ISTY
CALCUL HAUTE PERFORMANCE ET SIMULATION

Simulation de propagation d'un virus : PageRank

Encadré par:
Nahid Emad

Realisé par:
Mohamed Ilyes ELAJROUD
Karim KHARRAZ

Sommaire

Introduction

Dans le cadre du module "Calcul Haute Performance et Simulation", il nous a été demandé de modéliser le problème de la propagation de virus dans différentes populations.

Le but de ce projet est de déterminer un vecteur de vaccination “efficace” permettant de stopper la propagation de la maladie tout en tenant compte de la nature de la population et des caractéristiques du virus.

Pour ce fait, on a eu à modéliser le problème en un premier temps. Puis, de réaliser différentes simulations permettant ainsi d’avoir une vision global de la propagation de virus et de l’approche la plus performante pour l’éradiquer.

1. PageRank

L'algorithme PageRank, inventé par Sergeï Brin et Larry Page, les deux fondateurs de Google, s'inspire des travaux de Jon Kleinberg d'IBM. PageRank était à l'origine du classement des résultats du moteur de recherche Google.

L'idée de base de son fonctionnement est assez simple : Attribuer à chaque page web un score représentant son importance sur le Web . Une page a un *PageRank* d'autant plus important qu'est grande la somme des *PageRanks* des pages qui pointent vers elle (elle comprise, s'il y a des liens internes). Le PageRank est une mesure de centralité sur le réseau du web.

Mathématiquement , Le score pageRank d'une page P représente la probabilité qu'un utilisateur se baladant sur le net et se retrouve "par hasard" sur cette page P.

La formule équivalente est :

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

Avec :

α : *damping factor*

$1 - \alpha$: *Jumping factor*

P : *Matrice de transition*

A : *Matrice de probabilités*

G : *Matrice de google*

Ci-dessous l'équivalent des noms de variables correspondant à notre code :

Variables déclarées dans le code	Équivalent mathématique
`trans`	matrice de transition
`adj`	Matrice d'adjacence
`err`	Facteur de tolérance (utiliser pour arrêter le promeneur/itérations)
`G`	matrice de google

2.Modélisation

Le problème de la propagation d'un virus dans une population peut être assimilé au problème de la promenade aléatoire d'un individus sur internet.
Le problème d'une vaccination efficace peut être ramené au fait qu'il faut vacciner les personnes les plus susceptibles de propager le virus.
On peut donc penser à instaurer un système de Ranking permettant la détection des personnes ayant le plus de probabilité de transmettre le virus.
On se retrouve donc à assimiler notre problème de ranking de personnes à celui de ranking des pages web d'où notre choix de l'algorithme de PageRank.

- Internet → Une population
- Une page sur internet → Un individu dans la population
- Un promeneur → Un virus
- La promenade du marcheur → La propagation du virus
- Le rang d'une page est la probabilité de la présence du promeneur sur cette page → Le rang d'une personne est la probabilité qu'elle soit infecté.

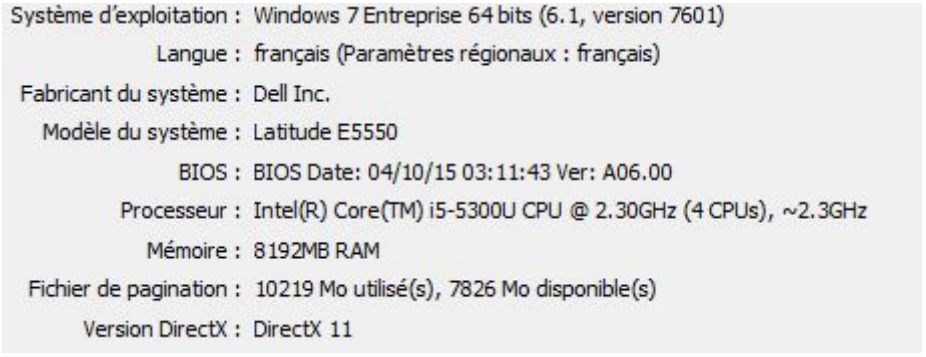
Variables déclarées dans le code	Équivalent logique
`initinfect`	Le ratio de personnes infectés initialement.
`contamprob`	La probabilité qu'une personne contamine une autre (voisins dans le graphe)
`cureprob`	La probabilité de guérison sans vaccination d'une personne.
`ratiovacc`	Le ratio du nombre de personnes vaccinées.
`countit`	La période de simulation (nombre d'itérations).

