MATH0499 : Projet d'Implémentation n°7

Alexandre Andries, Thomas Rotheudt

Table des matières

1 Problématique

Le projet consiste à représenter une population sous la forme d'un graphe. Chaque sommet du graphe correspond donc à un "citoyen". Les arêtes du graphe représentent quant à elles les liens d'interaction entre ces citoyens ; soit, si ceux-ci sont susceptibles d'entrer en contact physiquement. Le but du projet est de simuler la propagation d'une infection (virus, bactérie etc.) à travers une population. Une simulation type commence toujours avec une situation initiale où l'entièreté de la population est marquée comme *saine*, à l'exception d'un sommet (choisi aléatoirement parmi tous les sommets) qui est marqué comme *contaminé*. Il s'agit là du patient zéro.

La simulation évolue ensuite étape par étape, représentant une évolution dans le temps (temps discret). A chaque étape, on considère que les citoyens infectés entrent en contact avec leur réseau et infectent celui-ci. La propagation évolue donc de manière exponentielle à travers la population. Après plusieurs étapes successives en tant qu'infectés, les citoyens guérissent et obtiennent une immunité, les rendant invulnérables à l'infection.

2 Stratégie

On considère un graphe divisé en plusieurs communautés. Le graphe est non-orienté car les rencontres entre citoyens sont considérées comme symétriques. L'agent initiant la rencontre n'a pas d'importance, on s'intéresse seulement aux sommets adjacents des sommets dont le statut est *contaminé*. Afin de créer un environnement relativement réaliste représentant une situation plausible, on utilise un graphe comprenant plusieurs "partitions", celles-ci étant vues comme des communautés. Ces communautés peuvent par exemple représenter des familles, des entreprises où des collègues se croisent, des cercles d'amis. Ces "sous-graphes" ont des membres ayant une forte connexité entre eux. En revanche, les communautés ont quant à elles une plus faible chance d'avoir des arêtes reliant leurs sommets entre elles.

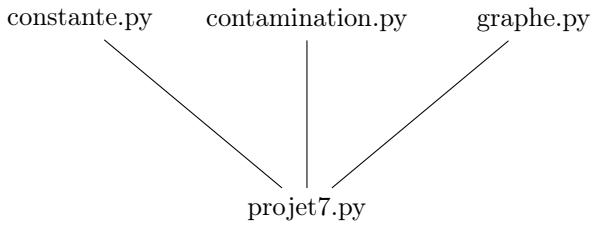
Considérons par exemple un citoyen standard possédant un groupe d'amis, une famille comprenant de nombreux cousins, et un lieu de travail. Ce citoyen sera représenté comme un sommet d'une des communautés du graphe. Il pourra partager de nombreuses arêtes avec sa communauté rapprochée, par exemple sa famille, mais également une arête avec un collègue et/ou un ami qui, à son tour aura des connexions avec des membres de son entourage. Le graphe permet donc de représenter réalistement les liens connectant les citoyens d'une population.

La mécanique de propagation est relativement simple. On considère chaque étape de la simulation comme équivalente à quelques jours, voire une semaine, au cours de laquelle les sommets infectés rencontrent et contaminent leurs contacts (i.e. les sommets qui leur sont adjacents). Ainsi, chaque rencontre permet la propagation de l'agent analysé, par exemple un virus comme la Covid-19. La simulation représente visuellement l'expansion exponentielle des contaminations. Partant d'un unique patient zéro, on observe qu'en seulement quelques étapes l'entièreté des sommets du graphe est contaminée.

La simulation suit également certaines règles. Tout d'abord, un sommet *contaminé* devient *guéri* après un certain laps de temps (nombre d'étapes prédéfini). De plus, un sommet *guéri* obtient également une immunité au virus et ne peut plus être contaminé à nouveau. Il est également possible de choisir l'option de rendre le virus mortel ; c'est à dire qu'un citoyen du graphe pourra aléatoirement guérir ou, malheureusement, décéder après un certain temps en tant qu'infecté.

