Rapport Projet C

Marc NGUYEN, Thomas LARDY École des Mines de Saint-Étienne

Notre programme vise à simuler la propagation d'une épidémie dans une population représentée par une grille de taille N×N où chaque individu représente une cellule de cette grille. Nous allons présenter dans ce document la manière dont nous avons conçu le programme, le plan correspondant à l'avancée chronologique de notre raisonnement. De plus, nous avons utilisé Gitlab pour pouvoir gérer et partager notre projet plus facilement, en voici le lien : *https://gitlab.emse.fr/marc.nguyen/projets6-marcnguyen-thomaslardy*

Algorithme principal

Structures de données

D'après l'énoncé, il est plus intéressant de faire une matrice d'adjacence, étant donnée que nous utilisons une grille de personne. De plus, cela permet de visualiser chaque tour plus facilement. Nous allons donc adopter une structure `Population`, contenant la côte de la grille d'adjacence et une grille de `Personne`. `Personne` est une structure contenant la propriété `State`, qui est un enum représentant l'état de la personne. State contient `SAIN`, `MALADE`, `MORT` et `IMMUNISE`.

Algorithme

La fonction centrale `jouerTour` fait passer la grille du temps t au temps t+1. Son principe repose sur le parcours des cellules de la grille : pour chaque cellule, nous énumérons les différents types de voisins et nous appliquons les lois de probabilité de changement d'état leur étant liées. Nous prenons garde de modifier une grille tampon, qui deviendra à la fin de la fonction la nouvelle grille, et non la grille au temps t, pour ne pas fausser les résultats. Pour faire tourner la fonction en boucle, nous vérifions deux choses : la première est que le temps maximum qui a été entré par l'utilisateur n'est pas dépassé et la deuxième est qu'il y a encore des gens contaminés dans la population (c.-à-d. `MALADE` ou `INCUBE`). La fonction `zombiePresent` permet de vérifier ce second point. Si ce n'est pas le cas, le programme doit s'arrêter et renvoyer les statistiques. La position du premier malade et la taille de la grille sont choisies par l'utilisateur, en fournissant les coordonnées x et y du patient zéro et la taille en paramètre du programme.

Analyse des résultats

Il est évident que la propagation varie avec la probabilité d'être `MALADE`. Il est aussi évident que plus la probabilité de mourir est élevé, plus l'infection est mortel. Il est également évident que plus la probabilité d'être immunisé est élevé, moins la maladie peut se répandre et tuer. Cependant, si la probabilité de mourir est égale à la probabilité d'être immunisé, nous obtenons 50% de immunisés et 50% de mort, ce qui permet de vérifier que l'algorithme fonctionne bien. Mais, même si égales, plus la probabilité est haute, plus le status MORT gagne en ampleur. Cela s'explique simplement par la priorité du code à exécuter, c.-à-d le calcul du statut MORT arrive en premier. (Si 50 % de probabilité d'immuniser, 50 % d'être mort et 100% malade, nous obtenons environ 66 % de mort et 33 % de immunisé. Si 10 % de probabilité d'immuniser, 10 % d'être mort et 100 % d'être malade, nous obtenons environ 51% de mort et 49% de immunisés.)

Limitations

Au vue de l'énoncé, nous avons choisi de faire une matrice d'adjacence, mais dans un cas plus général nous devrions utiliser une autre structure de donnée (par exemple, dans le cas où toutes les personnes n'ont pas le même nombre de voisins), comme un tableau de listes d'adjacence.

Extensions

Incubation

Hypothèses

Nous définissons les règles de l'incubation:

1. Toute personne contaminée passe par un stade d'incubation.
2. Une personne incubée devient malade après un temps fixé, identique pour chaque individu.
3. Une personne incubée ne peut ni mourir, ni guérir.
4. Une personne incubée peut transmettre la maladie, de la même manière qu'un malade.

