CHPS

Simulation de la propagation d'une épidémie

ISTY

Mamoun El Hajami IATIC5
2023/2024

Table of Contents

Introduction

Ce projet explore la simulation de la propagation d'une épidémie en utilisant l'algorithme de PageRank. L'objectif est de modéliser l'évolution d'un virus au sein d'une population, en variant certains paramètres pour prédire la propagation de l'épidémie. Nous avons abordé le concept de PageRank ainsi qu'un modèle épidémiologique, avant de plonger dans notre implémentation et l'analyse des résultats obtenus.

Le PageRank, habituellement appliqué pour évaluer la popularité des pages web, est détourné dans notre contexte pour simuler la propagation d'un virus. Cette approche permet de hiérarchiser les individus au sein d'un réseau, identifiant ainsi ceux ayant un rôle clé dans la diffusion de l'épidémie. L'adaptation de l'algorithme de PageRank à notre problème inclut la prise en compte des individus n'ayant aucun lien sortant (outlink) ou étant isolés, en intégrant un facteur d'amortissement pour simuler des déplacements aléatoires au sein du réseau.

Notre modèle épidémiologique repose sur l'hypothèse d'un contact homogène entre individus, permettant de simplifier la complexité du monde réel tout en fournissant une approximation utile pour la prédiction et le contrôle de la propagation des maladies infectieuses. Ce modèle, combiné à l'implémentation de PageRank, nous offre un cadre pour simuler des campagnes de vaccination ciblées et évaluer leur impact sur la vitesse et l'étendue de l'épidémie.

L'implémentation en Python s'appuie sur la manipulation de matrices pour modéliser le réseau d'individus et simuler la propagation du virus. Le choix de vacciner de manière aléatoire ou ciblée (en se basant sur les scores de PageRank) nous a permis d'observer différents scénarios de propagation et d'évaluer l'efficacité des stratégies de vaccination. Nos résultats mettent en évidence l'importance de la vaccination intelligente, capable de réduire significativement le nombre d'individus infectés et de ralentir la propagation de l'épidémie.

Ce projet illustre le potentiel de méthodes computationnelles avancées, comme l'algorithme de PageRank, dans la modélisation et la gestion des épidémies. Les insights obtenus soulignent l'importance d'une stratégie de vaccination bien pensée, offrant des pistes pour l'optimisation des ressources et la préparation face à des crises sanitaires futures.

Théorie :

PageRank

Le PageRank permet d'évaluer la popularité d'un site web, ou plus précisément, d'une de ses pages. Celui-ci permet ainsi, en théorie, d'augmenter la pertinence des pages trouvées. La recommandation d'une page importante compte plus que la recommandation d'une page moins importante.

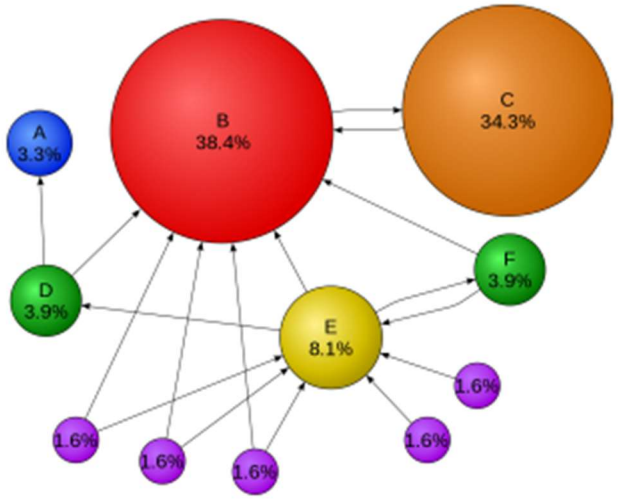


Figure 1- PageRank

Dans la figure 1, on observe les probabilités d'aller sur les destinations possibles. Nous allons utiliser une matrice de transition pour observer les probabilités de se rendre sur une page à partir d'une autre page.

$$P = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/3 & 1 & 0 \\ 1 & 0 & 1/3 & 0 & 1/3 \\ 0 & 1/2 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 1/3 \\ 0 & 0 & 1/3 & 0 & 0 \end{pmatrix} \end{array}$$

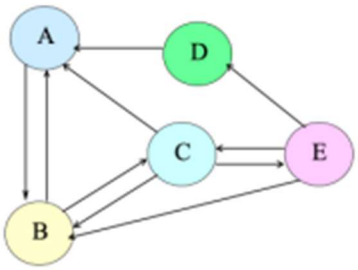


Figure 2- Matrice de Transition

Dans la figure 3, w_0 représente le vecteur initial, c'est la d'où on part dans le graphe.

