Sujet projet: Modèle Toy case SEIR multi-agent

Date de rendu : 05/04/2024 (provisoire, date de rendu à voir selon le déroulement), 60% de la note finale

1 Objectif

L'objectif est de développer un modèle multi-agent simple de propagation d'une maladie dans une population.

2 Le code

10 points

• On place des individus sur une grille en 2 dimensions. Les individus sont caractérisés par deux états discrets : leur statut et le temps écoulé dans ce statut, et par trois paramètres de durée, dE, dI, dR.

• Le statut peut prendre quatre valeurs : statut = S, E, I, R. (Susceptible, Exposed, Infected et Recovered)

• dE, dI et dR correspondent à la durée de vie du statut lorsqu'il est égal à E, I et R, respectivement.

• L'espace est de forme toroïdale (grille périodique). Plusieurs agents peuvent être situés sur la même cellule.

• Temps : le temps est discret. Un pas de temps correspond à un jour.

• Déplacement : À chaque pas de temps, tous les individus se déplacent aléatoirement vers une autre cellule. La nouvelle position est choisie au hasard dans la grille (pas nécessairement les cellules voisines).

• Infection : À chaque pas de temps, tous les individus susceptibles (statut = S) vérifient s'il y a un ou plusieurs individus infectieux dans leur voisinage (individus avec statut = I). Si des individus infectieux sont présents, l'individu susceptible devient exposé (statut = E) avec une certaine probabilité.

• Exposition : À chaque pas de temps, chaque individu en état E devient I si le temps écoulé dans l'état E est supérieur à la durée de la période d'exposition dE.

• Récupération : À chaque pas de temps, chaque individu en état I devient R si le temps écoulé dans l'état I est supérieur à la durée de la période infectieuse dI.

• Perte d'immunité : À chaque pas de temps, chaque individu en état R devient S si le temps écoulé dans l'état R est supérieur à la durée de la période d'immunité dR.

• Planification : L'ordre d'exécution des agents est aléatoire. Les changements d'état sont asynchrones, c'est-à-dire que le calcul du nouvel état d'un individu est effectué lorsque l'individu est choisi au hasard, il n'y a pas de mémorisation des états précédents des individus.

• Émergence : Le nombre d'individus avec statut S, E, I et R.

• Interaction : Un contact existe entre deux individus s'ils sont à proximité immédiate dans l'espace. Un individu est en contact avec tous les individus situés dans les 8 cellules autour de sa propre cellule (voisinage de Moore), plus ceux de sa propre cellule. L'espace est torique afin d'éviter les artefacts de limites.

• Stochasticité : Les individus se déplacent aléatoirement, sautant d'une cellule à l'autre n'importe où dans la grille.

• Infection : la probabilité p pour un individu susceptible S de devenir exposé E est calculée comme suit : $p = 1 - e(-0.5 * Ni)$ avec Ni le nombre d'individus infectieux dans le voisinage (8 cases, voisinage de Moore) et 0.5 la force de l'infection. Si un tirage dans la distribution uniforme est Uniform [0; 1] < p, alors l'individu avec statut S devient E. (Exemple : double proba = 1 - exp(-0.5*Ni) if (random < p then agent.state = Exposed)).

- Initialisation

• Nombre total d'individus : 20000.

• Taille de la grille : 300x300.

• Statuts (graines) pour le PRNG : 100 simulations, donc 100 statuts de départ si parallèle, ou un statut de départ si séquentiel. (Si simu de 15 minutes, alors 1500 minutes = 25 heures = 1 jour …). Vous pouvez faire tourner le code sur un cœur du serveur Ada de l'Isima par exemple, pendant 24h (voir la commande nohup), ou sur votre machine personnelle (à faire tourner 24h).

• Type de PRNG : Mersenne Twister

• 19980 individus ont leur statut égal à S.

• 20 individus ont leur statut égal à I.