3. Environnement du projet :

Nous avons choisi le langage python afin de faire la simulation de la propagation du virus.
Ce choix s'explique par le fait que ce langage est dans notre domaine d'expertise et qu'il facilite les opérations de calculs matricielles.
Nous avons utilisés un environnement virtuelle afin d'isoler notre environnement de travail de notre système existant .

Architecture du système :

La simulation à été réalisé avec les spécifications ci-dessous :



Considérations :

- Version Python : **3.7.6**
- Dépendances :
    - `numpy` : Optimise les opérations sur les matrices
    - `csv` : Permet la manipulation des fichiers CSV pris en entrées
    - `matplotlib` : Traçage des courbes
    - `tqdm` : Permet de générer une barre de progression

Installation d'un environnement virtuel :

`Les graphes utilisés pour la simulation sont volumineux. Il est donc impératif d'isoler leurs exécutions pour qu'elle ne soit pas bloquée par le système d'exploitation (kill process).`
Ceci est fait à travers les commandes suivantes:

`pip3 install virtualenv` : Téléchargement de l'environnement virtuel d'exécution.
`virtualenv venv && source venv/bin/activate` : Création de l'environnement virtuel.
`pip3 install -r requirements.txt` : L'ajout des dépendances nécessaires qui sont mentionnées dans le fichier "requirements.txt".

4. Graphes exploités :

4.1 Présentation des graphes

Afin de d'avoir une simulation proche de la réalité, nous avons choisi de travailler sur des réseaux de graphes réels mis en openSource dans le cadre du projet "Stanford Network Analysis Project" .
les graphes qu'on a choisi sont les suivants :

- **Bitcoin Alpha trust network :** Ce réseau est appelé aussi "who-trusts-whom network " . Les noeuds représentent les utilisateurs de la plateforme bitcoin alpha et les liaisons représentent si un utilisateurs a voté pour un autre utilisateurs . Ce réseau est utilisé pour construire un réseau de confiance entre les utilisateurs afin d'éviter les transactions frauduleuses .
- **email-Eu-core network :** Ce réseau représente les communications par mails des membres d'une grande institution de recherche célèbre . les noeuds représente les IDs des membres . une liaison est faite s'il y a un échange de mails établis par les deux parties
- **Gnutella peer-to-peer network, August 9 2002 :** Gnutella est un réseau d'échange de fichiers peer-to-peer . les noeuds représentent les utilisateurs de la plateforme . les liaisons représentent l'échange de fichiers .

Statistiques des réseaux		
Nom du réseau	nombre de noeuds	nombres de liaisons
Bitcoin Alpha trust network	3783	24 186
email-Eu-core network	1005	25 571
Gnutella peer-to-peer network, August 9 2002	8114	26 013

Le choix de ces graphes n'est pas pris au hasard : en effet , on voulait toucher a trois niveaux de populations (petite, moyenne , grande) . nous étions limités à la mémoire disponible (7.4 Gb de RAM) sur notre machine .

4.2 Manipulation des graphes :

les graphes sont livrés dans un fichier csv composé d'ensemble de lignes représentant une liaison sous la forme 'a b' ou 'a b' ou 'a,b' où a et b sont les noeuds .

Afin d'exploiter ce fichier , on a mis en place une fonction qui permet de traduire ces lignes en une matrice d'adjacence . Le principe est simple :

- stocker toutes les liaisons dans une liste 'edges' (ou on calcule en même temps le nombre de noeuds)
- on initie la matrice d'adjacence avec des zéros partout
- si on a une liaison (v1,v2) dans edges , on met a 1 la case (v1,v2) dans la matrice d'adjacence .