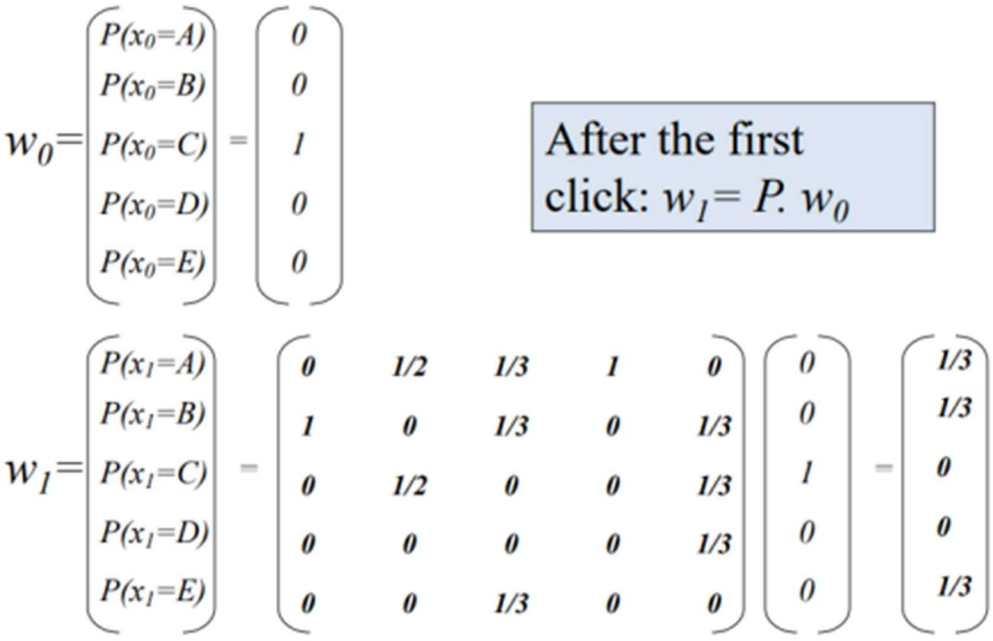


Figure 3- Evolution dans le graphe

Après k itération nous avons donc :

$$w_k = P \times w_{k-1} = P \times (P \times w_{k-2}) = \ldots = P^k \times w_0$$

Cependant, nous remarquons 2 problèmes dans l'algorithme de pageRank qui sont :

— Des pages qui n'ont aucun outlink.

— Des pages qui ont des liens mais aucun vers une autre partie du graphe.

C'est pour cela que nous utiliserons l'algorithme de PageRank amélioré dans notre implémentions. La solution proposée par Google est d'ajouter à la matrice de transition une autre matrice de transition qui représente le gout du promeneur :

$$A = \alpha \times P + (1 - \alpha) \times G$$

Avec α le dumping factor dans [0,1].

Modèle épidémique

L'objectif est d'obtenir une réponse rapide et un contrôle efficace de la propagation des maladies infectieuses afin d'aider les campagnes de vaccination dans les actions à réaliser par les organismes de santé. Nous allons utiliser donc utiliser un modèle épidémiologique homogène :

— chaque individu a un contact égal avec n'importe quel autre individu ;

— le taux d'infection est déterminé par la densité de la population infectée

Ce modèle va nous permettre de prédire le seuil épidémique. C'est-à-dire l'incidence d'une maladie à partir de laquelle il est possible de considérer qu'une épidémie est en cours, l'incidence étant le nombre de nouveaux cas sur une période. Ce modèle permet également d'avoir une approximation des lieux de propagation du virus à partir des personnes infectées. Nous nous retrouvons donc avec cette matrice de transition :

$$A = \alpha \times P + (1 - \alpha) \times \nu z^T$$

Avec :

— α le dumping factor dans [0,1] ;

— P la matrice de transition ;

— v le vecteur de téléportation ;

— z un vecteur unitaire ;

— (1 − α) le jumping factor, ici il représente que le virus se propage d'un individu à un autre individu quelconque du graphe.

