

# Technique d'I.A.

## Systèmes multi-agents

Compte rendus de projet

Par Aristide DULLIN et Hedwin BONNAVAUD

### Table des matières

Simulation de départ	2
Durée de contamination	2
Limite d'âge et du système	3
Fonctionnalités additionnelles	4
Impact de différent critères sur la propagation de l'épidémie	6
Confinement	6
Hôpital	7
Porteurs sains	8
Fuite des malades	9
Immunité et survie de la population	11
Références	11

### Résumé

Le but de ce projet sous Netlogo est de simuler la propagation d'une épidémie en fonction de différent critères. Netlogo permet la modélisation de systèmes multi-agent et de mettre en évidence les réactions d'un système par rapport aux variations de critères.

## Simulation de départ

Pour commencer ce projet de simulation de propagation de virus, nous avons pris comme point de départ un projet existant et disponible depuis NetLogo dans la rubrique Models Library>Biology>Virus.

Ce modèle est un système multi-agents faisant se déplacer des humains de manière complètement aléatoire.

Ces humains peuvent être infectés ou non, et dès qu'une entité infectée se déplace sur le même patch qu'une entité saine, cette dernière a une certaine probabilité d'être contaminée.

### Durée de contamination

Le temps depuis lequel une personne est contaminée est stocké dans ses informations. À chaque étape de son comportement, une entité malade va vérifier si cette période ne dépasse pas une durée maximale (qui n'est pas paramétrable).

Lors d'un dépassement de cette durée, l'entité aura une certaine probabilité de mourir. Cette probabilité peut être modifiée par l'utilisateur.

Si l'entité ne meurt pas, elle deviendra immunisée contre la maladie pendant 1 an.

En effet, cette simulation de départ cherche à être réaliste quant aux durées. Dans toute la simulation, on utilise la semaine comme unité de temps. Un tick (un cycle d'horloge) correspond à une semaine.

À chaque étape de la vie d'une entité, son âge est incrémenté de 1, ce qui correspond à une durée de vie de 1 semaine. En conservant cette logique, on peut définir la fin d'une immunité (dont nous avons parlé au paragraphe précédent) à 52, ce qui correspond au nombre de semaines dans une année.

Concernant la durée de vie d'une entité, cette système cherche à simuler une durée de vie variable et aléatoire d'une entité à l'autre. Pour cela, la simulation génère un âge aléatoire entre 0 et l'âge maximum "lifespan". Cette constante globale est initialisée avec la valeur  $50 \times 52$ . Etant donnée que l'unité de temps est la semaine et qu'une année contient 52 semaine, nous pouvons dire qu'une entité ne peut pas vivre plus de 50 ans.

## Limite d'âge et du système

Cette attribution de l'âge, aléatoire et uniforme entre 0 et 50, avec une mortalité à 50 ans, donne des durée de vie équivalente à une attribution de l'âge 0 à chaque entité lors de son initialisation, et une mortalité aléatoire et uniforme entre 0 et 50 (ce qui aurait été surement plus logique ainsi).

Cette comparaison nous permet de constater deux erreurs. Nous savons que la durée de vie d'une entité peut dépasser 50 ans, et qu'elle est n'est pas uniformément répartie entre 0 et 50 ans.

Comme on peut le constater sur le graphique ci-dessus, la répartition des durées de vie est très loin d'être uniforme entre 0 et 50 ans. Même si la distribution ne correspond pas exactement à une loi normale, cette dernière serait cependant bien adaptée.

Tous ces paramètres sont des éléments qu'il pourrait-être intéressant de changer pour en observer les conséquences. Cependant, pour obtenir une simulation plus réaliste, il existe d'autres fonctionnalités que nous pourrions ajouter. Dressons une liste des fonctionnalités à développer avant d'en analyser les conséquences.

## Fonctionnalités additionnelles

Voici les modifications que nous chercherons à ajouter au modèle de départ :

- Modifications des paramètres énoncés précédemment,
  - changement de la génération de l'âge
    - augmentation de la durée de vie des entités
    - modification de la répartition des durées de vie

Concernant la durée de vie des entités, l'âge de ces dernières est réparti aléatoirement entre 0 et l'âge moyen, et est initialisée à 0 pour les nouveaux nés.

