) à [évaluation stricte](https://fr.wikipedia.org/wiki/%C3%89valuation_stricte), [affectation unique](https://fr.wikipedia.org/wiki/Affectation_(informatique)) et [typage dynamique](https://fr.wikipedia.org/wiki/Typage_dynamique) [fort](https://fr.wikipedia.org/wiki/Typage_fort). Ca couche concurrente est fondée sur le [modèle d'acteur](https://fr.wikipedia.org/wiki/Mod%C3%A8le_d%27acteur) (Processus) .Erlang possède aussi des fonctionnalités de [tolérance aux pannes](https://fr.wikipedia.org/wiki/Tol%C3%A9rance_aux_pannes) et de mise à jour du code à chaud, permettant le développement d'applications à très haute disponibilité.

Des compagnies présentes dans le secteur des télécommunications, comme T-Mobile, Nortel ou encore plus récemment WhatsApp, utilisent Erlang comme il est aussi utilisé pour écrire le serveur XMPP *(Extensible Messaging and Presence Protocol)*, la base de données NoSQL CouchDB, le serveur HTTPYaws ainsi que le logiciel de modélisation 3D Wings 3D. (6, s.d.)

### 2.2 Pourquoi Erlang :

Un réseau social peut contenir des milliers et milliers d’acteurs ce qui est difficile à gérer quand on veut transformer un modèle de réseau en un paradigme acteurs, mais comme la principale force d’Erlang est la notion de concurrence, qui est basée sur les processus Erlang, cela va nous faciliter la tâche.

En effet une application est subdivisée en processus légers : comparé aux processus ou aux threads traditionnels, le coût des processus Erlang est faible en ressources machines. Ainsi, un nombre important d’entre eux peuvent être créés sans que les performances ne soient dégradées. En plus de ça les processus Erlang sont isolés ce qui implique qu’il n’y a pas de partage de mémoire, évitant ainsi les problèmes d’accès concurrentiels. Un identifiant est affecté à chaque processus lors de sa création.

L’exécution des processus Erlang se fait sur une machine virtuelle BEAM *(Bogdan Erlang Abstract Machine)*, ce qui lui permet, tout comme JAVA, d’assurer la portabilité de l’application vers n’importe quelle autre plateforme. Aussi, plusieurs nœuds peuvent être interconnectées et former un système distribué. (7, s.d.)

* **La communication entre les processus :**

Les processus erlang peuvent communiquer entre eux, également connus sous le nom de passage de messages. Cette procédure est asynchrone sous la forme que le processus d'envoi ne s'arrêtera pas après l'envoi du message et chaque processus possède une file d’attente (MailBox) où arrivent les messages en attendant d’être lu. (8, s.d.)

**Echange des messages :**

Un tel échange présente la syntaxe suivante : **Pid ! Message**

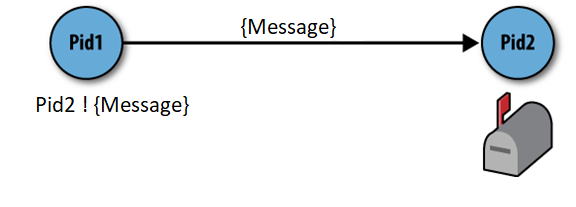


Figure 2 : Envoie de Message dans Erlang

Pid est l’identifiant du processus destinataire et Message est le message à transmettre. Ce message peut être une valeur de **tout type de données Erlang**. Si nous savons le nom d’un processus, nous pouvons lui envoyer un message.

**Réception des messages :**

Comme on la déjà mentionné chaque processus possède une file d’attente (MailBox) où arrivent les messages en attendant d’être lu.

En effet c’est la réception des messages qui met le processus dans un état actif, quand le processus reçoit un message il le traite, dans le cas où le processus n’a pas de message à traiter il sera suspendu jusqu’à ce qu’il reçoit un nouveau message.

Ces messages sont extraits de la boîte aux lettres à l'aide de la clause **receive**. La clause **receive** est une construction délimitée par les mots réservés **receive** et **end** et contient un certain nombre de clauses ou pattern. Chaque clause a une séquence d'expressions dans le corps (à droite de la flèche) à exécuter. (8, s.d.)

En général la clause **receive** a la forme suivante :