5. Simulation et interprétation

Afin d'avoir une analyse pertinente, il est essentiel de varier des variables et de fixer d'autres pour de détecter leurs impactes respectifs sur les différents résultats.
De ce fait on a fait le choix de diviser notre travail en deux :

5.1 Simulation 1

5.1.1 procédure de simulation :

On a fixer :

- Le nombre d'itérations= 200.
- Le pourcentage des individus infectés = 5% de la population.
- La probabilité qu'un individus contamine un autre = 30%.
- Le pourcentage de la population vaccinée: 25%.

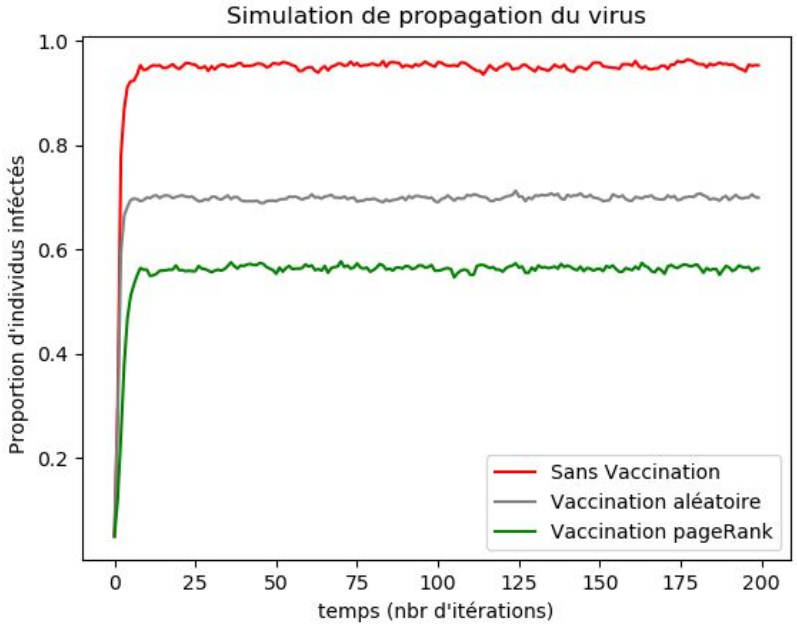
Et ou on fait varier:
Les populations :

- email
- bitcoin
- gnutella

On note que les graphes en questions varient en termes de nombres de noeuds et de nombres d'arcs.

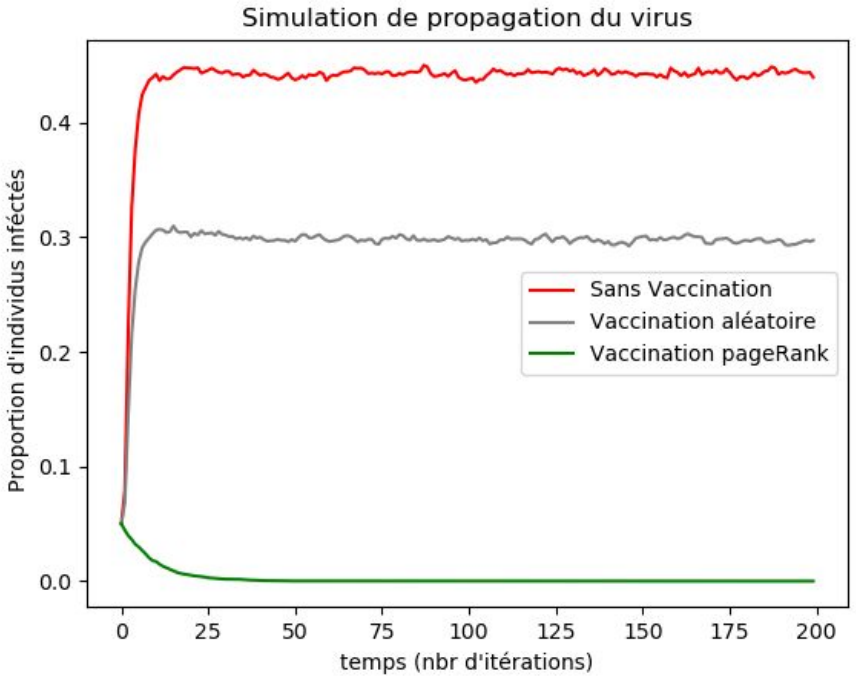
5.1.2 Résultats de simulation :