Méthodologie

Modèle de réseau : Pour simuler la propagation d'une épidémie au sein d'une population, nous utilisons un ensemble de données réel provenant de Twitter pour construire la topologie de notre réseau. Ce choix permet de modéliser un réseau social complexe où les interactions entre individus peuvent influencer la vitesse et la direction de la propagation de l'épidémie. Le réseau est représenté sous forme de graphe, où les nœuds correspondent aux utilisateurs de Twitter et les arêtes représentent les connexions entre eux, illustrant les interactions et les potentiels chemins de transmission du virus.

Modèle de simulation : Notre modèle de simulation repose sur plusieurs hypothèses pour simplifier le comportement complexe d'une épidémie réelle. Nous considérons des probabilités d'infection et de guérison fixes pour tous les individus du réseau. Au lancement de la simulation, 200 individus sont initialement infectés pour simuler l'introduction du virus dans la population. Ces conditions initiales permettent d'observer l'évolution de l'épidémie à partir d'un petit nombre de cas index.

Stratégies de vaccination : Nous examinons trois stratégies de vaccination distinctes pour évaluer leur efficacité à contrôler l'épidémie :

- Aucune vaccination : Ce scénario sert de référence pour observer la propagation du virus sans intervention.
- Vaccination aléatoire : Dans cette approche, les individus sont vaccinés de manière aléatoire. Cette stratégie ne prend pas en compte la position ou l'importance des nœuds dans le réseau, reflétant une campagne de vaccination sans priorisation spécifique.
- Vaccination intelligente : La stratégie de vaccination intelligente cible les nœuds les plus centraux du réseau, identifiés grâce aux scores de PageRank. Cette méthode suppose que vacciner les individus ayant le plus grand nombre de connexions ou occupant des positions stratégiques dans le réseau peut être plus efficace pour empêcher la propagation du virus. En concentrant les efforts de vaccination sur ces nœuds clés, nous cherchons à maximiser l'impact de la vaccination en coupant les chemins de transmission les plus probables du virus.

Ces différentes stratégies sont analysées à travers des simulations pour déterminer laquelle est la plus efficace pour ralentir ou arrêter la propagation de l'épidémie dans le réseau modélisé.

Implémentation

Dans ce projet, nous avons exploré différentes méthodes pour simuler la propagation d'une épidémie sur un réseau social, en nous appuyant sur l'algorithme de PageRank pour évaluer l'importance des nœuds (utilisateurs) dans le réseau. Initialement, l'approche envisagée était d'implémenter le modèle de simulation en C, afin de tirer parti de la performance et de l'efficacité de traitement des grands ensembles de données qu'offre ce langage. Cependant, cette première tentative a été confrontée à des défis significatifs liés à l'initialisation et à la gestion des structures de données représentant le graphe du réseau social, notamment en raison de la complexité de manipulation directe de la mémoire et des structures dans C.

Face à ces difficultés, et notamment les problèmes de gestion de mémoire lors de l'utilisation d'un graphe de 81 306 nœuds qui nécessitait plus de 24 GB de RAM, une transition vers Python a été opérée. Python, avec sa facilité d'utilisation et la richesse de ses bibliothèques pour le traitement de données et la manipulation de graphes, a permis une approche plus agile et moins consommatrice de mémoire grâce à l'utilisation de matrices creuses et de la bibliothèque SciPy.