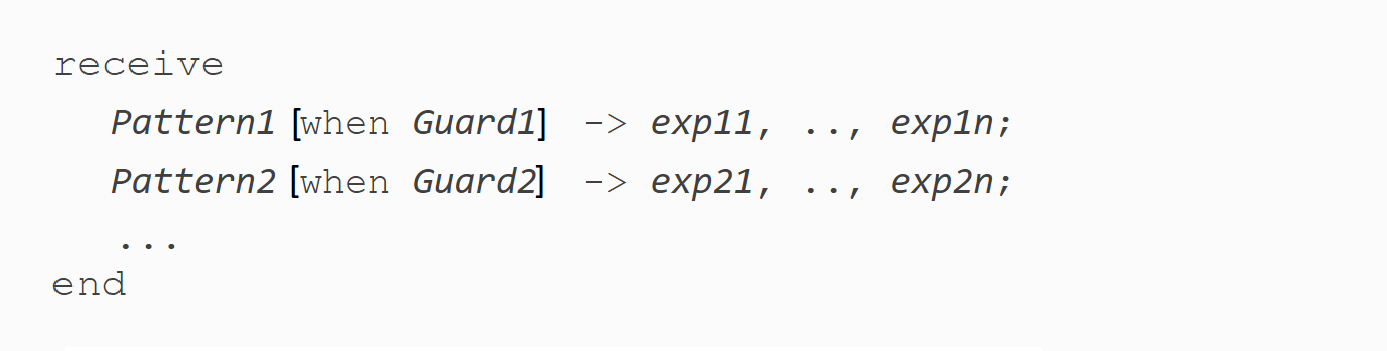


Figure 3 : clause de récéption des messages

### 2.3 Taches effectuer :

Avant de détailler ce qu'on a pu concrétiser jusqu’à maintenant, il est important de souligner qu'on a travaillé pour le moment indépendamment du premier groupe, car on n’a jamais développé avec Erlang, ce qui induit qu'un très bon apprentissage de ce dernier est nécessaire à la réussite de ce projet.

Cet apprentissage des bases et tout ce qui est syntaxe et spécificité du langage a nécessité de faire des exercices et des pratiques afin qu’on puisse se familiariser avec le langage avant d’entamé le vif du projet.

Après cette phase d’apprentissage on a commencé de travailler sur le paradigme acteurs sur Erlang en essayant de transformer des petits graphes en processus Erlang et faire des échanges entre eux (simulation d’interactions), pour savoir le comportement des processus Eralng et comment l’échange des messages se passent entre eux, en plus de ça on a essayé de garder des traces de la simulation dans des logs qui seront transmis au 3ème groupes pour les analyser et les afficher d’une manière cohérente et compréhensible

* **Lecture du fichier et instanciation des processus :**

Pour pouvoir instancier le graph qui nous sera transmis par nos camarades du premier groupe, on a commencé à réaliser des tests avec un fichier de configuration assez simple. Dans ce dernier on trouve tous les nœuds du graphe et pour chaque nœud on lui associe son propre poids et la liste de ces voisins. La structure du fichier est décrite comme suite :

*Noeud1 [Poids, {Voisins}]*

*…*

*NœudN [Poids, {Voisins}]*

Lors de l’instanciation on fournit à chaque processus la fonction qu’il va exécuter avec son poids et la liste de ses voisins. Comme ça chaque processus pourra diffuser un message à ses voisins.

* **La simulation :**

Comme on a vu, les messages sont extraits de la boite à lettre à l’aide de la clause **receive**. Cette dernière peut contenir un ou plusieurs patterns avec lesquels on essaye de faire une correspondance avec les messages reçus pour faire un certain traitement.

Dans notre cas on a décrit 2 pattern et un timeout dans le corps de la clause **receive**, le tableau suivant illustre l’action à exécuter pour chaque pattern et aussi pour le time out.

Tableau 2 : pattern définie dans la clause receive

|  |  |  |
| --- | --- | --- |
| Structure du pattern | Fonctionnalité | Description |
| {start} | Lancement de la simulation | Le lancement de la simulation sera déclenché à partir du terminal en invoquant une méthode qui va envoyer à tous les processus le message {**start**}.  Quand le message sera reçu le processus va exécuter la séquence des expressions correspondantes à ce message et qui consiste à envoyer une demande de calcule (Pattern2) à tous ses voisins |
| {calcul,Value,Pid} | Demande de calcule | Ce pattern contient l’atom **calcul**, le poids de l’émetteur **Value** et son identifiant **Pid**. Quand le processus reçoit ce message certain calcules vont être faite en se basant sur la valeur reçu de l’émetteur et son propre poids pour savoir si il doit adopter la valeur reçu ou bien garder sa propre valeur. |
| TimeOut N | Action à exécuter après un certain temps N (N est en millisecondes) | Ça sera notre cas d’arrêt si un processus passe un certain temps N sans recevoir aucun nouveau message le processus va s’arrêter |

* **Sortie de la simulation :**

Le résultat de notre simulation sera les logs dans lesquelles on a gardé des traces durant toute la simulation, ces logs seront analysés par le troisième groupe.

### 2.4 Perspectives :

Après s’être familiariser avec la programmation concurrente dans Erlang avec des tests simples on se basant sur des petits graphes, on va maintenant essayer de travailler sur des graphes plus grands et plus structuré (Réseaux sociaux) qui seront générer par le premier groupe et on va implémenter des modèles mathématiques de la diffusion d’influence dans les réseaux sociaux afin d’avoir des résultats réels ou qui sont proche de la réalité.

## Visualisation :

Pour la partie affichage de ce projet, nous avons décidé de travailler avec Python, notre choix s’est basé sur la simplicité du langage, et aussi les puissances qu’il offre surtout dans l’analyse de donnée, ce que nous serons emmenés à effectuer aussi.