Méthodes

Nous utilisons un nouvel état `INCUBE` pour les personnes ainsi qu'un compteur `duree_incube`, qu'il faut décrémenter à chaque tour, et qui correspond au temps passé en incubation. Si le compteur d'une personne vaut zéro, il faut changer son état en `MALADE`, respectant ainsi l'**hypothèse 2**. De plus, pour une personne saine, un voisin incubé est équivalent à un voisin malade. Il suffit donc de modifier la condition dans le test des voisins de la fonction jouerTour. Les **hypothèses 1, 3 et 4** sont ainsi respectés.

Analyses et Résultats

Nous remarquons que, dans le cas où tous les autres paramètres sont fixés, plus le temps d'incubation est élevé, plus le nombre de personnes touchées par la maladie (c.-à-d morte ou immunisée) est important. Ainsi le temps d'incubation est un paramètre crucial à prendre en compte dans les risques de propagation d'une maladie. Cela est cohérent puisque plus la durée d'incubation est longue, plus le patient incubé expose ses voisins longtemps à la maladie, celui-ci ne pouvant ni mourir ni guérir pendant ce temps (pas de symptôme, donc pas de diagnostic). De plus, nous verrons qu'un patient incubé ne peux pas être le point de départ d'une mise en quarantaine.

Discussions

Cette extension incubation retranscrit de manière assez réaliste un phénomène que l'on retrouve chez de nombreux virus, comme par exemple la grippe. Elle peut avoir un grand impact sur la propagation de la maladie, mais l'on pourrait cependant introduire la possibilité de guérir pendant la période d'incubation, ce qui réduirait cet impact.

Vaccination

Hypothèses

Nous définissons les règles de la vaccination :

1. Un vaccin doit d'abord être découvert (probabilité de découverte).
2. Le vaccin doit se propager (nous supposons que c'est en raison de la communication, ou transport des vaccins, ou préparation), et nous supposons que personne ne craint pas la vaccination (probabilité : 100 %).
3. Les personnes incubées ou malades ne sont pas guéries.
4. Les personnes immunisées sont quand même vaccinées.

Méthodes

Similairement à l'incubation, nous utiliserons un nouvel état : `VACCINE`.

Pour la découverte, nous utiliserons une variable `chance_decouverte_vaccin` pour respecter l'**hypothèse 1**, représentant, donc, la probabilité d'une personne `SAIN` ou `IMMUNISE` découvre un vaccin.

Pour la propagation du vaccin, nous changerons la logique de `jouerTour`, afin que toute personne `SAIN` ou `IMMUNISE`, voisine d'une personne `VACCINE`, soit aussi `VACCINE`, respectant ainsi les **hypothèses 2, 3 et 4**.

Analyses et Résultats

Pour une population de 50×50, avec le patient zéro à (25, 25), quarantaine 0 %, paramètres par défaut. Le paramètre `chance_decouverte_vaccin` varie.

La vaccination est le moyen le plus efficace pour stopper la propagation de la maladie, car la propagation du vaccins est très efficace.

Discussions

En prenant une probabilité élevée, la partie est pratiquement gagnée. Un réglage réaliste serait de prendre une probabilité proche de la réalité. En prenant, par exemple Ebola (2014 - 2016), nous avons 104 000 semaines environ, donc un meilleur réglage serait 1/104000 soit environ 0,001 %.

L'algorithme néglige certaines limites réalistes. Premièrement, nous avons supposé que la propagation est quasi-instantanée, nous pouvions ajouter une probabilité de propagation du vaccin, similaire à celui de la propagation de la maladie, pour représenter la probabilité qu'une personne soit convaincu/informé ou que le vaccin marche sur cette personne. Deuxièmement, nous aurions pu faire en sorte à ce que le vaccin ait une probabilité de tuer (par allergie, ou par mauvaise atténuation du pathogène), mais nous avons supposé que les chercheurs sont efficaces.