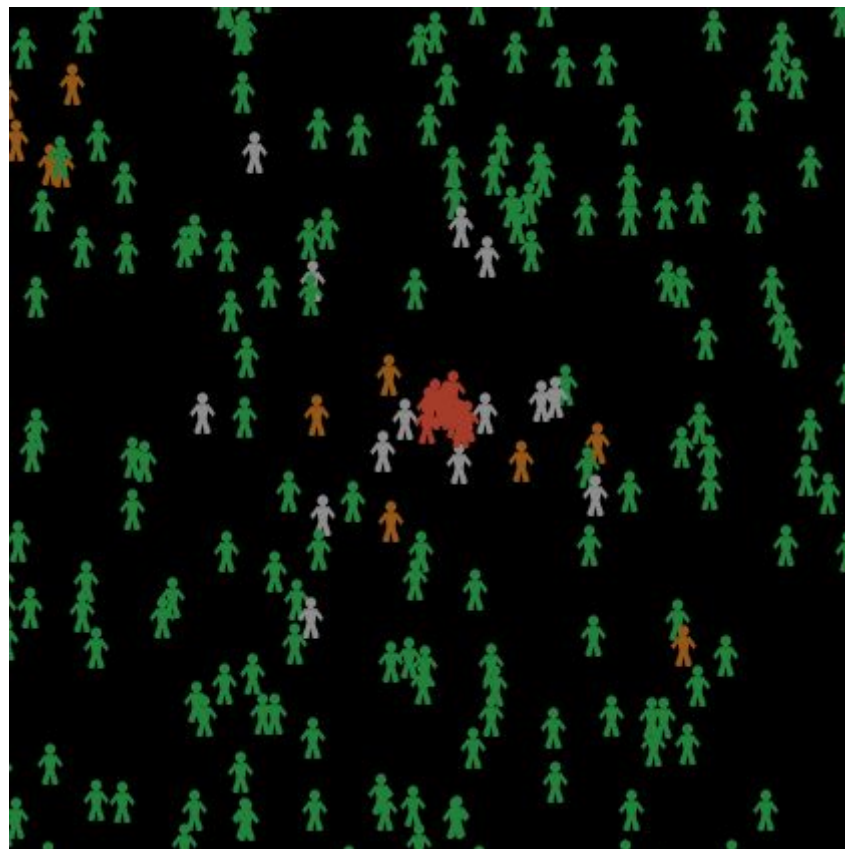
Lors de la génération d'une entité (dans les deux cas énoncés précédemment), la durée de vie de cette dernière est initialisée selon une loi normale de moyenne 82 ans et d'écart type 5 ans (même si ce n'est pas fidèle à la réalité puisque la durée de vie ne suit pas une loi normale, cela permettra d'avoir des évolutions plus réaliste).

code de l'initialisation des tortues au début de la simulation :

```
"set age random-float lifespan  
set max-age random-normal lifespan stand-deviation"
```

- Les personnes infectées sont mises en quarantaine dans des hôpitaux.

A chaque "tick", les tortues se déplacent aléatoirement. Pour simuler l'existence d'un hôpital, les tortues malades se déplaceront vers le centre de la carte.



Ainsi, nous pouvons simuler une quarantaine, période pendant laquelle la contamination est plus faible dans un groupe.

De plus, il est possible de paramétrer la probabilité de survie du virus dans l'hôpital en plus du cas général.

Pour savoir si une entité se trouve dans l'hôpital, il nous suffit de mesurer sa distance au point 0,0 avec la fonction `distancexy 0 0`. Si cette valeur est inférieure à 2, alors on considère que l'entité est dans l'hôpital.

- La population peut-être confinée par l'utilisateur.

Un taux de confinement est ajouté, entre 0 et 100, indiquant le pourcentage de temps moyen pendant lequel une personne reste immobile.

Une valeur de 0 indique aucun confinement, alors qu'une valeur de 100 indique un confinement totale.

Pour cela, nous avons modifié la fonction `move` permettant à une entité de se déplacer.