Email:



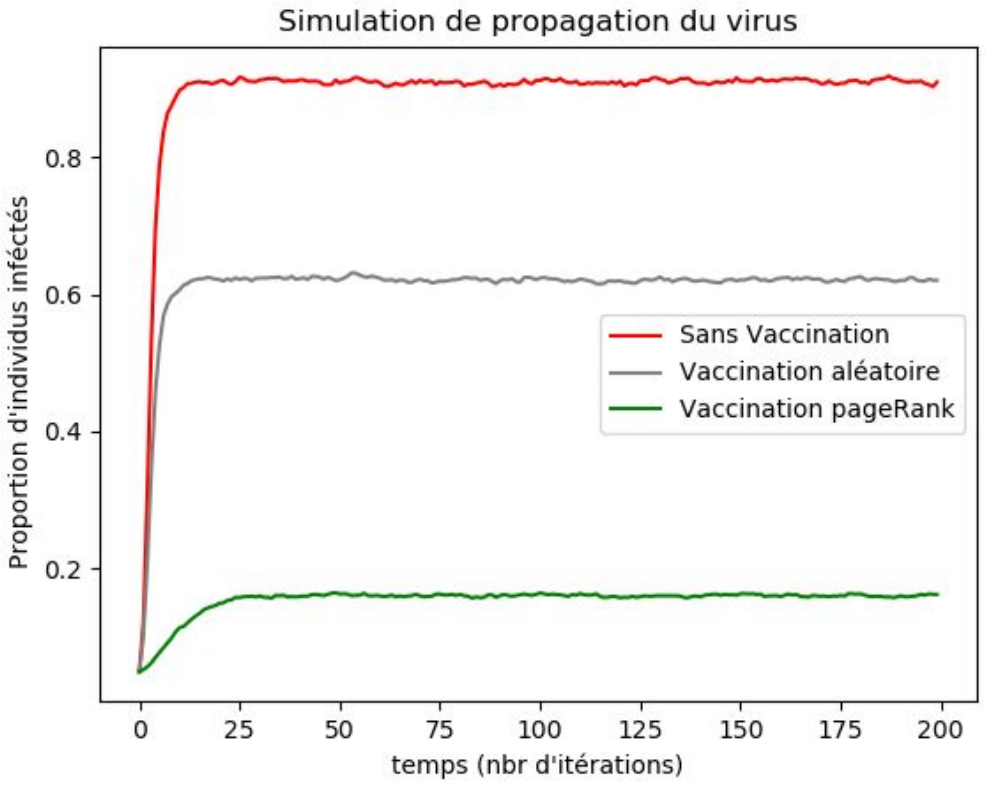

```
C:\Users\A755734\PycharmProjects\hpc\venv\Scripts\python.exe C:/Users/A755734/PycharmProjects/hpc/pagerankinfection.py
graph name : email-Eu-core.txt
number of nodes : 1005
number of edges  25570
100%|██████████| 25570/25570 [00:00<00:00, 1420470.36it/s]
génération de la courbe de propagation de la maladie sans Vaccination:
100%|██████████| 199/199 [00:01<00:00, 111.04it/s]
génération de la courbe de propagation de la maladie avec vaccination aléatoire:
100%|██████████| 199/199 [00:01<00:00, 192.07it/s]
génération de la courbe de propagation de la maladie avec vaccination pageRank:
100%|██████████| 199/199 [00:00<00:00, 272.96it/s]
```

