Université de Liège

Théorie des Graphes

PROJET 19 : SIMULATION DE LA PROPAGATION D'UNE MALADIE DANS UN RÉSEAU DE CONTACTS REPRÉSENTÉ PAR UN GRAPHE

GANOUBONG TINGUE Leslie Lucynda
BOUSTANI Mehdi

Table des matières

1. Enoncé

Il s'agit d'une simulation de propagation d'une maladie dans un réseau de contacts représenté par un graphe. Les agents du réseau sont marqués comme sains, infectés ou guéris. Initialement, quelques agents sont choisis au hasard comme infectés et les autres sont marqués comme sains. À chaque étape, chaque agent sélectionne aléatoirement un sous-ensemble de ses voisins dans le réseau et les rencontre. Si un agent sain rencontre un agent infecté, il devient infecté à son tour. Les rencontres sont symétriques, ce qui signifie que l'agent qui initie la rencontre n'a pas d'importance. De plus, les agents infectés se rétablissent après un certain nombre d'étapes.

2. Méthode

Pour simuler l'évolution de l'épidémie, nous utilisons la méthode suivante :

— Création d'un graphe représentant le réseau de contacts en utilisant la bibliothèque `NetworkX`.

— **Attribution des états aux agents** :

Chaque nœud du graphe représente un agent et est associé à un objet `Agent` qui comporte un état initial défini comme étant sain, infecté ou guéri. La période (nombre de pas) requise pour que les agents récupèrent est d'une semaine, soit 7 jours simulés.

Chaque état est associé à une couleur via un dictionnaire Python :

- Sain : Vert
- Infecté : Rouge
- Guéri : Bleu

Initialement, un nombre aléatoire d'agents est infecté pour démarrer la simulation.

— **Simulation de la propagation de l'infection** :

À chaque étape de la simulation, chaque agent (nœud) sain sélectionne aléatoirement un sous-ensemble de ses voisins dans le réseau pour simuler une rencontre. Si un agent sain rencontre un agent infecté, il devient à son tour infecté. Cette propagation se fait de manière aléatoire et symétrique, sans distinction entre l'initiateur de la rencontre.

— **Mise à jour des états et stockage des instantanés** :

Les états des agents sont mis à jour après chaque étape de la simulation. Ces états sont enregistrés à chaque étape pour constituer une série d'instantanés représentant l'évolution de la propagation de l'infection.

— **Visualisation de l'évolution de l'épidémie** :

Cette méthode utilise la bibliothèque `matplotlib.animation` pour générer une animation montrant l'évolution des états des agents (sains, infectés, guéris) sur le graphe de contacts au fil du temps. Chaque image de l'animation représente un instantané de l'état du réseau à une étape spécifique de la simulation.

3. Algorithme

L'algorithme de base fonctionne comme suit :

3.1. Initialisation

L'initialisation du système consiste à créer un graphe représentant le réseau des contacts entre les agents. Le graphe est créé, sous forme de class python, en utilisant la bibliothèque `NetworkX`. On spécifie le nombre d'agents, de communautés (ainsi que leur taille) et le degré minimum des noeuds. De plus, le nombre d'agents et de communautés est choisis par l'utilisateur via le terminal. Ensuite, on attribue un état initial à chaque agent. Au départ, seuls quelques agents sont infectés.

3.2. Boucle principale

La boucle principale de la simulation s'exécute un nombre de fois défini par la variable `frames`. À chaque itération de la boucle, on procède comme suit :

— On parcourt tous les agents sains.
— Pour chaque agent sain, on choisit un sous-ensemble aléatoire de ses voisins.
— Pour chaque voisin de l'agent sain, on vérifie si ce voisin est infecté.
— Si le voisin est infecté, l'agent sain devient infecté.

3.3. Animation

À la fin de chaque itération de la boucle principale, on stocke l'instantané de l'état actuel du graphe dans une liste nommée `snapshots`.

On utilise ensuite la bibliothèque `matplotlib.animation` pour animer le graphe en parcourant la liste des instantanés.

Plus précisément, la fonction `run_simulation()` effectue les étapes suivantes :

— Initialisation du graphe et des agents.
— Boucle principale de la simulation.
— Stockage des instantanés.
— Animation du graphe.

La fonction `animate_graph()` effectue les étapes suivantes :

— Création de la figure et de l'axe.
— Calcul de la position des noeuds.
— Définition des couleurs des noeuds.
— Création de la fonction d'animation.
— Exécution de l'animation.

4. Théorie

Soit G = (V, E) le graphe représentant notre réseau de contact.

4.1. Graphe non orienté

Il est évident que dans ce projet, nous devons manipuler un graphe non orienté. Si un agent est relié à un autre, cela signifie qu'ils se connaissent mutuellement et donc qu'ils peuvent potentiellement se rendre visite l'un l'autre.

Plus formellement, on a que E est une relation symétrique sur V : $\forall v_1, v_2 \in V : (v_1, v_2) \in E \Rightarrow (v_2, v_1) \in E$.

4.2. Connexité

En effet, notre graphe G est connexe étant donné que si un agent est infecté, tout le monde le devient. Cela signifie que d'un sommet, on peut rejoindre celui qu'on veut et cela malgré le grand nombre d'agents et de communautés.