Pour surmonter les limitations de performance inhérentes à Python, le programme de PageRank, initialement envisagé en C, a été intégré au projet sous forme de bibliothèque partagée. Cette hybridation a permis de combiner l'efficacité du C pour le calcul intensif de PageRank avec la flexibilité de Python pour la simulation de la propagation et la gestion des stratégies de vaccination. L'utilisation conjointe de C et de Python a ainsi permis d'exécuter la simulation sur un graphe de grande taille en environ 5 minutes, démontrant l'efficacité de cette approche mixte pour le traitement de données complexes sur des réseaux de grande échelle.

```c
int compareNodeScore(const void* a, const void* b) {
    NodeScore na = *(NodeScore*)a;
    NodeScore nb = *(NodeScore*)b;
    if (na.score > nb.score) return -1;
    if (na.score < nb.score) return 1;
    return 0;
}

void powerMethod(double* pagerank, Edge* edges, long long edgeCount, int nodes) {
    double* tempRank = calloc(nodes, sizeof(double));
    for (int iter = 0; iter < MAX_ITER; iter++) {
        for (long long i = 0; i < edgeCount; i++) {
            tempRank[edges[i].to] += pagerank[edges[i].from] / (double)nodes; // Simplification
        }
        double diff = 0.0;
        for (int i = 0; i < nodes; i++) {
            tempRank[i] = DAMPING_FACTOR * tempRank[i] + (1.0 - DAMPING_FACTOR) / nodes;
            diff += fabs(tempRank[i] - pagerank[i]);
            pagerank[i] = tempRank[i];
            tempRank[i] = 0.0; // Reset for next iteration
        }
        if (diff < CONVERGENCE_THRESHOLD) break;
    }
    free(tempRank);
}

// New function that wraps the PageRank calculation
void calculate_pagerank(Edge* edges, int edgeCount, int nodes, double* pagerank) {
    // Initialize PageRank
    for (int i = 0; i < nodes; i++) pagerank[i] = 1.0 / nodes;

    // Calculate PageRank
    powerMethod(pagerank, edges, edgeCount, nodes);
}
```

Figure 4- Implémentation Pagerank en C

Un aspect crucial de notre projet a été le traitement préliminaire des données du réseau social pour optimiser la simulation de la propagation de l'épidémie. La structure initiale des données, tirée du réseau twitter, présentait un ensemble complexe d'identifiants uniques pour chaque utilisateur, rendant le traitement direct et la manipulation du graphe difficile et inefficace en termes de mémoire et de performance de calcul.

Pour pallier ce problème, nous avons mis en œuvre une fonction de mapping qui réassigne à chaque identifiant unique un nouvel identifiant incrémentiel. Ce processus commence par charger les données initiales dans un DataFrame pandas, suivie de la création d'une liste unique d'identifiants à partir des colonnes "from" et "to", représentant respectivement l'émetteur et le destinataire des liens sociaux.

```python
import pandas as pd

# Load the data
df = pd.read_csv('twitter_combined.txt', sep=' ', header=None, names=['from', 'to'])

# Create a unique list of IDs
unique_ids = pd.unique(df[['from', 'to']].values.ravel('K'))

# Create a mapping of old IDs to new IDs
id_map = {old_id: new_id for new_id, old_id in enumerate(unique_ids)}

# Apply mapping
df['from'] = df['from'].map(id_map)
df['to'] = df['to'].map(id_map)

# Save the mapped data to a new file
df.to_csv('mapped_twitter_combined.txt', sep=' ', header=False, index=False)

```

Figure 5- Fonction de mapping en Python

La création d'un dictionnaire de mappage, qui associe chaque identifiant original à un nouvel identifiant numérique séquentiel, permet de restructurer le graphe de manière plus compacte et efficace. Ensuite, cette correspondance est appliquée aux colonnes "from" et "to" du DataFrame, résultant en un ensemble de données remappé où les identifiants sont désormais continus et commencent à 0, ce qui facilite grandement la manipulation du graphe et l'allocation de mémoire pour les structures de données comme les matrices creuses.

Le fichier remappé, enregistré sous un nouveau nom, sert de base pour les calculs de PageRank et la simulation de propagation, réduisant considérablement l'empreinte mémoire et améliorant les performances de calcul. Cette étape de prétraitement s'avère cruciale pour le succès de la simulation, permettant une gestion plus efficace et scalable du réseau social de grand encombrement.