Bitcoin:




```
C:\Users\A755734\PycharmProjects\hpc\venv\Scripts\python.exe C:/Users/A755734/PycharmProjects/hpc/pagerankinfection.py
graph name : soc-sign-bitcoinalpha.csv
number of nodes : 7605
number of edges  24185
100%|██████████| 24185/24185 [00:00<00:00, 833923.12it/s]
génération de la courbe de propagation de la maladie sans Vaccination:
100%|██████████| 199/199 [01:01<00:00,  3.25it/s]
génération de la courbe de propagation de la maladie avec vaccination aléatoire:
100%|██████████| 199/199 [00:29<00:00,  6.74it/s]
génération de la courbe de propagation de la maladie avec vaccination pageRank:
100%|██████████| 199/199 [00:00<00:00, 681.47it/s]
```

Gnutella :




```
C:\Users\A755734\PycharmProjects\hpc\venv\Scripts\python.exe C:/Users/A755734/PycharmProjects/hpc/pagerankinfection.py
graph name : p2p-Gnutella09.txt
number of nodes : 8114
number of edges  26012
100%|██████████| 26012/26012 [00:00<00:00, 867019.26it/s]
génération de la courbe de propagation de la maladie sans Vaccination:
100%|██████████| 199/199 [02:34<00:00,  1.29it/s]
génération de la courbe de propagation de la maladie avec vaccination aléatoire:
100%|██████████| 199/199 [01:13<00:00,  2.71it/s]
génération de la courbe de propagation de la maladie avec vaccination pageRank:
100%|██████████| 199/199 [00:19<00:00, 10.11it/s]

Process finished with exit code 0
```

5.1.3 .Analyse

La propagation du virus est importante sur les différentes populations (50% ~ 90%). Cette différence s'explique essentiellement par le nombre d'arcs des graphes.

Une vaccination par hasard a un faible impacte sur la propagation du virus sur l'ensemble des populations => Cette méthode de vaccination ne permettrait pas l'éradication de l'épidémie.

Une vaccination PageRank à des meilleurs résultats que la vaccination par hasard. Mais son efficacité dépend de la taille du graphe représentatif de la population ainsi que le nombres de liens qui relie ses individus.

On remarque qu'on obtient un meilleur résultat pour un nombre d'individus plus important. Ce résultat est même meilleur si on augmente le nombre de liaisons.
On explique cela par le fait que plus le graphe est grand plus la probabilité d'avoir un ranking de personnes plus dispersé est important. Et vu qu'on va vacciner un pourcentage fixe de la population et qu'on sélectionne ce pourcentage parmis les personnes ayant le plus de probabilité d'être infecté.=> On a plus de probabilité de stopper la propagation du virus.

5.2 Simulation 2

Pour cette deuxième partie, on choisit de se focaliser sur la dernière population (graphe gnutella) car c'est la ou on observe les meilleurs résultats de PageRank.

5.2.1 procédure de simulation :

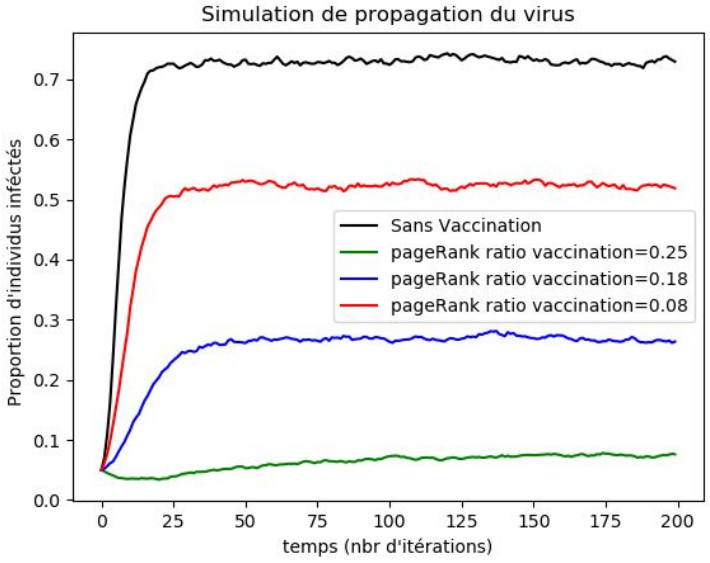
On a fixer :