Si une valeur aléatoire tirée uniformément entre 0 et 100 est supérieure ou égale à la valeur choisie par l'utilisateur, alors la tortue peut se déplacer. Cet effet n'affecte pas les entités malades se dirigeant vers l'hôpital.

Au premier abord, cela semble seulement ralentir la simulation. Mais étant donné que seul le temps de déplacement des entités est affecté, et pas les autres durées comme leur temps de guérison, alors on peut observer des vraies évolutions sur la propagation du virus.

- Une partie des entités malades sont des porteurs sains.

Afin de ne pas supprimer toute contamination possible avec une combinaison d'hôpital et de confinement, et de gagner en réalisme, nous ajoutons le moyen de choisir quel proportion des malades est porteur sain. Ces derniers ne seront pas affectés par l'effet d'hôpital.

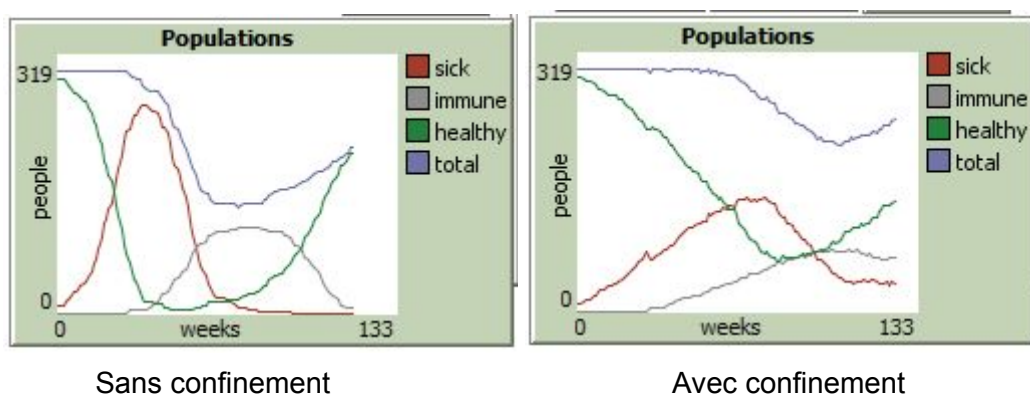
Ce sont donc des malades qui se déplacent comme tout le monde en respectant le confinement imposé par l'utilisateur.

## Impact de différent critères sur la propagation de l'épidémie

### Confinement

Dès la réalisation de la fonctionnalité de confinement des entités, nous avons pu remarquer des conséquences similaires à celles qui nous ont été expliqué concernant le confinement que nous vivons actuellement.

Voici l'évolution de la pandémie (en rouge), à gauche pour une simulation sans confinement, et à droite pour une simulation avec confinement.



Tout d'abord, il est important de souligner que seul le paramètre de confinement a évolué entre ces deux situations, et qu'elles ont été réalisées sur la même durée.

Nous pouvons alors remarquer que la courbe des entités contaminées est "aplatis". En effet, les entités se déplaçant moins, elles entrent moins en contact, et une entité malade risque moins de contaminer une entité saine, donc il y a un moins gros pic d'infectés.

Nous pouvons également observer que l'impact de l'épidémie sur la courbe de la population totale est moins marqué dans le cas d'une simulation avec confinement.

Cela est dû au fait que l'épidémie progressant plus lentement, le nombre de mort croit lui aussi plus lentement et de ce fait la population a plus le temps de se régénérer avec des naissances.

Ces deux simulations ont été effectuées sans hôpital, donc il n'influence pas le taux de guérison.

Enfin, nous pouvons remarquer que l'épidémie se maintient plus longtemps avec un confinement que sans confinement (voir la courbe rouge). En effet, la population ayant le temps de se régénérer avec de nouvelles naissances, le virus a toujours de nouvelles personnes à contaminer, alors que dans le cas sans confinement, son impacte sur la population est trop important. Les entités qui n'ont pas succombé à l'épidémie sont alors soit infectées, soit immunisées. Le virus finit alors par disparaître. Nous pouvons nous en rendre compte en observant la courbe des entités saines (courbe verte).