Quarantaine

Hypothèses

Nous définissons les règles de la quarantaine :

1. Une quarantaine doit être décidée (probabilité d'action).
2. Toute personne visiblement MALADE ou MORTE est incluse dans la quarantaine.
3. Toute autre personne proche d'une personne MALADE ou MORT sont inclus dans la quarantaine faisant office de cordon sanitaire.
4. La mise en place de la quarantaine est instantanée.
5. Les personnes étant dans la quarantaine continuent de "jouer un tour" mais n'affecte pas ceux en dehors de la quarantaine. Et inversement.
6. La quarantaine n'est pas éternelle.

Méthodes

Cette fois-ci, nous n'allons pas ajouter un état, mais ajouter une propriété à `Personne` qui se nomme `est_quarantaine` afin de pouvoir respecter l'**hypothèse 5**.

Pour l'**hypothèse 1**, nous utiliserons, cependant, la même logique que pour la mise en place de la quarantaine, donc nous mettons en place la variable `chance_quarantaine`.

Pour l'**hypothèse 6**, nous utiliserons la même logique que pour la mise en place de l'incubation, donc nous mettons en place la variable `duree_quarantaine`.

Pour les **hypothèses 2, 3 et 4**, nous allons adapter un algorithme de parcours de graphe, précisément l'algorithme de parcours en profondeur. Les modifications sont :

- La variable `cordon_sanitaire` décroît si la personne à marquer n'est ni `MORT`, ni `MALADE`.
- Toute personne non marquée a `cordon_sanitaire` égal à 0, et donc, ceux qui sont marquées l'ont strictement supérieur à 0.

Notez également qu'il s'agit de la version avec récursivité et non itérative, car la callstack suit la logique LIFO, ce qui simplifie le code.

Analyses et Résultats

Pour une population de 50×50, avec le patient zéro à (25, 25), vaccination 0 %, paramètres par défaut (`duree_quarantaine = 20` et `cordon_sanitaire = 5 = duree_incube + 1`). Le paramètre `chance_quarantaine` varie.

Plus la probabilité est élevée, moins la propagation de la maladie est possible. Entre 50 % et 100 %, la maladie n'arrive généralement pas à se propager. Entre 50% et 0.1%, la quarantaine peut laisser passer la propagation en raison du "mauvais" réglage de `cordon_sanitaire`. Elle arrive, cependant, à limiter la propagation. Si nous incrémentons `cordon_sanitaire` de 1, dès la mise en place de la quarantaine, la propagation est totalement limitée. Si nous réglons de telle sorte à ce que `duree_quarantaine` soit faible (ce qui est en réalité absurde), la propagation se fait dans tous les cas (`cordon_sanitaire = 7` et `chance_quarantaine = 1`).

Discussions

La modélisation est plutôt réaliste, mais nous pouvons discuter sur quelques points. Premièrement, la quarantaine est instantanée. Nous aurons pu propagation de la quarantaine en prenant les même logiques que la propagation de la maladie et le rayon du cordon sanitaire pour simuler la propagation et la taille de la zone de quarantaine.

Tableau de bord et Graphique

Méthodes

Nous utilisons les structures `Data` qui contient un tableau dynamique `liste_statistiques` et le nombre de tours. `liste_statistiques` est une liste de structure Statistique. Statistique est une structure contenant `nb_IMMUNISE`, `nb_MALADE`, `nb_VACCINE`, `nb_MORT`, `nb_SAIN` et `nb_INCUBE`, tous en `long`. Une liste chainée aurait bien pu être utilisée, mais le choix a été complètement arbitraire, car la complexité n'est pas importante pour faire un simple graphique.

Résultats

Les métriques les plus intéressantes sont les répartitions des états `SAIN`, `MORT`, `IMMUNISE`, `VACCINE` et le nombre de tours. Nous pouvons aussi rajouter la vitesse maximale de l'apparition d'un état précisément `IMMUNISE`, `INCUBE`, `MALADE` et `MORT` (nous rappelons que `VACCINE` se propage de manière quasiment instantanée, précisément +4n par tour où n est le nombre de personnes vaccinées.