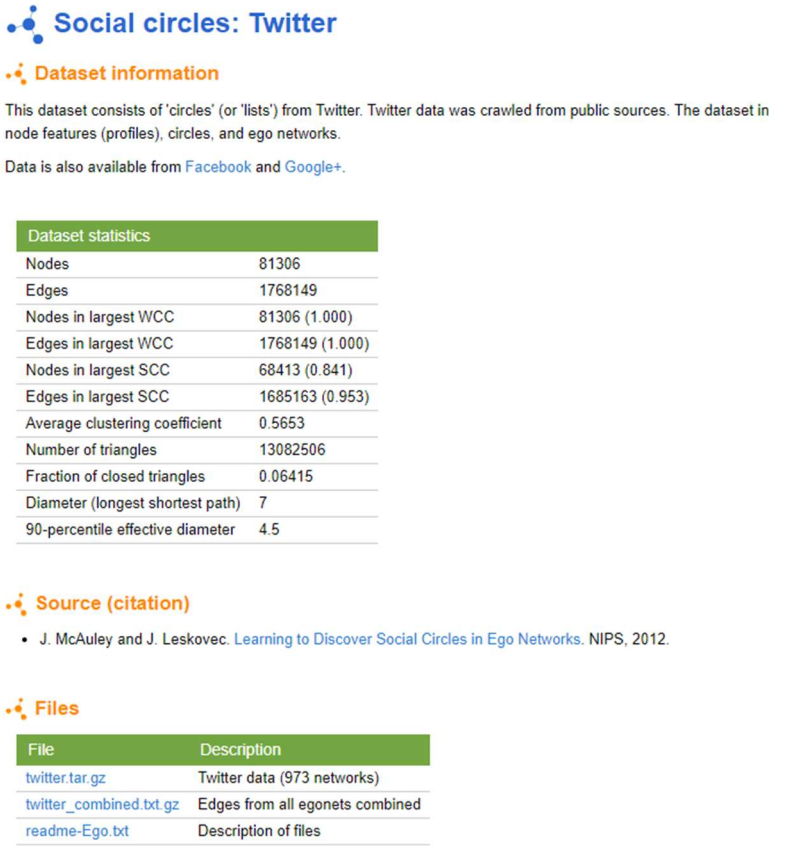
```python
def get_data():
    data = []
    max_id = 0  # Initialize to keep track of the maximum node ID
    with open('./mapped_twitter_combined.txt') as f:
        for item in f:
            parts = item.split()
            from_id, to_id = int(parts[0]), int(parts[1])
            max_id = max(max_id, from_id, to_id)  # Update max_id if necessary
            data.append([from_id, to_id])
    print(f"Maximum node ID in data: {max_id}")
    return data, max_id


def init_matrix_from_mapped_data(filepath, size):
    # Initialize an empty LIL matrix for efficient element-wise operations
    matrix = lil_matrix((size, size), dtype=int)
    with open(filepath, 'r') as file:
        for line in file:
            from_node, to_node = map(int, line.split())
            matrix[from_node, to_node] = 1
            matrix[to_node, from_node] = 1  # Assuming undirected graph

    # Convert to CSR format for efficient mathematical operations afterward
    return matrix.tocsr()


# Parameters
```

Figure 6- Fonction get_data() et Init_matrix_from_mapped_data()

La fonction **get_data()** extrait les données à partir d'un fichier texte où chaque ligne représente une arête entre deux nœuds dans un réseau. Elle itère à travers chaque ligne du fichier, divise la ligne en deux identifiants de nœuds, et les ajoute à une liste data. En même temps, elle maintient à jour la valeur de max_id pour connaître l'identifiant de nœud le plus élevé. Cela aide à déterminer la taille du graphe pour la création ultérieure de matrices.

La fonction **init_matrix_from_mapped_data()** prend un chemin de fichier et une taille de matrice pour initialiser une matrice creuse de type LIL (List of Lists), qui est optimisée pour les opérations d'ajout d'éléments. La matrice est remplie en parcourant à nouveau le fichier, en ajoutant un 1 pour chaque arête indiquant une connexion entre les nœuds. La matrice LIL est ensuite convertie en format CSR (Compressed Sparse Row), qui est plus efficace pour les opérations mathématiques rapides et est idéal pour représenter un graphe sous forme de matrice de transition.