- Le nombre d'itérations= 200.
- Le pourcentage des individus infectés = 5% de la population.
- Le pourcentage de contamination une première fois à 30%
- Le pourcentage de contamination une première fois à 10%

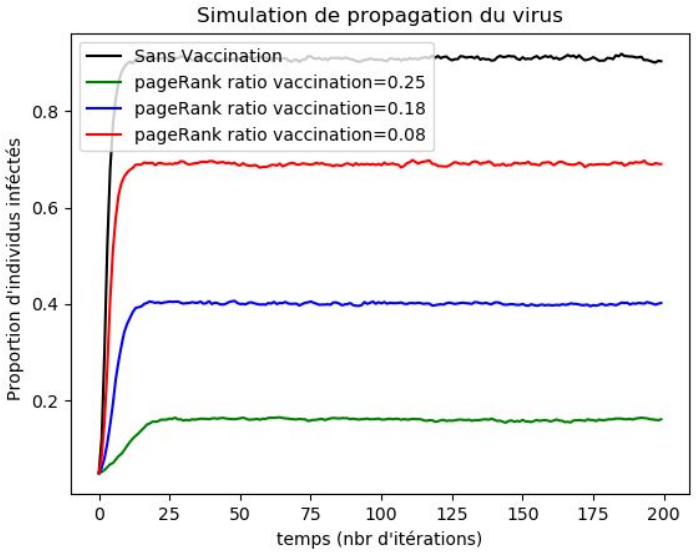
Et ou on fait varier:
Le pourcentage de vaccination= 25%, 18% et 8% pour les deux pourcentage de contamination (30% et 10%).

5.2.2 Résultats de simulation :

contamination = 10%



contamination = 30%



5.2.3 Analyse

On remarque pour les deux pourcentages de contamination initial (10% ou 30%). Les résultats de la vaccination PageRank sont étroitement liées au pourcentage de la vaccination.
On remarque aussi qu'on obtient des résultats satisfaisant à partir d'un taux de vaccination bien déterminé (25%) et qu'il est inutile donc de vacciner toute la population pour éradiquer la maladie.
Ce taux de vaccination dépends primordialement des caractéristiques du virus et de sa propagation :

- Le pourcentage des personnes infectés initialement.
- La probabilité de contamination.
- La probabilité de guérison.

On ne peut donc pas généraliser. Chaque épidémie est unique et nécessite donc une approche spéciphique.

Conclusion

L'algorithme de PageRank est encore utilisé par google mais couplées à plusieurs autres techniques d'indexation encore secrète . Ceci montre la puissance de cette technique malgré un algorithme basé sur une idée assez simple .

Malgré les limites hardware de notre machine, nous avons pu faire différentes interprétations sur cette algorithme avec une application dans la vie réel (propagation d'un virus dans une population)

Afin d'exploiter le maximum de potentiel de PageRank , on préconise l'utilisation de graphes plus grand . Ceci nous expose à plusieurs problèmes principalement des problèmes hardware. Pour un calcul plus rapide , le code doit supporter le multi-threading (le code actuel utilise un seul thread par défaut) afin de répartir la charge et diviser le temps de calcul par le nombre de threads . idéalement du multi-processing pour avoir des performances plus intéressantes .Sur ce genre de manipulation, on doit faire attention sur les variables partagés .

Concernant la RAM, le programme peut vite être stopper dès qu'il demande une quantité importante de Ram. si on prend un exemple de graphe contenant 55000 noeuds , on aura besoin d'une matrice de 3 025 000 000 cases de floats soit 220 Gb de Ram ! Dans ce cas on doit utiliser un système de fichiers HDFS afin de fetcher a chaque fois une partie , faire les manipulations nécessaires et les restocker .

Avec ce genre d'améliorations nous espérons avancer plus sur cet algorithme et exploiter son potentiel