Le tableau ressemble donc à :

```
-------------------------------------------------------------------
| Tours |   SAIN   |   MORT   |   IMMU   |   VACC   |   Total   |
| ----- | -------- | -------- | -------- | -------- | --------- |
|    45 |      105 |      714 |       81 |        0 |       900 |
|       |  11.66 % |  79.34 % |   9.00 % |   0.00 % |     100 % |
|-----------------------------------------------------------------|
| Vit. IMMU max | Vit. INCUB max | Vit. MAL max | Vit. MORT max |
| ------------- | -------------- | ------------ | ------------- |
|             7 |             21 |           13 |            54 |
-------------------------------------------------------------------
```

Le graphique représente également la répartition des états. Ayant peu de moyen pour faire un graphique de bonne qualité, nous avons préféré faire un graphique ASCII, permettant ainsi de l'afficher sur le terminal. Chaque colonne représente un tour. Un caractère représente une proportion d'individu de la population dans cet état. Nous obtenons ceci :

```
.....................**********$$$$$$$$$$$$$$
.................................**************
..............................uu.uuu
............................oou  u
.........................oouuu
.......................ouuuu
.....................ooouuu
...................oouuuu
.................ouuuuu
................oouuuuuu
..............ouuuuu
.............ouuuu
...........uouuuu
..........ouuuu
..........ouuuu
........uuuu u
.......uuuuuu
......uuu uu
.....uuu
.
```

Le graphique n'est évidemment pas précis étant donnée l'arrondi. Pour régler cela, les variables `tours`, `limite` et `hauteur` sont mises à dispositions.

Conclusion

Ce projet nous a permis de nous familiariser avec la simulation de propagation d'épidémie, et notre programme final représente de manière cohérente ce phénomène. Nous avons pu intégrer, dans un graphe, une population ainsi que des règles de simulation permettant de jouer à un "jeu de la vie". L'ajout d'extensions a été particulièrement intéressant, notamment la réflexion sur la manière d'essayer de retranscrire des phénomènes réels, telle que la mise en quarantaine, en algorithme. Au final, ce projet, en plus d'être instructif, a été plaisant à réaliser.

Annexe

Aide du programme

```
Projet Semestre 6, Propagation d'une épidémie dans une population par Marc NGUYEN et Thomas LARDY
en Mar-Apr 2019

Usage: ProjetS6-MarcNGUYEN-ThomasLARDY <x> <y> <cote> [options...]

Population Options:
  -t,  --tours            tours max de la simulation              [défaut: 100]

Simulation Options Générales:
  -b,  --mort             [0, 1] proba de mourir par la maladie    [défaut: 0.5]
  -g,  --immunise         [0, 1] proba d'être immunise             [défaut: 0.1]
  -la, --malade           [0, 1] proba de contamination            [défaut: 1.0]

Output Options:
  -od, --data             nom de données brutes               [défaut: data.txt]
  -og, --graph            nom du graphique                 [défaut: graphique.txt]
  -ot, --tableau          nom du tableau de bord   [défaut: tableau de bord.txt]
  -li, --limite           limite de char/ligne du graphique ASCII   [défaut: 80]
  -ha, --hauteur          hauteur du graphique ASCII                [défaut: 20]

Extension Incubation:
  -di, --duree-incube     durée d'une incubation                     [défaut: 4]

Extension Vaccin et Quarantaine:
  -q,  --quarantaine      [0, 1] proba d'une quarantaine          [défaut: 0.1]
  -dq, --duree-quarantaine  durée d'une quarantaine                [défaut: 20]
       --cordon           taille du cordon sanitaire                 [défaut: 5]
  -v,  --vaccin           [0, 1] proba de développer un vaccin   [défaut: 0.001]

Autres:
  -h,  --help             Affiche ce dialogue
```