Cas test



Social circles: Twitter

Dataset information

This dataset consists of 'circles' (or 'lists') from Twitter. Twitter data was crawled from public sources. The dataset in node features (profiles), circles, and ego networks.

Data is also available from Facebook and Google+.

Dataset statistics	
Nodes	81306
Edges	1768149
Nodes in largest WCC	81306 (1.000)
Edges in largest WCC	1768149 (1.000)
Nodes in largest SCC	68413 (0.841)
Edges in largest SCC	1685163 (0.953)
Average clustering coefficient	0.5653
Number of triangles	13082506
Fraction of closed triangles	0.06415
Diameter (longest shortest path)	7
90-percentile effective diameter	4.5

Source (citation)

- J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

Files

File	Description
twitter.tar.gz	Twitter data (973 networks)
twitter_combined.txt.gz	Edges from all egonets combined
readme-Ego.txt	Description of files

Figure 7- Jeu de donnée utilisé

Le cas test central de notre projet consistait à analyser la propagation d'une épidémie au sein d'un réseau social réel, en l'occurrence Twitter. Les données, extraites de sources publiques, comprenaient des éléments de réseau tels que des profils d'utilisateurs, des "cercles" ou listes d'amis, ainsi que les réseaux égo-centrés de chaque utilisateur.

Pour notre étude, nous avons utilisé un jeu de données comprenant 81 306 nœuds (utilisateurs) et 1 768 149 arêtes (connexions), reflétant l'intense activité et les interactions complexes entre les utilisateurs sur la plateforme. Les mesures de centralité du réseau, telles que le nombre de triangles fermés et le coefficient de clustering, offraient des insights supplémentaires sur la tendance des utilisateurs à former des groupes serrés, un facteur pertinent dans la diffusion d'informations ou de maladies.

Le diamètre du réseau, la plus longue des plus courtes distances entre deux nœuds, était de 7, indiquant que n'importe quel utilisateur pouvait théoriquement atteindre un autre en traversant au maximum sept liens. Cela démontre la petite taille du monde du réseau Twitter, un concept souvent cité pour décrire le degré de séparation réduit dans les réseaux sociaux.

En se basant sur ces données, nous avons pu simuler la diffusion d'une maladie hypothétique à travers le réseau, en mettant en œuvre différentes stratégies de vaccination pour observer leurs impacts sur la vitesse et l'étendue de la propagation. Cette approche nous a permis d'évaluer non seulement

l'efficacité des stratégies de vaccination mais aussi de mesurer les performances computationnelles des simulations sur des réseaux de grande taille.

Tous les tests ont été réaliser sur ma machine personnelle.

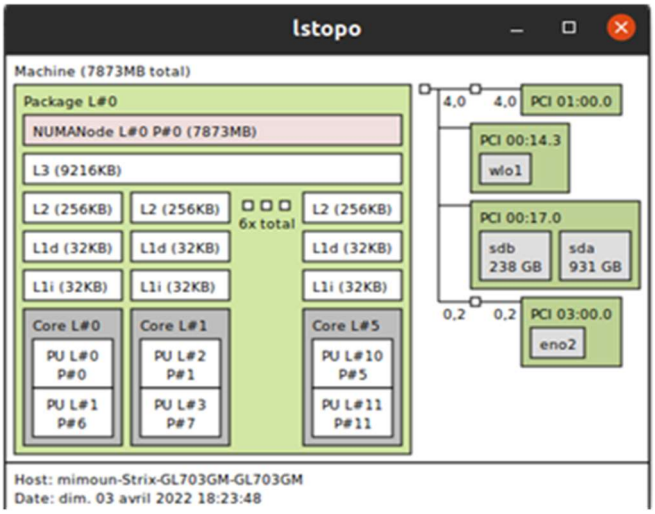


Figure 8- lstopo

La figure 9 représente la propagation de l'épidémie dans notre graphe suivant les différents paramètres définis précédemment, avec un facteur de propagation de 0.05 comme le covid une probabilité de guérison de 0.1, un facteur de damping de 0.85, une vaccination aléatoire qui vaccine 40% de la population aléatoirement et une vaccination intelligente qui vaccine 40% des tops meilleures PageRank. On voit clairement l'utilité de vaccinée intelligemment comme on peut le voir on stoppe la vitesse de propagation de l'épidémie de façon assez importante. En effet, cela peut permettre aux autorités responsables de mieux gérer l'épidémie sans créer forcement une pression sur le système hospitalier