3 Architecture et Logique du Code

Le code du projet est écrit en PYTHON 3 et est principalement construit sur base des fonctionnalités offertes par les libraires *NetworkX* et *matplotlib*. L'architecture du code est relativement simple : une unique fonction *animate()* fait office de boucle principale. Cette fonction est implémentée sur base de plusieurs autres fonctions implémentant chacune un sous-problème de la représentation du graphe. La fonction *animate* est définie dans le fichier principal (main) du code, nommé PROJET7.PY. Les autres fichiers du code sont CONTAMINATION.PY, GRAPHE.PY et CONSTANTE.PY. Dans ces fichiers annexes sont définies les variables globales et fonctions permettant le bon fonctionnement de *animate()*.



Le code consiste en l'utilisation de la fonction *FuncAnimation* offerte par la librairie *matplotlib*. Celle-ci permet de produire une animation visuelle enregistrée sous format GIF, où chaque "frame" de l'animation est en fait la représentation d'une itération d'une fonction qui met à jour l'état des sommets du graphe. Les différents fichiers de l'implémentation et leur contenu sont décrits ci-dessous.

3.1 constante.py

Le fichier CONSTANTE.PY contient les variables globales sur lesquelles se basent la construction du graphe et la propagation des contaminations. Parmi ces variables on retrouve notamment le nombre de sommets que contient chaque partition du graphe, ainsi que le nombre de partitions que l'on souhaite représenter dans le graphe. De plus, comme la génération du graphe est effectuée de manière aléatoire par une fonction de la librairie *NetworkX*, on définit également les données requises par cette fonction. Ceci inclut la connexité entre sommets d'une même communauté, ainsi que la connexité entre différentes communautés du graphe. Les données ci-dessus sont respectivement représentées dans le fichier par les variables NBR_SOMMETS, NBR_COMMUNAUTES, CONNEXITE, et INTERCONNEXITE. D'autres variables servant au bon fonctionnement du programme sont également définies dans le fichier.

3.2 contamination.py

Ce fichier contient les définitions des trois fonctions servant à gérer la propagation de l'infection entre sommets adjacents. La fonction *coloriage()* sert à colorier les sommets du graphe en fonction de leur propriété "weight" ; un sommet dont la propriété est "S" pour "sain" est colorié en gris, si la propriété vaut "C" pour "contaminé", sa couleur devient rouge, et enfin pour les propriétés "G" et "D" (respectivement "guéri" et "décédé"), la couleur devient vert, ou violet.

La fonction *nettoyage_anciens_conta()* reprend quant à elle la liste des sommets précédemment infectés et la nettoie afin d'éviter de répéter des contaminations ne respectant par les règles du graphe.

Enfin, la fonction *etapes_contamination()* permet de déterminer quels sont les sommets du graphe adjacents à des sommets contaminés à l'étape précédente afin de changer leur statut à "C" pour "contaminé". Cette fonction reprend les deux précédentes et gère le processus d'infection des sommets du graphe ainsi que leur guérison après plusieurs tours en tant qu'infectés.

3.3 graphe.py

Le fichier GRAPHE.PY sert à l'initialisation du graphe. Dans le code, le graphe est enregistré comme une variable qui est successivement utilisée à chaque étape du programme. Les deux fonctions définies dans le fichier sont *init_graphe()* et *patient_zero()*. La première permet de générer aléatoirement un graphe partitionné grâce à la fonction *random_partition_graph()* provenant de la librairie *NetworkX*. Cette dernière crée un graphe contenant plusieurs communautés dont les sommets sont très connexes entre eux. Les arguments de cette fonction sont les variables globales décrites précédemment dans le fichier CONSTANTE.PY. La fonction *init_graphe()* permet également de retenir la disposition graphique de la représentation du graphe afin de pouvoir utiliser successivement le même "schéma" dans l'animation.

La fonction d'initialisation fait appel à la seconde fonction définie dans le fichier : *patient_zero()*. Celle-ci sélectionne aléatoirement un sommet parmi tous les sommets du graphe et le définit comme infecté. Ce sommet correspond donc au "patient zéro" de l'épidémie, et sera à l'origine des infections des autres sommets, à commencer par ceux avec lesquels il partage une arête. Les autres sommets du graphe sont tous initialisés comme sains.