4.3. Degré minimum

Dans notre graphe, nous avons décidé d'imposer un degré minimum aux noeuds pour accélerer la propagation et simuler les relations entre agents d'une manière plus flexible. En effet, cela améliore donc la connexité de G.

Nous imposons également que sa valeur soit > 1 pour pouvoir manipuler plus facilement l'aspect aléatoire du choix des voisins de chaque sommets. De plus, si sa valeur était nulle, il y aurait des agents non impactés par la maladie (sommets isolés).

5. Commande

Pour lancer le programme, il faut utiliser la commande suivante :

```
python3 main.py
```

5.1. Execution

En exécutant cette commande, le programme demande à l'utilisateur de choisir le nombre d'agents (i.e le nombre de sommets du graphe) et le nombre de communautés dans le terminal.

Soit G = (V, E) le graphe représentant notre réseau de contact et soit C l'ensemble des communautés du graphe. Il est en effet primordiale de donner un intervalle au nombre de sommets et de communautés sinon on risquerait de manipuler des données absurdes.

Donc, nous avons décider de choisir ces intervalles :

— $\#V \in [1, 700]$

— $\#C \in [1, 10]$

tels que $\#C \leq \#V$

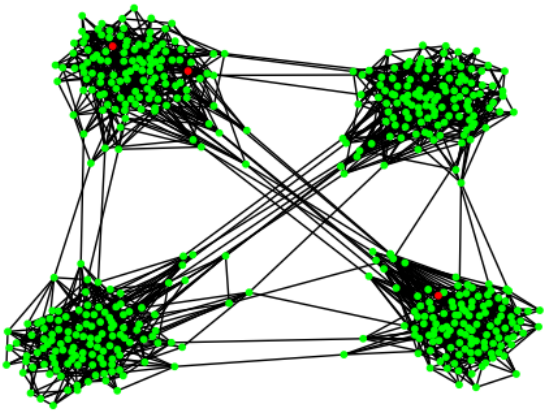
En effet, avoir 0 sommets/communautés nous donnerait un graphe vide et un graphe sans sommets ne contient aucune information sur les relations entre les agents.

De plus, avoir un majorant de 700 sommets et de 10 communautés est largement suffisant pour simuler notre propagation.

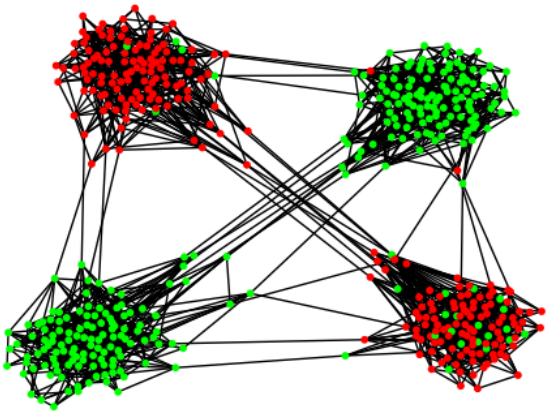
6. Exemples

Je choisis arbitrairement : #V = 500, #C = 4

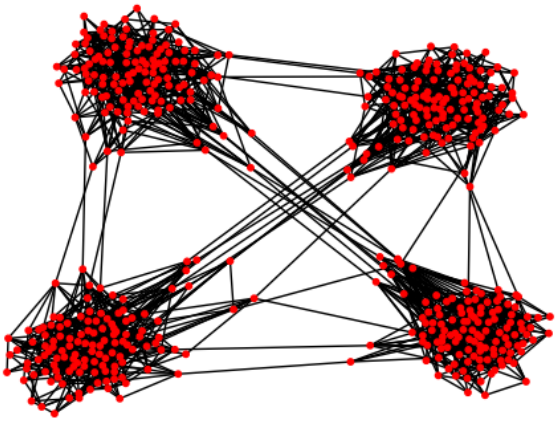
Voici quelques instantanés représentant la propagation :



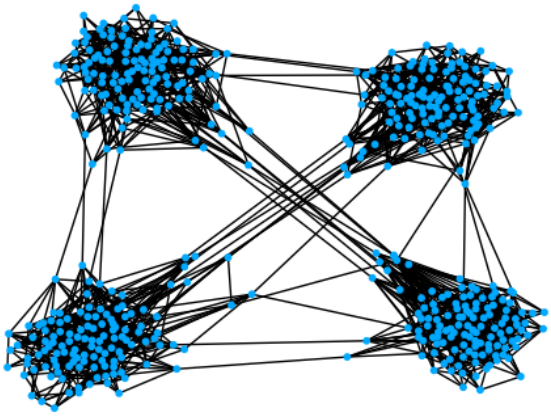
Tous les agents sains sauf quelques uns



La propagation commence



Tous les agents sont infectés



Après quelques jours (7), les agents guérissent progressivement

7. Bibliographie

https://community.wolfram.com/groups/-/m/t/1907703

https://thescipub.com/pdf/jcssp.2023.75.86.pdf

https://dataleek.io/index.php/2017/11/28/khan-academy-interview-project/