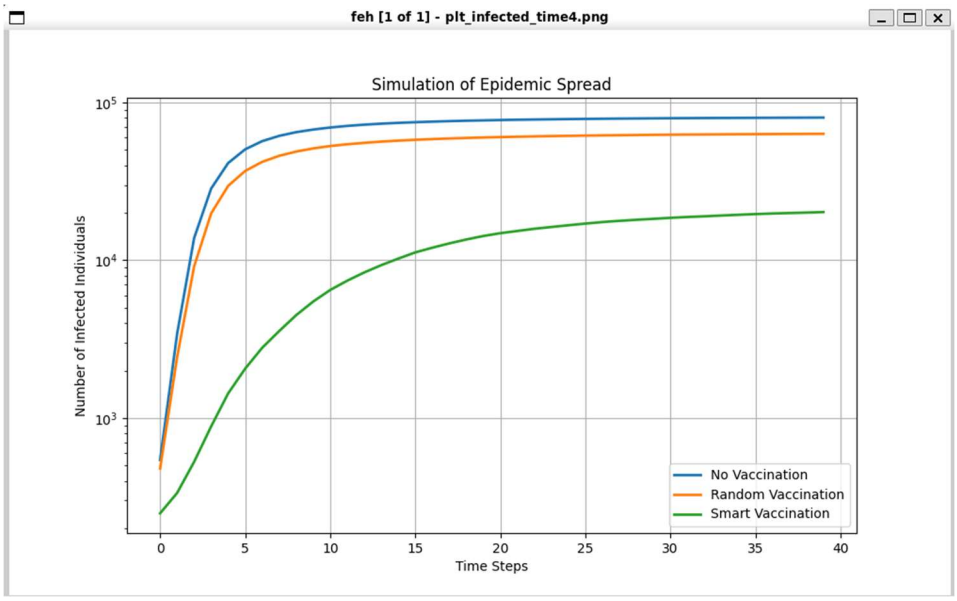


Figure 9- Résultat 1

La figure 10 garde les mêmes paramètres à part pour la vaccination intelligente qui au lieu de vacciné le top 40% on vaccine le top 70% pour avoir une immunité collective plus rapidement et on voit bien que c'est encore plus efficace.