Au vu des dernières observations, il semble indispensable, pour obtenir une simulation plus réaliste, de simuler la présence d'un hôpital. En effet, si les chances de survies sont plus importante au sein d'un hôpital qu'en dehors, nous devrions pouvoir observer des conséquences du confinement plus net et plus réaliste sur la population.

## Hôpital

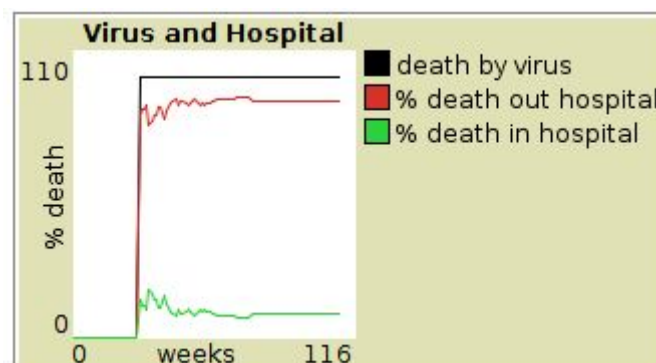
En imaginant ce que serait une simulation avec hôpital et afin de conserver la fiabilité de notre simulation, nous nous sommes rendu compte que les détails suivants doivent être respecter.

Tout d'abord, les entités se rendant à l'hôpital peuvent contaminer toutes les personnes sur leur chemin. Il faut donc que les entités se déplacent vers une zone prédéfinie dans laquelle elles seront mises en quarantaine.

Ensuite, si les entités malades sont déplacées dans les hôpitaux et que les entités saines se retrouvent confinés, la maladie devrait disparaître automatiquement puisqu'il n'y aurait plus aucun contact entre les personnes saines et les personnes malades.

Pour gagner en réalisme, nous avons donc pensé à un facteur de malades se rendant à l'hôpital. C'est à dire qu'une partie des malades se rendront à l'hôpital pour se faire soigner, alors que les autres non, soit parce qu'ils estiment ne pas être suffisamment malades pour celà, soit parce qu'ils sont porteurs sains.

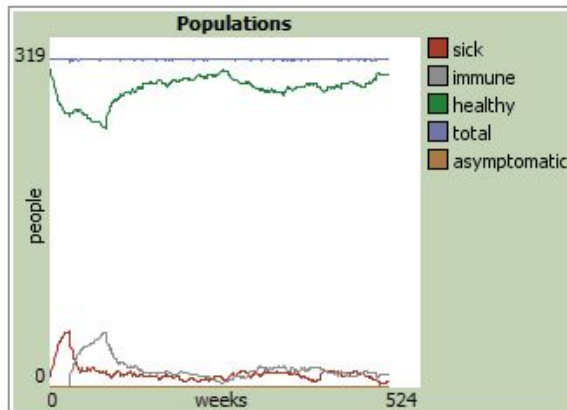
Les personnes malades qui sont dans l'hôpital ont moins de chances de mourir.



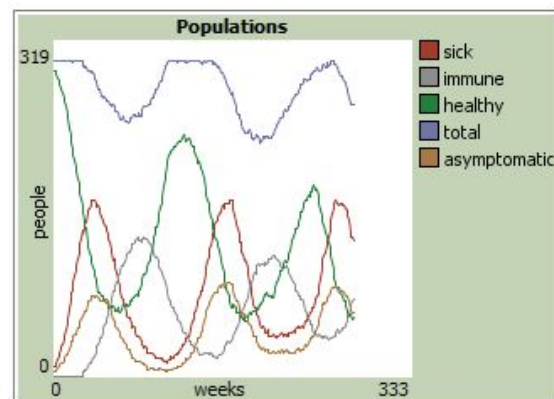
## Porteurs sains

Pour avoir une idée de l'effet de la présence de porteurs sains, nous allons réaliser deux simulations. Une avec hôpital et sans porteurs sains, et une avec hôpital et avec porteurs sains (50% des malades). Les deux se déroulant sans confinement.