4 Benchmarking

Cette section reprend des exemples de simulation. On peut ainsi comparer les différentes étapes de la propagation de l'agent infectieux, mais également différentes échelles. On observera également des cas particuliers, ou des cas extrêmes jugés intéressants.

4.1 Exemple 1 : Propagation à grande échelle

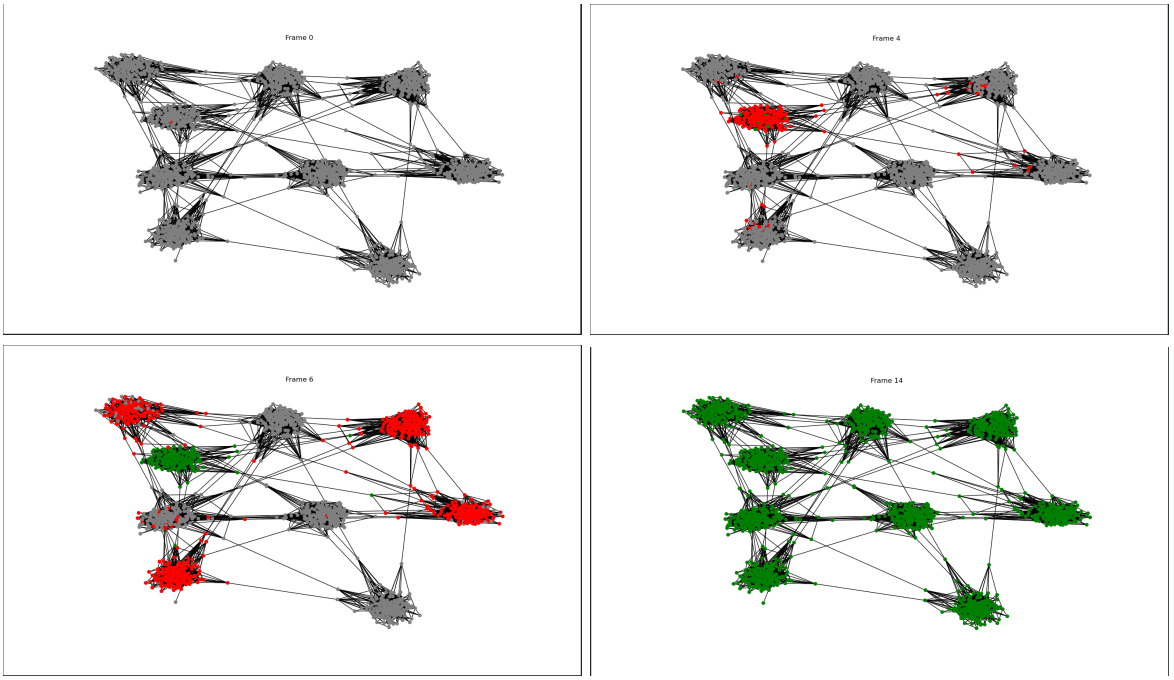


FIGURE 1 – Propagation à grande échelle (7 com., 170 sommets/com.).

La série d'images ci-dessus correspond au cas standard de la simulation. On y observe la propagation graduelle de l'épidémie. Notons l'apparition de cas de patients guéris dans la communauté de départ avant même que toutes les autres communautés n'aient été touchées. Au final, après un certain nombre d'étapes, la totalité des sommets apparaît comme guéris, ce qui implique que la totalité de la population a été préalablement infectée.