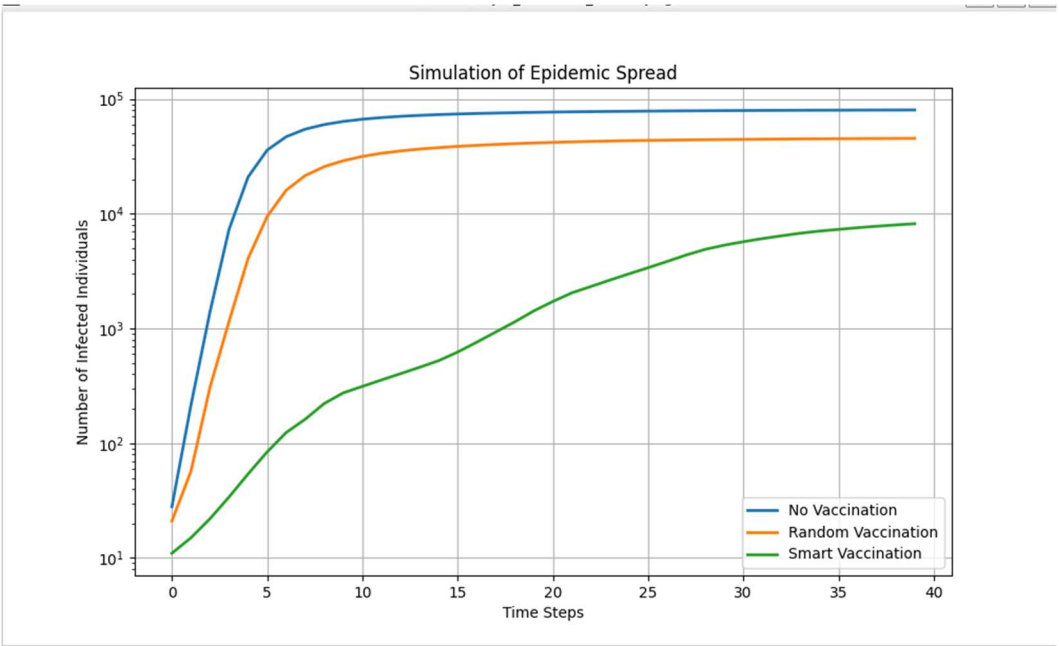


Figure 10- Résultat 2

Cela représente difficilement la réalité car la logistique requise et l'infrastructure industrielle pharmaceutique requise pour vaccinée 70% de la population la plus susceptible de contaminée n'existe pas encore je pense. On pourrait améliorer la simulation pour faire en sorte de graduellement vacciner la population cela aurait été une étude plus intéressante. Mais malheureusement cela augmente considérablement le temps d'exécution du programme. En effet, en mettant en place des métriques pour mesurer le temps d'exécution du calcule de pagerank, et des simulations, on constate que le calcule de pagerank pour un graphe de 81306 nodes est d'environ 4secondes par contre les simulations des différentes stratégies de vaccination prennent le plus de temps environ 300secondes.

On peut retenir principalement d'après l'étude de ces graphes graphe ci-dessus on se rend bien compte que la propagation de la maladie est très importante si aucune stratégie de vaccination n'est mise en place. Une majorité écrasante de la population est alors infectée très rapidement. On remarque également que la propagation de l'épidémie est ralentie si une politique de vaccination est mise en place ; Ce ralentissement est d'autant plus important si la vaccination des personnes est intelligente. Pour une vaccination intelligente moitié moins de personnes se retrouvent contaminées qu'avec une vaccination aléatoire au départ de l'épidémie.

Conclusion

Ce projet a été une opportunité d'explorer en profondeur l'application des algorithmes de PageRank au-delà de leur utilisation traditionnelle dans les moteurs de recherche, en les appliquant à la simulation de la propagation d'épidémies. En adaptant cet algorithme pour identifier les nœuds les plus influents dans un réseau, nous avons pu simuler différents scénarios de vaccination pour observer leur impact sur la propagation d'une épidémie. Cette approche nous a permis d'appréhender la complexité des réseaux sociaux réels et d'explorer des stratégies de vaccination plus efficaces, en se concentrant sur les individus les plus connectés.

Nous avons rencontré des défis techniques significatifs, notamment la gestion de la mémoire pour traiter un grand graphe de 81 306 nœuds, ce qui a exigé une adaptation de notre approche initiale. En combinant Python et C, nous avons réussi à surmonter ces obstacles, démontrant l'importance de la flexibilité et de l'ingéniosité dans la résolution de problèmes complexes. Notre méthode de vaccination progressive, simulant une approche plus réaliste de la vaccination en augmentant le pourcentage de la population vaccinée à chaque étape, a illustré une manière potentiellement plus efficace de contrôler les épidémies dans la réalité.

Les résultats obtenus confirment l'efficacité des vaccinations ciblées basées sur le PageRank pour ralentir significativement la propagation de l'épidémie, offrant des perspectives précieuses pour l'élaboration de stratégies de vaccination dans des situations réelles. Cette recherche ouvre la voie à de futures études pour affiner ces stratégies et les adapter à divers scénarios épidémiologiques, contribuant ainsi à notre compréhension et à notre capacité à gérer les épidémies dans des populations connectées.

En conclusion, ce projet a non seulement renforcé notre compréhension des algorithmes de PageRank et de leur applicabilité dans des contextes non traditionnels mais a également souligné l'importance de l'innovation dans les stratégies de santé publique pour lutter contre les épidémies. Les leçons apprises ici ont un potentiel significatif pour influencer positivement les efforts de prévention des maladies à l'avenir.