Voici, à gauche, le résultat de la simulation sans porteurs sains, et à droite, le résultat de la simulation avec porteurs sains :



sans porteurs sains



avec porteurs sains

On peut alors remarquer que la présence d'un hôpital contraint le virus à survivre en contaminant les quelques personnes qui s'y rendent. Cependant, ce dernier parvient à survivre car la population étant moins contaminée, elle est plus nombreuse (courbe grise en haut du graphique), et comme elle est plus nombreuse elle permet mieux au virus de se propager, et ainsi de suite.

Ce phénomène peut expliquer les vagues du graphique de droite. Chaque courbe dépend des autres. Cependant, les oscillations sont plus marquées à droites et la stabilisation de ces évolutions met plus de temps car les personnes asymptomatiques contaminent beaucoup de personnes dans la populations alors que les personnes malades en hôpital servent de générateurs à personnes immunisées pour maintenir un niveau de population correcte, et donc un taux de naissances suffisamment élevé pour permettre la propagation du virus.

Suite à ces observations, nous avons cherchés à préciser le comportement des tortues avec deux nouvelles fonctionnalités.

Tout d'abord, nous savons que de manière générale, une personne malade ne vas pas directement à l'hôpital. La plupart des personnes atteinte par des symptômes grippaux de vont pas systématiquement à l'hôpital (Dans le cas du covid-19 en France, 25000 personnes se trouvent à l'hôpital pour 55253 cas avec symptômes soit 45% des cas à l'hôpital).



Nous avons donc ajouté un paramètre ajustable “sick-peoples-goes-to-hospital” indiquant la proportion des cas se rendant à l'hôpital.

En ajustant ce paramètre, nous faisons varier l'utilité de l'hôpital, et rapproche les résultats de notre simulation à ce qu'ils étaient avant l'ajout de l'hôpital, mais cela nous donne une simulation plus réaliste.

## Fuite des malades

Ensuite, nous nous sommes intéressés aux comportements des personnes saines. En effet, certaines personnes saines évitent de s'approcher d'une personne présentant des symptômes visibles.

Nous avons donc ajouté un paramètre ajustable “peoples-escape-virus” permettant de choisir la proportion de personnes saines étant phobique des personnes présentant des symptômes du virus.

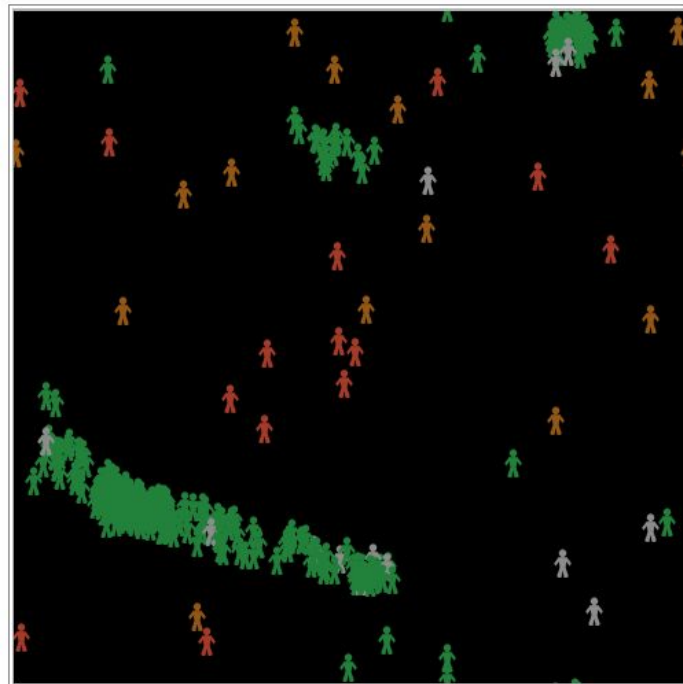
C'est à dire que ce caractère touchera une partie de la population, et que ces personnes éviteront de rentrer en contact avec une personne malade présentant des symptômes. Elles ne feront cependant rien à la vue de personnes malades asymptomatiques, puisqu'elles n'auront aucun moyen de savoir que ces personnes sont malades.