• Pour tous les individus, les paramètres de durée de vie du statut sont tirés aléatoirement dans la distribution exponentielle avec les paramètres 3, 7 et 365 pour E, I et R respectivement. Par définition, les valeurs des paramètres ne changent pas pendant la simulation.

```
negExp(double inMean)

    return -inMean * log(1 - genrand_real2());

dE = negExp(param)
```

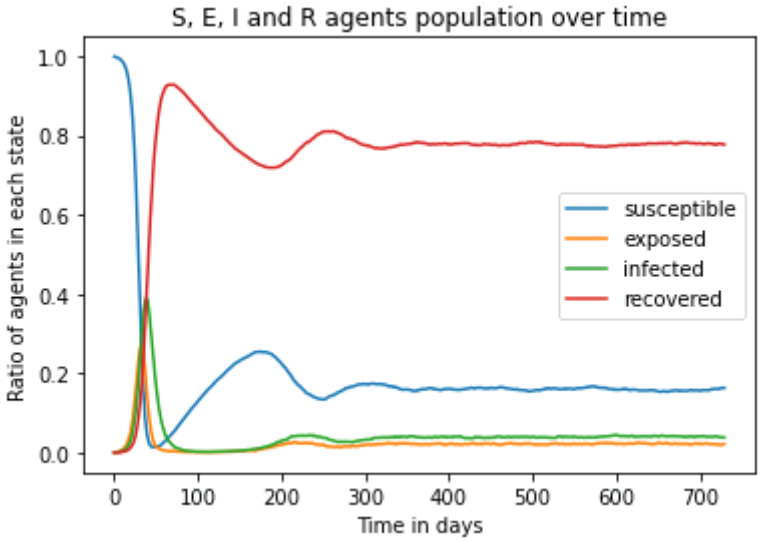
• Les agents sont placés aléatoirement sur la grille.

3 Rendu des résultats

10 points

Votre code devra générer des fichiers contenant les résultats du nombre d'agents dans l'état S, E, I et R pour chaque itération (730 itérations).

Vous devrez ensuite utiliser un notebook Jupyter afin de lire ces résultats, et afficher un graphique des moyennes. Les résultats devraient ressembler à cela :



Vous générerez des fichiers CSV pour chacune de vos réplications (100 réplications).

Vous affichez les 100 courbes (pour chaque réplications), et la courbe de la moyenne.

Vous devrez rédiger un rapport pour le projet, en vous basant sur les exemples de rapports fournis.

Le Jupyter Notebook peut servir pour faire une partie du rapport, voir le rapport en entier.

4 Details

Travail en groupe de 2 ou 3 personnes.

Vous devez rendre un lien github/gitlab, avec un readme indiquant les noms des personnes dans le projet.

Le lien doit contenir les codes sources Java/C++, et Jupyter notebook, avec les figures, etc.

Indiquez le PRNG utilisé, et la manière dont on peut reproduire les résultats (si le Jupyter notebook est bien fait, la reproduction des résultats est simple).

(Vous pouvez utiliser le "!" pour executer une commande bash. Par exemple : "!make", puis "!./runExperiments.sh", puis vos analyses en Python)

Si vous n'avez pas utilisé github ou gitlab pour votre projet, vous pouvez tout m'envoyer dans un fichier zip (mais je vous conseille git).

Vous pouvez coder en Français ou en Anglais.

Explications pour dE, dI et dR, qui seront des attributs de la classe Individu/Humain :

Ces trois attributs sont définis lors de la création de l'individu, donc directement dans le constructeur. Ils permettent de savoir combien de temps l'individu va rester dans l'état E, I et R. Ce sont donc des valeurs entières (int), qui correspondent à un nombre d'itérations (jours).

Une fois défini pour un individu, ces valeurs ne changeront jamais.

Pour calculer ces valeurs, vous utilisez une loi exponentielle négative :

function negExp(double inMean) { return -inMean * log(1 - genrand_real2()); } // Avec genrand_real2() retournant un nombre aléatoire [0;1[.
Dans le constructeur de Individu : dE = (int) random.negExp(3); dI = (int) random.negExp(7); dR = (int) random.negExp(365);

Pour la transmission de l'infection (si un individu S, lors de son déplacement, doit devenir E) :

double probability = $1 - \exp(-0.5 * \text{NbVoisinsMalade})$;

Important :

Vous devez faire attention à initialiser votre PRNG (le générateur de nombre aléatoire que vous utilisez) avec une seed (init_genrand ou initby_array).

Pas avec time ! Faites attention si par défaut votre PRNG s'initialise avec le time comme seed, ce sera 1 ou 2 points en moins.

Faites également attention au comportement de votre PRNG : Le mieux est probablement d'avoir une instance globale de votre PRNG pour toute l'application pour éviter tout recouvrement.

Si vous faites vos 100 réplications en séquentiel, c'est suffisant. Si vous faites en parallèle, alors il faudra initialiser différemment chaque réplication.