### 3.1 Python :

Python est un langage de programmation à typage dynamique conçu par Guido van Rossum.il a été conçu pour être facilement lu par les programmeurs. En raison de son grand nombre d'adeptes et de ses nombreuses bibliothèques, Python peut être implémenté et utilisé pour tout, des pages Web à la recherche scientifique.

### 3.2 Bibliothèques utilisés :

Pour la réalisation de notre travail, nous avons fini -après plusieurs recherches – par choisir les bibliothèques python suivantes pour tout ce qui est affichage dans le projet, car on aura à afficher les graphes des réseaux sociaux et faire des simulations dessus, ce qui sera traduit par des animations sur le graphe et des statistiques sur les données qu’on peut tirer de ces réseaux.

Pandas :

Bibliothèque souple et puissante d'analyse et de manipulation de données pour Python, fournissant des structures de données étiquetées similaires aux objets R data.frame, aux fonctions statistiques .

* Points fort de pandas :

Un objet **DataFrame** rapide et efficace pour la manipulation de données avec indexation intégrée;

Outils pour **lire et écrire des données** entre des structures de données en mémoire et différents formats : fichiers CSV et texte, fichier Json, bases de données Microsoft Excel, SQL et le format HDF5 rapide;

N[umpy](https://docs.scipy.org/doc/numpy/reference/index.html" \l "module-numpy" \o "(disponible dans NumPy v1.14)) :

C’est une bibliothèque *numérique* apportant le support efficace de larges tableaux multidimensionnels, et de routines mathématiques de haut niveau (fonctions spéciales, algèbre linéaire, statistiques, etc.).

NetworkX :

NetworkX est un package Python pour la création, la manipulation et l'étude de la structure, de la dynamique et des fonctions de réseaux complexes

NetworkX est conçu pour fonctionner sur les grands graphes du monde réel, c'est-à-dire par exemple, des graphes de plus de 10 millions de nœuds et 100 millions d'arêtes. En raison de sa dépendance vis-à-vis de la structure de données en « dictionnaire de dictionnaire » (pur Python), NetworkX est raisonnablement efficace, très évolutif, faisant de lui un outil intéressant dans le cadre de l'analyse des réseaux sociaux ce qui semble adapté parfaitement à notre cas, car on est bien dans le thème analyse des tendances dans les réseaux sociaux.

Matplotlib :

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.

**Justification de choix de bibliothèques**

* Ces bibliothèques sont très avantageuses, car elles fournissent :
* Des outils pour l’étude de la structure et de la dynamique des réseaux sociaux, biologiques et d’infrastructures ;
* Une interface de programmation standard et une implémentation graphique qui convient à de nombreuses applications ;
* Ça permet de manipuler des graphes de très grandes tailles, ce qui convient au cas des réseaux sociaux
* Ça permet de faire de belle présentation grâce aux animations et aux couleurs
* La capacité de travailler sans douleur avec de grands ensembles de données non standard.

# Organisation du travail

Diviser pour mieux régner c’est la stratégie qu’on a adoptée dans notre projet. Cette stratégie va mener à un bon déroulement du projet et au même temps à un avancement rapide. Comme chaque groupe au début travail indépendamment des autres groupes on a pas encore utilisé des outils de travail collaboratif.

En effet comme notre projet et constituer de 3 sous parties on a décidé de les partager entre nous. Au début chaque groupe va se contenter de travailler sur sa partie et après on va assembler tout le travail réaliser.

1er Groupe « Représentation et Génération des réseaux sociaux »

KHARCHI Anis et BENAMIROUCHE Malek

2ème Groupe « Système à acteurs »

LAKHLOUFI Anass etHamed Khodja ABDELHAK

3ème Groupe « visualisation »

# Liste des figures

[Figure 1 : création d'un lien avec R-MAT 9](#_Toc531208521)

[Figure 2 : Envoie de Message dans Erlang 11](#_Toc531208522)

[Figure 3 : clause de récéption des messages 12](#_Toc531208523)

# Liste des tableaux

[Tableau 1 : Exemple 9](#_Toc531208507)

[Tableau 2 : pattern définie dans la clause receive 13](#_Toc531208508)

# Référence

1. (s.d.). Récupéré sur https://fr.wikipedia.org/wiki/R%C3%A9seau\_social#cite\_ref-2

2. (s.d.). Récupéré sur https://groupefmr.hypotheses.org/3724

3. (s.d.). Récupéré sur http://barabasi.com/f/622.pdf

4. (s.d.). Récupéré sur https://cs.stanford.edu/people/jure/pubs/kronecker-jmlr10.pdf

5. (s.d.). Récupéré sur D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A Recursive Model for Graph

6. (s.d.). Récupéré sur https://fr.wikipedia.org/wiki/Erlang\_(langage)

7. (s.d.). Récupéré sur https://medium.com/%C3%A9cosyst%C3%A8me-des-langages-de-programmation/erlang-le-langage-qui-permet-de-voir-grand-da12a059bcf9

8. (s.d.). Récupéré sur https://www.safaribooksonline.com/library/view/erlang-programming/9780596803940/ch04.html