4.2 Exemple 2 : Propagation à petite échelle

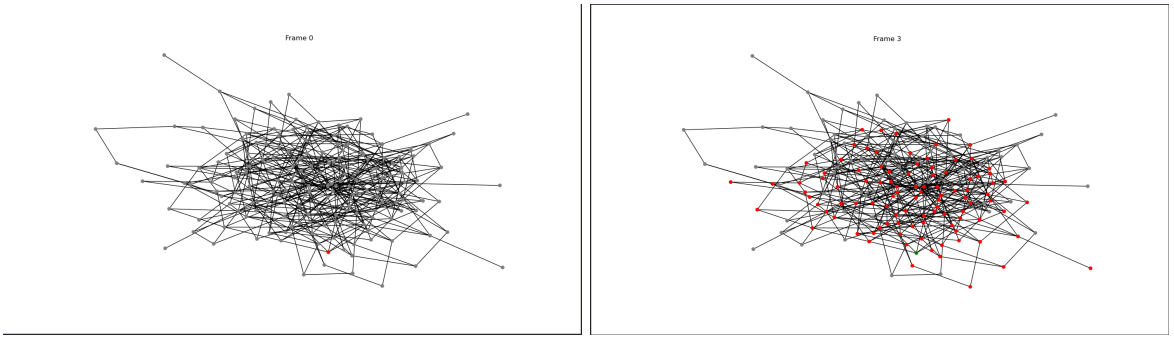


FIGURE 2 – Propagation à petite échelle (1 seule communauté à 170 sommets).

L'analyse d'une plus petite population ne possédant pas de partition permet d'observer les contaminations au sein d'une même communauté. Ici, en seulement quatre étapes, la quasi totalité des sommets a été contaminée. Ceci prouve le caractère exponentiel de l'épidémie.

4.3 Exemple 3 : Virus mortel

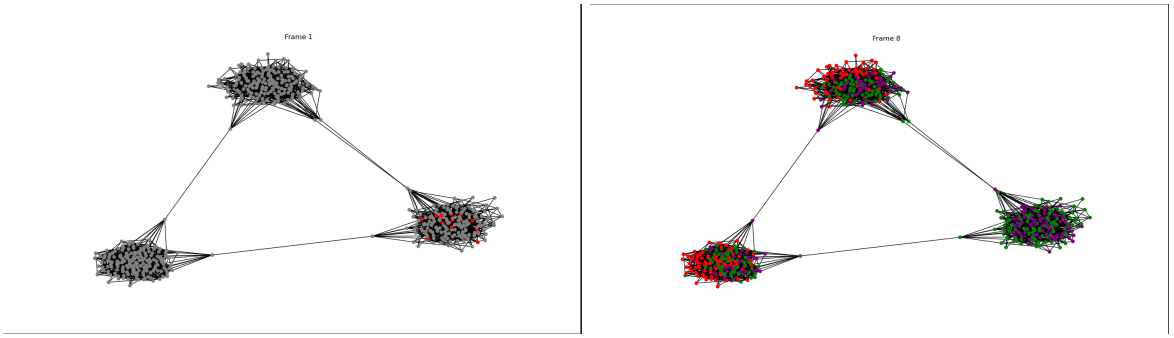


FIGURE 3 – Cas de virus mortel, les sommets violets sont décédés post-infection.

Les captures ci-dessus montrent le cas d'une simulation ou le virus est mortel. On observe ainsi qu'au final, une partie considérable de la population est décédée des suites de l'infection.

4.4 Exemple 4 : Isolement presque total

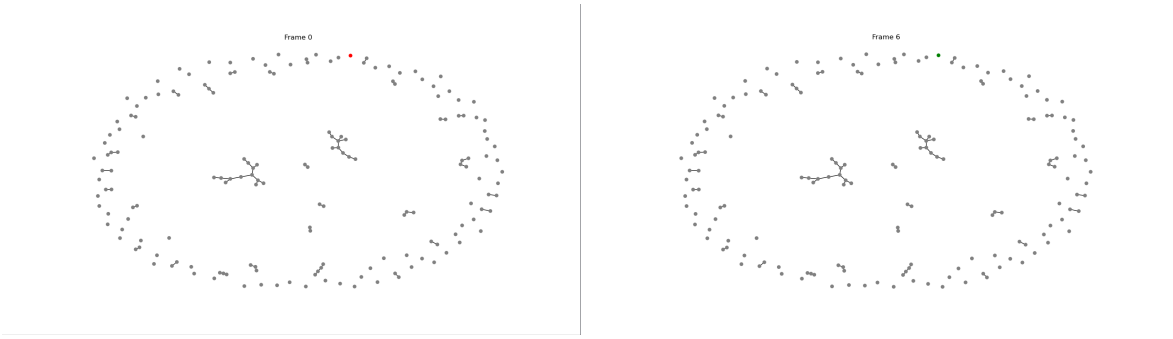


FIGURE 4 – Isolement quasi total des sommets, l'épidémie ne progresse pas.