Pour demander à une tortue de fuir une personne malade, nous exécutons le code suivant,

```
let closest min-one-of turtles with [sick? and not asymptomatic?] [distance myself]
ifelse closest = NOBODY
[
  ;; if closest doesn't exist because there's no sick turtles left
  move-randomly
]
[
  ;; else : fly from the closest sick
  face closest ;; face the closest sick
  fd -1 ;; Go backwards
]
```

La méthode consiste à trouver la personne malade la plus proche, puis de lui faire face et reculer.

Ceci nous permet d'avoir des groupements de personnes saines. C'est à dire que la majorité des personnes saines vont finir par se tenir au même endroit, puis à suivre le même mouvement car elles seront toutes soumises aux mêmes personnes malades.



Ce comportement, en plus de ne pas être réaliste, apporte un inconvénient pour la population saine. En effet un groupement peut-être encerclé de personnes malades, et finir contaminé. Les personnes étant très proches les unes des autres, la contamination du groupe est quasiment instantanée pour l'ensemble du groupe.

Cependant, l'avantage de fuir la maladie reste très positif et efficace pour la population saine.

Afin d'avoir un comportement plus réaliste, nous cherchons à empêcher la formation de groupements, qui ne devraient pas exister avec pour seul raison que les personnes fuient la maladie. Nous avons donc pensé que la formation de groupes était due au fait que trop de tortues saines adoptent un comportement identique trop facilement.

En effet, avec le code présenté précédemment, les tortues saines détectent les tortues malades les plus proches sans limites de distance. Ainsi, si il se crée une zone assez large ne contenant pas de tortues malades, toutes les tortues saines fuyant la maladie dans cette zone vont adopter le même comportement jusqu'à la contamination.

En remplaçant la condition de fuite par :

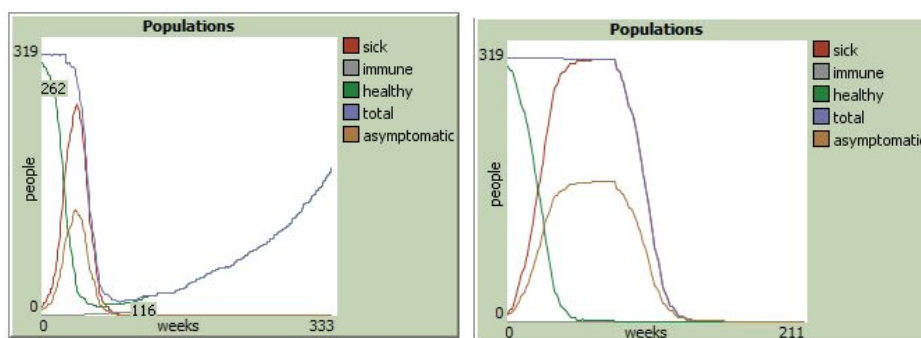
```
ifelse closest = NOBODY or distance myself > 5  
[move-randomly]
```

Les tortues ne fuiront les tortues saines les plus proches que si elles sont suffisamment proche pour être vu. Nous avons estimé que la distance de 5 donnait une simulation plus réaliste.

## Immunité et survie de la population

En plus des remarques faites précédemment, nous avons pu remarquer que le développement de l'immunité des entités contre le virus est indispensable à sa survie. Nous avons pu remarquer que, dans le cas générale, les personnes immunisées peuvent permettre le maintien d'un niveau de population correcte, et permettre ainsi au virus de continuer à se propager. S'il n'y a pas d'immunité, la taille de la population chute, et le virus disparaîtra alors. La survie de la population dépend alors principalement de la durée de vie du virus sur un individu.

Voici deux simulations avec à gauche, une durée de vie du virus sur un individu normale, et à droite, une durée de vie anormalement grande :



ces résultats sont très probablement dû au fait que notre modèle manque de précision.

Grâce à Netlogo, nous avons pu créer de nombreuses simulations dans des contextes différents et observer rapidement les réactions qu'elles causaient au système

## Références

<https://www.cairn.info/revue-economique-2008-5-page-941.htm#>

Lien vers une revue économique étudiant la durée de vie humaine, source du graphique présenté dans la partie décrivant la simulation de départ.