En cas d'isolement presque totalement respecté par les citoyens, on observe que l'agent infectieux n'est pas capable de se propager de proche en proche. L'unique cas malade guérit après quelques étapes et devient donc inoffensif. L'isolement total est donc la meilleure solution pour endiguer l'épidémie sans contaminer tous les sommets du graphe.

4.5 Exemple 5 : Isolement des communautés

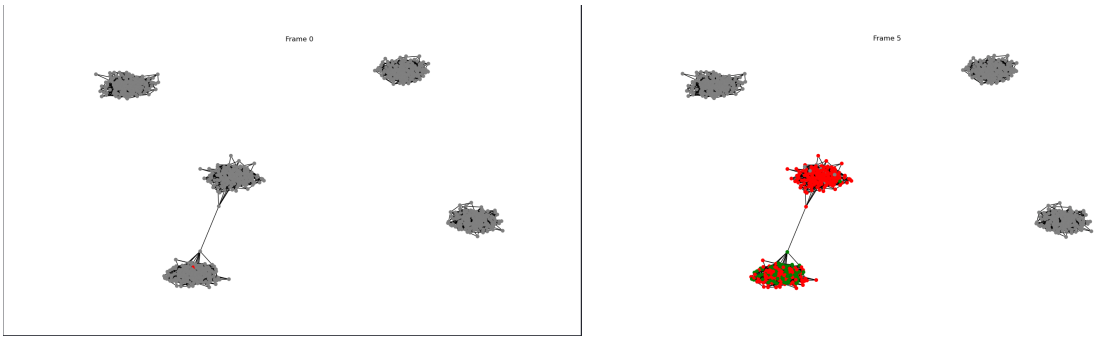


FIGURE 5 – Isolement inter-communautaire.

Dans le cas où seules les communautés du graphe s'isolent entre elles (mais pas les citoyens de ces communautés entre eux), la portée de l'épidémie est également amoindrie, bien qu'on observe que les partitions à l'origine de l'épidémie restent fortement affectées par le virus.

5 Difficultés Rencontrées

Le principal challenge rencontré lors du développement de la simulation fut de mettre en place l'animation du graphe. La librairie *NetworkX* permet des manipulations avancées des graphes et possède une documentation très complète facilitant grandement son utilisation. En revanche, la création d'animation sur base d'un graphe fut plus complexe à mettre en place. Il existe plusieurs extraits de code disponibles sur internet (voir sources) qui implémentent l'animation d'un graphe, et ceux-ci servirent de base à la création du programme, bien qu'il fut nécessaire de développer un code unique et distinct afin de parvenir au produit final concernant notre analyse assez spécifique.

6 Conclusion

L'animation produite permet de visualiser clairement la propagation **exponentielle** d'un agent infectieux à travers une population. Il est particulièrement pertinent d'utiliser des outils tels que l'animation graphique dans le cadre actuelle de la pandémie Covid-19. Un outil visuel offre la possibilité de montrer à quel point un citoyen ne respectant pas des mesures d'isolement en cas de contamination peut très facilement infecter l'entièreté de son entourage et de son réseau et propager un virus de manière exponentielle. Cette simulation est une base solide comme démonstration de l'efficacité des mesures préconisées par les experts en cas de pandémie : s'isoler et limiter ses contacts.

Bibliographie

[Rig10] Michel Rigo. *Théorie des Graphes*. Université de Liège, 2010.

[Deva] Matplotlib Developers. *Matplotlib Documentation*. URL : `https://matplotlib.org/stable/api/_as_gen/matplotlib.animation.FuncAnimation.html`. (accessed : 09.12.2021).

[Devb] NetworkX Developers. *NetworkX Documentation*. URL : `https://networkx.org/documentation/stable/reference/index.html#`. (accessed : 09.12.2021).

[Wol] Christopher Wolfram. *Agent-Based Network Models for COVID-19*. URL : `https://community.wolfram.com/groups/-/m/t/1907703`. (accessed : 09.12.2021).